Revisiting SSL: A Large-Scale Study of the Internet's Most Trusted Protocol

Johanna Amann[*], Matthias Vallentin[§],
Seth Hall[*], and Robin Sommer[*ǂ]

TR-12-015

December 2012

Abstract

Much of the Internet's end-to-end security relies on the SSL protocol along with its underlying certificate infrastructure. We offer an in-depth study of real-world SSL and X.509 deployment characteristics from an unprecedented vantage point, based on a data set of more than 1.4 billion SSL sessions collected at the border of five operational sites. Our contributions are two-fold: First, we revisit results from past work with a recent data set that allows us to reassess previous findings and identify recent trends. Second, we provide a detailed study on a range of further SSL/X.509 deployment properties that have not yet seen the attention they deserve, including the intricate web of intermediate certificate authority (CA) relationships, characteristics of SSL session reuse, usage of vendor-specific protocol extensions, and non-standard CA root hierarchies. While in general we find that today's SSL deployment functions well, we also identify new support for the inherent weaknesses of the system. Along the way, we gain deep insight into specifics and oddities of SSL/X.509 usage, including a surprising difficulty of aligning certificate validation with what typical browsers do.

*International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, California, 94704
§ University California, Berkeley, 2200 University Drive, Berkeley, California 94720
ǂ Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, California, 94720

This work was partially funded by the Deutscher Akademischer Austausch Dienst (DAAD) through a postdoctoral fellowship.

1. INTRODUCTION

The Secure Socket Layer (SSL) protocol provides a centerpiece of the Internet's end-to-end security, along with its underlying X.509 certificate infrastructure. However, most of that technology predates the modern Web, and the current system exhibits well-known weaknesses, such as its questionable trust bootstrapping which relies on >100 root certificates shipping with common browsers. Past studies—both academic and otherwise—have examined aspects of the SSL/X.509 combo, most comprehensively by Holz et al. [17] recently at IMC 2011. Our work continues this line of work by consolidating past efforts with a recent, broader data set that allows us to reassess previous findings and identify trends. Going further, we study a range of additional SSL/X.509 deployment properties that have not yet seen the attention they deserve, including the intricate web of intermediate CAs, vendor-specific protocol extensions, SSL session reuse, and independent CA root hierarchies used outside of the web context. While in general we find that today's Public Key Infrastructure (PKI) deployment functions rather well—a result the more surprising the further one digs—we also identify new support for the inherent weaknesses of the system. Along the way, we gain deep insight into specifics and oddities of SSL/X.509 usage, including a surprising difficulty of aligning certificate validation with typical browsers.

Our data[1] comprises approximately 1,000 hours of SSL activity from about 115K users at five research institutions, yielding a total set of about 1.4 billion SSL sessions, extracted from live network traffic independent of their TCP ports. With these sessions come about 1.8M certificates and about as many unique certificate chains. About 8% of the certificates validate against the Mozilla root store, while most of the remaining ones use an alternative root hierarchy. Due to privacy constraints we cannot release the connection-level data, however we make all the web certificates available to the community in the form of a public online notary service, with an interface compatible to existing systems.

From a broader perspective, we also see our work as a case-study on collaborating successfully with network operations. The challenge of getting access to real-world network data is a recurring theme in our scientific community, as researchers often find operators reluctant to provide input for their studies. Our work provides evidence that one can overcome such hurdles by addressing the constraints that the operations community face.

We structure the remainder of this paper as follows: §2 summarizes related work. In §3, we discuss our analysis methodology, including the collection process we deploy for this study; and §4 introduces our data set. In §5, we examine the data in the context of previous efforts to understand similarities and trends; and we then dig deeper into not yet explored aspects in §6. §7 presents the public notary service we set up. Finally, §8 discusses our experiences working with operations, before we conclude in §9.

[1]Please note that this technical report shows the state of our work as of April 2012.

2. RELATED WORK

The past years witnessed a noticeable increase of security incidents involving certificate authorities, rendering the global PKI infrastructure an attractive subject of study. The Electronic Frontier Foundation (EFF) popularized the study of SSL infrastructure by publishing certificate data sets obtained from actively scanning the entire IPv4 address space on port 443 in mid 2010 and, again, in early 2012 [12], yielding 5.5M [17] and 7.2M [25] distinct certificates, respectively.

Holz et al. [17] provide the most comprehensive academic treatment of the SSL infrastructure we are aware of. In addition to incorporating the 2010 EFF data set, the authors collect certificates by crawling the Top 1M Alexa domains from different vantage points over a set of time periods, as well as gather certificate and connection data by passively monitoring the uplink of a German research network. We revisit the findings of this research in §5. As they also provide background on SSL and X.509, we omit repeating a similar discussion and refer to their publication for such context.

Lenstra et al. [25] perform an analysis of 11.7M public keys from a variety of sources, including SSL certificates, PGP keys, and SSH host keys. The authors find that 4% of the public keys in certificates and PGP keys share an RSA modulus, which enables recovery of the corresponding private key. Nearly all affected public keys belong to embedded devices [15], and the sharing of moduli presumably results from poor entropy at key generation time (e.g., system startup). Rather than studying one aspect of public keys, our work has a broader goal: we offer a detailed study of many facets of the SSL protocol and X.509 certificate mesh work. However, we do indeed find that the reported key exponent distribution exactly matches our dataset for the first top 5 entries, which account for 99.9% of their data.

Vratonjic et al. [33] also crawl the Top 1M Alexa sites and extract 300K certificates (including 48% duplicates), of which only 16% validate, i.e., the browser does not show a warning. Holz et al. [17] find this number to be 18%. However, both studies do not cache intermediate certificates as part of their validation strategy, which skews the results.

Mishari et al. [27] study 30K SSL certificates from randomly scanning `.com` and `.net` domains, crawling the Top 10K Alexa sites, and probing well-known phishing and typo-squatting domains. After a brief characterization of the data set, the authors devise a classifier that separates malicious certificates from benign, based on 9 features including the certificate signature algorithm in use, if the subject appears to have default names (e.g., `ST=somestate`), if it is self-signed, if the subject common name is similar to the host name, or a "normal" value of the validity period. While our work does not assess certificates, it explores the feature space and can thus benefit such approaches.

Perspectives [35] offers an alternative trust model to the existing PKI infrastructure. Today, CAs make a trust decision on behalf of the users by signing a site's certificate. However, users cannot choose to anchor their trust with a different authority for a given site: a certificate is a one-to-one mapping between a CA and a site. Perspectives removes this "trust lock-in" by replacing a CA with one or more *notaries*, which lookup a requested certificate from a different vantage point and thus allow users to compare the certificate presented by the site against the ones seen by the notaries. While the original intention was to thwart man-in-the-middle (MITM) attacks via multiple paths to a site [29], the ability to switch to a user-initiated trust model gained much wider traction in the community. The basic implementation of Perspectives suffers from completeness (only the initial SSL connection is subject to inspection), performance (notary lag, and lack of caching), and privacy issues (leaking browsing history), which the follow-up Convergence [1] aims to address. We provide a Convergence compatible backend to the community to make our certificate data accessible.

The EFF proposes to solve the problem of malicious CA certificates by using semi-centralized timeline servers utilizing an cryptographic append-only data structure [13]. This data structure would make MITM attack virtually impossible and clients could automatically use alternative paths to a site when they detect any tampering.

Soghoian and Stamm [31] present the threat of compelled certificate creation attacks, in which governments may force a CA under their jurisdiction to issue malicious certificates for MITM attacks. The authors evaluate several theoretical scenarios in which such man in the middle attacks might be carried out and propose to solve the problem by displaying a warning if the CA is situated in a different country than the entity for which the certificate was generated. Our notary service offers an alternative framework to assess certificate trustworthiness.

3. METHODOLOGY

We begin our discussion with an overview of our methodology in terms of the features we extract from SSL traffic (§3.1), the collection setup now in operation at five network sites (§3.2), and our strategy for validating chains (§3.3).

3.1 Features

Generally, we assume the vantage point of a site's upstream border link where we passively monitor live SSL/TLS [2] traffic to extract a set of features for later offline analysis. From a high level perspective, the data we collect splits into two parts: *(i)* connection-level features of individual SSL sessions, and *(ii)* the aggregate set of X.509 certificates we see across them.

