Exploring Tor's Activity Through Long-term Passive TLS Traffic Measurement

Johanna Amann
International Computer Science Institute

Robin Sommer
International Computer Science Institute &
Lawrence Berkeley National Laboratory

ABSTRACT

Tor constitutes one of the pillars of anonymous online communication. It allows its users to communicate while concealing their location as well as the Internet resources they access from observers. Since its first release in 2002, Tor has enjoyed an increasing level of popularity, with now commonly having more than 2,000,000 simultaneous active clients on the network. However, even though Tor is widely popular, there is only little understanding of the large-scale behavior of its network clients. In this paper, we present a longitudinal study of the Tor network based on passive analysis of TLS traffic at the Internet uplinks of four large universities inside and outside of the US. We show how Tor traffic can be identified by properties of its autogenerated certificates, and we use this knowledge to analyze characteristics and development of Tor's traffic over more than three years.

1. INTRODUCTION

Anonymous online communication has become a paramount interest for both researchers and the Internet community at large. Tor represents the most popular system to that end, allowing users to communicate with Internet servers while keeping their identity and location private. While many conceptual aspects of Tor's communication have been studied in the past, details about its network-level properties—such as, especially, the clients' behavior—remain scarce. Most of what the community knows about the Tor network comes from public *directory information*, which it uses to maintain the network. However, as Tor purposefully limits this knowledge, there is hardly any information about real-world usage patterns of Tor clients.

By default, Tor uses the SSL/TLS[1] protocol suite to establish encrypted connections between participating nodes, just as it is commonly used by web browsers, email clients, etc. In difference to other services using TLS, Tor does not partake in the global PKI with its trusted Certificate Authority system. Instead, Tor nodes automatically generate X.509 server certificates, which they rotate frequently. It turns out, however, that Tor's current certificate algorithm leaves them identifiable through pattern matching, enabling passive observers of the TLS data stream to distinguish Tor connections from other TLS connections.

In this paper, we exploit this characteristic to present a measurement study of the Tor network using passively collected TLS session information. We *(i)* identify Tor sessions; *(ii)* compare the connections against publically available information from Tor directory authorities and; *(iii)* use metadata from the TLS protocol layer to infer properties of clients and servers.

Our data set consists of passively collected information of all outgoing TLS sessions from 4 university networks with, in total, more than 300,000 users, spanning a period of more than 3 years. Of the 115 billion total sessions in that set, Tor contributes more than 30 million.

We organize the remainder of this paper as follows: §2 gives a short overview of the related work. §3 summarizes the Tor protocol and introduces our data set. §4 discusses the methodology of our measurement study. §5 takes a look at the properties of outgoing Tor connections in our data set while §6 follows up to examine characteristics of Tor servers. §7 discusses our results and concludes this paper.

2. RELATED WORK

There are a number of works that measure different parts of the Tor infrastructure. In 2009, McCoy et al. [22] measured the Tor network by joining in as exit and relay nodes. Their results show that non-interactive protocols consume a disproportionate amount of bandwidth; that substantial Tor communication involves clear-text protocols (including transmitting user passwords); and that at least one exit node examined the content of user payloads. In 2010, Chaabane et al. [6] did slightly different measurement work taking the same approach.

Loesing measures the relay as well as the client side of the Tor network using information from the Tor directory authorities [20, 19], showing trends from 2006 to 2009. The studies examine the number, bandwidth and country distribution of relays and clients, and offer an estimate of the number of requests that the network transfers.

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

Dhungel et al. [9] measure and examine delays introduced by guard relays using active probing.

While there is further a wealth of work examining anonymity in the Tor network [15, 18, 21], proposing updates to the Tor routing algorithms [31], and measuring specific details like underground marketplaces [7] and child pornography trafficking [16], to the best of our knowledge no prior effort has studied the encrypted traffic between Tor nodes.

3. BACKGROUND

We begin our discussion by summarizing the background, starting with an overview of the inner working of Tor with a focus on its communication protocol. For this we first introduce Tor's different node types in §3.1, followed by an overview of their communication in §3.2. Finally, §3.3 introduces the data set from the ICSI Notary service that we use throughout this paper.

