Comparative Analysis of DoT and HTTPS Certificate Ecosystems

Ali Sadeghi Jahromi
Carleton University
alisadeghijahromi@cmail.carleton.ca

AbdelRahman Abdou
Carleton University
abdou@scs.carleton.ca

***Abstract*—The Internet's Public Key Infrastructure (PKI) has been used to provide security to HTTPS and other protocols over the Internet. Such infrastructure began to be increasingly relied upon for DNS security. DNS-over-TLS (DoT) is one recent rising and prominent example, whereby DNS traffic between stub and recursive resolver gets transmitted over a TLS-secured session. The security research community has studied and improved security shortcomings in the web certificate ecosystem. DoT's certificates, on the other hand, have not been investigated comprehensively. It is also unclear if DoT client-side tools (*e.g.,* stub resolvers) enforce security properly as modern-day browsers and mail clients do for HTTPS and secure email. In this research, we compare the DoT and HTTPS certificate ecosystems. Preliminary results are so far promising, as they show that DoT appears to have benefited from the PKI security advancements that were mostly tailored to HTTPS.**

I. Introduction

The Domain Name System (DNS) is critical to the Internet; every communication that uses a domain name starts with a name resolution to discover the associated IP address. Due to the lack of security in DNS, a large number of privacy and security problems, such as large-scale monitoring [31], worldwide DNS manipulation, and censorship [22], [8], [36] arose. The proliferation of numerous IoT devices exacerbates these problems; 25 billion IoT devices are expected to be Internet-connected by 2025 [28]. Such devices resolve domain names to communicate with their cloud backends. The lack of privacy in DNS enables intermediate entities, such as Internet Service Providers (ISPs) and other Autonomous Systems (ASes), to identify the type of IoT devices that a user owns [7].

DNS-over-TLS (DoT) was standardized in 2016 to protect DNS messages between the stub- and recursive- resolver from manipulation, eavesdropping, and privacy leaks [42], [12]. When configured with persistent connections, DoT adds negligible overhead to vanilla DNS [27]. Page load times with DoT are faster than other alternatives, *e.g.,* DNS-over-HTTPS (DoH) [23]. DoT has been adopted by large public DNS resolvers such as Cloudflare, Google, and Comcast [15]. Popular platforms, such as Android *"Pie"* and Linux *"Systemd-resolved"*, among others, have implemented DoT stub-resolvers [16]. As a result, DoT queries have increased over the Internet since 2018 [27]. Similar to securing web (HTTPS) and email (S/MIME), DoT in the Internet relies on the Internet's Public Key Infrastructure (PKI) and associated Certification Authorities (CAs) for signing and delivering resolvers' standard X.509 certificates.

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2021
25 February 2021, Virtual
ISBN 1-891562-67-3
https://dx.doi.org/10.14722/madweb.2021.23027
www.ndss-symposium.org

On security, we investigate whether DoT would be susceptible to classic (or historic) PKI shortcomings, such as invalid/self-signed certificates, weak cryptographic parameters, or fraudulent certificates issued by compromised CAs. Over time, browsers enhanced HTTPS security by stringent certificate validation and indispensable demand of security features, like the placement of certificates in Certificate Transparency (CT) logs [38], [6]; CAs have accordingly been stepping-up their issuance standards. It is unclear how many of the browser-implemented reinforcements for HTTPS are adopted in DoT, and how the relatively lax security in DoT affects issued certificates. For example, successful authentication [4] and encryption are unnecessary in DoT depending on the client's configured usage profile (see opportunistic mode in RFC 7858 [42]).

We present results upon comparing a random sample of DoT and HTTPS certificates collected from Rapid7 [32]. Particularly, this paper contributes results upon comparing DoT and HTTPS certificates for the following aspects:

- Distribution and characteristics of certificate issuers (Sec. IV).
- Certificate parameters, including validity windows and cipher-suites (Sec. V).
- Proportion and distribution of certificates in CT-logs (Sec. VI).

Our results highlight non-major differences between both ecosystems, including differences in: the dominant CA, certificate validity, and cryptographic properties. The proportion of invalid certificates appears almost similar in both ecosystems, likewise the expiry windows and cryptographic functions. We also found almost equivalent rates of CT-log inclusion in both ecosystems. These results suggest that so far, the deployment and usage of DoT certificates in practice appears promising, and not significantly affected by the lack of strict security checks in sub-resolvers.