Table 1 summarizes the features we collect. Their choice accounts for privacy concerns (see §8). In particular, we do not record any information that identifies a client system directly. We however log one-way hashes of the pairs (client, server) and (client, SNI), which allow us identify all sessions involving the same endpoints. We also hash SSL session IDs to avoid recording payload. Finally, we configure the live analysis to inspect only *outgoing* connections to protect local servers and avoid a potential site bias.

3.2 Collection and Processing

We use Bro 2.0 [28, 4] to extract the features from live traffic. Bro detects and parses all SSL connections independent of any transport-layer ports by inspecting their payload. [8] We wrote a custom script in Bro's configuration language that implements extraction and logging, which we give to operators at participating sites. The Bro script continuously records all features to a file, which it uploads in regular intervals to a central server at our research institute. The five sites in our study are all operating ongoing Bro installations and added our script to existing setups.[3]

On the central server, we import all uploaded data into a PostreSQL database. We validate all certificate chains at import time (see §3.3) and perform further consistency checks similar to what browsers are doing, such as matching the subject's *Common Name* (CN) against both the SNI value and all *Subject Alternative Names*. We spent significant time to optimize our import scripts for handling large numbers of SSL connections. At peak times, our data providers upload more than a million new connections per hour. During bulk imports, we measured the script's maximum rate at about 100K connections/minute for a single thread, running on a Intel Xeon E5630 CPU. As certificate validation accounts for most the work, the processing parallelizes well across CPUs.

We point out that our data set exhibit artifacts of the collection process that are beyond our control. As we leverage operational setups that run our analysis on top of their normal duties, we must accept occasional outages, packets drops (e.g., due to CPU overload) and misconfigurations. As such, we deliberately design our data collection as a "best effort" process: we take what we get, however we can generally not quantify what we miss. However, given the large total volume across the five sites, we consider the aggregate as representative of many properties that real-world SSL activity exhibits.[4] Furthermore, due to the passive nature in which our data was generated, our results are specific to the networks which allowed us to harvest their data.

3.3 Certificate Validation

When validating certificates, we aim to match the results that a typical browser would come to. That, however, turns out surprisingly involved as we find a large variety in what certificates SSL servers include. In particular, many chains

[2] For the remainder of the paper, we will refer to either SSL or TLS as "SSL".

[3] At three of the sites, members of our team did the set up after receiving administrative approval.

[4] As support, we note that generally we do not find unexpected qualitative differences between sites other than those we explicitly point out.

Feature	*Description*
Available ciphers	Lists of ciphers offered by client and server, respectively.
Client SSL extensions	List of SSL extensions the client sent.
Hash(client, server)	Hash of client and server IP addresses.
Hash(client, SNI)	Hash of client IP address and SNI, if available. (see §6.7)
Hash(client session ID)	Hash of the client-provided session ID. (see §6.8)
Hash(server session ID)	Hash of the server-provided session ID. (see §6.8)
Selected cipher	Cipher negotiated between client and server.
Server certificates	Complete server-side certificate chain (see §3.3).
Server Name Indication	Value of the SNI extension header, if available (see §6.7).
Ticket lifetime hint	Suggested lifetime of session's reuse ticket, if available. (see §6.8)
Timestamp	Time of the connection's first packet.
Version	SSL protocol version.

Table 1: Features we collect in real-time for each SSL connection at our data collection sites.

remain incomplete, which tools like OpenSSL cannot handle directly. As we have not found this process documented elsewhere, we report the specifics of our approach in the following. Here, and for the remainder of our discussion, we use the Mozilla root store as our trusted base.

When validating a certificate C from a server chain CH_S, we incrementally assemble a temporary validation chain CH_V that leads from C to one of the roots. Once we have such a sequence complete, we use OpenSSL to verify its correctness.[5] If successful, we mark all certificates in CH_V as *validated*. We then likewise consider CH_S as validated.

We first attempt to build CH_V exclusively from certificates contained in CH_S. First, we match C's *Issuer* against the *Subject* of all root certificates. If we find a match, this completes CH_V. Otherwise, we examine CH_S for a matching intermediate certificate. If found, we insert that into CH_V and proceed recursively. If not, we search for a matching intermediate across *all* already validated CA certificates in our data set that have not yet expired. This step matches the behavior of typical browsers, which *cache* intermediate CAs they have already encountered in past sessions in their local certificate store.[6] Of all valid certificate chains in our data set, 7.37% are incomplete and hence require this step. If we have come to a complete chain CH_V, yet find that OpenSSL does not accept it, we attempt to extend it further. This addresses a specific case where the name of an intermediate certificate matches that of a root, even though their keys differ.

We observe that certificate chains frequently contain *more* certificates than necessary for their validation. 19.70% include a certificate that is already part of the Mozilla root store. Excluding those as well as cross-signing and intermediate certificates unnecessary for validation, we see further extra certificates in 2.84% of the chains, including ones that are expired, self-signed, or duplicated. On the website of a security researcher, we encounter an extra, unused certificate with a 28-bit(!) key at the top of the chain. Browsers ignore such additional certificates as they stop after the first valid end host. Indeed, we find ourselves unable to *display* that extra certificate with any of the standard browsers.

The presented approach seems to generally match well with what a browser returns. While an exact comparision is technically difficult[7], we manually checked that among the certificates our method leaves unvalidated, there are no frequent cases that a standard browser would accept.

4. DATA SETS

We now describe our data set in more detail. Five operational network sites of different sizes provide us with SSL data for our study, captured at their upstream network links. Table 2 summarizes the data from each site. Our contributors requested to remain anonymous. They are all research environments, with four of the five located in the U.S.. As we added the sites incrementally to our study, the individual sets span different time periods. For comparision, we list the total hours observed at each site (non-continuous due to occasional outages).

As Table 2 shows, our data set includes a total of 1.9M unique X.509 certificates and 1.8M unique certificate chains. 149K (8.0%) of the certificates validate against the Mozilla root store. The data comprises a total of 1.4G individual SSL connections, of which 709.3M (50.42%) and 152.9K (41.77%) come with valid certificates and chains, respectively. To not introduce measurement errors, we exclude connections from our analyses (and from Table 2) for which Bro reports missing packets or analysis failures (e.g., violations of the SSL state machine). These are in total 23.6M and 22.1M, respectively.

Grid traffic accounts for a significant share of our data set (81.05%/80.07% of all certificates/chains, yet only 2.43% of connections). While Grid infrastructure uses SSL extensively, it deploys an independent root hierarchy and hence its certificates, according to our definition relative to the Mozilla

[5]We use an a EFF-developed OpenSSL patch that allows to validate a certificate retrospectively at the time of the corresponding connection.

[6]This behavior makes validation dependent on a browser's state for incomplete chains. Indeed, we noticed that the web server of a major research institution failed to include a necessary intermediate certificate and thus remained inaccessible with a fresh browser installation.

[7]Specifics of the validation logic tend to be hidden deeply in a browser's code, with no easy way to split it out, or access from external. Furthermore, browsers themselves can disagree in individual cases.

set, do not validate. We instead identify Grid activity by separately looking for corresponding certificates. The International Grid Trust Federation (IGTF) accredits Grid CAs worldwide, and maintains a root store. However, we notice that if we use that set directly, we miss out on a significant number of certificates that, based on their subjects and issuers, appear Grid-related yet do not trace back to any of these roots (likely local setups not integrated into the global infrastructure and certificates for which we are missing chain elements). Hence, we build a list of Grid certificates manually by matching on common Grid-related name patterns, and consider that our base set for identifying Grid activity. We have verified that of all non-valid certificates not tracing back to this set, none would do so with the IGTF store. In other words, we catch more Grid certificates, but do not miss any.In the remainder of the paper, we generally split Grid activity out separately; and we look at Grid specifics in §6.3.

5. EVOLUTION

Although our community encourages reappraisal of past work, we only see few instances of such research in practice. In this section, we revisit findings from Holz et al. [17] (from now on referred to as Holz) by juxtaposing them with our data. In their study, Holz use two types of data: *(i)* 16 active port-443 scans of the Alexa Top 1M hosts, executed from different geographic locations; and *(ii)* two packet traces recorded passively at the Munich Scientific Research Network (MWN) in Germany. The authors released the former on their web site [16], and we use it for our comparision. In §5.1 we begin with comparing certificate aspects, and then analyze connection properties in §5.2.

5.1 Certificates

Our data set comprises a total of 1.9M unique certificates, of which 366.2K (19.92%) are non-grid certificates. Holz's active scan data set consist of 556K unique non-grid certificates [8]; their passive data set comprises two scans of 163K and 102K, respectively. Holz generally exclude Grid certificates, so we break them out separately in the following.