3.1 Tor Node Types

The Tor network consists of different types of nodes. Users run a *Tor client* that allows them to access the Tor network. They use a web browser, or other local software, to access the network via a proxy port that the Tor client opens on their machine. Clients connect to *relays*, which forward their information to other nodes or the Internet at large.

Information about all currently available relays is publically available from semi-trusted *directory authorities*, which the Tor client software hardcodes. At the time of writing, the Tor network offers 9 directory authorities. After retrieving relay information from a directory authority, clients connect to the network by connecting to typically three *guard relays*. The Tor network chooses guard relays through an automated process that favors stable and reliable nodes [33]. Clients keep connecting to the same set of guard relays for about 4 to 8 weeks—a design that protects against attackers controlling nodes only for shorter periods while aiming to correlate timing information [26, 35]. Next, *exit relays* forward connections to the public Internet, with a relay's administrator deciding if the node may act in this role. Experience shows that running an exit relay carries a risk of facing police investigation when criminal activity originates from its IP address [36].

When a Tor client wants to connect to a host on the Internet, it picks a random path through the Tor network, starting at one of its guard relays and ending at the destination. Neighboring relays on that path establish connections between each other, forming *circuits* that allow clients to reach the destination. The same circuit can be re-used by a client for several connections to the same target server. The time limit for a circuit's reuse depends on the Tor version, and tends to lie between 10 minutes and 2 hours [28]. Finally, *bridges* represent a further class of relays. Their IP addresses remain private to allow clients from censored countries or networks to access the Tor network, even if those countries block all Tor relays listed by the public directory authorities. Bridge IP addresses can, e.g., be obtained via Tor's website, which enforces rate limits and uses captchas.

3.2 Tor Node Communication

Tor supports two ways of communication: *(i)* using the traditional Tor protocol; and *(ii)* using pluggable transports. When using the traditional Tor protocol [10], Tor nodes connect to each other using a TLS connection. Depending on the Tor protocol version, the way in which a node establishes the TLS connection varies slightly, but with all modern versions of Tor the server presents an automatically generated X.509 certificate. The nodes start using the Tor communication protocol after finishing the setup of the TLS connection.

The second way of connecting to the Tor network uses pluggable transports [37], which enable tunneling Tor through other protocols. Tor supports several such transport protocols, including obfs2 [24] and obfs3 [25] (protocol obfuscation layers for TCP protocols), WebSockets [12], and Meek, which uses domain fronting to hide inside innocuous-looking HTTP requests to CDNs [23].

For our study, we examine only Tor communication over TLS, not any pluggable transports.

3.3 The ICSI SSL Notary

For our study, we use data from the ICSI SSL Notary [17, 2], which passively collects TLS session and certificate information from currently seven research and university networks, covering activity of approximately 390K users in total. To date, the Notary has recorded more than 123G TLS connections, and more than 3 million unique certificates[2]. The first data providers started contributing data to the Notary in February 2012. Our data providers run the open-source Bro Network Monitoring System [5, 27] on their gateway links. We provide them with a custom Bro analysis script that collects details from each outgoing SSL connection. For more details about the setup, we refer to [2]. For this paper, we use data from four of our seven data providers, choosing universities with large user populations that have been contributing consistently. Table 1 shows aggregate information about these four providers. This subset covers more than 300,000 users on two continents for a period of more than 3 years.

4. METHODOLOGY

In this section we introduce our measurement methodology, including our approach to identify Tor certificates. We also present the features that we consider for each Tor connection.

[2] Not counting Tor and Grid Computing certificates.

Site		Certificates		Connections		Time
Label	Users	Filt.	Tor	Total	Tor	Days
N1	90K	2.2G	3.6G	52G	8.2M	1,135
N2	50K	1.1G	9.5G	18G	20M	921
N3	170K	1.2G	636K	41G	1.1M	853
X1	12K	209K	247K	2.9G	364K	861
Total	391K	3.0G	14G	115G	30M	—

Table 1: Summary of data set properties from contributing sites. N = North America, X = rest of world. *Total* reflects the number of *unique* items across the sites. *Filt.* counts certificates after filtering Tor and Grid Computing certificates.