II. Related Work

Numerous previous literature analyzed HTTPS certificates. In 2012, Durmuric *et al.* [19] actively collected HTTPS certificates in the wild, and found approximately 12.8% invalid certificates. Chung *et al.* [14] used Rapid7's dataset, plus active collections of certificates, and found that 65% of web certificates were invalid. More recently, Tehrani *et al.* [39] studied the security and HTTPS certificates of alerting authorities in USA and found 15% are not-existing or invalid certificates.

DoT was standardized 16 years after HTTPS. In 2019, Lu *et al.* [27] discovered 1.5K open DoT resolvers over the Internet, and 25% of the used certificates by these resolvers were invalid. Others studied security and overhead of DoT [41], and DoT's resistance to traffic analysis attacks [35], [24].

III. Data Collection and Analysis Methodology

We employ a passive measurement methodology, where we used two certificate datasets gathered by Rapid7's Project Sonar [33] in March 2020: one for HTTPS, the other DoT. Project Sonar actively scans the Internet for different services, similar to Censys [18]. Project sonar scans for a specific service port over the IPv4 address space using Zmap [32]. Further information and protocol metadata are then extracted by communicating with the IP addresses over the open ports, analogous to banner grabs [18]. The gathered data then gets released on the Open Data web page [32].

The DoT and HTTPS datasets that we used were gathered by Rapid7 on the 4^{th} and 23^{rd} of March 2020, respectively. They consist of X.509 certificates and their fingerprints collected from scanning DoT resolvers (port 853) and HTTPS webservers (port 443), respectively [32]. The DoT dataset consists of ∼10K certificates, the HTTPS ∼9.3M. However, ∼54% of the DoT certificates (collected over ports 853) were also found in the HTTP certificate dataset (collected over port 443). The overlapping certificates belonged to less than a quarter of the DoT certificate owners (942 of 4632 unique subjects); these owners apparently chose to use the same certificate for both DoT and HTTPS.

We used the OpenSSL toolkit, `SCTcheck`, and the `Pyopenssl` library to parse and process certificates in both datasets. For certificate validity, we used OpenSSL's *verify* utility. Validating a certificate also involves chain validation [11]. For that, we relied on Mozilla's intermediate CA certificate lists released on April 2020 [29],[1] and the certificate root store of Ubuntu 18.04. OpenSSL's *verify* utility stops validating certificates when the first invalidity reason is encountered. For example, if the certificate is self-signed and expired, the tool would report only one of those two invalidity reasons and quit. As such, for every certificate in our dataset, we had to write our own scripts to list all reasons of invalidity.

Finally, we noticed that many certificates had an IP address placed as a string in the *Subject* and *Issuer* fields. Those were mostly self-signed certificates. We used MaxMind's IP/Country database to geolocate these addresses.[2]

[1]The difference between the scanning time and the intermediate certificates release time might lead to a negligible inaccuracy as Mozilla releases the list of intermediate certificates daily [29].

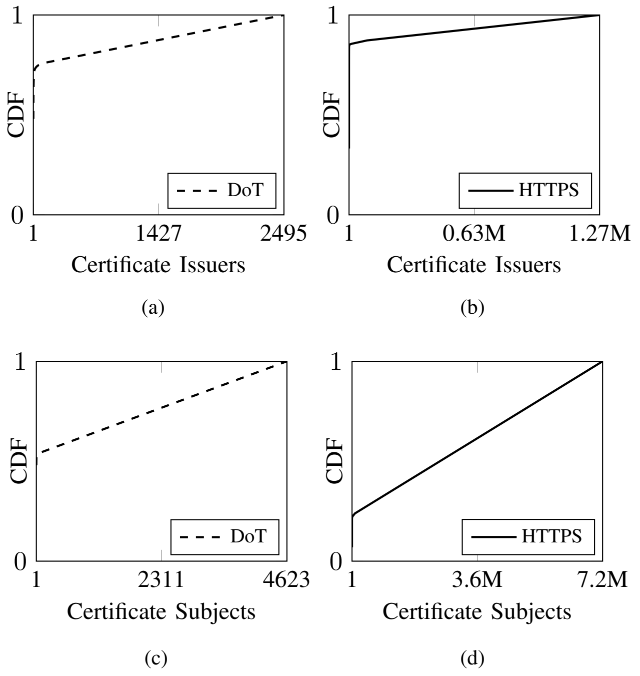

Fig. 1. Distribution of Certificate Issuers and Subjects in DoT and HTTPS.

IV. Certificate Issuers

In this section, we investigate the distribution of certificate issuers and subjects among DoT and HTTPS certificates.