For comparison, EFF's Feb 2012 scan of the entire IPv4 address space on port 443 contains 7.2M certificates [25]. Passive monitoring catches a subset that, while smaller, is the one most relevant to users as it captures what they use, excluding numerous embedded devices, middle-boxes, and other rarely accessed hosts on the Internet.

Cryptographic properties. The strength of a public key depends on its type and bit length. The dominating public key family in X.509 certificates is RSA; we only see 23 ECDSA and 11 DSA keys in our data set. Holz did not examine the public key families, but for certificate signature algorithms, they find that DSA signatures are virtually non-existent, which each DSA signature algorithm having a frequency of 0.1% or less. This concurs with our findings: we find 13 DSA and 114 ECDSA signatures in total.

[8]Number generated by revisiting their raw data.

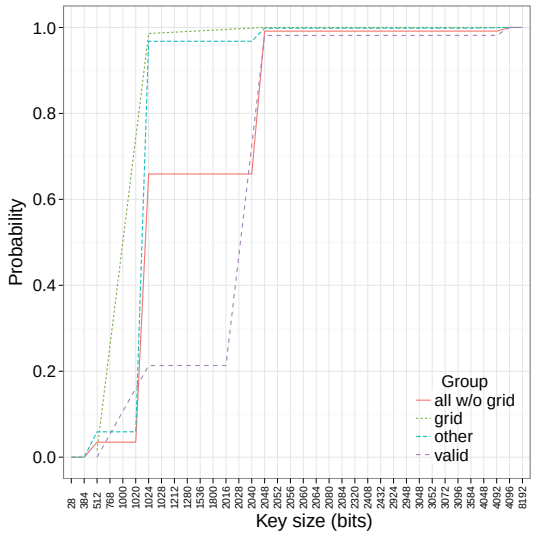

Figure 1: ECDF of the key lengths of certificates with an RSA public key.

For RSA, the cryptography community discourages key lengths less than 1024 bits [3]. However, Holz finds that their latest scan still includes 55% certificates with smaller lengths, while earlier scans include 20% more than that. Using the data set they made available, we in addition examine the subset of valid certificates in there and find that in the latest set, 49% have a key length of 1024 bits or lower. Figure 1 shows key lengths in our data set for non-grid, Grid, validated, and remaining certificates. For non-grid certificates, we find that 64.71% have a key length of 1024 bits or less, which is 11% higher compared to Holz. However, when restricting to valid certificates, only 9.03% have a key length of 1024 or less. We assume that some of the difference comes from the active vs. passive approach. Overall, while we cannot generally confirm the positive trend of increasing key length observed by Holz, we find that only 1 in every 10 valid certificates contains a weak key.

Chain Lengths. We define a certificate's *chain length* as one of two values, depending on whether the certificate validates. If it does, we use the the number of steps required to do so. This number may be smaller or larger than the number of certificates in the chain sent by the server (see §3.3). For example, a certificate signed directly by a root has length 1. If a certificate does not validate, we use the number of certificates in the chain directly, not counting the host certificate. Thus, a self-signed certificate usually has length 0 as it does not come with intermediates or roots.

Holz reports that 50–70% of certificate chains have length 0, depending at the time and location of the scan.[9] We re-

[9]We note that Holz uses a slightly different method to determine the chain lengths, where they exclude all self-signed intermediates from their observations, but do not exclude other unrelated certificates present in valid chains. However, such cases are rare and only impose a small bias towards larger lengths.

Site			Certificates			Connections			Time	
Label	Type	Est. Users	Total	Grid [3]	Validated	Total	w/ Certs [1]	Validated	Hours	Period
US1	University	60,000	1,719,600	1,457,772	102,720	615,277,101	357,514,744	319,360,230	1,163	02/22–04/17
US2	Research site	250	48,450	38,519	8,793	9,372,928	5,047,586	3,916,195	1,382	02/17–04/16
US3	Research site	4,000	40,769	8,960	28,879	83,880,442	34,603,002	34,018,252	882	02/22–03/31
US4	University	50,000	177,727	74	95,772	697,431,016	359,822,923	351,751,087	1,207	02/22-04/16
X1	University	3,000	1,270	2	1,003	548,265	225,205	215,272	232	03/14–03/23
All [2]		117,250	1,857,036	1,505,184	148,984	1,406,509,752	757,213,460	709,261,036	—	—

[1] Connections with no certificates tend to be session reuse; see §6.8.
[2] The total reflects the number of *unique* items across all sites.
[3] Certificates related to Grid infrastructure; see §6.3 for details.

Table 2: Summary of data set properties from contributing sites. For validation, we use the Mozilla root store.

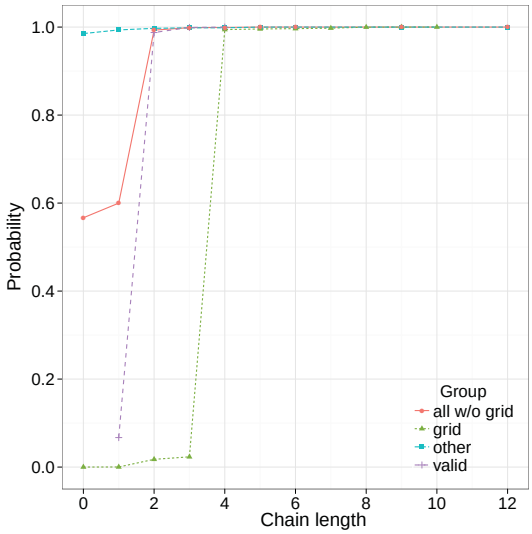

Figure 2: ECDF of keychain length.

examine their newest active scan and find 49.0% of the certificate chains to have a length of 0 when counting only unique chains. When limiting to valid certificates, we find 19.6% certificate chains of length 1 (one end host certificate directly validating to a certificate in the root store). We further observe 38.0% valid chains with length 2 (one intermediate), 27.8% with length 3, and 14.2% with length 4 or greater.

Figure 2 shows the chain lengths we encounter in our data. For non-grid certificates we see that the chain length generally is very low, with more than 99% having a length of 2 or less. For valid certificates, the median chain length is 2 (one intermediate certificate before arriving at a certificate in the root store). This reflects good practice, as certificate providers should use an intermediate to sign end-user certificates (see §6.2). For the remaining non-valid chains, we observe almost always a chain length of 0 (98.6%), which corresponds to self-signed certificates. Most Grid certificates exhibit a length of 4 (97.1%).

Chain Validation. Holz finds that about 60% of the chains from their active scans validate. We apply our extended validation method from §3.3 to two of their scans and find that it yields an additional 4%, indicating that a typical browser would accept more certificates than the earlier estimate.

When looking at the number of different non-grid certificate chains that servers send, we see 366.2K unique chains of which 41.77% validate, which is at least 20% less than what Holz observed in their active and passive scans (depending on the scan). The reason lies in the high number of automatically generated non-validating certificates that we find in our data, which we detail in §6.4. Because of the high number of Grid certificates, the number of chains is also high (1.5M).

Note that the above analysis counts each unique certificate exactly once, but each certificate can occur in one or more connection. In fact, when not considering Grid connections, 51.68% of our connections validate and 1.27% do not. The remainder of the connections do not transmit certificate data.

Validity Period. We also examine the validity period of the certificates we encounter. For validating certificates, the median validity period is 2 years. For non-validating certificates it is 1 year, with a long tail. The tail of non-validated certificates comes from applications that use self-signed certificates with unusually high lifetimes. We find a number of modes that correspond to lifetimes of specific applications. For example, *PandoClient* uses a lifetime of 1000 years and the *Windows Media Player Network Sharing Service* generates 100-year certificates, with the NetBIOS name of a user inside the CN subject field. These numbers match with the findings of Holz.

For Grid certificates, the median we encounter is 3 days, which is higher than the median validity periods of grid certificates reported by Holz (11 hours). As an odd case, we see 90 certificates that have a negative lifetime (their expiry date is smaller than their start date).

Signatures. Holz encounters about 17% certificates that use the weak MD5 hash in their 2009 active scans, and about 10% or less in their 2011 active scans as well as in their 2010/2011 passive scans. Our data reflects this positive trend: we see only 1.51% non-grid certificates using MD5. However, 44.05% of the Grid certificates use MD5.