Collected Features

Timestamp	TLS extension value lengths
TLS Version	Client SNI [1]
Server certificates	*Hash*(Client session ID)
No. client certificates	*Hash*(Server session ID)
Server IP & port	*Hash*(Client IP, Server IP, Salt)
Client available Ciphers	*Hash*(Client IP, SNI, Salt)
Selected cipher	Client & Server ALPN [13]
Server ticket lifetime [30]	TLS Alerts
Selected EC curve	Client EC point formats
Connection Duration	Client EC curves
Sent & Received bytes	DH parameter size

Table 2: Features collected by our TLS data collection. Shaded features are used in this study.

For studying Tor sessions, we need to distinguish traffic between Tor nodes from other TLS communication. Examining Tor's payload, as well as its TLS source code [38], reveals that the certificates that Tor servers generate exhibit characteristics that renders them unique. By default, both the issuer and the subject of the Tor's certificates represent random Common Names, consisting of the components `www.`, a random 8 to 20 letter alphanumeric domain name, and a `.com` or `.net` ending (e.g., `www.4dpbq2neblawq7lbq.net`, `www.iqo3xm6iukfa4qf.com`). The subject and issuer fields are generated independently and thus differ from each other. Neither subject nor the issuer fields contain further information that is commonly found in certificates (and mandated by Certificate Authorities), such as location or company names.

These properties allow us to identify Tor connections by parsing the X.509 certificates in our data set and then matching a corresponding regular expression on their subject and the issuer fields. Through a set of semi-automated cross-checks, we verified that our data set contains no non-Tor TLS sessions with certificates matching this heuristic.

One potential pitfall of identifying Tor connections this way stems from TLS session resumption, which skips most of the TLS handshake, including the certificate exchange, for consecutive connections to the same TLS server. However, the Tor specification states that Tor clients and servers must not implement session resumption (sec .2.2 of [10]), hence avoiding that challenge.

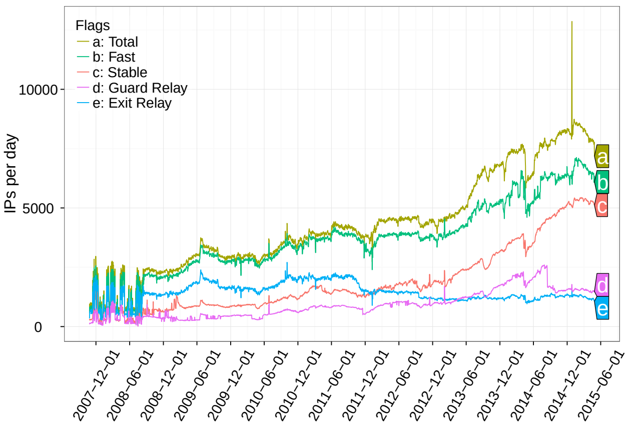

Figure 1: Relay types derived from CollecTor data.

Table 2 summarizes the features that our notary data set provides.[3] Most of the collected information concerns the TLS handshake, such as the supported cipher list a client sends, the server selected cipher or different TLS extensions. In addition to this, the Notary information also contains the IP address and port of the server. To retain anonymity of users at contributing sites, we hash client IP addresses with Server IP (and SNI if present), along with a site-specific secret salt unknown to us. This enables us to identify unique client/server pairs while keeping client IPs private.

5. TOR SERVER CONNECTIONS

As a first step in our exploration of the Tor network, we compare the passively collected data from our measuring points with publically available information from the Tor network. The Tor Project releases a set of statistics containing information about the relays and bridges in the network on its CollecTor webpage [8].

5.1 Tor Consensus Information

For our subsequent analysis, we use CollecTor information about the Tor network status consensuses. These network status consensuses contain all the relays in the Tor network as agreed on by the semi-trusted Tor directory authorities (see §3.1). Among others, the data contains the IP addresses, ports, and Tor versions of all public relays, as well as the relay flags (like *guard* relay, *exit* relay, *stable*, *fast*). This data is available since the end of 2007 with hourly granularity.

