An Evaluation of X.509 Certificate Revocation and Related Privacy Issues in the Web PKI Ecosystem

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 June 2023, accepted 14 July 2023, date of publication 27 July 2023, date of current version 2 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3299357

RESEARCH ARTICLE

An Evaluation of X.509 Certificate Revocation and Related Privacy Issues in the Web PKI Ecosystem

DIANA GRATIELA BERBECARU, (Member, IEEE), AND ANTONIO LIOY
Politecnico di Torino, Dipartimento di Automatica e Informatica, 10129 Turin, Italy

Corresponding author: Diana Gratiela Berbecaru (diana.berbecaru@polito.it)

The work of Diana Gratiela Berbecaru was supported in part by the Italian Ministry of University and Research (MUR) D.M. 1062/2021—Programma Operativo (PON) "Ricerca E Innovazione."

ABSTRACT Supporting users to transact with websites securely in a privacy-preserving manner has become more challenging than ever in the web ecosystem based on public key infrastructures. While establishing TLS (Transport Layer Security) secure channels to web servers, the X.509 certificates are typically used for server authentication. Such certificates must be correctly *validated* by the clients upon use. This paper discusses first the X.509 certificate format and the main entities (like standardization bodies, browser vendors, and organizations) involved in the definition, management, and processing of X.509 certificates. Subsequently, we concentrate on certificate revocation status checking (part of certificate validation) and the related privacy aspects. Through experiments, we show that some common web browsers still incorrectly or incompletely perform certificate revocation (status) checking, mainly for the non Extended Validation (non-EV) certificates, although the certificates contain useful extensions, and the web browsers implement partly this task. To this aim, we analyzed first the certificates in the Alexa Top 1 Million (Top1M) list containing the most widely accessed websites in August 2021. Then, we exploited a local testbed to assess common browsers' behaviour while checking the revocation status of EV and non-EV certificates. For non-EV certificates, the *soft-fail* approach was typically encountered, meaning the web browsers established TLS connections with the web server even if the revocation data was not available. For the EV certificates, the browsers implemented stricter controls. We discuss privacy issues related to certificate status checking, outlining that the so-called OCSP stapling mechanism may respond better to client latency and user privacy concerns. Finally, we analyze the adoption of the OCSP stapling mechanism and the support for Google's Certificate Transparency project in the Majestic Top1M list of website certificates in 2022. This work bridges the gap between X.509 standards/guidelines and real-world applications' behaviour in applying recommendations while handling certificates.

INDEX TERMS X.509 certificates, revocation, OCSP stapling, security, measurement, privacy.

I. INTRODUCTION

Invented back in 1978 by Kohnfelder while working on his bachelor of science thesis [1], digital certificates have been further refined and improved and are exploited nowadays in many security systems and protocols. The X.509 certificates (taking the name from the standard defining their format [2]) are widely used on a large scale, such as for establishing secure TLS channels [3] for safe browsing and payment transactions, for creating advanced formats of digital signatures and even signatures with legal value, for secure e-mail, and to support security at the IP (Internet Protocol) level or in critical infrastructures. The X.509 certificates play a prominent role also in the digital identity systems, [4], [5], [6], since secure data formats and protocols like the Security Assertion Markup Language V2.0 [7], or OpenID Connect [8], heavily rely on X.509 certificates.

Digital certificates are issued and managed by specific entities (commercial or public) named Certification Authorities (CAs) in the frame of Public Key Infrastructures (PKIs).

The associate editor coordinating the review of this manuscript and approving it for publication was Luca Bedogni.

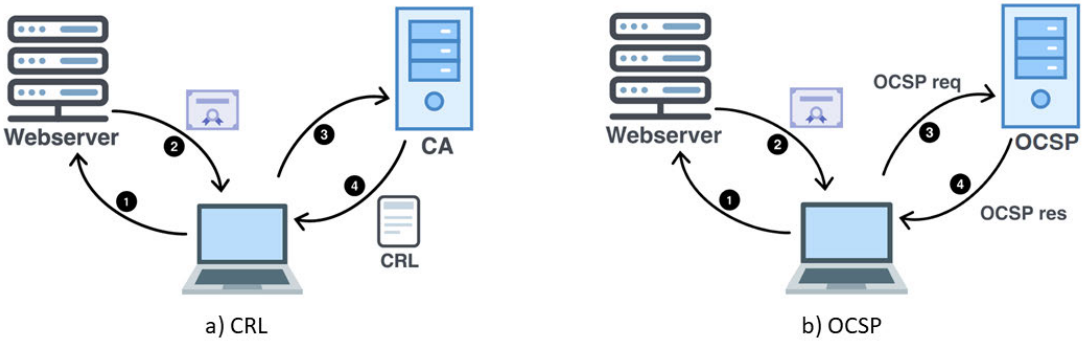

FIGURE 1. CRL and OCSP certificate revocation mechanisms.

The overall goal of a PKI is to issue certificates and uphold end users (applications) to check the validity of a certificate used in a transaction, in a secure channel, or for securing a data format. Thus, the PKI incorporates various functions such as user registration, key generation, management of root CA certificates, key recovery, key update, and cross-certification, to support different trust models, certificate revocation, and distribution of certificate revocation information.

End-user applications (including web browsers) interact with the PKI to perform management operations, such as obtaining a certificate from a CA. Moreover, PKI-enabled applications are also supporting other certificate-related functions, such as importing the certificates (either CA or user certificates, along with the key pair) in a dedicated *certificate store* [9], which can reside either in the application itself (in a specific database), or in the operating system (OS) [10]. For example, traditionally, Microsoft allowed users to import certificates in the OS so that all the other applications could retrieve and use them when needed. However, recently, Microsoft announced [11] changes to be adopted in the Microsoft Edge browser in certificate processing and for the Certificate Trust List (CTL), which is the certificate storage defined by the Microsoft Trusted Root Certificate Program [12]. More precisely, future versions of this browser would have a *certificate verifier module* and the CTL inside the browser,this decoupling the verifier and the list from the host OS platform.

The above example emphasizes a relevant aspect of certificate usage: certificate verification, which must be performed by the relying parties (i.e. the applications that receive the X.509 certificates). Most of the time, the certificate verification process is transparent to the users, because the applications perform it on their behalf. Upon receiving a certificate, *any* PKI-enabled application (including web browsers) must perform several operations to ''process'' (or validate) it. These operations include knowledge of the PKI trust topology, certificate formats, and profiles, execution of algorithms for certificate path construction and validation, and checking of certificate revocation via specific protocols. In particular, the X.509 standard [2] and the IETF PKIX Working Group [13] defined both the X.509 certificate format and the processing steps for certificate validation. However, some certificate fields or extensions are optional, so they might be absent. Even though the CA/Browser (CAB) Forum [14] has indicated stricter requirements for publicly available certificates, some violations [15] are still encountered, and we mention some of them in our work.

Certificate revocation checking. Although each certificate has a limited time validity, however, the issuing CA can revoke a certificate (before its expiration) due to different reasons, e.g., leak of the private key corresponding to the public key in the certificate, or change of the certificate owner's role inside an organization. Consequently, to distribute the certificate revocation data, the CAs must continuously keep track of the revoked certificates and make certificate revocation information public and (as much as possible) available in time. To this aim, one of the first methods used by the CAs are the so-called Certificate Revocation Lists (CRLs), which are published periodically (e.g., hourly, daily, or monthly) according to the CA policies.

Since each certificate is uniquely identified by a serial number, it is sufficient for a CA to indicate in the CRL the serial number and the revocation date for each revoked certificate, along with other fields and extensions including the lifetime and the issuer of a CRL. It is the responsibility of the relying party to check the revocation status upon using a certificate, that is to retrieve the CRLs from the various CAs (Fig. 1 a)). The CRL is digitally signed by the issuing CA, to its integrity and authentication of the CRL. This signature must be verified before using the CRL.

Since the CRLs tend to grow in size, they are not efficient in terms of cost and latency, especially if they are used on light or resource-constraint devices. The OCSP (Online Certificate Status Protocol) [16] was proposed as an alternative scalable method to determine the revocation status of certificates. The OCSP client (integrated into the PKI application) may contact an online OCSP responder (Fig. 1b), which provides the current certificate status, expressed as good, revoked, or unknown. The OCSP server can be managed either by the CA itself or by a delegated third party, which can provide

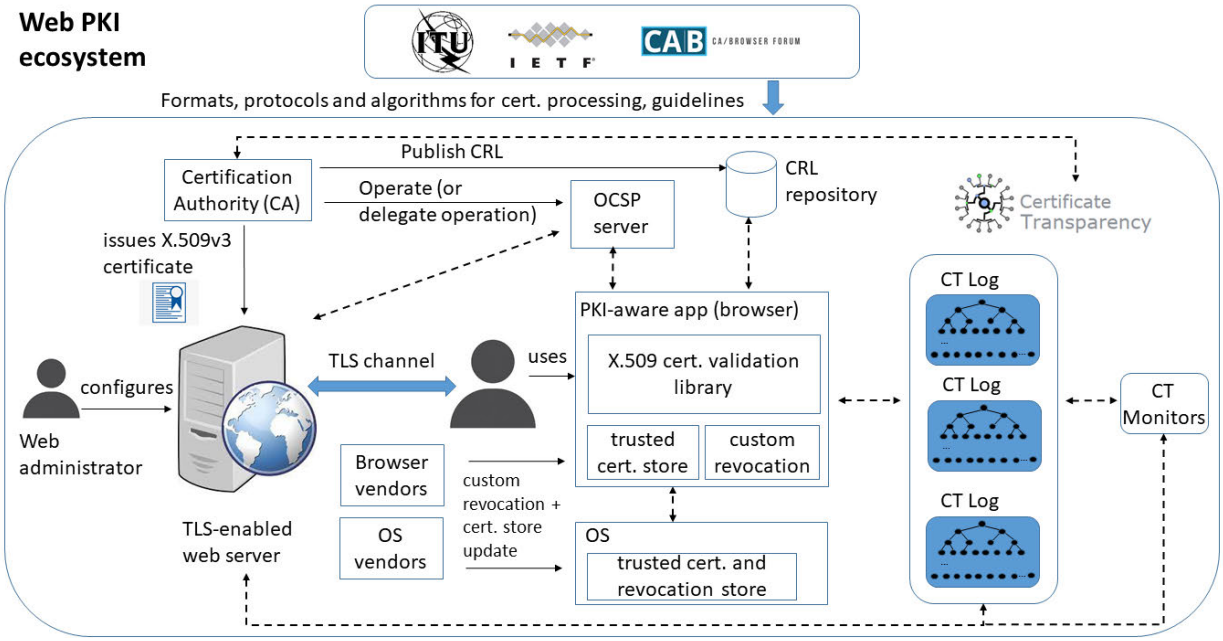

FIGURE 2. Web PKI ecosystem components.

revocation information for certificates issued by different CAs [17].