Figures 1(a) and 1(b) show CDFs of certificate issuers. For DoT, 105 of 2495 issuers were responsible for ∼76% of DoT certificates. In HTTPS, ∼91K of ∼1.27M issuers were responsible for 87% of HTTPS certificates. Although many issuers were untrusted, about two-thirds of the certificates in both ecosystems were issued by trusted issuers. The long tails in Figures 1(a) and 1(b) represent issuers that issued very few certificates. The majority of those ($>$ 90% for both ecosystems) were either non-existent issuers or untrusted CAs, and they mostly issued one certificate. Collectively, these have issued $<$ 37% of certificates in both ecosystems.

Figures 1(c) and 1(d) show a CDF of the *Subject* field, which identifies the certificate owner (the entity that owns the public key). Many (∼50%) DoT certificates in our dataset are owned by a single entity. This organization is *incapsula.com.* Imperva Incapsula is an American cybersecurity company, which provides cloud-based security solutions to its customers. We assume that as a part of their security solutions, they deployed these DoT resolvers for their clients. The distribution was less skewed in HTTPS, with *Technicolor* network devices owning ∼7% of HTTPS certificates in our dataset.

Figure 2 shows the top five DoT issuers and the proportion of HTTPS certificates that they have issued. *GlobalSign* is

[2]Securely geolocating these addresses [3] is out of the scope of the current paper.

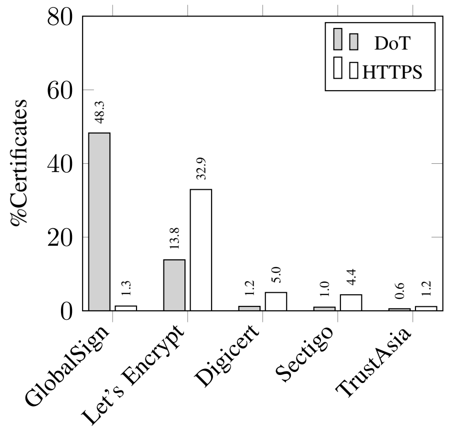

Fig. 2. The proportion of certificates issued by the 5-most popular CAs in DoT, and the proportion of HTTPS certificates issued by these CAs.

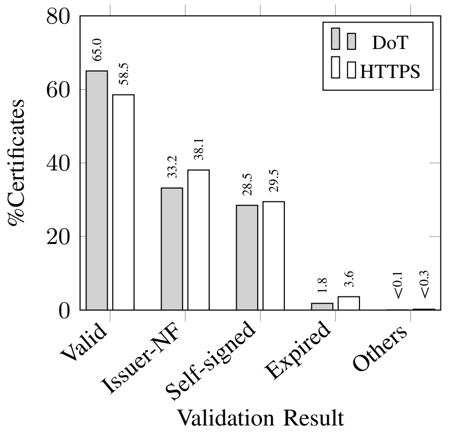

Fig. 3. Proportion of valid and invalid certificates in our datasets, and the reasons of invalidity.

the dominant CA in DoT. However, the majority of the certificates *GlobalSign* has issued belong to a single owner (*incapsula.com*), rather than being distributed amongst many DoT services. The ∼10K DoT certificates in our dataset belong to ∼4.6K unique owners; the numerous DoT certificates that *GlobalSign* has issued belong to only 31 of the 4.6K owners. In contrast, the 13.8% of DoT certificates issued by *Let's Encrypt* belong to 1348 owners. So *Let's Encrypt* can be considered equally popular in the DoT ecosystem, despite issuing significantly fewer DoT certificates (13.8%). For HTTPS, *Let's Encrypt* remains the dominant certificate issuer, which aligns with previous literature [34], [2], whereas *GlobalSign* has only issued 1.3% of the HTTPS certificates in our dataset.

Summary. *GlobalSign* and *Let's Encrypt* are the two most popular CAs in the DoT ecosystem. The former was founded in 1996 [20], and the latter has established a strong security reputation worldwide. Both CAs employ the Automatic Certificate Management Environment (ACME) protocol [10], which automates certificate renewal. Together they are responsible for almost two-thirds of the DoT certificates in our dataset. Our analysis on certificate issuers, thus, shows no outstanding signs of security concerns in the DoT ecosystem.

V. Certificate Parameters and Characteristics

We now move to certificate parameters, specifically focusing on the characteristics of invalid certificates (Sec. V-C to Sec. V-A), expiry windows (Sec. V-D), and cryptographic properties (Sec. V-E).

Figure 3 shows the proportion of valid and invalid certificates in both ecosystems, with invalid certificates clustered by reasons of invalidity. As a certificate can be invalid for multiple reasons (*e.g.,* expired and has an untrusted issuer), the sum of the proportions exceeds 100% on the vertical axis. As shown, 65% of the DoT certificates in our dataset were valid, compared to 58% for HTTPS. We now analyze and compare the top three invalidity reasons across both ecosystems.