Figure 1 shows a plot of the consensus information showing all relays as well as specific subclasses of relays having the *exit*, *guard*, *stable* and *fast* status flags set. A single relay can hold several flags simultaneously to

[3] Since we have extended our data collection script over time, information about older connections does not contain all the listed attributes.

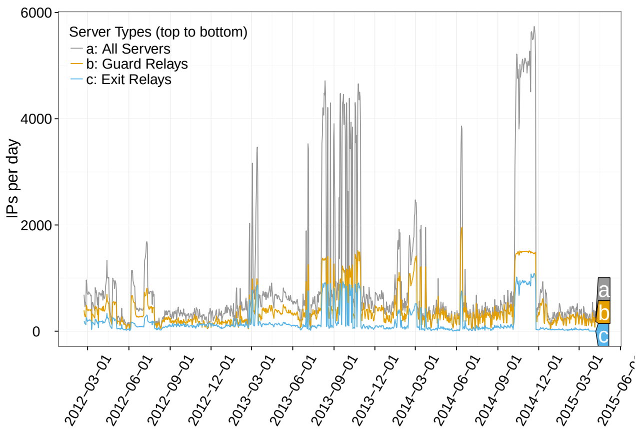

Figure 2: Connections to differing node types at N1.

represent, e.g., both a guard and an exit node.

As the graph shows, the Tor network size has been rising slowly over the recent years. However, this is not true for all node types. While the average number of relays per day increased from 3,984 in 2011 to 7,524 in 2015 (i.e., 89% more), and the number of guard nodes increased from 793 to 1,911 (141%), the number of exit relays indeed decreased by 37% from 1,965 to 1,243 per day. We assume this corresponds to an increasing awareness that Tor exit node maintainers may find themselves facing legal challenges (see §3.1). However, this also means that in 2014, each exit node routed a larger fraction of the traffic than in 2011—which makes operating an exit node more interesting to malicious participants aiming to examine outgoing traffic.

The *stable* flag signals that a node has remained reliable over time; it constitutes a requirement for becoming a guard node. Tor considers a relay stable when either its mean time between failures (MTBF) is at least the median of all known active relays or its weighted MTBF (definition in [34]) is more than 7 days [34]. The number of stable Tor relays has increased by 183% from 2011 to 2014, from an average of 1,466 relays to 4,171. This might correlate with permanent Internet connections becoming more available to end-users.

5.2 Connection Classification

Generally, in any large end-user network, we would expect most Tor nodes to act as clients. Hence, most outgoing connections should connect to guard relays. To check this, we match all outgoing connections to the Tor network consensus information of CollecTor.

Figure 2 shows the total number of external relay IP addresses seen each day at site N1, also indicating which of them act as guard and exit relays. Over the time period of the measurement, 50% of all connections (5,251,896 of 10,593,541) terminated at guard nodes. Considering that the average number of guard nodes in the Tor network is just 20% (all-time; 25% in 2014), this indicates that there is a sizeable fraction of clients running at this institution.

The graph also contains several distinct peaks during which the ratio of of guard nodes per day is much lower. During these times, most connections terminate at "normal" relay nodes on the Tor network that are neither exit nor guard relays. We suspect that the peak between August and November of 2013 can be attributed to the Mevade Botnet, which caused a massive global rise in the number of active Tor users, going from approximately 1 million daily users to nearly 6 million [14]. We are not aware specific reasons for the other spikes, the most notable spanning October to December 2014. However, as we do not see similar artifacts at our other sites, and taking into account that most connections do not target guard servers, we speculate that a local user was running a Tor relay during these times, offering the university's excellent bandwidth to the Tor network. To verify that hypothesis, we analyze the TLS fingerprints of the connections from this site to the Tor network. In particular, we focus on two bits of information that each client sends in its TLS *client hello* message: the lists of cipher suites and TLS extensions that it supports, which both depend on the interplay between the versions of Tor and OpenSSL. This analysis reveals that the spikes in December 2014, the Mevade spike between August and November 2013, the spike in February 2014, and the spike in March 2013 all map to specific TLS fingerprints, indicating a single software responsible for each.