Certificate revocation status verification has proved to be more complex than expected because various factors influence its correct operation, as discussed in Section II-B. The revocation mechanisms differ in terms of scalability (i.e., they have different costs based on the size of downloaded data and the amount of processing required), availability (online service vs. CRLs downloaded and stored locally on the client), vulnerability to some security attacks (e.g., denial of service, or replay attacks), and even in terms of user privacy (possibility to track user's web activity). In addition to the standard CRL and OCSP methods, the browser vendors have proposed other custom methods, like Google Chrome's CRLSets [18], or Mozilla's OneCRL [19]. Nevertheless, the browsers may skip the revocation checking if the certificate revocation data is not available or reachable. In practice, this is a commonly encountered situation. Moreover, the browsers may decide to exploit an alternative revocation method (e.g., OCSP) in case one (e.g., CRL) fails.

Many researchers have indicated the importance of the certificate revocation process [20], [21]. For example, [19] states:

"*Whatever system handles revocation has to be fast, reliable, accurate, and privacy-preserving; and it has to be able to respond to hundreds of thousands of requests in a cost-effective way*".

At the same time, several studies (including [19], [22]) underlined that the certificate revocation we are using nowadays is completely broken. More on this, authors in [23] indicate:

"*Certificate revocation is therefore a critical component of any Public Key Infrastructure (PKI), and yet recent studies have found revocation to be woefully inadequate in the web's PKI. Website administrators revoke certificates at paltry rates; no browsers fully check for certificate revocation when connecting to TLS servers; and certificate authorities host Certificate Revocation Lists (CRLs) that are untenable large*".

In the Web PKI ecosystem, several entities (shown in Fig. 2) take decisions or perform actions that have an impact on the certificate revocation (and implicitly on the privacy issues), as well as on the entire certificate validation process: a) the standardization bodies and working groups defining the X.509 certificate format, like the International Telecommunication Union (ITU), or the IETF PKIX Working Group (WG); b) influential consortium groups (like the CA/Browser Forum) defining the requirements for publicly-trusted certificates [14] and the guidelines for highly EV trusted certificates; c) the CAs that should be compliant to the above-mentioned standard(s) and recommended guidelines in the process of certificate issuance and revocation; d) the web administrators who are in charge of properly configuring the relevant certificate and revocation data on the web servers; e) the developers of the PKI applications, including famous browser vendors like Mozilla, Chrome, and OS vendors, like Microsoft, or Apple and f) the implementers of the certificate validation libraries integrated into the browsers. Additionally, the Certificate Transparency (CT) system [25], proposed and deployed by Google several years ago to quickly spot compromised domain certificates, is another significant element operating in the Web PKI ecosystem since billions of devices

(both desktop and mobile) run browsers interacting with the CT system.

As remarkably noted in [15], the Web PKI "*bears the marks of compromise: too few constraints on what certificates can express and too many parties yielding too much authority*". Significant work has been done to protect users from possible attacks due to incorrect or incomplete certificate verification, especially for high-assurance certificates. At the same time, driven by regulations, such as the General Data Protection Regulation (GDPR) [26], increasing attention has been paid to user privacy. In this sense, the TLS protocol has been extended with extensions for OCSP to support user privacy [27], the result being the OCSP stapling mechanism. We analyze how the Web PKI actors address both the certificate revocation processing and the user privacy issues.

We make the following contributions:

- we extensively discuss first the Web PKI ecosystem components. We overview the X.509 certificate format, we analyze the role of the standardization bodies and working groups (such as the CA/Browser Forum) in defining specific X.509 certificate fields, the certificate extensions for high-assurance certificates, and the rules for certificate processing in the browsers. We detail the revocation methods (standard and custom ones supported by some browsers) and the CT system.
- we analyze the X.509 certificate profiles for the websites listed in the Alexa Top1M domain list in August 2021. For some certificate extensions, we indicate violations that have (still) occurred with respect to the CAB requirements of 16th August 2021 [28].
- we document the support for revocation checking with CRL and OCSP mechanisms in the most widely used web browsers, like Google Chrome, Mozilla Firefox, Internet Explorer, Microsoft Edge, Safari, and Opera, both for non-EV and EV certificates, with the help of experimental tests.
- we analyze the support for OCSP stapling in the certificates of the websites in the Majestic Top1M domain list (in 2022), as well as the support for the CT system. We compare our results with other related previous works to discuss the trend from the user privacy perspective.

Throughout our work, we aim to adhere to ethical standards and aim for our work to be fully reproducible. We share the dataset and scripts used for performing the tests [29].

Organization. The paper is organized as follows: Section II presents the main concepts and data relevant to our study. We detail the classical fields and X.509 certificate extensions, as well as the additional TLS extensions and custom revocation mechanisms proposed for efficiency and security concerns (like Google's CRLSets) and for privacy reasons (like OCSP stapling). Section IV details selected related works on estimating certificate revocation in the Web PKI. Section V gives a panorama of the supported extensions in the certificates of the Alexa Top1M list domain, and Section VI provides details on the testbed and the results obtained while testing the certificate revocation support in the considered web browsers. In the same section, we present additional results documenting the support for the OCSP stapling mechanism in the Majestic Top1M list certificates and support for fake certificate detection backed by the CT logs. Finally, Section VIII concludes the paper and indicates possible future work.

II. ANALYSING WEB PKI ECOSYSTEM COMPONENTS

A. X.509 CERTIFICATES

An X.509 certificate is a data structure digitally signed by an *issuer*, which binds a *subject* to an asymmetric *public key*. The subject could be a person, a network node, or even a generic entity identifiable through an identifier like an email address, an IP address, a DNS name, or even a general identifier. An end-entity certificate is part of a so-called certificate chain, where the first certificate is a root (self-signed) CA certificate, the last certificate is the end entity certificate called also *leaf* certificate (sometimes called *endpoint*), while in between there are typically one or more intermediate CA certificates.

In the web PKI, the issuers of leaf certificates are famous CAs such as Let's Encrypt, Verisign, or GoDaddy, who have their own CA certificates. Clients (like web browsers or PKI-enabled applications) must obtain the root CA certificates securely. Typically, the browsers and the operating systems ship with a root CA certificates bundle, which is updated periodically in the client's certificate store. The major root certificate stores are Apple, Microsoft, Mozilla, and Android [30]. Certificates in a certificate store are normally kept in some kind of permanent storage such as a disk file or the system registry,[1] in a protected OS account (sometimes called the trust store) which can be written only with admin privileges [31] or in the application itself. For example, Microsoft announced recently the integration of the trusted certificates (CTL) directly into the Edge browser [11], rendering thus the certificate trust store independent of the platform over which the application runs. We note that the certificate store must be (adequately) protected on the client side: if an attacker manages to insert his own (root CA) certificate into the client's certificate store, such as through malware or via a social engineering attack, then he can issue fake certificates for *any* web domain and the client is subject to man-in-the-middle attacks (as in the DigiNotar attack documented in Section III). No major attack has been documented so far in which attackers managed to compromise an entire main root certificate store. Some cases occurred against specific platforms and specific certificates, such as Dell's eDellRoot.[2] In practice, we can not exclude that with sufficient knowledge and resources, an adversary might be able to compromise such stores for specific certificates or on selected platforms.

[1] https://learn.microsoft.com/en-us/windows/win32/seccrypto/managing-certificates-with-certificate-stores

[2] https://arstechnica.com/information-technology/2015/11/dell-does-superfish-ships-pcs-with-self-signed-root-certificates/

1) DV AND EV X.509 CERTIFICATES

The basic X.509 certificate format for a web server is the Domain Validation (DV) certificate, which is issued when the CA has attempted to confirm that the applicant controls the domain for which the X.509 certificate will be issued [32]. Although there is no standardized procedure for conducting this confirmation, the following validation methods are typically employed: DNS validation, Email validation, Web-based validation, and TLS handshake with Server Name Indication (TLS-SNI) [33], [34]. For example, the famous Let's Encrypt CA (issuing certificates free of charge for nodes registered in DNS) checks the server identity via DNS validation, Web-based validation, and TLS-SNI [34].

It is known that these certificates should only be relied upon at most for domain-related information, and not for information about a specific organization. The Organization Validation (OV) certificates provide more assurance than the DV certificates because the CAs make additional validation checks before issuing them. Since such certificates provide more details about the company owning them, they are considered in terms of assurance in between the DV certificates and the Extended Validation (EV) certificates [35] that provide the highest level of assurance. The EV certificates, firstly adopted in 2010 and sometimes referred to as EV SSL certificates [36], provide more guarantees about the (public or private) organization running a web server. They have been created in response to the increasing number of online frauds, capable of eroding the security of consumers in web-based transactions.

Back in 2005, a group of CAs, browser vendors, and other interested parties formed the CA/Browser (CAB) Forum.[3] In 2007, the CAB consortium ratified the guidelines that formalize the management and validation procedures for the EV certificates. The CAB guidelines have been continuously updated, and the current version was released in November 2022 [37]. The EV certificates provide organizational information of known quality and display this information to the user [34]. Such certificates are issued nowadays by commercial CAs like GlobalSign,[4] Comodo,[5] GoDaddy,[6] or DigiCert.[7] The EV certificate issuance process has been designed to ensure that only private organizations, government entities, or business entities having a physical location (business presence) in the real world can obtain such a certificate, excluding those listed on any government-prohibited list or denial list. According to [32], the EV certificates have five required fields identified through specific Object Identifiers (OIDs): organization name, domain name, jurisdiction of incorporation, registration number, and address of the place of business. Thus, a user may view, for example, the address of a specific company using an EV certificate (as registered with the CA), whereas this information would be unknown under a DV certificate.

2) CERTIFICATE FIELDS

The X.509 certificate format is structured and contains a specific set of fields and extensions [2]. The *Version* field indicates the version of the standard to which the certificate refers. More specifically, the value of this field is an integer between 0 and 2, where value i indicates version i + 1, for 0<=i<=2. The version number is very useful because it provides information on how to parse the following fields. The next field, namely the *Certificate Serial Number* is an integer allowing the unique identification of the certificate within the CA that issued it. As the standard requires, a CA can never issue two certificates with the same serial number. The *Signature Algorithm Identifier* field indicates the algorithm employed to create the digital signature applied on the certificate, which guarantees integrity and authentication of the data in the certificate itself. Based on the algorithm indicated in this field, one or more initialization parameters could be present.