A. Issuer-not-Found

The most common invalidity reason is Issuer Not Found *"Issuer-NF"*, meaning the leaf certificate issuer could not be found in the list of trusted (intermediate) issuers or root certificates. The majority of the DoT issuer-NF certificates were issued by *Fortinet* network devices, which also constitute a considerable proportion of issuer-NF certificates in our HTTPS certificate dataset. But the most popular issuer of issuer-NF HTTPS certificates was *Technicolor*. These certificates had long expiry-windows (⩾5 years), which raises further concerns about their security. Chung *et al.* [14] explained that many client-side network devices regularly generate invalid certificates that increase the proportion of invalid certificates in an ecosystem. Overall, issuer-NF certificates in both ecosystems were mostly issued by network devices rather than CAs, and had relatively long expiry windows.

B. Self-Signed Certificates

A self-signed certificate is one where the Common Name (CN) field under *Issuer* is the same as the CN field under *Subject*. Self-signed certificates not present in a trust root store expose users to several attacks, including MitM, resolver impersonation, and pharming attacks. From Figure 3, the proportion of self-signed certificates in our data set was almost equal in both ecosystems: 28.5% of DoT certificates and 29.5% of HTTPS.

There were ∼2.4K unique issuers responsible for issuing the 2.85K self-signed DoT certificates (the 28.5% in Fig. 3)—no significant dominant of self-signed certificates in DoT. The two most popular CN strings in self-signed DoT certificates were "Server" and "localhost," respectively constituting 8% and 5% of the self-signed DoT certificates. In contrast, ∼1M unique issuers have issued the 2.7M self-signed HTTPS certificates (29.5% in Fig. 3), suggesting that the distribution of issuers among self-signed HTTPS certificates is more skewed than in DoT. The most popular two self-signed HTTPS certificate issuers were "Vigor Router" (5%) and "192.168.1.1" (4%).

Of the self-signed certificates, 61% (DoT) and 15% (HTTPS) had a string that mimics an IP address structure in the CN fields. Table I shows that using IPv4 addresses as the subject field is only present in self-signed certificates. From Table II, public IP addresses form $> 80\%$, and $> 99\%$ of the found IP addresses in HTTPS and DoT self-signed certificates, respectively.

Upon geolocating the public IP addresses, we found that DoT's addresses map to 17 countries and HTTPS' to 201, which expectedly illustrates the size and distribution difference between both ecosystems. Half the HTTPS self-signed certificates with an IP address in the CN fields are located in the DoT's 17 countries. Therefore, despite minor differences between DoT and HTTPS self-signed certificates in our dataset, their proportion, distribution, and characteristics generally appear comparable.

TABLE I
SUBJECT FIELDS CONTAINING IP ADDRESSES, AND THEIR COUNT AMONG SELF-SIGNED AND NON-SELF-SIGNED CERTIFICATES.

Data Set	*Subject Field*	*Self-Signed*	*Not Self-Signed*
DoT	IP address	1,736	0
	Non IP address	1,100	7,119
HTTPS	IP address	404,231	0
	Non IP address	2,344,244	6,573,467

TABLE II
DISTRIBUTION OF PUBLIC AND PRIVATE IP ADDRESSES.

IP address	*HTTPS(%)*	*DoT(%)*
Public	324,571 (80.4)	1,724 (99.3)
Private	79,160 (19.6)	12 (0.7)

C. Expired Certificates

A certificate has a *Not-After* field indicating the date and time when the certificate validity ends. We analyze the proportion of expired certificates in both datasets. We classify a certificate as expired if the *Not-After* field is before the time that Rapid7 has collected said certificates.

From Figure 3, although the portion of expired certificates in HTTPS is double that in DoT, they generally constitute a small proportion in both ecosystems. This can be attributed to CAs increasingly employing the ACME protocol. Investigating the top trusted CAs in both DoT and HTTPS, we observed from our dataset that all of these CAs except GoDaddy (as of this writing) are supporting ACME based certificate issuance and auto-renewal mechanisms to avoid unexpected expiration of certificates.[3]

We observe that the majority of expired certificates in both ecosystems are issued by Let's Encrypt: ∼78% for DoT, ∼60% for HTTPS. Despite being more popular among HTTPS certificates, the proportion of the expired certificates issued by Let's Encrypt in HTTPS is smaller compared to DoT. As such, there appears no positive correlation between Let's Encrypt-issued certificates and expired ones. Let's Encrypt enforces an expiry window of $\leqslant$ 90 days, which suggests that Let's Encrypt DoT certificates may not have been configured for auto-renewal. Next, we shed light on the certificate expiry windows in both ecosystems.