Looking at our other sites, site N3 and site X1 exhibit a generally low level of Tor connections (1,286 and 418 connections per day on average, respectively) in comparison to site N1 (9,366/day). Connections there mostly terminate at guard nodes in the Tor network (80% and 75% of connections respectively), suggesting client activity. Site N2 has the largest number of connections into the Tor network among all of our sites (21,675/day on average), with connections steadily increasing from 2,818/day in February 2013 to 88,666 in February 2015. The distribution of connections changes starting in mid-2014, going from 72% terminating at guard nodes in January 2014 to just 38% in January 2015. We again assume this to be a case of having well-established Tor servers inside the network of this university.

5.3 Connection durations

Another piece of information available to a passive observer of the Tor network is the duration of connections going to Tor relays. Table 3 gives an overview of the durations to guard nodes that we encountered at our 4 sites. At each of our sites, we see a few very long connections, with at least one connection having a duration of more than 6.8 days in each case. However, the distribution of durations is highly skewed towards very

Site	1st Qu.	Med.	Mean	3rd Qu.	Max
N1	3.0	3.0	9.5	10.0	9,839
N2	3	5.5	18	15.5	22,280
N3	1.2	3.0	7.2	3.1	16,370
X1	3.0	3.0	7.8	3.3	10,090

Table 3: Summary of guard relay connection durations for each site in minutes. Qu. = Quantile, Med. = Median.

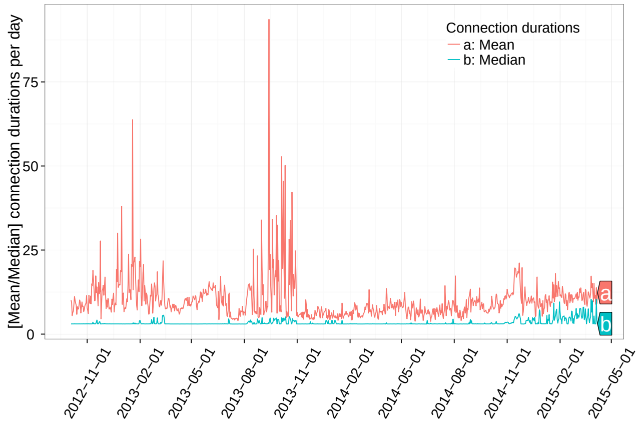

Figure 3: Median and mean guard relay connection durations at site N1 in minutes.

short connections. Depending on the site, the median connection duration across the data set is between 3.0 and 5.5 minutes, with the mean being a bit higher at 7.2 to 18 minutes. Figure 3 shows a comparison of the *daily* mean and medium durations at site N1, illustrating that the mean remains stable over time while the median fluctuates more, potentially due to local user activity.

We find a partial explanation for this behavior by examining how Tor relays establish connections between each other. When two Tor relays set up a circuit, they keep the TLS session alive for up to three minutes to potentially reuse for followup requests; only if there are no further circuits going over this connection during that time, they will tear it down [3]. However, from the literature we could not identify an explanation for the even shorter duration that we see frequently as well. Their high number (19%, 9%, 35% and 17% of all connections for N1, N2, N3, and X1, respectively) points towards a systematic reason. Possible explanations include Tor clients using short-lived connections for internal housekeeping, independent of user activity (e.g. to update their relay lists); and implementation artifacts.

6. SERVER CHARACTERISTICS

In this section, we take a look at the server side of the Tor, beginning with an examination of the server version changes in §6.1, followed by a look at the server-chosen cipher suites in §6.2.