The *Issuer Name* field identifies the CA that issued a certificate. The X.509 standard requires the use of a unique identifier, which takes the form of a Distinguished Name (DN) whose main fields are:

a) O: identifies the organization;
b) OU: identifies a specific unit within the organization;
c) CN: is the Common Name, i.e., the name of the subject or entity;
d) L: indicates the locality, generally, it is the name of a city;
e) ST: indicates the state or province;
f) C: indicates the country to which the subject or entity belongs.

For publicly-trusted certificates, not all fields in the Issuer DN are mandatory [15]: the organization and country are required, whereas the Common Name is optional.

The *Validity* field establishes the validity period of a certificate. Each certificate contains two different dates: the first one (*Not Before*) indicates the date from which the certificate is valid, the second one (*Not After*) indicates the date after which the certificate is no longer valid. Although theoretically any time validity period could be specified, in practice the certificates are nowadays valid (on average) for one year. As recommended by CA/Browser Forum [14], the maximum validity period for website certificates dropped to 39 months on April 2015, whereas for public certificates issued after 1 September 2020, it cannot exceed 398 days. For EV certificates as well [37], the maximum validity time is 398 days.

The *Subject* field contains the details of the subject or entity asking for a certificate from a CA. The X.509 standard requires the use of unique identifiers for each applicant. The identity of the applicant establishes who is the owner of

[3] https://cabforum.org/
[4] https://www.globalsign.com/
[5] https://www.comodo.com/e-commerce/ssl-certificates/ev-ssl-certificates.php
[6] https://www.godaddy.com/
[7] https://www.digicert.com/

the private key corresponding to the public key contained within the certificate. The format of the Subject, as for the Issuer Name field, takes the form of a DN, such as: ''C = Italy, O = Polito, OU = DAUIN, CN = Paolo Rossi''. However, the CN value may vary depending on the entity requesting the certificate or the purpose for which the certificate has been issued. Nevertheless, the X.509 standard places no constraints on the value entered in the CN. Therefore, for a natural person, the CN can contain a person's name and surname, a person's national identification number, a pseudonym, or initials. In the case of a web server, the attribute contained in the DN may be a name that parallels the DNS (Domain Name System) name in the *Subject alternative name* (SAN) extension presented further below, but the implementations are not required to convert such names into DNS names [13]. The Subject field may even be empty, but in this case, the SAN extension must be present and must be marked critical. In fact, in the publicly-trusted certificates following the CA/Browser forum guidelines, the Common Name is deprecated (if present it must contain a single IP or FQDN), Organization is optional, Location (covering the Street Address, Locality, State, and Postal Code) must appear if Organization name is present (otherwise it mustn't), Country is required if Organization is present [15]. A new field named ''Registration'' (holding Business Category, Incorporation Locality/State/Country) must appear for EV certificates, while it may not appear for the other types of certificates.

The *Subject Public Key* field contains data related to the public key of the subject who requested the certificate. This field allows determining the asymmetric algorithm to which the key refers and any initialization parameters in addition to the binary sequence containing the public key. The last field in the certificate is the *Signature*, which contains a digital signature obtained from the predetermined hashing function (e.g., SHA256) applied to the whole certificate body. This signature is made by the CA by using its private key. In this way, the public key is transmitted in clear text in each issued certificate and can be extracted from the message by any user. This signature guarantees the integrity and authenticity of the issued certificate, and it can be verified by an application by using the issuing CA public key. Other fields have been added subsequently to distinguish between domain validation and extended validation certificates, as discussed above.

3) X.509 CERTIFICATE EXTENSIONS

In 1993-1994, when X509-based implementations started to be deployed on a wide range, it became evident that the first certificate versions (namely, v1 and v2) were not suitable anymore. Thus, some modifications to the certificate format have been performed. Among the most important issues (regarding the certificate format) that have emerged, we recall:

a) each user can have several certificates for different public keys; it's thus indispensable to have a mechanism to distinguish (unambiguously) the various certificates belonging to the same owner;

b) in some application contexts, the certificates must carry additional information regarding the owner beside his/her name (for example, the e-mail address);

c) the certificates can be issued according to different certification policies and for different purposes; for example, the CA could issue a certificate to a user used only for encrypting operations and not for digital signing purposes in high-value transactions;

d) the CAs establish mutual trust relationships; it must be possible to regulate the transfer of trust from one security domain to another.

e) compromise of CA certificates must be identified as soon as possible by the web domain holders and by the applications.

To respond to such requirements, the ITU standardization body and the IETF PKIX WG (specifying the certificate profile for Internet applications) defined the so-called X.509v3 certificate *extensions*, that allow associating additional attributes to the public key in the certificate. Unfortunately, even though the IETF PKI WG has tried to remove some choices and to give clear indications for certificate processing, some of the optional extensions still leave space for interpretation, as underlined also in [56]. Consequently, for certificate revocation checking, for example, it is possible to encounter different behaviours of the certificate-enabled applications, due to different interpretations of the optional or non-critical extensions.

The X.509 certificate extensions are very broad in their applicability. To support the development of interoperable implementations in X.509v3 systems for Internet use, it was necessary to specify a profile for the use of the X.509v3 extensions tailored for the Internet. The goal of the RFC 2459, its successor RFC3280, and the latest stable RFC5280 [13] was to facilitate the use of X.509v3 certificates within Internet applications, including the web browsers, the secure electronic mail, user authentication via digital certificates, and IPsec.

Each extension can be either ''critical'' or ''non-critical''. Any relying party (system, application) using X.509v3 certificates can ignore a non-critical extension if it is not recognized and reject the certificate if a critical extension is not recognized. Therefore, critical extensions must be carefully asserted in certificates, as problems may arise in their use. The main X.509 certificate extensions defined in RFC5280 [13] are as follows.

The *Authority Key Identifier* is used when a CA has more than one public key for signing the issued certificates, either due to multiple concurrent key pairs or due to key changeover. This extension serves to identify the public key corresponding to the private key used for signing the certificate. Identification can be based either on the key identifier or on the CA name and the serial number.

The *Basic Constraints* extension specifies whether the subject of the certificate is a CA and the length of the certificate chain. If this extension is present, the pathLenConstraint (which indicates the maximum number of CA certificates

forming the ''certificate chain''), must be greater than or equal to zero.

The *Certificate Policy* extension contains the sequence of certificate policies according to which the certificate was issued and for what purposes it can be used.

The *Key Usage* extension is used to define the purpose of the key contained in the certificate, more precisely, to limit the use of a key. In particular, a public key can be exploited for digital signature, non-repudiation, key encryption, data encryption, and key agreement. Moreover, the CA certificates have keys for certificate signing and CRL signing. It is not forbidden to use the key for other purposes than the ones specified in the certificate. However, it is the responsibility of the relying party (RP), namely of the application using the certificate, to ensure that the ''key usage'' is respected, and to block the operations for which the key is not allowed.

The *Extended Key Usage* extension indicates one or more purposes for which the public key can be used, in addition to or instead of the basic ones in the Key Usage extension, such as code-signing, S/MIME, or document signing.

The *Issuer Alternative Name* extension is used to associate the respective identities to whoever issues the certificate.

The *Name Constraints* extension, used only in a CA certificate, indicates limitations to which all the names of the subjects in subsequent certificates within the chain are subject.

The *Policy Constraints* extension can be used in CA certificates to restrict the validation of the certificate chain.

The *Private Key Usage* time period allows the issuer of the certificate to specify a validity period for the private key, different from the validity of the certificate.

The *Subject Alternative Name (SAN)* extension is a very important one. It allows CAs to indicate additional identities for the subject of the certificate. In this extension, other identification information not entered within the DN of the Subject field can be used, such as email address, DNS name, IP Address, Directory Name, X.400 names, and Uniform Resource Identifier (URI), but there are other options as well, including local names. It is typically used when a certificate is for a host with multiple names, sometimes called also multi-domain certificate.[8] Further, as indicated in [13] (Section 4.2.1.6.), multiple instances of each name form may be included in this extension. For example, a wildcard certificate for a domain example.com (whose Common Name begins with an asterisk, i.e., *.example.com) can be used for two virtual hosts (namely, one.example.com and two.example.com) on the same server.[9] This extension is always critical if the Subject field is empty, and the CAs must check all its parts, as the SAN is tied to the public key.

The *Subject Key Identifier* extension is used to identify certificates containing a particular public key, more specifically it contains the hash of the public key. Also, to facilitate the construction of the certificate chain, this extension must be present in all CA certificates. The value placed in the subject key identifier of a CA certificate must be the value placed in the authority key identifier of the certificates issued by the subject of that CA certificate.

The *CRL Distribution Points (CRLDP)* extension specifies locations from where the CRL can be obtained. It can be an e-mail address, a URL, or an entry of one directory. The *Authority Information Access (AIA)* extension indicates the URL of the alternative OCSP responder(s) that can be used to check the revocation status of a certificate. Moreover, it contains a URI pointing to the issuer's CA certificate, which can be helpful in case the CA certificate is missing from a presented TLS chain or is not available on the RP system.

Other X.509 extensions have been defined to support the OCSP Must-Staple mechanism (discussed in Section II-C), and the logging of the certificates and precertificates in the CT log servers: the Signed Certificate Timestamp (SCT) list (typically viewed as OID 1.3.6.1.4.1.11129.2.4.2 in the browser window showing the certificate details), and the *poison* X.509 certificate extension (with OID 1.3.6.1.4.1.11129.2.4.3) used in the precertificates submitted by the CAs to the CT logs [38] (discussed briefly in Section III).

B. PROS AND CONS OF OCSP VERSUS CRL, CRLSets

Certificate status checking with CRLs achieves privacy since the browser does not reveal user-visited websites, as it simply retrieves CRLs from the respective CAs (based on the URL in the CRLDP extension). Nevertheless, several works [17], [39] observed that from the scalability point of view, the CRLs are impractical because they tend to grow in time. As an alternative, a CRL might be split into smaller parts, allowing the application to download first a full CRL (called base CRL) and later on to download only smaller CRLs containing changes to the base CRL. Moreover, a new extension called *Freshest CRL* has been defined to support the smaller CRLs (called delta CRLs). With this mechanism, the size of the CRLs could decrease but anyway, significant processing is required on the user side to combine the base CRLs and the delta CRLs.

We note that the OCSP mechanism is not suitable for determining the certificate status in the past. Moreover, other possible drawbacks or attacks may occur as well. For example, an attacker could render the OCSP responder unavailable, e.g., through a denial of service attack, to impede clients from checking the certificate status. In general, the (online) OCSP responders are more exposed to attacks compared to the CAs that can keep the CRL signing process on a system disconnected from the network and transfer the digitally signed CRLs to a public server.