[3]GoDaddy provides independent mechanisms for automatic certificate renewal [26], [21].

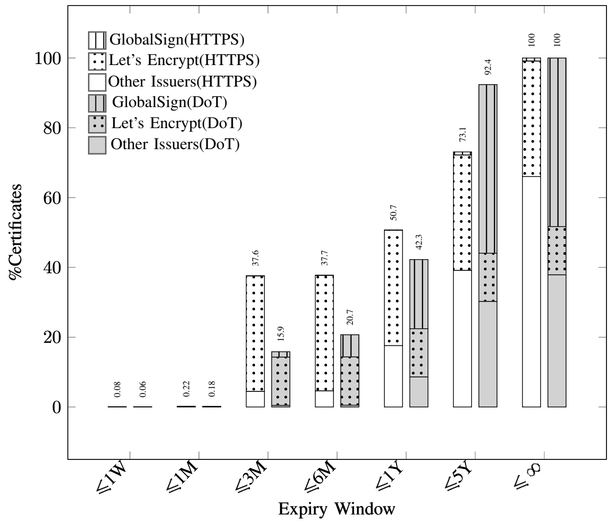

Fig. 4. Distribution of expiry windows in DoT and HTTPS certificates. White bars represent HTTPS, gray DoT. The proportion issued by each of the two most popular issuers (namely, GlobalSign and Let's Encrypt) is illustrated.

D. Expiry Windows

Shorter certificate lifespan is a healthier security practice [40], but might lead to functional and security problems as, *e.g.,* management and operational costs increase [13]. Over the past several years, the security community has been advocating for shorter certificate expiry windows for the web. For example, before 2018, browsers required certificates to have an expiry window of 39 months or shorter, and later (from March 2018) agreed on accepting certificates with a maximum lifetime of 825 days (∼2 years) [1], [30]. Google then announced the maximum accepted certificate lifespan by the Chrome browser would be 397 days (∼1 year) [37]. On September 2020, Apple followed suit, accepting a maximum certificate life span of 398 days (∼1 year) [5].

Figure 4 shows a cumulative histogram of various expiry windows for the certificates in our dataset. A negligible number of certificates were issued with a lifespan of less than a month in both DoT and HTTPS certificates. Unexpectedly, a few number of (invalid) certificates had a negative expiry window—their *not-after* date was before their *not-before*. Very few certificates had less than one month expiry window. For HTTPS, 50% of certificates in the $\leqslant$1M bin have been issued by trusted CAs, such as GTS, Sectigo, and Digicert. In contrast, all DoT certificates in that bin were issued by untrusted issuers. That is a minor plus point for HTTPS, but is potentially negligible due to the significantly small proportion of certificates in that bin for both ecosystems. For the $\leqslant$3M bin, the HTTPS proportion is more than double the DoT, which can be attributed to the larger proportion of Let's Encrypt-issued certificates in HTTPS (see Sec. IV); the proportion of certificates the CA has issued in this bin is almost equal across both ecosystems—∼88% for HTTPS, ∼87% for DoT. Analogous patterns can be observed for the remaining bins.

Overall, certificates with less than one-year lifespan are common in HTTPS, which can be attributed to (1) browsers (including Chrome and Safari) enforcing such shorter lifespans, and (2) the popular CA in HTTPS, Let's Encrypt, only issues certificates with $\leqslant$ 90 days lifespan. This appears to have not been carried over to the DoT ecosystem—over 60% of the certificates in DoT were issued with a lifespan of $\leqslant$ 2 years, and Let's Encrypt has only issued ∼14% of DoT certificates in our dataset (*cf.* Figure 2). A corollary of the above two points, although there is no short certificate lifespan enforcement in DoT client-side tools, we observe that the majority of the DoT certificates were issued with relatively short expiry windows. The majority of certificates issued by the dominant DoT CA, GlobalSign, had an expiry window between one and two years. As such, DoT certificate life spans appear to not suffer severe security weaknesses.

We also noticed that a large portion of invalid certificates in HTTPS had a lifespan of more than five years; however, the portion of these enormous lifespan certificates in DoT is roughly three times smaller.