5.2 Connections

Holz' passive data set consists of two traces and includes

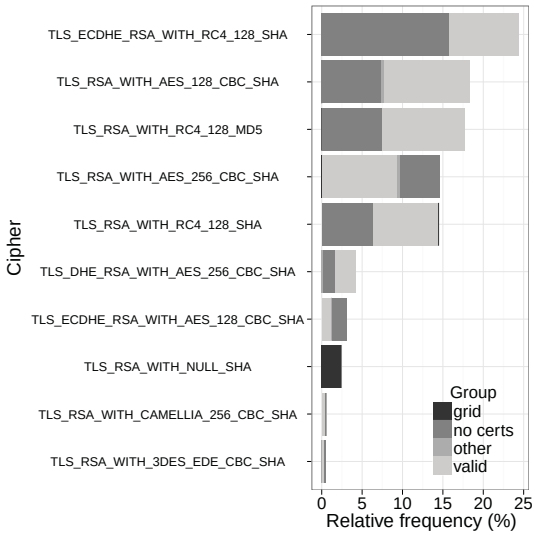

Figure 3: Observed ciphers in all connections except Grid.

249M connections total. The data spans two two-week periods in September 2010 and April 2011. In contrast, our data set includes 1.4 billion connections from 5 different sites recorded over the course of about 1000 hours each, though at different times. Note that, different than in the certificate analysis above, Holz did not exclude Grid traffic from their connection analyses.

Ciphers. Holz observe as the two most frequent ciphers *(i)* RSA with RC4 and MD5 (about 22 and 30% of all connections, respectively, in each of their two passive runs), and *(ii)* RSA with AES-128 and SHA (about 18% and 21%). We provide a breakdown of the ciphers we encounter in Figure 3, which shows the top 10 used in all connections. These ciphers cover more than 98.8% of all connections. Compared to Holz, a new cipher (ECDHE with RC4) takes the top place. Looking at it more closely, more than 99.5% of the corresponding connections involve servers in Google ASes; hence, this difference seems a result of Google's switch to ECDHE in November 2011 to provide forward secrecy. [22] For other connections, RSA with AES-128 and SHA is now most common. In general, each insecure cipher mentioned by Holz occurs less frequent in our data set. The only exception to this is the null cipher which dominates Grid connections.

6. NEW PERSPECTIVES

In this section we examine a number of aspects that past studies have not further explored. We discuss overall traffic properties like AS and port distributions (§6.1), intermediate CA relationships (§6.2), Grid activity (§6.3) and other non-validating certificates (§6.4), X.509 extensions (§6.5), SSL extensions (§6.6), the SNI extension specifically (§6.7), session reuse (§6.8), differences between vantage points (§6.9), and finally reuse of private keys (§6.10).

6.1 Traffic Characteristics

Port	Protocol	Percentage
443	HTTPS	93.8%
993	IMAPS	1.5%
5223	Apple Push	1.1%
995	POP3s	0.60%
5228	Android Push	0.13%

Table 3: Top 5 TCP ports. Percentages relative to all connections.

AS	Description	Percentage
15169	Google	32.3%
2914	NTT Communications	8.36%
32934	Facebook	8.05%
209	QWest	5.43%
2152	California State University	5.36%
714	Apple	4.76%
13414	Twitter	3.79%
8075	Microsoft	3.60%
14618	Amazon AES	2.55%
16509	Amazon 02	1.39%

Table 4: The 10 most popular server ASes. Percentages relative to all connections.

Across all connections in our data set, we see a total of 17,743 unique TCP destination ports using SSL, i.e., about a third of the available spectrum. Table 3 summarizes the 5 most common ports, which cover 97.10%. As expected, HTTPS accounts for most of the activity by far. Potentially more surprising, the Apple and Android *Push* services make the list as well. We identify them by inspecting a random sample of the corresponding certificates. From the 6th most common port onward, we see a number of ports that seem related to Grid activity or site-monitoring. The next port we can identify is 465 (SSMTP) with 0.03% of connections (27th most common port).

Table 4 breaks the connections down by server AS. We see that Google accounts for almost a third of all SSL traffic. Much of the ISPs' traffic, and also that of Cal State, appear due to Akamai (see §6.7). For Facebook and Twitter, we assume that their APIs trigger most of their activity.

Finally, we examine the versions of the SSL and X.509 standards we observe. Across all connections, we see 94.0% TLSv1 and 5.9% SSLv3. The remainder is split between TLSv11 (0.006%), SSLv2 (0.0007%) and TLSv12 ($3.6 \cdot 10^{-6}$ %). For X.509, version 3 is the de-facto standard with SSL and, accordingly, we see that version for 99.8% of all certificates. However, beyond v3, we indeed also see version 1 (0.12%), version 2 (exactly one!), and version 4 (0.03%). The latter unfortunately does not exist and hence we assume software is setting the version field incorrectly.[10]. Of the 577 "v4" certificates, Linksys/Cisco devices delivered 133, and Grid proxies account for 438 (at different institutions) The remaining 6 appear associated with custom applications (e.g., *Kaleidescape* for one). No root CA signed any of the "v4"

[10]The version fields counts from 0 (for v1), and hence one might specify an intended v3 with the value of 3 (i.e., v4).

certificates, nor the one v2.

6.2 Intermediary CAs

A well-known weakness of SSL concerns its insufficient trust model, which leads users to implicitly trusting a large set of CAs they have never heard of. The core of the problem is the large size of a typical browser root store, Underappreciated, however, is the role that *intermediate CAs* play who receive trust delegations from a root. We use our certificate data set for building a graph that spans the dominant parts of the global trust structure for gaining insight into the relationships between CAs.

In this section, we limit our analysis to certificates and connections that validate against the Mozilla root store, which currently includes 135 top-level certificates; we observe 76 of them in at least one valid certificate chain. We do not see all because some CAs have multiple roots and use only a subset thereof, or use them for purposes other than the web. Other roots, in particular if from non-US CAs, see less frequent use and remain unlikely to oocur at the sites we monitor. Among the roots we see in use, we find CAs from 26 different countries, including institutions that users may have concern trusting ("The DHS shouldn't be able to sign certificates for Chinese sites or vice-versa" [26]). Nevertheless, a browser will allow SSL connections to any site that presents a certificate signed by any of them.

Each root CA may chose to delegate trust to intermediate CAs, which then likewise become fully trusted with the same authority as any root: intermediates can issue arbitrary certificates and further delegate signing themselves.[11] In total, we see certificates of 502 intermediate CAs. Some intermediate CAs use more than one certificate, though, with differences in specifics such as validity periods. Counting just unique (*issuer*, *subject*) pairs yields a set of 451 unique intermediate CAs. Adding in the roots, this results in 586 CAs authorized to sign globally trusted certificates.[12]

For root CAs, it is in fact good practice to not sign certificates directly but via intermediaries. Accordingly, of the 76 roots we observe, 68 indeed delegate to an intermediary ; 29 of them to exactly one. However, even among those delegating, 10 do sign certificates directly as well. The largest example in terms of intermediaries is the USERTRUST Network with 30 client CAs and 2,292 signed certificates. Excluding roots that do not delegate, the median number of intermediaries is 2, the mean 4.5, and the maximum 45 (GTE CyberTrust).

The oldest intermediary certificate we encounter is from VeriSign, its validity period started on April 16, 1997.[13] The newest is a SECOM Trust intermediate CA, starting Feb 16, 2012. In general, the average validity period we observe is 10 years (minimum 2, maximum 25). However, the lifetime of a certificate is also constrained by the minimum lifetime over all edges in the intermediary graph. For example, if a root R creates intermediaries I_1, I_2, and I_3 with respective lifetimes 10, 5, and 30 years, then a certificate issued by I_3 cannot have a lifetime beyond 5 years because it would stop validating eventually.

Quantifying Impact. In order to quantify the influence of intermediate CAs, we devise an *impact* measure along three dimensions: $I_{subca}(ca)$ represents the ratio between sub-CA certificates recursively signed by ca (including ca itself) and all CAs; $I_{cert}(ca)$ extends this definition to also include non-CA certificates; and $I_{conn}(ca)$ represents the ratio between valid connections that have a certificate in $I_{cert}(ca)$ and all valid connections.

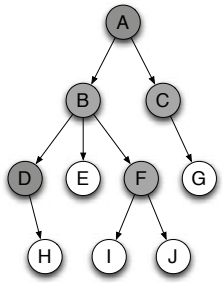

Figure 4: An exemplary CA trust tree.