For efficiency or security reasons (such as to protect the OCSP responder's private key) the OCSP responder can provide *cached OCSP responses* instead of the ones created in real-time. According to [14] the OCSP responses must have

[8] https://www.digicert.com/tls-ssl/multi-domain-ssl-certificates
[9] https://cwiki.apache.org/confluence/display/HTTPD/NameBasedSSLVHosts

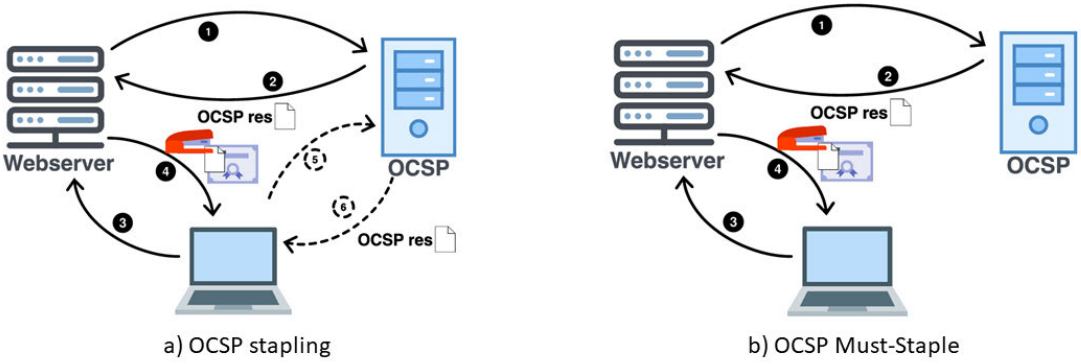

FIGURE 3. OCSP stapling and OCSP Must-Staple models.

a validity interval greater than or equal to eight hours, and less than or equal to ten days. In case cached OCSP responses are employed, the OCSP responder is not exposed anymore to online attacks but an attacker might provide old (replayed) OCSP responses instead of genuine ones, which means that replay attacks are possible.

The OCSP method introduces potential *privacy* concerns because a client visiting a website must ask the OCSP responder if the certificate for that web server is still valid. Consequently, the responder (and implicitly the entity running the responder, typically a CA) comes to know to which website a client has connected. A more privacy-preserving solution would be the adoption of the OCSP stapling mechanism presented in Section II-C.

As an alternative to the CRL and OCSP mechanisms, the browser vendors have designed and started to support in the browsers their custom revocation methods. For example, the Chromium project proposed alternative CRL-like lists named CRLSets[10] supported in the first place by Google Chrome. Mozilla adopted the OneCRL method (starting from Firefox version 37 released in 2015), which covered intermediate CA certificates.[11] A CRLSet is a small (size at most 250kB [39]), pre-populated list of revoked certificates updated and sent to client out-of-band to a) reduce the cost of checking revocation information at page load time; b) avoid cases in which the browser cannot reach the CA's servers (such as in the case of captive portals allowing users to sign in but blocking traffic to all other sites); and c) to avoid attacks in which an adversary blocks access to the revocation information in case of online revocation checking. Thus, in the CRLSet approach, since revocation updates are pushed to the browser periodically, the client does not need to perform an online revocation check[12] and consequently: a) from the privacy point of view, the considerations observed for the OCSP method do not apply, so we may consider it a more privacy-preserving method b) is not subject to attacks in which an adversary tries to block these checks during the attack. On the other hand, CRLSet does not cover all possible certificates, but only those selected and pre-populated into the list.

C. OCSP STAPLING

In the last years, the OCSP stapling mechanism has been proposed to allow web browsers connecting via TLS protocol to a web server to verify, in a more *privacy-preserving* way, the revocation status of the web server's certificate. OCSP stapling can also solve the latency of the clients because almost 30% of the TLS overhead comes from checking whether the certificate has been revoked [40]. Note that OCSP stapling (formally named TLS Certificate Status Request extension) is an extension of the TLS protocol [27], not of the OCSP protocol or of the X.509 certificate format.

In OCSP stapling, the web server *periodically* obtains OCSP responses for its certificate, caches them, and "staple" these cached responses in the TLS handshake in a dedicated TLS extension (Fig. 3a). The overhead on the server side is low, equivalent to the time spent on OCSP transactions. Thus, when a web client (browser) connects to the TLS-enabled web server, it sends a TLS extension of type "status_request" in the `client hello` message of the TLS handshake, along with an indication of the trusted OCSP responder. If the server supports OCSP stapling, it sends back to the browser in the TLS handshake the server's certificate, its chain (except the root CA certificate), and a cached OCSP response. In this way, the web client does not need to make an additional network request (for the corresponding OCSP response, or CRL) to find out the revocation status of the server's certificate, thereby mitigating the privacy and latency costs the clients need to face. We may note that OCSP stapling only allows the revocation status verification for the leaf certificate, even though a web client would want to check the revocation status of *all* certificates in the chain. An extension has been proposed for this scope in RFC6961 [41], but it has been subsequently deprecated in TLS 1.3 [3].

We note three issues. First, if the OCSP response is not returned to the client in the TLS handshake, the client can

[10] https://www.chromium.org/Home/chromium-security/crlsets/

[11] https://blog.mozilla.org/security/2015/03/03/revoking-intermediatecertificates-introducing-onecrl/

[12] https://www.imperialviolet.org/2012/02/05/crlsets.html

still fetch it with the classical OCSP approach (dotted lines in Fig. 3a). Thus, the same privacy concerns applying to OCSP could (potentially) occur also in this case. Second, the *freshness* of the revocation information depends on how often the webserver contacts the OCSP responder to obtain a response for his certificate. So, hypothetically, there is a window of exposure (webserver certificate appearing as "still valid" although it has been revoked by the CA), but paranoic clients may anyway choose to fall back on using classical OCSP. Third, since OCSP stapling is not mandatory, the client application typically chooses to continue a TLS connection in case an OCSP response is not received. This behaviour, named *soft-fail*, is often adopted by the browsers [42]. To indicate that a client application must receive a stapled OCSP response in the TLS handshake whenever it sees a certificate, a dedicated X.509 certificate extension, called Must-Staple, was introduced in 2015 [43]. This extension acts like a signal: the client application must *hard-fail* (i.e., reject a certificate) if the server does not provide a valid OCSP response in the TLS handshake. From the privacy point of view, the OCSP Must-Staple is probably the best option, because the client does not connect directly to the OCSP responder (Fig. 3b). Unfortunately, OCSP Must-Staple is not used in practice nowadays, according to the experimental tests described in Section VII.

III. CERTIFICATE TRANSPARENCY

In 2011, a famous critical event in web history occurred, which affected a Dutch CA, namely DigiNotar [44]. In particular, attackers were able to compromise this CA and issued fake rogue certificates for hundreds of websites, including Google and Skype.[13] Since DigiNotar was present as a trusted CA in the CTL or anyhow in the trusted root CA certificates database of many browsers, this event had a significant impact on the clients because they were unable to detect that they were exposed to man-in-the-middle attacks.

Following this event, Google proposed the Certificate Transparency (CT) system [25], an open, global monitoring system composed of append-only public logs that collect certificates or precertificates for the web servers. The core idea of the CT is to make it impossible or, at least very difficult for an attacker to issue certificates for a web domain without making them publicly visible to the domain owner. Google launched the first CT log in March 2013, and IETF has released an Experimental RFC [38] corresponding to the currently running CT system. Subsequently, in December 2021, an updated CT specification was released [45].

The CT system uses a data structure, namely the Signed Certificate Timestamp (SCT), and a framework for issuing SCTs and auditing and monitoring purposes. Anyone can submit certificates or pre-certificates to a log server, although the majority are sent by the CAs. Noticeably, the CAs may send either a pre-certificate to the CT logs (that is *before* they issue a certificate) or a certificate (that is, *after* the certificate

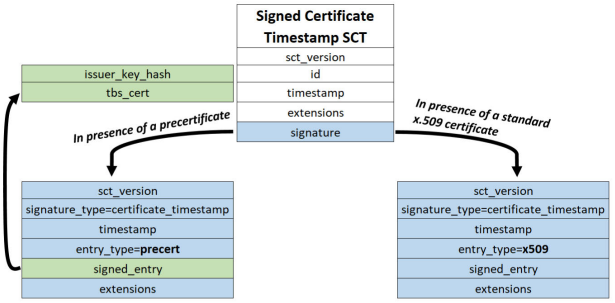

FIGURE 4. SCT structure.

for a web domain has been issued, with the condition that the certificate issuer is one of the root CAs published by the log). As mentioned, upon sending a valid certificate or pre-certificate to a CT log server, the latter answers with a kind of promise, the SCT, whose structure is shown in Fig. 4. If the CA has asked for an SCT via a pre-certificate, it will include it in a dedicated X.509v3 certificate extension in the issued certificate. Thus, the SCT will be part of the X.509v3 certificate for its entire lifetime.

We must remember that a pre-certificate is different from the X.509 certificate, and it contains a critical *poison* extension to ensure that it will not be treated as a standard certificate by the TLS clients. Every browser vendor has its policy about how many SCTs a certificate must have [39]. For example, Apple (Safari) asks for at least two distinct log operators (so the certificate must have at least two SCTs from different log servers), whereas Google (Chrome) asks one SCT from a Google-operated log and one from a non-Google one.

Anyone can query the logs (via HTTPS) to verify that it is well-behaved, or that a specific TLS certificate has been properly added to the log server (publicly auditable). The original CT architecture has foreseen three main components: the Log Servers, the Monitors, and Auditors [24]. The Auditors, likely built into the browsers or operating as an independent service, could communicate with the log server asking whether the SCTs (and corresponding certificates) have been legitimately added to a CT log. It was not entirely clear whether an Auditor could discover or keep track of visited websites for the end entity/user that received a web certificate with an SCT inside, which could cause privacy concerns similar to the classical OCSP mechanism. The current CT architecture mentions the following actors: Logs, Monitors, and User Agents [25]. However, some solutions for SCT auditing are being investigated [39]. The CT monitors, on the other hand, are useful to detect the misbehavior of the CAs. Some of the most prominent examples of CT monitors available currently are crt.sh[14] and sslmate's Cert Spotter.[15] They allow users to quickly view the details of the logged CT entries for millions of web domains. Other CT Monitors currently listed

[13] https://www.enisa.europa.eu/media/news-items/operation-black-tulip/

[14] crt.sh.

[15] https://sslmate.com/certspotter

on CT project page [25] are managed by companies like digicert, Cloudflare, Entrust, Hardenize, Keytos, Facebook, or ReportURI.