E. Public-key and Hash Function

In this section, we investigate public keys and hash function algorithms used in the DoT and HTTPS certificates as a part of their cryptographic properties. We rely on NIST recommendations for key management [9]. For asymmetric-key algorithms, $\leqslant$ 80 bits, such as RSA-1024 and ECC (160-223), is considered weak. For hash functions, SHA-1 is also considered weak, and was formally deprecated in 2011 [9].

Figure 5 shows the asymmetric key algorithms used in both certificates in our datasets. RSA-2048, which has sufficient strength of 112 bits, is used by the majority of certificates in both ecosystems, with ∼87% in HTTPS and ∼75% in DoT. The weaker variant, RSA-1024, is five-times more popular in DoT. The HTTPS ecosystem thus appears to have a slight advantage over DoT.

TABLE III
MAIN HASH FUNCTIONS

Hash Algorithm	*DoT*	*HTTPS*
SHA-256	∼98%	∼96%
SHA-1	∼2%	∼4%

Regarding hash functions, Table III shows that $> 95\%$ of certificates in both ecosystems used SHA-256, which has a security strength of 125 bits and is considered secure [9]. Only 2% of DoT certificates used SHA-1, but the proportion was double in HTTPS. Therefore, regarding the used hash functions, DoT certificates appear in a slightly better position. We note that there was a negligible proportion of $\leqslant 0.1\%$ of other deprecated and non-deprecated hash functions, such as MD5 and SHA-384, in both ecosystems.

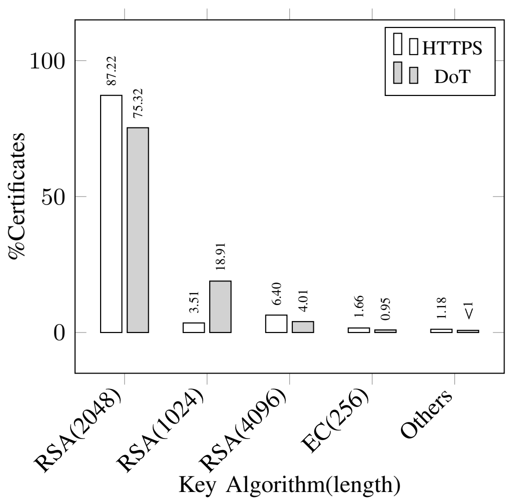

Fig. 5. Asymmetric key algorithms (length) in DoT and HTTPS certificates.

Summary. The characteristics of the DoT certificates appear to be comparable to HTTPS, especially in hash functions, certificate validity periods, and invalidity reasons. Minor exceptions exist in used public keys and expiry windows. However, similar to the previous section, we find no concerning evidence of lack of security in DoT certificates.

VI. INCLUSION IN CERTIFICATE TRANSPARENCY LOGS

CT-logs are append-only and publicly auditable databases of certificates [25]. Certificate issuers append their issued certificates to CT-logs. Upon the receipt of a certificate, CT-logs return a Signed Certificate Timestamp (SCT). The included certificate becomes publicly accessible, and the SCT is signed by the CT-log's private key to verify that the certificate has been included. Domain owners thus become aware of certificates issued for their domain, allowing them to detect bogus activity. We now compare two aspects: the proportion of certificates included in CT-logs, and the distribution of certificates among different CT-logs.

From our dataset, ∼61.4% and ∼66.8% of HTTPS and DoT certificates respectively had SCTs. Approximately 62% of the DoT certificates that contained SCTS were issued by either GlobalSign or Let's Encrypt. The top issuers adopted automatic inclusion of certificates in the CT-logs, and DoT certificates also benefited from this advancement.

Figure 6 shows a CDF of the distribution of certificates with SCTs among CT-logs. Both ecosystems exhibit a similar pattern. One notable point is that Digicert's CT2 log had been deactivated on May 2020 due to possible private key leakage [17]. In Figure 6, the line between points a and b, and points c and d illustrates the proportion of DoT and HTTPS certificates logged into Digicert CT2 log. As the proportion in Digicert CT2 log is greater in DoT, deactivation of this CT-log would negatively affect the DoT ecosystem. Regarding popularity, Google *Xenon2020* is the top CT-log among HTTPS certificates, whereas Sectigo *Mammoth* is the top among in DoT. Overall, the top three Google's CT-logs in HTTPS included more portion of certificates compared to the top three Google's CT-logs in DoT.

Summary. As browsers, like Chrome and Safari, enforced the inclusion of web certificates in CT-logs, CAs increasingly implemented CT-log inclusion policies [38], [6]. Therefore, approximately all of the valid certificates in both ecosystems have been logged. The forced CT-log inclusion in the HTTPS ecosystem appears to have boosted the inclusion of DoT certificates.