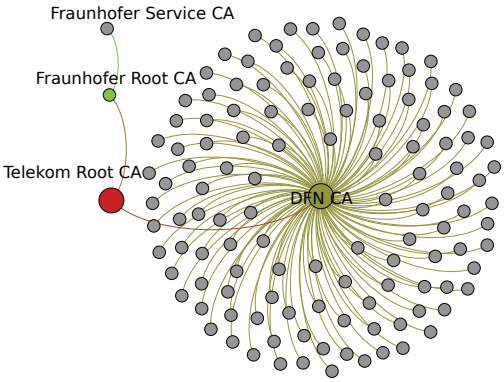

Figure 5: Deutsche Telekom trust tree.

To illustrate, consider the CA graph depicted by Figure 4. Shaded nodes represent a CA certificate, and white nodes are leaf certificates. The total number of nodes is 10, of which 5 are CAs. Then, we have $I_{subca}(B) = 3/5$, $I_{subca}(A) = 1$, and $I_{cert}(B) = 7/10$. If 42 connections out of 84 connections included a certificate signed by a CA within the sub-tree

[11] There is a small exception: not all intermediate CAs may sign *Extended Validation* certificates.

[12] This aligns with the EFF's result of "650-odd organizations that function as CAs trusted (directly or indirectly) by Mozilla or Microsoft", which they derived from active scans. [12]

[13] Note that we do not know when the certificate was *generated*, as we have seen several versions of it with validity periods ranging from 2004 to 2016. We assume that the versions with later expiration dates have been generated more recently.

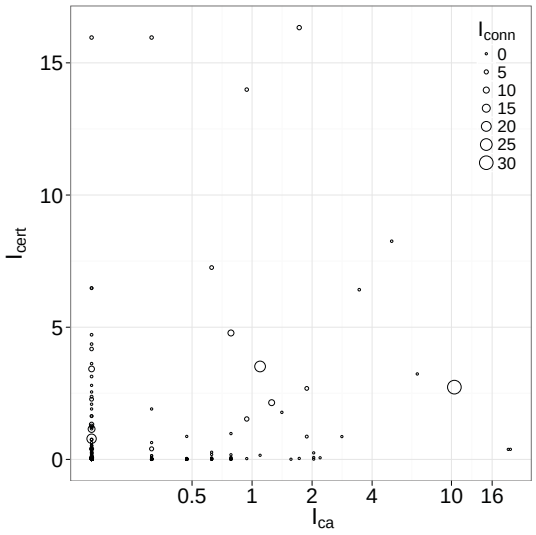

Figure 6: Impact of root and intermediate CAs. We show I_{subca}, I_{cert}, and I_{conn} for all root and intermediate CAs.

rooted at B, we would have $I_{conn}(B) = 1/2$. Note that I_{conn} depends most on the vantage points of our data collection.

Figure 6 shows a scatter plot of all root and intermediate CAs we have observed, plotting I_{subca} on the x-axis (log-scale) and I_{cert} on the y-axis. The plot further scales the size of each point proportionally to I_{conn}. Table 5 summarizes the root CAs maximizing these measures and, hence, show the largest "footprint" in our data.

We see that a VeriSign certificates is the root that authorizes (directly or indirectly) the most certificates in our data set (16.3%). Furthermore, Deutsche Telekom AG (DT) has the largest CA impact ($I_{subca}(DT) = 19.7\%$). Figure 5 shows the corresponding subgraph of the CA hierarchy with 126 intermediates. We see that DT authorizes almost all intermediates only indirectly via the German research network DFN-Verein ($I_{subca}(DFN) = 122$). Upon inspecting that one's client CAs, we find all of them to be German research institutions, presumably operating their own CAs to issue certificates for local users and services. Any of these could in principle issue a malicious certificate for independent domains. Furthermore, a compromise at *DFN-Verein* would require replacing the CA certificates at all these institutions.

In theory, it is possible to restrict certificates that an intermediate CA can issue by including constraints in a X.509 *Name Constraint* extension. For example, a parent CA could specify a limited DNS name space that restricts the certificates an intermediary can generate. In practice, however, this extension seems hardly used; we encounter it in exactly one intermediate CA certificate (for *WISeKey*, restricting it to `*.icc-cpi.int` and `*.icc.int`). We tried to connect to one server that deploys a certificate from *WISeKey* and found that both Firefox and Opera refuse to load the page. Safari, Internet Explorer, and Chrome did, yet it is hard to tell if they honor name constraints at all. The lack of traction for this extension is unfortunate as better support would help to contain the impact of malicious CAs. Returning to *DFN-Verein* example, it could likely limit most of its client CAs to their corresponding institutions without interfering with their operation.

6.3 Grid Infrastructure

The Grid infrastructure provides distributed computing environments for data-intensive applications, typically in a scientific setting [9]. Although grid connections only account for 2.43% in our data set, they contribute 81.05% of the unique certificates. This unexpected discrepancy motivates a separate discussion; clearly, Grid software must be deploying SSL differently than the web.

The Grid infrastructure uses an independent certificate hierarchy. The International Grid Trust Federation (IGTF) accredits Grid CAs worldwide; as of this writing, the IGTF root store includes 87 CAs, of which we see 41 in use in our data set.

The Globus project [14] introduced the Grid Security Infrastructure (GSI) protocol suite [10], which evolved as a popular middle-ware for Grid implementations. GSI employs SSL and X.509 certificates to provide user authentication and fine-grained access control in Grid deployments, tailored to their batch-oriented architecture where users submit jobs that harness a specific subset of the shared compute infrastructure. Managing such dynamic use cases requires *delegating* trust to specific Grid services, which then act on behalf of the user. To this end, users generate short-lived *proxy certificates* from their permanent long-term certificate [34].

A proxy certificate uses the standard X.509 format along with a `ProxyCertInfo` extension that encapsulates a user-specific delegation policy, such as how often delegation may occur and the delegate's capabilities. Validation of proxy certificates occurs in two phases. First, one validates the entity's permanent certificate according to default X.509 rules in RFC 3280. The second step validates the proxy certificate according to RFC 3820, which involves *(i)* checking that the `ProxyCertInfo` extension is present, *(ii)* that the subject is derived from the issuer, *(iii)* that the specified maximum path length still holds, and *(iv)* that the delegated capabilities apply to the usage scenario.

The delegation process begins with a mutually authenticated SSL handshake for which the connection originator uses its generated proxy and the responder its own (non-proxy) certificate. After the handshake, the responder creates a public/private key-pair and sends back a certificate signing request. The originator signs the request and sends the new proxy certificate to the responder, who can now use the delegated certificate to act on behalf of the originator. Because no private keys travel over the channel, the delegation procedure does not require encryption. Indeed, we find that 98.04% of all Grid connections use a NULL cipher. The clear-text delegation is one reason for that, another that many applications disable encryption to maximize the performance of data

I_{subca}		I_{cert}		I_{conn}	
19.7%	Deutsche Telekom AG	16.3%	VeriSign (1)	30.2%	GTE Corporation
10.3%	GTE Corporation	16.0%	Go Daddy Group	22.4%	Equifax
6.7%	GlobalSign nv-sa	14.0%	GeoTrust	9.9%	DigiCert
5.0%	USERTRUST Network	8.3%	USERTRUST Network	9.7%	VeriSign (2)
3.4%	AddTrust AB	7.3%	Thawte	5.6%	VeriSign (3)
2.8%	StartCom Ltd.	6.4%	AddTrust AB	5.6%	VeriSign (1)
2.2%	RSA Security	4.8%	DigiCert	4.0%	Entrust.net
2.0%	PM/SGDN, France	3.5%	Equifax	4.0%	Thawte
2.0%	Baltimore	3.2%	GlobalSign nv-sa	3.4%	GeoTrust
1.9%	Entrust.net	2.7%	GTE Corporation	3.4%	Go Daddy Group

Table 5: Top 10 impact of root and intermediate CAs.

transfers.

With the delegation model, the roles of Grid endpoints do not match well with a standard client/server model. For example, in a GridFTP transfer between two servers, a user may first delegate a proxy certificate to each GridFTP server, which the two then use to connect to each other. As a consequence, we find Grid user certificates in our data set although at the SSL level, we do not record any client-side information. That also explains the large number of Grid certificates we see in total, as generally each delegation involves creating a new certificate. Accordingly, we often see many unique certificates for a Grid server; in one, case we encounter more than 87K certificates with a single host.