IV. RELATED WORK

Several works, like [9], [20], and [46], have performed large-scale measurements of worldwide HTTPS ecosystems by collecting for a long time (more than a year) and ultimately analyzing the certificates for a large number of servers or IP addresses. These works provide a broad overview and a comprehensive analysis of X.509 certificates in the wild. Previous research emphasized that although the X.509 certificate format and the revocation mechanisms are well known nowadays, the web TLS ecosystem is populated with many invalid certificates [48], and the certificate revocation checking is still problematic. Such a situation is not due to a single entity or source but is determined by different actors, including the CAs that are in charge of the issuance and revocation of certificates, or the web administrators that must handle the configuration of proper certificates of the websites. At the same time, both the clients (the browser applications) or the cryptographic libraries integrated into X.509 aware applications still lack proper or complete support for certificate processing [21], [42]. We briefly describe some of these works, closely related to our analysis.

Certificates' revocation status in the Web. Liu et al. [42] took a closer look at certificate revocation in the Web's PKI in 2015. They empirically evaluated the extent to which all the parties involved in this process met their (certificate) revocation responsibilities. They have considered the following parties: web administrators, CAs, and browsers. To perform the analysis, they collected 38,514,130 unique TLS certificates through full IPv4 port 443 (HTTPS) scans between October 30, 2013, and March 30, 2015. For what it concerns *website administrators' behaviour*, they observed that the administrators failed to disable or remove invalid certificates, and a considerable percentage of revoked certificates (> 8%) was configured/exploited on the web servers. Moreover, the website administrators infrequently enabled OCSP stapling. In particular, only 3% of certificates were served by hosts supporting OCSP stapling. Regarding the CAs, they measured that the distribution of certificate revocation information via CRLs imposed a significant overhead and high latency on the clients. More in detail, they showed that some CAs have not adopted smaller CRLs, enforcing clients to download very large CRLs (up to 76 MB in size) before fully establishing the TLS connection.

A large-scale study on HTTPS certificate deployment in China reported some common mistakes and misconfigurations [20].

Speaking about *browsers' practices* in handling certificate revocation checking, Liu et al. [42] found that the fraction of times that revocation information was checked was surprisingly low. Browser developers often applied a *soft-fail* approach, deciding to trust certificates even when they could not download revocation information. This reaction might be good for usability but creates problems for PKI security, so it is advisable to use a *hard-fail* approach instead of the soft-fail one. The same study observed that many browsers do not interpret (correctly) the unknown OCSP responses. Moreover, not all browsers support the OCSP stapling, and many do not respect the revoked staple. Mobile browsers, instead, did not perform any form of certificate revocation checking. Finally, the authors have also studied the CRLSet mechanism built into Google Chrome to spread revocations, finding that CRLSet only covered 0.35% of all certificate revocations (in 2015).

Broken certificate validation in applications and libraries. About ten years ago, Georgiev et al. demonstrated that TLS (formerly known as SSL) certificate validation was completely broken in many client-side security-critical applications and libraries [47]. For completeness, we recall that certificate validation is a process composed of several steps including time validity checking, certificate policy and naming checking, key usage checks, and certificate revocation checking, as described in the algorithm given in RFC 5280. Thus, certificate validation failures are due to several possible reasons, not only due to the revocation status of the certificates processed. Affected applications included programs for managing cloud-based storage and computation, Windows-based cloud storage clients, and Java-based Web-service middleware. Broken libraries included payment service modules from Amazon or PayPal. Even though steps forward have been performed since then to render certificate validation a safer process [49], some misunderstandings still exist between TLS libraries (like OpenSSL, or its variants) and applications that are ''especially worrisome'' [15]. For example, the applications expect the TLS libraries should perform more advanced checks, e.g., the compliance of certificate extensions or fields with CAB requirements [14]. At the same time, the TLS libraries may delegate too many certificate verifications back to the applications.

V. ANALYSIS OF X.509 CERTIFICATES IN ALEXA Top1M DOMAIN LIST

A. DATA COLLECTION

To analyze the support for X.509v3 fields and extensions in the most widely used websites, we downloaded the Alexa Top1M list on the 26th of August 2021. To collect the certificates, we have implemented two scripts in Python: the first one scrolled the Alexa Top1M list and, one at a time, it passed the link of every listed web server to the second script, which tried to establish a TLS connection with the web site passed in as an argument. Then, it saved locally the entire certificate chain received in the TLS connection. In case of failure, the script went ahead with the next entry in the list.

This procedure took about 20 days, during which we collected 442,331 certificates corresponding to 645,332 websites. The number of certificates is smaller than the entries listed in the Alexa Top1M list because several servers have used the same certificate with several values in the SAN

extension (this is a classical procedure to support virtual domains). Additionally, we have encountered many connection errors with several servers, determining fewer collected certificates. We have further analyzed the collected certificates to understand:

- a) which X.509v3 certificate extensions and revocation methods were supported
- b) which CAs (among the most popular ones) have issued the collected web server certificates
- c) how many X.509v3 certificates were inadequately used because they were either revoked or expired

All this information is reported further below.

B. ANALYZING X.509 CERTIFICATE FIELDS AND EXTENSIONS

In this section, we detail the support for the main X.509 certificate fields and extensions in the collected certificates, including the ones helpful for revocation checking. We implemented scripts in Python language to process the certificate extension values more efficiently.

Basic Constraint (BC) extension: We have divided the entire set of approximately 440,374 collected certificates into two (sub)sets, namely `Leaf` and `CA`. The first ones are the end-entity certificates belonging to the target web servers, while the second ones are CA certificates. After analyzing the BC extension, we have obtained the following results:

- 1098 CA certificates (0.25%) had the BC extension set to *true*, meaning thus that these certificates are part of the `CA` set.
- 436,193 (99.00%) certificates had the BC extension set to *false*, so we have considered them part of the `Leaf` set.
- 1590 certificates (0.36%) did not have the BC extension. In this case, the script has raised an 'Extension not found' exception, but we have considered these certificates in the `Leaf` set too.
- 1 certificate raised a Value Error exception because the `path_length` value of the BC extension was not set, while the CA value was false.

In the `CA` set, 628 certificates ($\approx$ 58%) had a critical BC extension, and 470 certificates ($\approx$ 42%) had a non-critical one, as shown in Fig. 5. This fact represents a violation of the CAB requirements [28] because the CA certificates must have the BC extension present and must be marked critical. In the Leaf set instead, 405,363 certificates ($\approx$ 93%) had the BC extension marked critical, while 31,928 certificates ($\approx$ 7%) had the extension marked non-critical, as shown in Fig. 6. According to [28], this extension is not mandatory in endpoint certificates, so the leaf certificates were in line with the CAB requirements.

Certificate Issuer: By processing the Issuer field, we observed that more than 55% of the `Leaf` set certificates were issued by the famous Let's Encrypt CA, as shown in Fig. 7, followed by Sectigo and other famous CAs.

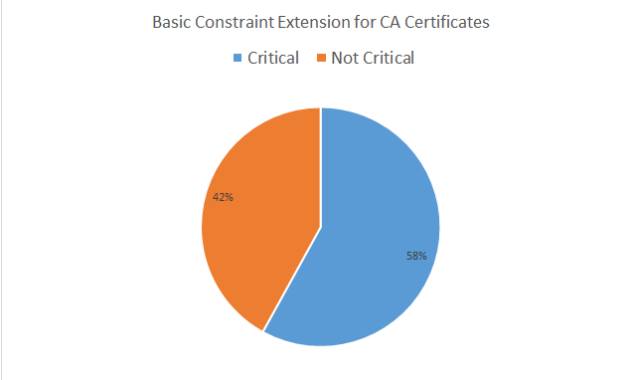

FIGURE 5. Basic constraint extension (critical vs. non-critical) in the collected CA certificates. According to [28], in the CA certificates, this extension must appear and must be critical.

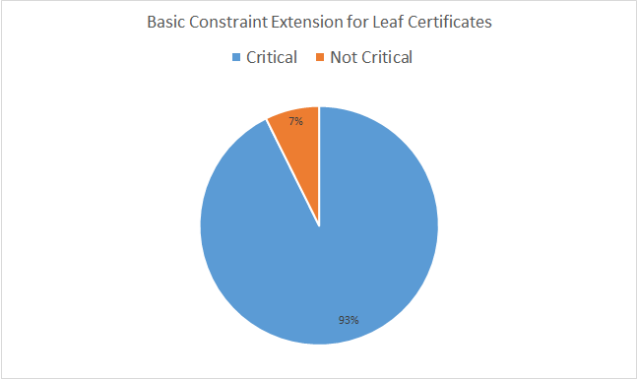

FIGURE 6. Basic constraint extension (critical vs. non-critical) in the Leaf set. According to [28], in the leaf certificates, this extension is optional.

Authority Key Identifier (AKI) extension: We then checked the support for the AKI extension in all the certificates. In the `Leaf` set, 99.77% of certificates contained this extension and only 0.23% did not hold it. In the `CA` set, 87.13% of the CA certificates had an AKI extension, and 12.87% certificates did not support it. All certificates had the extension marked non-critical, and only one certificate (in the `Leaf` set) raised a value error exception.

Subject Key Identifier (SKI) extension: The SKI extension was present in 99.74% leaf certificates, and 95.21% CA certificates. In all certificates, the extension was marked non-critical, and only one certificate in the Leaf set has raised a value error exception.

Key Usage (KU) extension: The KU extension was present in 99.19% leaf certificates and 52.88% of CA certificates. In 99.90% leaf certificates, the extension was critical, while in 8.56% CA certificates, it was non-critical. Thus, a small percentage of CAs have not followed the CAB requirements [28] stating that the extension must be critical both in intermediate and root CA certificates.

Extended Key Usage (EKU) extension: The EKU extension was present in 99.43% of leaf certificates and also in the 24.49% of CA certificates. According to [28], the EKU

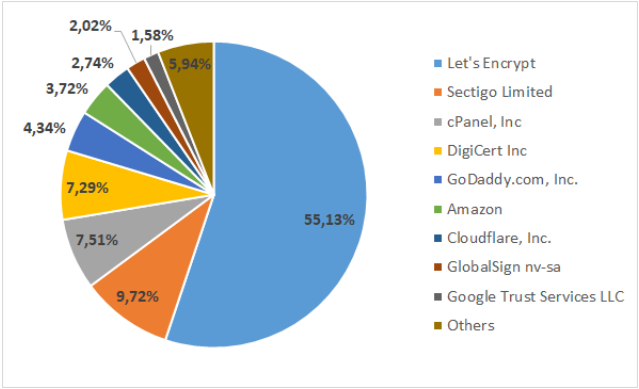

FIGURE 7. CA Issuers of the certificates in the CA set.