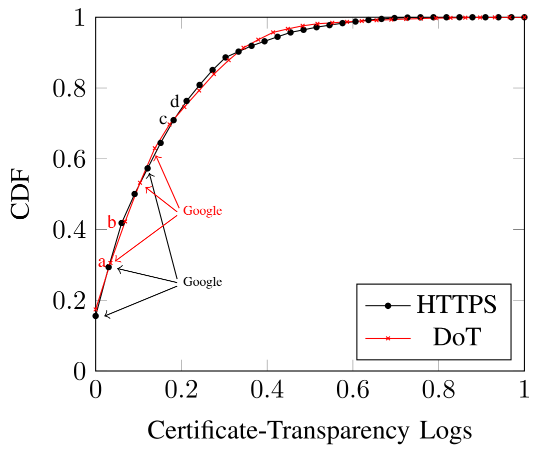

Fig. 6. Embedded SCTs in DoT and HTTPS certificates. The distance between red marked points with a, b, and the black points marked with c, d are representing the portion of certificates in DoT and HTTPS that were logged to the Digicert CT-2 log.

VII. Conclusion

The security of the HTTPS ecosystem has improved over time, with browser vendors, industry standards, CAs, and researchers coordinating efforts to advance aspects like CT-log inclusion, short certificate expiry windows, and improved cryptographic properties. Our analysis herein shows that the DoT ecosystem appears to have seized the momentum, reaping the benefits of the advancements made in HTTPS. We observed some differences between the invalid certificates of both ecosystems. Valid certificates, on the other hand, exhibit almost similar characteristics. Some certificates existed in both the DoT and HTTPS datasets. Those overlapping certificates are presumably stronger than ones present only in the DoT dataset because they (the overlapping) should satisfy the strict constraints of the HTTPS ecosystem. It would be unfair to judge the security of the DoT ecosystem with the DoT-only (non-overlapping) certificates because the overlapping ones are still used to establish DoT sessions in practice. As such, we took the conscious decision of conducting our analysis in both datasets as-is, without removing the overlapping certificates from the DoT dataset. A notable concern is that client-side DoT tools in the *opportunistic* mode can undermine the observed security strength of the DoT ecosystem. Client-side DoT tools must thus mimic the tenacity of modern browsers. Our research herein constitutes a work-in-progress, reflecting part of our journey towards a comprehensive evaluation of DoT security. Next, we will focus on evaluating client-side DoT tools, including a comparative assessment involving DoT stub-resolvers, web browsers, and mail clients, and their security features.

VIII. Acknowledgments

The second author acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant.

References

[1] "Baseline requirements for the issuance and management of publicly-trusted certificates, v.1.2.3," in *CA/Browser Forum*, 2014.

[2] J. Aas, R. Barnes, B. Case, Z. Durumeric, P. Eckersley, A. Flores-López, J. A. Halderman, J. Hoffman-Andrews, J. Kasten, E. Rescorla, S. Schoen, and B. Warren, "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web," in *ACM CCS*, 2019.

[3] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Location verification on the Internet: Towards enforcing location-aware access policies over Internet clients," in *IEEE Conference on Communications and Network Security (CNS)*, 2014.

[4] F. Alaca, A. Abdou, and P. C. van Oorschot, "Comparative Analysis and Framework Evaluating Mimicry-Resistant and Invisible Web Authentication Schemes," *IEEE Trans. Dependable and Secure Computing (TDSC)*, vol. 18, no. 2, pp. 534–549, 2021.

[5] Apple, "About upcoming limits on trusted certificates," 03/03/2020. [Online]. Available: https://support.apple.com/en-ca/HT211025

[6] ——, "Apple's certificate transparency policy," 2019. [Online]. Available: https://support.apple.com/en-ca/HT205280

[7] N. Apthorpe, D. Reisman, and N. Feamster, "Closing the blinds: Four strategies for protecting smart home privacy from network observers," *arXiv preprint arXiv:1705.06809*, 2017.

[8] S. Aryan, H. Aryan, and J. A. Halderman, "Internet censorship in Iran: A first look," in *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[9] E. Barker, "Recommendation for key management: Part 1 – general," National Institute of Standards and Technology, 2020.

[10] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, "Automatic Certificate Management Environment (ACME)," RFC 8555, 2019.

[11] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, 2008.

[12] S. Bortzmeyer, "DNS Privacy Considerations," RFC 7626, Aug. 2015.

[13] L. Chuat, A. Abdou, R. Sasse, C. Sprenger, D. Basin, and A. Perrig, "SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust," in *IEEE European Symposium on Security and Privacy (EuroSP)*, 2020.