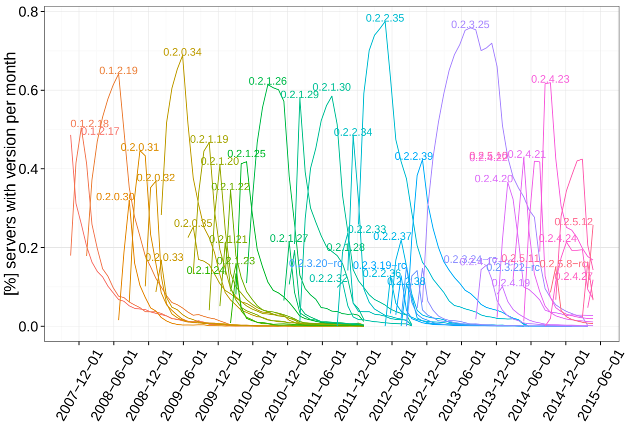

Figure 4: Tor versions used by relay nodes, according to CollecTor network status consensus information. Graph does not include versions with peak usage < 10%.

6.1 Tor Server Versions

The Tor network consensus introduced in §5.1 provides the software versions for all running Tor relays. We extracted these and show their distribution over time in Figure 4. While generally the uptake of new server versions is rather fast, we see a long tail of servers that remain on older releases for a significant period of time. From a deployment perspective, this makes sense; unlike for the Tor client software which, when used in form of the Tor Browser bundle, comes with an autoupdate functionality, administrators install Tor relay servers either manually or via the package management system of their operating system. Considering this, we deem the update rate surprisingly good, suggesting a high level of motivation among server operators to update the software diligently, likely due to their interest to protect Tor users' privacy as much as possible. Furthermore, it certainly helps that Tor's developers tend to be well-connected within the OS community, with some of them being, e.g., also Debian developers.

Inspecting the data in more detail reveals that a large number of Tor versions never sees widespread adoption. In total, we observe 314 different versions in the consensus data set. Of these, only 46 versions ever reach a usage level of more than 10% of all relays. Of the 269 versions with a maximum usage level below 10%, 249 are alpha or release candidate versions. As Figure 4 shows, there are only 6 versions of Tor that exhibit a combined use of more than 60% of all relay nodes. There is a repeating pattern of specific versions like, e.g,. 0.2.2.36 to 0.2.2.38, do not see any widespread use, while their parent version keeps enjoying popularity (which however then ends rapidly eventually). This kind of behavior suggests that OS distributions may not include certain versions of the software, preventing it from seeing widespread adoption.

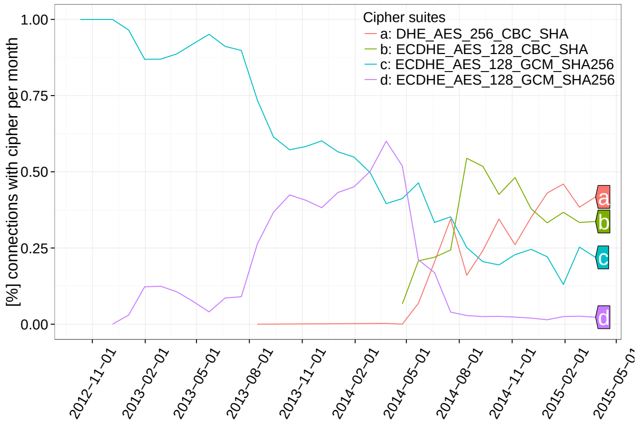

Figure 5: TLS cipher suites in connections at site N3.

6.2 Server Cipher Suites

With this knowledge, we can take a deeper look at the Notary data set. Another piece of information present in our data is the cipher suite that a server chooses in its TLS *server hello* message, which represents the encryption algorithm that the remainder of the TLS session will then use.

Figure 5 shows the main cipher suites that the outgoing connections at site N3 selected, excluding those that only a small fraction of all connections used (< 10%). The figure suggests a number of encouraging conclusions. Tor, in general, chooses secure cipher suites: all server selections we see use ephemeral keys and are thus perfectly forward secure. This indeed matches one of the original design goals of Tor, contributing also to its choice to avoid session resumptions (see §4).