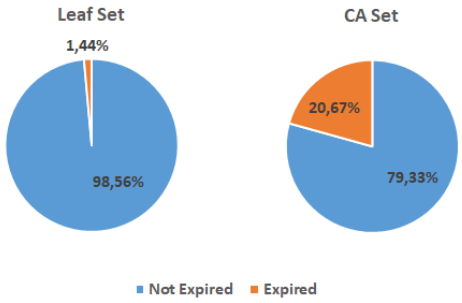

FIGURE 8. Percentage of expired/non-expired certificates in the Leaf and CA sets.

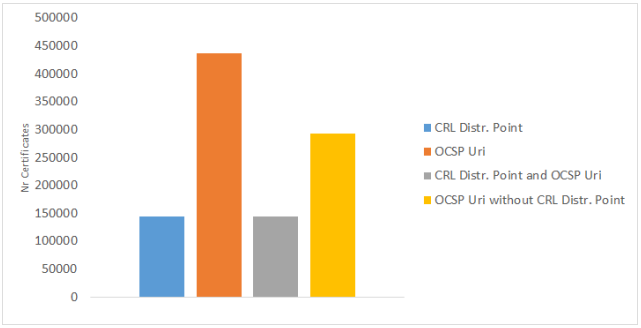

FIGURE 9. Presence of X.509 certificate extension for revocation checking (CRLDP and AIA) in the collected leaf certificates (Leaf set).

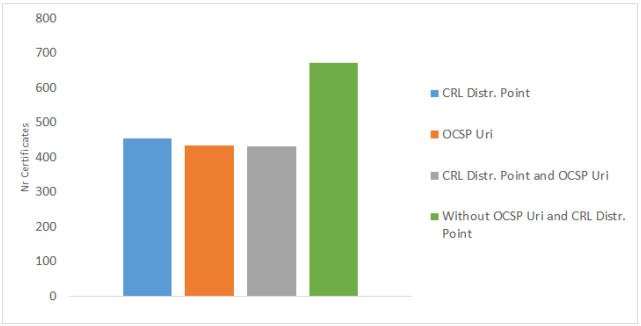

FIGURE 10. Presence of X.509 certificate extension for revocation checking (CRLDP and AIA) in the collected CA certificates (CA set).

must not appear in Root CA certificates and is optional in (intermediate CA) cross-certificates that share a Subject DN and subject public key with a root certificate. In all the other intermediate CA certificates the extension must be present and should not be marked critical.

Certificate Policies extension: Certificate policies extension was present in 99.08% leaf certificates and 38.52% CA certificates. In all certificates (i.e., in both Leaf and CA sets), the extension was marked non-critical. According to the CAB requirements [28], this extension must appear in endpoint certificates and intermediate CA certificates and should not be marked critical. In the root CA certificates, the extension should not be present. Thus, except for a small percentage, the leaf certificates have followed the requirements.

Subject Alternative Name (SAN) extension: it is present in 99.57% certificates in the `Leaf` set. In all leaf and CA certificates, this extension is non-critical, while only one certificate in the `Leaf` set had the extension marked critical.

Expiration date: We checked the certificate's expiration date for both sets against the 26th of August 2021 (when we downloaded the certificates). in the Leaf set, 1.44% of the certificates were expired, while, surprisingly, about 21% of the CA certificates were outdated too (Fig. 8).

Table 1 summarizes, in brief, the above findings. We note that, in general, the majority of certificates have been issued in accordance with the CAB requirements, although some (minor) deviations (except for the validity of CA certificates) have been encountered.

C. ANALYZING X.509 CERTIFICATE EXTENSIONS FOR REVOCATION CHECKING

We have further analyzed the X.509 extensions for revocation checking in the collected certificates.

CRL Distribution Points (CRLDP) extension: it is used to check the certificate status through the CRL method. In the sets analyzed, this extension appeared in about 40% certificates of the `CA` set and in ≈32% certificates of the `Leaf` set. According to the CAB requirements (August 2021), this extension was optional in endpoint certificates, so about 58% leaf certificates did not contain the extension but they were in line with the CAB requirements [28].

Next, to establish how many `Leaf` set certificates supporting CRL have been revoked in practice, we wrote a script (in Python), which downloaded the CRL (from the URL in the CRLDP extension), then checked the certificate status against the downloaded CRL. We measured that 278 out of the 143,550 certificates having a CRLDP extension were **revoked**, while 142,335 certificates had the status **good**. We have also observed that downloading the corresponding CRL failed for 937 certificates.

Authority Information Access (AIA) extension: This extension appeared in 99.08% of the `Leaf` set, while only 38.42% CA certificates contained it. All certificates had this extension marked non-critical, and only one certificate in the `Leaf` set

TABLE 1. Extensions of Root CA, Intermediate CA, and leaf certificates according to CAB requirements (August 2021). Measured compliance of the analyzed certificates in Alexa Top 1M list to the CAB requirements. Deviations from the CAB requirements are marked in red.

Certificate type	Certificate extension/field	Requirements	checked certificates compliant to CAB Requirements (in %)
Leaf	Certificate Policies	must appear, should not be marked critical	99.08% extension present
	Basic Constraints	optional	93% (extension present, critical) 7% (extension present, not critical)
	Key Usage	optional if present, certificate and CRL signing must not be set	99% extension present
	CRL Distribution Points	optional must not be critical if present must contain the HTTP URL of CA's CRL service	32% extension present
	AIA	must appear, must not be critical	99.08% extension present 0.92% extension not present
	Subject Alternative Name	must appear must only include DNS names and IP addresses should not contain local names or reserved IP addresses wildcard FQDN are permitted	99.57% extension present 0.42% extension not present
	Time Validity	valid (for 398 days)	98.56% not expired 1.44% expired
Intermediate CA	Certificate Policies	must appear, should not be marked critical	38.52% (of CA set) present
	Basic Constraints	must appear, must be critical	58% (of CA set) present, critical; 42% extension present, not critical
	Key Usage	must appear, must be critical	52.88% (of CA set) present
			8.56% (of CA set) non-critical
	Ext Key Usage	optional/required	24.49% (of CA set) not present
	CRL Distribution Points	must appear, must not be critical	40% (of CA set) present
	AIA	should appear, must not be critical	99.08% (of CA set) present
	Time Validity	valid	≈21% (of CA set) expired
Root CA	Certificate Policies	should not be present	see value above (for CA set)
	Basic Constraints	must appear, must be critical	see value above (for CA set)
	Key Usage	must appear, must be critical	see value above (for CA set)
	Ext Key Usage	optional/required	see value above (for CA set)

raised a value error exception. To establish how many leaf certificates with an AIA extension were revoked, we wrote a script to check the status by querying the OCSP responder in the AIA extension. After running the script, for the `Leaf` set, we obtained the status `revoked` for 600 certificates, `good` for 409.435 certificates, and `unknown` for 11 certificates. For the other certificates, the OCSP responder could not be contacted due to connection errors.

As shown in Fig. 9, OCSP was the preferred method for checking revocation status in the `Leaf` set certificates because ≈ 99.08% contained an OCSP responder URL in the AIA extension. Only six certificates with no AIA extension had a CRLDP extension, and 0.92% of them (4054 certificates) did not support any revocation status mechanism because they didn't contain either a CRLDP or an OCSP responder URI in the AIA extension. We have also measured that 32.54% of the `Leaf` set certificates listed a CRLDP extension, and almost the same number of certificates supported both the CRL and OCSP methods. On the other hand, 66.54% of certificates listed an OCSP responder URL (in the AIA extension) but did not provide a CRLDP extension.

As depicted in Fig. 10, 59.54% (671 certificates) of the CA certificates did not provide a way for checking revocation status with CRL or OCSP, because they didn't hold a CRLDP extension or an OCSP responder URI in the AIA extension. In the same set, 38.42% of CA certificates (433) contained an AIA extension with an OCSP responder. One certificate indicating an OCSP responder in the AIA extension did not have (instead) a CRLDP extension. At the same time, 40.37% of the CA certificates had the CRLDP extension, and 38.3% out of them had (additionally) an OCSP responder URI in the AIA extension; only 2.04% (23 certificates) had the CRLDP set but did not support OCSP since the AIA extension was absent.

VI. TESTING X.509 REVOCATION IN DESKTOP WEB BROWSERS: SOFT-FAIL OR HARD-FAIL?

We analyzed the revocation-checking behaviour in recent versions of desktop web browsers under different conditions. We measured whether the browsers support "fallback" revocation methods, e.g. if OCSP responder is not available, we looked at whether the browser automatically attempts to download the CRL for certificate status checking. Moreover, we have assessed browsers' behavior when the CRL could not be downloaded, or the OCSP responder was not available, meaning that the certificate revocation (status) could not be determined. We encountered the following behaviors:

1) the browser accepts the certificate anyway and goes ahead with the TLS connection (*soft-fail*);

2) the browser rejects the certificate and (implicitly) the TLS connection (*hard-fail*).

We have formulated and tried to respond to the following questions:

- Q1. How do desktop browsers behave when the OCSP server or the CRL repository is unavailable or

TABLE 2. Web browser and OS versions used in conducting the tests (Win = Windows 10, Linux = Kali Linux 2020.4, macOS = MacOS 11).

Browser	Version	Operating System (Platform)
Google Chrome	95.0.4638.54	Win/macOS/Linux
Mozilla Firefox	93.0	Win/macOS/Linux
Opera	80.0.4170.63	Win/macOS/Linux
Internet Explorer	20 H2	Win
Microsoft Edge	95.0.1020.30	Win
Safari	13.1.2	macOS

unreachable? Do they fall back to an alternative revocation method when the first method fails?

- Q2. Do browsers refuse the TLS connection to the testbed (web) server if they cannot retrieve revocation information for each certificate in the chain (starting from the web server certificate)? Do they apply a hard-fail or a soft-fail approach?

When revocation data is unavailable or the OCSP responder is unreachable, the web browsers may have different behavior, depending on their implementation (browser vendor), the underlying certificate validation library exploited, and the OS on which they run [21]. This situation occurs because the PKI standards state how to identify an invalid (revoked) certificate but do not state what to do when part of the information (like revocation information) is not known. So, in this case, the decision is left to the application developer. Since some X.509 extensions are optional or marked non-critical, like CRLDP or AIA, each application decides how to treat that information. We selected the browsers based on the statistics from Netmarket-share [50], updated on 22 October 2021. Data show that Google Chrome is the most used browser among users (69.28%), followed by Edge (7.75%), Firefox (7.48%), and Internet Explorer (5.21%). In our tests, we used popular browsers, such as Chrome, Mozilla, Edge, Internet Explorer, Safari, and also Opera. Table 2 summarizes the browsers we chose for conducting the tests together with the OS under which they run.