A proxy certificate's validity period is typically in the order of hours to limit the consequences of an account compromise; longer-lived jobs however also require longer intervals. [34, 20] In our data set, we find a median lifetime of 2.98 days among all Grid certificates.

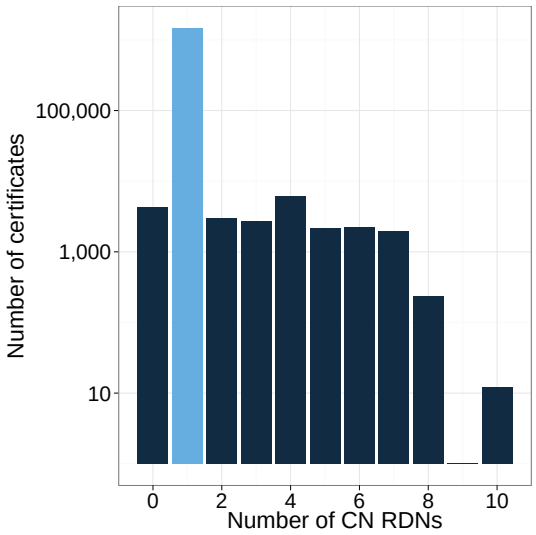

Figure 7: Delegation path length measured by the number occurrences of CN fields in the subject of Grid certificates.

The Grid implementations we see use the certificate's subject field in a very specific way. In each delegation phase, the new certificate's subject inherits the value of the issuer and refines it by appending an additional value of `CN=proxy` or `CN=limited proxy` in order to comply with the above mentioned validation procedure. We can exploit this property to determine the delegation path length as the number of `CN` fields occurring in the subject field. Figure 7 shows the path lengths for all Grid certificates. Most paths have length 1, and path lengths 2–7 range from 2,000 to 6,000.

Certs	Issuer
455	U.S. Government
145	CA Cert Root CA[1]
143	PlanetLab[2]
101	Government of Korea
94	Electronic Arts OTG3 Certificate
66	KISA Korea Certification Authority
62	Government of Korea Root CA
38	Autoridade Certificadora Raiz Brasileira
28	Intel Root CA
25	T-Mobile USA, Inc. Root CA

[1] CA Cert [5] is a community driven root that issues free certificates.
[2] Aggregating 143 certificates from four CA certificates with the same subject.

Table 6: Top 10 private CAs with number of child certificates we see.

44.05% of the Grid certificates we encounter use MD5 signatures, which is considerably higher than the 1.51% among all certificates. This property results from some Grids using old software. Indeed, the IGTF is currently working to move CAs to SHA-2 by January 2013 [19].

6.4 Other Certification Authorities

If we ignore Grid certificates, there are 202.8K certificates in our data set that do not validate against the Mozilla root store. In §6.7, we already find 90.83% of these to belong to Tor, which leaves us with 18.6K certificates not yet classified. Of these, 40.81% are self-signed non-CA certificates, and 16.39% are CA certificates. The latter, however, also contain self-signatures; we inspect some and find self-signed CAs associated with *VMware*, *Parallels* and *Plesk*.

Table 6 shows a break-down of the remaining other certificates by the top 10 alternative certificate roots. Interestingly, the PlanetLab certificates are the main source of elliptic curve keys in our data set. Their certificates have an IP address and a port in their subject field, corresponding to the server/port

Extension	% Certs	% Valid Certs	Purpose
keyUsage	87.5 %	90.0 %	Defines purpose of the certificate key.
extendedKeyUsage	83.9 %	98.2 %	Extends key purpose in addition to keyUsage.
crlDistributionPoints	83.3 %	99.99%	Defines how to obtain CRL information.
basicConstraints	81.1 %	94.1 %	Identifies a CA certificate.
authorityInfoAccess	7.7 %	93.3 %	Specifies how to access issuer's certificate.
authorityKeyIdentifier	7.3 %	83.9%	Defines the public key used to sign the certificate.
subjectKeyIdentifier	6.3 %	72.5 %	Identifies the key of the certificate.
certificatePolicies	5.9 %	71.1 %	Specifies terms under which certificate has been issued.
subjectAltName	5.4%	64.3 %	Defines additional identities (mostly domains) for the subject.
proxyCertInfo	1.8 %	0 %	Indicates proxy certificates and restrictions.
logoType	1.6 %	19.4 %	Embeds logo into certificate.

Table 7: X.509 extensions observed in > 1% of all certificates. Percentages relative to all certificates and valid certificates, respectively. All extensions are defined in RFC 3280, except *proxyCertInfo* (RFC 3820) and *logoType* (RFC 3709).

tuple of the underlying connection.

Finally, when eliminating another 20.23% certificates that have been expired, we are left with a small remainder of 22.55% non-validating certificates. They include self-signed certificates for which *Issuer* and *Subject* do not match (which we see, e.g., with some Western Digital products), and certificates for which we cannot follow the chain because of missing or expired intermediates, damaged certificates, or missing keys.

6.5 X.509 Extensions

Since version 3, the X.509 standard allows for the inclusion of custom certificate extensions (RFC 5280). Each extension includes an *Object Identifier (OID)* to define its type, and an ASN.1 data structure. Across all certificates, we saw 57 unique extensions in use; however the number decreased to 21 in validated certificates. Table 7 shows the most common ones. We note that, in general, the precise semantics of many extensions remain ill-defined and subject to different interpretations. [32].

6.6 SSL Client Extensions

Since TLS 1.0, the SSL protocol also allows for custom extensions. At the beginning of a connection, client and server exchange a list of the extensions they each support in the clear.We observe clients announcing support for a total of 13 extensions, summarized in Table 8. Some of the extensions are still in draft status and difficult to identify.[14] In 74.37% of all connections we see support for at least one client extension. Likely the most interesting extensions are *server name indication* (enabling virtual hosting) and *stateless session ticket* (enabling resuming past sessions without a further key exchange). We discuss them separately in §6.7 and §6.8, respectively. We cannot inspect the server-side because Bro 2.0 does not extract extensions from server traffic.

6.7 SNI Support

The *server name indicator* SSL extension (RFC 6066) addresses a long-standing problem: traditionally, SSL requires servers to associate only a single certificate with each pair (*IP address*, *port*), as otherwise they could not tell which to pass along to a client. That, however, prevents web servers from virtually hosting more than one site. The SNI extension solves the restriction by enabling clients to specify a target host name, similar to the `Host` header in clear-text HTTP 1.1. SNI also helps with another problem: while a certificate can specify multiple host names, their number remains limited and any change requires issuing a new certificate. With SNI, one can instead use separate certificates for each name.

Percentage	Domain
36.5%	Google
15.9%	Facebook
11.7%	Akamai
4.7%	Twitter
2.6%	*(Not set)*
2.9%	Apple
2.3%	Microsoft
1.0%	Netflix
0.99%	Yahoo
0.46%	Mozilla

Table 9: Most popular SNIs – top 20 grouped by entity (see text). Percentages are relative to all connections using the extension.

However, for a web server to rely on SNI requires comprehensive client support, as otherwise it would exclude a significant user population. We examine SNI availability in our data set and find support with 82.02% of all client/server pairs. (82.38% when limiting to HTTPS port 443; recall that we cannot do a per-client analysis). For such a rather important feature, we deem this number low, in particular given that first implementations started offering SNI as early as 2004 and the IETF standardized the extension in 2006.

Examining client requests, we observe a total of 1M unique SNI values, referring to 868.5K 2nd-level domains. Table 9 shows the most popular ones; for clarity, we aggregate the top 20 2nd-level domains manually into groups representing the main entity they belong to (e.g., *Google* includes `google.com`, `google-analytics.com`, `youtube.com`, and others).[15] Oddly, 2.6% of all connections set the SNI exten-

[14]The SSL protocol refers to extensions by numerical values but the official list that IANA maintains does not include most of the drafts.

[15]We removed one site at 0.55% that was accessed mainly from one of our collection sites and hence could reveal its identity.