The plot also shows that Tor connections started to switch from Diffie-Hellman (DH) key exchange to Elliptic Curve Diffie-Hellman (ECDH) in December 2012. The process has proceeded only slowly though and is still ongoing: 20% of the connections still use DH nowadays. Examining the DH key exchange in more detail reveals that its parameter size is always 1024 bits; Tor apparently never uses larger parameters. We assume that the reason for the continuing use of DH key exchanges lies in the OpenSSL versions that are installed on Tor servers. Some operating system providers have excluded ECDH support from their OpenSSL libraries for a long time due to fears of patent claims [11], making DH key exchanges the only viable alternative for perfect forward secrecy. While 1024 bit keys are not yet considered insecure, their use is discouraged. Since a sizeable percentage of connections is still using DH key exchanges, Tor should consider switching the parameter size to 2048 bits.

For ECDH connections, we at first see an uptake of connections using AES-128 with SHA1 in cipher block chaining (CBC) mode, which then rapidly switches to either AES-256 with SHA1 and CBC, or AES-128 using Galois/Counter-Mode (GCM) and SHA-256. The reason for this is probably that OpenSSL only supports GCM starting with OpenSSL 1.0.1. Version 1.0.0, which is still maintained, cannot use this cipher mode. Since GCM is the preferential choice of cipher suites, we assume that Tor falls back to CBC if not available. EC connections almost exclusively use the secp256r1 curve, which also is the most commonly supported curve on web servers [4]. Interestingly, we do not see any connections that would use the RC4 stream cipher. Considering its known weaknesses [29], this confirms the Tor developers proactive approach to key security.

The results at the other three sites show similar behavior, including a matching cipher distribution.

7. DISCUSSION AND CONCLUSION

This paper presents a longitudinal measurement study of Tor's network-level activity, derived from passively collected TLS connection information at four large-scale network sites over the course of more than 3 years. Generally, our study confirms that Tor pays attention to choosing TLS security parameters carefully, including ensuring forward secrecy, avoiding broken ciphers and picking modern cryptographic primitives. However, we also notice that a significant number of servers keeps using a Diffie-Hellman key exchange with a parameter size of 1024, which could become a security risk soon. Our analysis also shows that while server operators tend to update their software quickly, there is a a significant long-tail of systems that keep using outdated versions for significant periods of time.

For the reader not intricately familiar with Tor, one surprising result might be the ease with which one can identify Tor connections on the network by their characteristic use of X.509 certificates. For environments aiming to block Tor traffic—common not only from a censorship perspective, but also inside many enterprise environments—this suggests an alternative route to the standard approach of tracking Tor relays through blacklists, which need frequent updates. Interestingly, Tor switched to the current certificate scheme precisely to avoid such detection. As [32] discusses, earlier versions used "funny-looking certs [that] made Tor pretty easy to profile". With Tor 0.2.0.20, they switched to the current scheme to better blend in. However, as our study shows, detection remains an arms race, and an attacker with the ability to match regular expressions against certificates on the wire can easily identify Tor traffic today. Going forward, Tor could raise the bar further by avoiding the tell-tale signs that our detector picks up on. However, longer term, their strategy to rely on pluggable transports promises a better chance to render their users invisible again.

8. REFERENCES

[1] 3rd, D. E. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066, Jan. 2011.

[2] Amann, J., Vallentin, M., Hall, S., and Sommer, R. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Tech. Rep. TR-12-014, International Computer Science Institute, Nov. 2012.

[3] Biryukov, A., Pustogarov, I., and Weinmann, R.-P. TorScan: Tracing Long-Lived Connections and Differential Scanning Attacks. In *Proc. ESORICS* (2012).

[4] Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. Elliptic Curve Cryptography in Practice. In *Financial Cryptography and Data Security* (2014).

[5] Bro Network Monitoring System. https://www.bro.org.

[6] Chaabane, A., Manils, P., and Kaafar, M. A. Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *Proc. NSS* (2010).

[7] Christin, N. Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace. In *Proc. WWW* (2013).

[8] CollecTor: Your friendly data-collecting service in the Tor network. https://collector.torproject.org/.

[9] Dhungel, P., Steiner, M., Rimac, I., Hilt, V., and Ross, K. Waiting for Anonymity: Understanding Delays in the Tor Overlay. In *Proc. P2P* (2010).