A. DATA FOR CRL AND OCSP TESTING

To run the tests, we have used the OpenSSL library [51], more precisely, the OpenSSL 1.1.1l stable version. We have generated a CA, custom web server certificates and chains, and the relative CRLs. More in detail, we have created certificate chains composed of 3 certificates: a self-signed root CA certificate, an intermediate CA certificate issued by the root CA, and a leaf certificate issued by the intermediate CA. Next, we created CRLs for the CA and leaf certificates. We have installed the testbed CA and the OCSP responder on a PC running Kali Linux (version 2020.4).[16] Then, we activated the OCSP responder implemented in OpenSSL. We imported the root CA and intermediate CA certificates into the corresponding browsers' stores, namely the root CA and intermediate CA stores. Additionally, we have configured a service allowing to download the CRLs from the testbed hosts.

OpenSSL uses a configuration file named `openssl.cnf` to handle the contents of generated X509 certificates. We created different copies of `openssl.cnf` file for EV and non-EV root and intermediate CA certificates. OpenSSL configuration file contains a sort of ''profiles'' for managing the content of leaf and CA certificates. By changing and adding some fields in the configuration file, we generated different types of certificates, such as for the CRLDP and AIA extensions. The EV certificates have a specific profile. In particular, the CA issuing the EV certificates must add a specific OID value to the certificate format. Most popular CAs disseminate OID values of the EV certificates so that web browser developers can identify them. In tests, we used the DigiCert OID value in a new OpenSSL configuration file named `validationEV.cnf` to simulate a CA authorized to issue EV certificates.

B. TESTBED SETUP

With all testing certificates and the CRLs available, and with the OCSP responder up and running, we set up an experimental testbed composed of three different nodes running Windows, Linux, and MacOS. On these PCs, we installed web browsers connecting via TLS to a web server having a certificate (and the corresponding chain) described in Section VI-A. Consequently, by configuring the TLS server certificate (and chain) accordingly, we assessed browsers' behaviour under different contexts.

For the TLS-enabled web server, we used a desktop PC running Windows 10 and Microsoft Internet Information Services enabled for HTTPS on port 49152. On this server, to perform the tests in Table 3, we installed different certificate chains containing revoked or good certificates for the leaf, intermediate CA, and root CA. We installed Wireshark[17] on the PC hosting the CA, the CRLs, and the OCSP responder to monitor the network traffic with the browsers. In particular, we looked at whether the browsers have generated the OCSP requests, or if they have tried to download the CRLs.

To perform the tests for Linux, we used Kali 2020.4 running on a virtual machine and Mozilla Firefox, Google Chrome, and Opera web browsers. Finally, to launch the tests from MacOS, we used another computer (connected to the same LAN) on which we installed the selected browsers.

C. CRL AND OCSP SUPPORT IN BROWSERS

To study desktop browsers' practice more in detail, we have performed the tests with the browser versions and platforms shown in Table 2. We have changed the configuration of the webserver (with revoked certificates at different levels), as well as the availability of the CRL and OCSP responder, as listed in Table 3. We comment further below on the obtained results that are resumed in Table 4.

[16] https://www.kali.org/

[17] http://www.wireshark.org

TABLE 3. Test cases performed on the selected web browsers and OS platforms, by varying the configuration of the server, intermediate CA, and root CA certificates in the testbed, as well as the availability of certificate revocation data (CRL and OCSP).

Certificate	CRL	OCSP
Root CA	CRL not available	OCSP not available
	Certificate Revoked	Certificate Revoked
Intermediate CA	CRL not available	OCSP not available
	Certificate Revoked	Certificate Revoked
Leaf	CRL not available	OCSP not available
	Certificate Revoked	Certificate Revoked

Reject unknown status
Fallback on status checking with CRL when OCSP fails

1) GOOGLE CHROME

Chrome exploits the NSS library [52] for TLS, and a platform-dependent library for certificate validation [53]. Chrome behaves differently under Windows, macOS, and Kali Linux when validating the EV and non-EV certificates. On Windows, for non-EV certificates, the browser did not attempt to perform revocation checking since it did not either generate OCSP queries or attempted to download the CRLs. Consequently, the browser silently established a TLS connection with the experimental web server even when a revoked non-EV certificate was present in the certificate chain. For EV certificates, Chrome generated OCSP requests to check the revocation status of each certificate in the chain. It first sent OCSP requests via HTTP GET, but the requests were rejected because the testbed OCSP responder didn't support them. Subsequently, the browser sent the OCSP requests with HTTP POST, correctly processed by our OCSP responder. If the browser received an unknown response, it then requested a CRL for each certificate in the chain. If at least one certificate in the chain was revoked, the browser refused the TLS connection. If the OCSP responder was not available, Chrome sent a request to retrieve the corresponding CRL.

On macOS, for non-EV certificates, similarly to the reaction on Windows, the browser did not retrieve any revocation information through either OCSP or CRL. Thus, a TLS connection was established with the test web server when a non-EV revoked certificate was present in the chain (at any level). For EV certificates instead, Chrome sent OCSP requests for checking the revocation status of the leaf and intermediate CA certificates, but the OCSP responder answered with a ''malformed status'' response because the requests were carried through HTTP GET. Differently from the behavior of Chrome on Windows, on macOS, the browsers did not try to obtain an OCSP response through HTTP POST, and it did not fall back on using CRLs in case the OCSP responder was not available. Thus, a TLS connection was established with success when a revoked certificate was present in the chain (at any level). On Kali Linux, for non-EV and EV certificates, Chrome did not request the revocation information either with CRL or with OCSP. In this scenario, the browser just accepted the TLS connection even if the certificate chain had a revoked certificate (at any level), as installed in the experimental web server. In case both an OCSP responder and the corresponding CRL were not available, Chrome silently accepted the TLS connection for the EV and non-EV certificates on all platforms.

2) MOZILLA FIREFOX

Firefox, similarly to Chrome, exploits the NSS library [52] and its behaviour is consistent on the tested platforms, behaving in the same way on Kali Linux, Windows, and macOS for EV and non-EV certificates. Regarding certificate revocation status checking for the testbed server's certificate, the browser queried the OCSP responder only for the leaf certificate, but it did not verify the revocation status for the other certificates in the chain. Consequently, the TLS connection was refused when a revoked leaf certificate was present; in contrast, the TLS connection was accepted in case a revoked intermediate or root CA certificate was in the chain. If the OCSP responder was not available, Firefox did not fall back to downloading the CRL(s), but it silently accepted the TLS connection. Upon receiving an unknown OCSP response, the browser correctly showed a security warning.

3) OPERA

Opera (born from Chromium project [53]) behaved differently on the selected platforms for the non-EV and EV certificates. On Windows, in the case of non-EV certificates, it did not fetch certificate revocation information either with the OCSP method or with the CRL. Thus, Opera established a TLS connection with the experimental web server even when a revoked certificate was present in the chain. For the EV certificates instead, Opera sent OCSP requests for checking the root CA, intermediate CA, and leaf certificates. Since the requests were sent via the HTTP GET method, the OCSP responder answered with a ''malformed status'' error. Subsequently, the browser automatically tried to get OCSP revocation information through HTTP POST and Opera correctly refused the TLS connection with the testbed server if a revoked certificate was in the chain; it attempted to fall back on CRL if the OCSP responder was not reachable, and it requested a CRL if it received an unknown response from the OCSP responder.

On macOS, for non-EV certificates, it did not check the revocation status of the certificates in the chain. Thus, the browser accepted a TLS connection with the experimental web server when at least one certificate (at any level) in the chain was revoked. For the EV certificates, it sent firstly OCSP requests by using the HTTP GET method to retrieve revocation information for the leaf and intermediate CA. As in the previous tests, the testbed server did not accept such requests. Next, we observed that the browser did not attempt to contact the OCSP responder through the HTTP POST method (as happened on Windows 10). Moreover, it did not try to download the CRLs. Consequently, a TLS connection was established with the testbed server without displaying any security warning when a revoked certificate was present in the chain (at any level). Since OCSP requests

TABLE 4. Results obtained for the tested browsers in handling revocation checking (with CRL and OCSP) in the prototype testbed (certificate chains composed of a leaf certificate, one intermediate CA, and one root CA). X: browser failed the test in all cases; V: browser passed the tests in all cases; EV: browsers passed the tests only for EV certificates; X*: browser did not attempt to retrieve certificate revocation information (neither with OCSP nor with CRL); **: OCSP requests were sent via HTTP GET, but the browser did not send the request via HTTP POST if HTTP GET failed: in this case, it is not possible to evaluate correctly the behaviour; L: browser passed the test only for Leaf certificate; L/I means browser passed the tests for intermediate CA (I) and leaf (L) certificate.

Simulated Tests	Chrome			Firefox	Opera			IE	Edge	Safari
	Linux	MacOS	Win	Linux/MacOS/Win	Linux	MacOS	Win	Win	Win	MacOS
OCSP not available	X*	X	EV	X	X*	X	EV	V L/I	EV	X
CRL not available	X*	EV** L/I	EV	L	X*	EV** L/I	EV	V L/I	EV	EV** L/I
Certificate Revoked	X	EV** L/I	EV	L	X	EV** L/I	EV	V L/I	EV	EV** L/I
Fallback on CRL	X	X	EV	X	X	X	EV	V L/I	EV	X
Reject unknown status for Leaf certificate	X*	EV**	EV	V	X*	EV**	EV	Fallback on CRL	EV	EV**
Reject unknown status for intermediate CA certificate	X*	EV**	EV	X	X*	EV**	EV	Fallback on CRL	EV	EV**
Reject unknown status for rootCA certificate	X*	X	EV	X	X*	X	EV	X	EV	X
Close TLS connection in absence of revocation information	X	X	X	X	X	X	X	X	X	X

failed because of transmission via the HTTP GET method, we couldn't test the browser's behavior when receiving an unknown status from an OCSP responder. On Kali Linux, both for non-EV certificates and for EV certificates, Opera did not request revocation information either through CRL or OCSP. The tests showed that the browser has accepted a connection to the server even though a chain with a revoked certificate (at any level) has been installed on the server.

4) INTERNET EXPLORER

Internet Explorer (IE) treated the non-EV and the EV certificates in the same way. When the browser established a TLS connection to the testbed web server, it tried to fetch certificate revocation information for the intermediate CA and leaf certificates. The browser has requested firstly revocation information for the leaf and intermediate CA certificates to the OCSP responder via HTTP GET. As in the previous tests, the requests were not accepted by the testbed responder. Subsequently, the browser automatically sent the OCSP requests via HTTP POST, and the testbed OCSP responder processed them correctly. IE correctly detected when at least one revoked certificate (at any level) was present in the server's certificate chain, and it (correctly) refused the TLS connection by displaying a security warning. The browser did not fetch revocation information regarding the root CA certificate. Thus, it silently accepted a TLS connection with the experimental web server when the installed root CA certificate was revoked. In case the OCSP responder was not available, it attempted to fall back on CRLs. It also requested the CRL when receiving the unknown status from the OCSP responder. However, in case both the OCSP responder and the appropriate CRL were not available, IE silently accepted the connection to the testbed web server, both for EV and non-EV certificates.