[14] T. Chung, Y. Liu, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measuring and applying invalid SSL certificates: The silent majority," in *ACM IMC*, 2016.

[15] S. Dickinson, "DNS Privacy Public Resolvers," Nov 2019. [Online]. Available: https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Public+Resolvers

[16] ——, "DNS Privacy Clients," Dec 2020. [Online]. Available: https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Clients#DNSPrivacyClients-DOT

[17] Digicert, "Moving forward: What DigiCert's CT2 log retirement means for you," May 2020. [Online]. Available: https://www.digicert.com/dc/blog/digicert-statement-on-ct2-log/

[18] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *ACM CCS*, 2015.

[19] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *ACM IMC*, 2013.

[20] GlobalSign, "GlobalSign, About Us," 2001. [Online]. Available: https://web.archive.org/web/20010205010600/http://globalsign.net/company/index.cfm

[21] GoDaddy, "Renewing my SSL Certificate." [Online]. Available: https://ca.godaddy.com/help/renewing-my-ssl-certificate-864

[22] C. Grothoff, M. Wachs, and M. Ermert, “NSA’s MORECOWBELL: Knell for DNS,” 2015.
[23] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, “Analyzing the costs (and benefits) of DNS, DoT, and DoH for the modern web,” in *ACM Applied Networking Research Workshop*, 2019.
[24] R. Houser, Z. Li, C. Cotton, and H. Wang, “An investigation on information leakage of dns over tls,” in *ACM International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*, 2019.
[25] B. Laurie, A. Langley, and E. Kasper, “Certificate Transparency,” RFC 6962, 2013.
[26] Let’s Encrypt, “Let’s Encrypt Certificates on GoDaddy Hosting.” [Online]. Available: https://letsencrypt.org/docs/godaddy/
[27] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, “An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?” in *ACM IMC*, 2019.
[28] A. Mourad, R. Yang, P. H. Lehne, and A. De La Oliva, “A baseline roadmap for advanced wireless research beyond 5G,” *Electronics*, vol. 9, no. 2, p. 351, 2020.
[29] Mozilla, “CA/Intermediate Certificates,” Apr 2020. [Online]. Available: https://wiki.mozilla.org/CA/Intermediate_Certificates
[30] M. Špaček, “Maximum HTTPS certificate lifetime to be 1 year soon,” 2020. [Online]. Available: https://www.michalspacek.com/maximum-https-certificate-lifetime-to-be-1-year-soon
[31] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson, “Global measurement of DNS manipulation,” in *USENIX Security Symposium*, 2017.
[32] Rapid7, “About Open Data.” [Online]. Available: https://opendata.rapid7.com/about/
[33] ——, “An Introduction to Project Sonar.” [Online]. Available: https://www.rapid7.com/research/project-sonar/
[34] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, “The rise of certificate transparency and its implications on the Internet ecosystem,” in *ACM IMC*, 2018.
[35] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, “Encrypted DNS→Privacy? A Traffic Analysis Perspective,” *arXiv preprint arXiv:1906.09682*, 2019.
[36] K. Singh, G. Grover, and V. Bansal, “How India Censors the Web,” in *ACM Conference on Web Science (WebSci)*, 2020.
[37] R. Sleevi, “Ballot SC22: Reduce Certificate Lifetimes,” 2019. [Online]. Available: https://github.com/cabforum/documents/pull/138/commits/2b06f7839daa45f685e37c51d74e255ef4ff9d75#diff-7f6d14a20e7f3beb696b45e1bf8196f2R1454
[38] E. Stark, R. Sleevi, R. Muminovic, D. O’Brien, E. Messeri, A. P. Felt, B. McMillion, and P. Tabriz, “Does certificate transparency break the web? measuring adoption and error rate,” in *IEEE Symposium on Security and Privacy (SP)*, 2019.
[39] P. F. Tehrani, E. Osterweil, J. H. Schiller, T. C. Schmidt, and M. Wählisch, “Who ya gonna call? (Alerting Authorities): Measuring Namespaces, Web Certificates, and DNSSEC,” *arXiv preprint arXiv:2008.10497*, 2020.
[40] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh, “Towards short-lived certificates,” *Web 2.0 Security and Privacy*, 2012.
[41] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, “Connection-oriented DNS to improve privacy and security,” in *IEEE Symposium on Security and Privacy (SP)*, 2015.
[42] L. Zhu, Z. Hu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, “Specification for DNS over Transport Layer Security (TLS),” RFC 7858, May 2016. [Online]. Available: https://tools.ietf.org/html/rfc7858