Extension	Percentage	Purpose	Definition
server_name_indication	77.9 %	Client-provided server name.	RFC 3546
elliptic_curves	75.3%	Support for elliptic curve cryptography.	RFC 4492
ec_points_formats	75.3%	Support for elliptic curve cryptography.	RFC 4492
session_ticket	43.4 %	Stateless session reuse.	RFC 5077
renegotiation_info	40.0 %	Key renegotiation secure against MITM.	RFC 5746
next_protocol	25.4 %	Support for Google's SPDY.	Internet Draft, expired 07/2010. [21]
status_request	24.0 %	Requests OCSP response from the server.	RFC 6066
cert_type	0.0049 %	Support for other certificate types.	RFC 6091
encrypted_client	0.0033 %	Support for encrypted client certificates.	Internet draft, expired 04/2012. [23]
origin_bound_certificate	0.0033 %	Dynamic certs to replace login cookies.	Internet Draft, expired 03/2012. [2]
signature_algorithm	0.0022 %	Indicate signature/hash algorithms.	RFC 5246
extended_random_values	0.0013 %	Extended bit-length for random values.	Internet Draft, expired 03/2009 [30]
heartbeat	0.00062 %	Implements keep-alives.	RFC 6520

Table 8: Observed SSL client extension support. Percentages relative to all valid connections.

sion but do not provide a hostname. Comparing the table with the ASes in Table 4, we note that the SNI break-down reveals insight into the *content* rather than the *provider* (e.g., much of the NTT AS-traffic is due to Akamai).

The set of 1M SNI values exhibits a long tail: less than 1% of all connections use 99% of the values. Inspecting a sample manually, we find many clients that specify what appears to be a random host name, such as `www.fl4i5z3cpys.com`, which explains the large total number. The corresponding connections come with certificates exhibiting the same regular structure: *Issuer* and *Subject* match the pattern `CN=www.[randomstring].[tld]`. It turns out that Tor is behind these: we checked about 1,500 hosts returning such certificates and found them all to be Tor nodes. We speculate that Tor uses the randomization to evade detection. Excluding the Tor SNIs leaves us with 258.3K values; the top 100 SNIs in there however still account for more than 92%.

To understand whether servers indeed use a supplied SNI, we examine IP addresses that serve more than one valid certificate chain (excluding Grid traffic where this is common; see §6.3). Among the 4.4K IPs (1.73%) that we find, 57.92% are from Google's AS 15169. It appears that Google respects the SNI to serve separate certificates; however, in its absence they return different ones that instead include a larger number of alternative DNS names. Excluding Google, we examine a random sample of further IP addresses, yet do not find other hosts using SNI to serve different certificates. What we instead commonly see is a server returning different certificates that are however all valid for the same IP. We assume that these are load-balancer setups. Finally, we also notice further cases of multiple certificates because an older one has expired. In conclusion, we do not find significant reliance on SNI outside of Google.

However, we do see certificates with a large set of seemingly independent *Common Names*, which suggests that people are working around the lack of direct support for virtual hosting. If we look at the longest certificates in our data set (in terms of bytes), we indeed find several of them at the top. In one example, a service provider lists 199 independent host names—and thereby gives away its customer list.

6.8 Session Reuse

Our data set provides us with insight into the deployment of *SSL session reuse*, which constitutes a mechanism for skipping a full SSL key exchange by recycling previous state for follow-up sessions between the same client/server pair. Reuse primarily helps the server to shed load[16]

SSL defines two methods to reuse a session. One of them is part of the main protocol specification (RFC 2246) and leverages a unique, server-generated *session identifier* that the client may chose to retain for sending back later. This approach requires both client and server to maintain a mapping between the identifier and the negotiated security parameters. The second method introduces an SSL protocol extension that, from the server's perspective, enables *stateless* reuse (RFC 5077). In this case, the server sends a *session ticket* to the client that includes all it later needs to resume, encrypted with a server-only key. When a client wants to reuse a session, it provides the ticket back to the server.

In total, we observe reuse in 35.07% of all SSL connections. 54.82% of all servers reused sessions (53.49% when only considering port 443). Due to protocol intricacies that we omit for brevity, we cannot directly decide which type of reuse a connection employs without having access to server-side extensions (which Bro does not extract in its current version). However, in 9.39% of all connections we see the server providing a stateless session ticket, from a total of 54.82% of all servers . As such, we assume that a large share of reused sessions do indeed use the newer, stateless variant.

For stateless reuse, we can inspect the ticket lifetimes that servers provide to give a hint to the client how long to store a ticket. We encountered 272 different lifetimes. The most common life times are 28h, 0, 4h, 86,000 seconds (23h:53m:20s), and 5m. (0 zero means "unspecified"). We see the 28h value almost exclusively with Google servers , which account for 5.83% of all the lifetimes. The maximum lifetime we encounter is 108.35 years, the minimum lifetime other than 0 is 60 seconds. The average lifetime is just over 80 hours, the

[16][6] finds that "RSA computations are the single most expensive operation in TLS, consuming 13-58% of the time spent in the web server".

	US1	US2	US3	US4	X1
U_i valid	**46.12%**	6.49%	11.78%	**39.87%**	11.76%
U_i grid	99.99%	99.87%	99.06%	—	—
U_i other	74.15%	72.11%	38.79%	49.98%	76.83%
C_i valid	**31.80%**	0.38%	2.28%	**25.63%**	0.07%
C_i grid	96.84%	2.55%	0.58%	—	—
C_i other	58.17%	0.40%	0.56%	20.18%	0.22%

Table 10: Vantage point comparison by uniqueness U_i and contribution C_i per contributing site. We omit grid values for US4 and X1 due to a lack of sufficient data.

median is 28 hours.

6.9 Vantage Point Comparision

Unlike active scanning, passive monitoring leads to data sets that reflect user activity at specific sites. To quantify diversity across our collection points, we compare our $n = 5$ sites with two metrics, *uniqueness* and *contribution*. For a site i, the uniqueness U_i represents the ratio of certificates *only* seen there, $\widehat{c}_i$, to all its certificates c_i:

$$U_i := \frac{|\widehat{c}_i|}{|c_i|} \quad \text{where} \quad \widehat{c} := c_i - \bigcup_{i \neq j} c_j \quad \forall i \in \{1, \ldots, n\}.$$

A high value of U_i means that at site i, a large share of the certificates remain specific to that location.

The contribution C_i of site i represents the percentage of certificates $\widehat{c}_i$ that site i adds to the *whole* data set:

$$C_i := \frac{|\widehat{c}_i|}{|\bigcup_{i=1}^{n} c_i|}.$$

I.e., a high contribution indicates that the site's certificates add significant value to the aggregate data set.

Table 10 shows uniqueness and contribution for valid, Grid, and remaining (i.e., other) certificates across our sites. We see that among the valid certificates, 46.12% are unique at US1 and 39.87% at US4. As expected, the majority of certificates at smaller sites constitutes a large subset of those at larger sites. Conversely, the other remaining certificates tend to be more site-specific. Uniqueness of Grid certificates approaches 100%, as many of these serve as one-time authentication tokens. Regarding the contribution, US1 and US4 contribute the majority of valid and remaining certificates to our data set (31.80% and 25.63%, respectively). Nearly all Grid certificates stem from US1, which we speculate might be due to a particular Grid service in use there that deploys delegation extensively (see §6.3).

6.10 Private Key Reuse

SSL best-practice recommends using a new public/private key pair for each certificate, even though technically multiple certificates can share a key. While reusing keys is not a security problem per se, it increases the impact of a key compromise. We measure the amount of key reuse across all our certificates and find it at the expected low level overall: only 5.9K non-CA[17] certificates share their private key with at least one other instance, 1.4K keys share across at least 10 certificates, and 1K across at least 100. In total, we find 2K private keys in more than one certificate.

Inspecting the reuse cases more closely, we find Google to account for a large share. 1.1K Google certificates share their key with others; often with *many*. In one case, the same key belongs to 223 certificates, all with a different set of Google-themed *Common Names*. However, we also see different keys in use for the same *Common Name*. There are 64 Google certificates that do not share a key.

Besides Google, the next largest key re-user is an e-commerce solution provider that operates online shops for customers. Here, we find one key in use with 77 different web sites, presumably all under the control of the provider (the *Common Names* resolve to 73 IPs within a continuous block of twelve /24 subnets). However, even when assuming that their customers have no direct access to the private key, any compromise at the provider would affect all the sites. We find several further examples of similar hosting setups. While we cannot know their motivation for reusing keys, we suspect its mostly the convenience of not having to manage separate keys per customer.