[10] Dingledine, R., and Mathewson, N. Tor Protocol Specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt.

[11] Enable Elliptical Curve Diffie-Hellman (ECDHE) in Linux, July 2013. https://www.internetstaff.com/enable-elliptical-curve-diffie-hellman-ecdhe-linux/.

[12] Fifield, D. Flash Proxies. https://crypto.stanford.edu/flashproxy/.

[13] Friedl, S., Popov, A., Langley, A., and Stephan, E. Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension. RFC 7301, July 2014.

[14] Hopper, N. Challenges in Protecting Tor Hidden Services from Botnet Abuse. In *Proc. Financial Cryptography and Data Security* (2014).

[15] Hopper, N., Vasserman, E. Y., and Chan-TIN, E. How Much Anonymity Does Network Latency Leak? *ACM Trans. Inf. Syst. Secur. 13*, 2 (Mar. 2010), 13:1–13:28.

[16] Hurley, R., Prusty, S., Soroush, H., Walls, R. J., Albrecht, J., Cecchet, E., Levine, B. N., Liberatore, M., Lynn, B., and Wolak, J. Measurement and Analysis of Child Pornography Trafficking on P2P Networks. In *Proc. WWW* (2013).

[17] ICSI SSL Notary. http://notary.icsi.berkeley.edu.

[18] Le Blond, S., Manils, P., Chaabane, A., Kaafar, M. A., Castelluccia, C., Legout, A., and Dabbous, W. One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users. In *Proc. LEET* (2011).

[19] Loesing, K. Measuring the Tor Network, Evaluation of Client Requests to the Directories to Determine total Numbers and Countries of Users. Tech. Rep. 2009-06-002, The Tor Project, June 2009.

[20] Loesing, K. Measuring the Tor Network from Public Directory Information. Tech. Rep. 2009-08-002, The Tor Project, Aug. 2009.

[21] Manils, P., Abdelberi, C., Blond, S. L., Kâafar, M. A., Castelluccia, C., Legout, A., and Dabbous, W. Compromising Tor Anonymity Exploiting P2P Information Leakage. *CoRR abs/1004.1461* (2010).

[22] Mccoy, D., Bauer, K., Grunwald, D., Kohno, T., and Sicker, D. Shining Light in Dark Places: Understanding the Tor Network. In *Proc. PETS* (2008).

[23] Meek. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[24] obfs2 (The Twobfuscator). https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt.

[25] obfs3 (The Threebfuscator). https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt.

[26] Overlier, L., and Syverson, P. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2006), SP '06, IEEE Computer Society, pp. 100–114.

[27] Paxson, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks 31*, 23-24 (1999).

[28] Perry, M. Tor-dev mailing list: Tor Browser 4.5a5 will change circuit expiry to 2hrs. https://lists.torproject.org/pipermail/tor-dev/2015-March/008548.html.

[29] Popov, A. Prohibiting RC4 Cipher Suites. RFC 7465, Feb. 2015.

[30] Salowey, J., Zhou, H., Eronen, P., and Tschofenig, H. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077, Jan. 2008.

[31] Tang, C., and Goldberg, I. An Improved Algorithm for Tor Circuit Scheduling. In *Proc. CCS* (2010).

[32] Tor Wiki – TLS History. https://trac.torproject.org/projects/tor/wiki/org/projects/Tor/TLSHistory.

[33] Tor Blog – The Lifecycle of a New Relay. https://blog.torproject.org/blog/lifecycle-of-a-new-relay.

[34] Tor Directory Protocol, Version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt.

[35] Tor FAQ. https://www.torproject.org/docs/faq.

[36] The Legal FAQ for Tor Relay Operators. https://www.torproject.org/eff/tor-legal-faq.html.en.

[37] Tor Pluggable Transport Specification. https://gitweb.torproject.org/torspec.git/tree/pt-spec.txt.

[38] Tor TLS Connection Source code. https://doxygen.torproject.org/tortls_8c_source.html#l01178.