7. NOTARY SERVICE

The data collection we describe in §3 is an ongoing effort that we intend to maintain and extend to further sites. We deem the collected SSL data a valuable resource that we want to make accessible to the community. However, we need to account for sensitivity concerns at our data providers (see §8) and hence cannot release the raw data. As a trade-off, we instead focus on the set of web certificates we observe, which is likely the most valuable part for the broader community as it provides clients with a measure of certificate reputation.

In the spirit of existing efforts—such as Perspectives [29], Convergence [1], and the EFF SSL Observatory [11]—we operate a public SSL notary service that offers an online API to query whether any of our collection sites has observed a specific certificate. [18] The interface is DNS-based and we provides a superset of the information that was available in Google's (now defunct) SSL catalog [24] so that existing clients (including Convergence servers) can easily leverage it. When queried with the SHA1 of a certificate, the notary replies with a `TXT` record indicating the day we first saw the certificate (relative to 1/1/1970), the most recent day we saw it anywhere, the number of days in between when at least one site reported it, and the information if the certificate currently validates against the Mozilla root store. For example, to check for one of Google's certificates using `dig`:

```
# dig txt +short C19[..]58.notary.icsi.berkeley.edu
"version=1 first_seen=15387 last_seen=15450 \
times_seen=64 validated=1"
```

[17]We exclude CAs here as we find them sometimes reusing keys across different versions of a certificates.

We include all certificates into the notary that we observe at any of our collection points, except those related to Tor and Grid activity. We exclude the latter primarily for privacy reasons, yet also deem them less interesting for typical usage scenarios. In the future, we plan to offer an alternative interface that also reports a certificate's validity at the time of the query, as determined by our validation process (see §3.3). We host the notary with a setup similar to Team Cymru's Malware Hash Registry [7], and it can handle a large numbers of certificates on the server side.

8. COLLABORATING WITH OPERATIONS

Researchers in our community often find operators reluctant to provide real-world data for scientific studies. For this effort, we work with five operational environments that all have agreed to instrumenting their networks for recording and exporting information we extract from their traffic in real-time, with full payload access for the analysis application. As such, we see our work also as a case study on overcoming the hurdles that researchers often face when interacting with operations, and we discuss our experiences in the following.

Generally, at the sites included in this study, we found operators with an *interest* in supporting our research effort and contributing value to the larger community. We consider such support a crucial prerequisite as it provides sites with a motivation to invest time into the collaboration. With that perspective, participation becomes a question of realistically trading off benefits with risks. None of the sites took that decision lightly, yet they all eventually approved going ahead.

The key to a successful collaboration lies in accepting and addressing the constraints that operators face. Most importantly, we design our study to minimize the risk of exposing sensitive site information, at the cost of some analyses we cannot perform, such as per-client statistics. Specifically, we *(i)* limit the collected features to a subset generally deemed low sensitive; *(ii)* clearly separate between data for *internal* analysis and *public* access via the notary; and *(iii)* accept that we generally do not control the collection setups and may hence experience artifacts such as non-continuous time intervals and packet loss.

When operators were assessing the features we collect, their main concern was the risk of revealing sites their users access. For the notary, we account for that by leaving all sources anonymous and providing coarse-grained, aggregated information. For the raw data the risk of a deanonymizing attack remains larger: if attackers got access, they could check for specific (*client*, *server*) pairs, and likely also target individual clients by brute-forcing the IPv4 address space. Our collaborators are aware of this risk, and they ultimately deemed it sufficiently low. However, such an assessment remains an environment-specific decision that can go either way. Indeed, we also talked to a site not included in this study that eventually could not approve their participation.

9. CONCLUSION

In this work we offer a large-scale study of SSL traffic collected passively in five operational network environments. Continuing the spirit of the community's past work, we reassess previous findings with a broader data set, examine a range of further aspects that have not yet seen much attention, and contribute to improving end-user security by making the collected web certificates available to the public in the form of an online notary service.

Overall, we observe that real-world SSL/X.509 deployment exhibits a complex structure, with many specifics remaining ill-defined and left to interpretation. Examples include the process of validating certificates and interpreting semantics of protocol fields and extensions. As such, the most appropriate perspective on SSL may be a pragmatic "it works, if it works", with a corollary observation that some features aiming to improve weaknesses are not yet sufficiently supported across implementations (e.g., enforcing constraints on certificates that intermediate CAs can issue; virtual hosting of SSL servers). We conclude that, from a broad perspective, the system in fact works better than one might expect, mainly because SSL implementations tend to work around technical deficiencies. That observation however does not change some of the inherent weaknesses SSL exhibits, most importantly its insufficient trust model that depends crucially on a much too large set of players. Our analysis of CA relationships demonstrates the risks involved, yet also shows that there is structure that a fine-granular permission model could exploit.

We are continuing our data collection and intend to add further sites to the set. Doing so will further increase coverage and allow for a long-term analysis showing trends and developments over time. We also see our work as a case study on working successfully with operations to provide research studies with real-world data, and we will explore extending our feature collection to other portions of the sites' traffic.

10. REFERENCES

[1] Convergence. `http://www.convergence.io`.

[2] D. Balfanz, D. K. Smetters, M. Upadhyay, and A. Barth. TLS Origin-Bound Certificates. RFC Draft, May 2012.

[3] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery. On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. Technical report, Cryptology ePrint Archive, Report 2009/389, 2009.

[4] Bro IDS Web Site. `http://www.bro-ids.org`.

[5] Cacert.org. `http://www.cacert.org`.

[6] C. Coarfa, P. Druschel, and D. S. Wallach. Performance analysis of tls web servers. *ACM Trans. Comput. Syst.*, 24(1):39–69, Feb. 2006.

[7] T. Cymru. Malware Hash Registry. `http://www.team-cymru.org/Services/MHR/`.

[8] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic Application-Layer Protocol

Analysis for Network Intrusion Detection. In *USENIX Security Symposium*, 2006.

[9] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[10] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92. ACM, 1998.

[11] E. F. Foundation. The eff ssl observatory. https://www.eff.org/observatory.

[12] E. F. Foundation. The EFF SSL Observatory. www.eff.org/observatory.

[13] T. E. F. Foundation. The sovereign keys project. https://www.eff.org/sovereign-keys.

[14] Globus. http://www.globus.org.

[15] N. Heninger. New research: There's no need to panic over factorable keys—just mind your Ps and Qs. https://freedom-to-tinker.com/blog, February 2012.

[16] Holz et al. active scan data sets. http://pki.net.in.tum.de/node/1.

[17] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The ssl landscape: a thorough analysis of the x.509 pki using active and passive measurements. In *Proc. 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011.

[18] ICSI Certificate Notary. http://notary.icsi.berkeley.edu/.

[19] EUGridPMA meeting notes, January 2012. https://www.eugridpma.org/meetings/2012-01/summary.txt.

[20] D. Kouril and J. Basney. A Credential Renewal Service for Long-Running Jobs. In *Proc. IEEE/ACM International Workshop on Grid Computing*, 2005.

[21] A. Langley. Transport Layer Security (TLS) Next Protocol Negotiation Extension. RFC Draft, Jan. 2010.

[22] A. Langley. Google Online Security Blog – Protecting data for the long term with forward secrecy, Nov. 2011. http://googleonlinesecurity.blogspot.com/2011/11/protecting-data-for-long-term-with.html.

[23] A. Langley. Transport Layer Security (TLS) Encrypted Client Certificates. RFC Draft, Oct. 2011.

[24] B. Laurie. Google Online Security Blog – Improving SSL certificate security, Apr. 2011. http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html.

[25] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, Whit is right. 2012.

[26] M. Marlinspike. SSL And The Future Of Authenticity. http://blog.thoughtcrime.org/ssl-and-the-future-of-authenticity.

[27] M. A. Mishari, E. D. Cristofaro, K. M. E. Defrawy, and G. Tsudik. Harvesting SSL Certificate Data to Mitigate Web-Fraud. *CoRR*, abs/0909.3688, 2009.

[28] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[29] Perspectives project. http://perspectives-project.org/.

[30] E. Rescorla and M. Salter. Extended Random Values for TLS. RFC Draft, Nov. 2008.

[31] C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against ssl.

[32] C. Tonga. X.509 Certificates—PKI: The OSI of a new generation. http://www.cypherpunks.to/~peter/T2a_X509_Certs.pdf.

[33] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux. The Inconvenient Truth about Web Certificates. In *The Workshop on Economics of Information Security (WEIS)*, 2011.

[34] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *3rd annual PKI R&D workshop*, volume 14, 2004.

[35] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference*, 2008.