5) MICROSOFT EDGE

Microsoft Edge treated the non-EV certificates differently from the EV ones, like the cases tested for Google Chrome and Opera for Windows. For non-EV certificates, Edge did neither send OCSP requests nor download the corresponding CRLs for the certificates in the chain. Consequently, the browser silently established the TLS connection with the experimental server even when one revoked certificate was present in the chain. For EV certificates, Edge sent OCSP requests to obtain the OCSP revocation status for each certificate in the chain. The OCSP requests were initially sent through the HTTP GET method. Since they were rejected by our testbed OCSP responder, they were sent again (automatically) via the HTTP POST method. Edge correctly refused the TLS connection with the experimental server in the presence of a revoked certificate in the chain; it fell back to CRL in case the OCSP responder was unavailable, and requested a CRL if it received an unknown OCSP response. In case both OCSP response and CRL were not available, Edge silently established a connection for EV and non-EV certificates.

6) SAFARI

Safari, like to Edge and Chrome for Windows, handled the non-EV certificates differently from the EV ones. For non-EV certificates, it did not retrieve revocation information for every certificate in the chain, neither with OCSP nor with CRL. Consequently, it established a successful TLS

connection with the experimental server even though the certificate chain contained a revoked certificate. For the EV certificates instead, Safari has sent OCSP requests for the intermediate CA and leaf certificates. Nevertheless, since the OCSP requests were sent via HTTP GET, they have been considered malformed by our OCSP responder. Next, the browser did not attempt to obtain the OCSP responses through HTTP POST, and it did not attempt to fall back on using the CRLs instead. So, in this case, it was not possible to evaluate correctly the browser's behavior. For the same reason, we could not assess the behavior when receiving an OCSP response containing an unknown status, since the browser was able to send OCSP requests only via HTTP GET, so they were rejected by the testbed server.

VII. ANALYZING CERTIFICATES IN THE MAJESTIC Top1M LIST

Similarly to the analysis performed for the certificates in the Alexa Top1M, we have exploited a dedicated script in Python to analyze the Majestic Top1M domain certificates [54].[18] Besides Alexa and Cisco Umbrella, this list is the third top list widely used in scientific research [55]. In particular, we have looked for support for the OCSP stapling and the CT system.

Methodology. The script took as input the list of domains to be analyzed. For each of them, it performed a different TLS connection, by using the OpenSSL `s_client` application, to retrieve information about support for OCSP stapling. Next, for each certificate, the script searched for the SCTs embedded in the certificate, and it checked whether the *TLS Feature* extension was present because that information was relevant for detecting the support for the OCSP Must-Staple. Moreover, the SCTs have been further analyzed, by comparing their log IDs to the ones contained in a JSON (JavaScript Object Notation) file we have created ad-hoc. This JSON file contains the log servers' IDs recognized by the Chromium Project, grouped by log provider. There are six CT log providers managed by Google, Digicert,[19] Let's Encrypt,[20] Sectigo,[21] TrustAsia,[22] and Cloudflare.[23] In this way, we could count how many SCTs were embedded in the certificates in the Majestic Top1M list domain and to which CT log servers they were referring. We used the resulting data to automatically generate charts expressing the usage of CT log servers, along with the percentage of support for OCSP stapling and OCSP Must-staple.

The script collecting information from the Majestic Top1M domains has run in parallel on two Ubuntu machines for more than a week. A fraction (271,817) of the domains in the Majestic ranking were unreachable. Consequently, the charts and results illustrate data for about 72.82% domain certificates in the mentioned list. Table 5 shows the adoption of OCSP stapling and OCSP Must-Staple mechanisms in 2018 and 2019 as reported in [56], and the comparison with the results we measured in 2022. Only 31.47% of the status request performed during the TLS handshake returned an OCSP response. None of the certificates retrieved from the domains contained the TLS Feature extension. The most used CT logs were the Google ones referred by 674,630 SCTs in the Majestic Top1M certificates (Fig. 11). The second most used CT provider was Digicert, with 371,223 SCTs pointing to its CT log servers. Third, we found Cloudflare with 331,719 SCTs, followed by Let's Encrypt with 289,492 SCTs. The less-used CT operators were Sectigo (with 68,495 SCTs) and TrustAsia (with 161 SCTs).

As already mentioned, every certificate can contain an arbitrary number of SCTs. The second chart produced by the

TABLE 5. OCSP stapling and OCSP Must-staple support in 2018, 2019 (Source: [56]). Comparison with our results measured in 2022.

Year	OCSP stapling	OCSP Must-Staple
2018	19%	0.04%
2019	27%	0%
2022	31%	0%

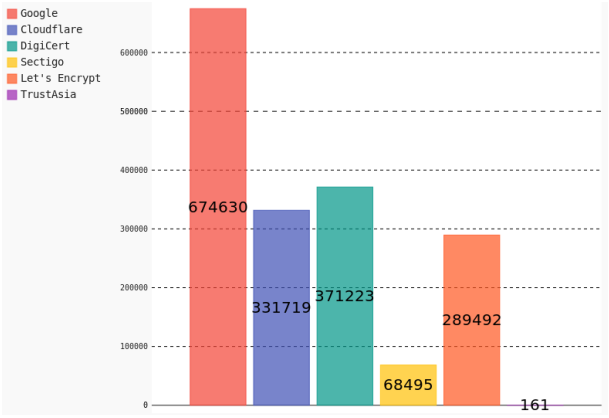

FIGURE 11. CT log providers and the total number of SCTs stored into the CT log providers for the web certificates in the Majestic Top1M list.

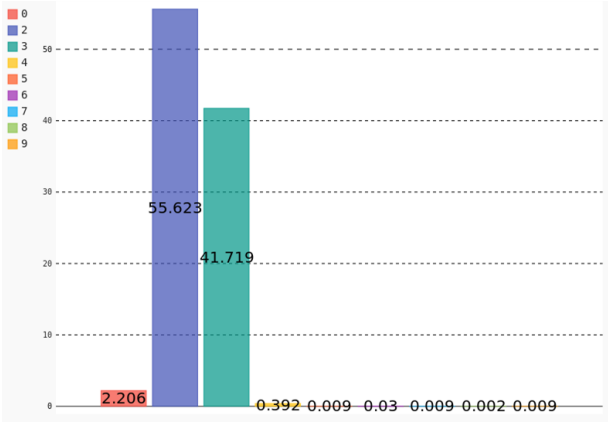

FIGURE 12. Number of certificates in the Majestic Top1M domain list (in %) holding from 0 to 9 SCTs.

[18] We used the Top1M domain list ranking from Majestic because the Alexa Top1M domain list ranking went out of support in 2022.

[19] https://www.digicert.com

[20] https://letsencrypt.org

[21] https://sectigo.com

[22] https://www.trustasia.com

[23] https://www.cloudflare.com

TABLE 6. Number of SCTs stored in the specific CT log servers (for the web server certificates of Majestic Top1M list).

Log server	No. of SCTs
Google Argon2022 log	64,056
Google Argon2023 log	326,431
Google Argon2024 log	0
Google Xenon2022 log	77,134
Google Xenon2023 log	207,001
Google Xenon2024 log	0
Google Icarus log	0
Google Pilot log	2
Google Rocketeer log	4
Google Skydiver log	2
Cloudflare Nimbus2022 Log	76,465
Cloudflare Nimbus2023 Log	255,254
Cloudflare Nimbus2024 Log	0
DigiCert Log Server	188
DigiCert Log Server 2	2
DigiCert Yeti2022 Log	34
DigiCert Yeti2023 Log	204,593
DigiCert Yeti2024 Log	0
DigiCert Yeti2025 Log	0
DigiCert Nessie2022 Log	7,688
DigiCert Nessie2023 Log	158,184
DigiCert Nessie2024 Log	0
DigiCert Nessie2025 Log	0
DigiCert Yeti2022-2 Log	534
Sectigo Sabre CT log	14,507
Sectigo Mammoth CT log	53,988
Let's Encrypt Oak2022 log	52,692
Let's Encrypt Oak2023 log	236,800
Let's Encrypt Oak2024H1 log	0
Let's Encrypt Oak2024H2 log	0
Trust Asia Log2022	15
Trust Asia Log2023	146

script (Fig. 12) shows how many different SCTs have been embedded within a single (domain) certificate. We observed that most certificates had two or three SCTs. More precisely, about 55.62% of the domain certificates had two SCTs, and about 41.72% of them had three SCTs. We have also found a small number of certificates ($\tilde{2}$.20%) that did not have any SCT. Then, 0.392% domain certificates contained four SCTs, 0.009% domain certificates had 5 SCTs, 0.030% had 6 SCTs and 0.009% had 7 SCTs. In the end, 0.002% of the certificates had 8 SCTs, and the last 0.009% domain certificates used the maximum number of SCTs found during this analysis, i.e., 9 SCTs.

Table 6 details the usage of the SCTs in the domain server certificates of the Majestic Top1M list, along with the CT log servers considered trusted by the Chromium Project.

VIII. CONCLUSION AND FUTURE WORK

Despite their wide adoption, X.509v3 certificates are still not adequately verified against their revocation status when used in common applications, like web browsers. Moreover, when using certificates, some privacy issues might occur. Herein, we described the main actors impacting the definition and processing of X.509v3 certificates, as well as the technologies and protocols employed for certificate revocation checking, like the CRL or the OCSP protocol.

Although this work does not have the ambition to provide a full-fledged scanning of all the certificates populating the Web PKI, it fills the gap between theoretical requirements and genuine implementation for certificate revocation checking in web browsers. Furthermore, it provides some interesting points (e.g., for user privacy aspects) that need further investigation and research in the HTTPS ecosystem.

By analyzing the website certificate profiles from the Alexa Top1M list in August 2021, we noticed that some violations with respect to the CAB baseline requirements [28] still occurred, such as the use of expired CA certificates or the criticality bit not set in the BC extension of CA certificates. Regarding the support of certificate revocation extensions (CRLDP and AIA), the situation is encouraging: the majority of the analyzed certificates supported at least one certificate revocation method (namely, OCSP), and about 30% of them supported both CRL and OCSP methods. However, since the X.509v3 certificate extensions for revocation checking are optional or non-critical, the applications can adopt different strategies for their processing. Thus, we analyzed how common browsers process the CRLDP and AIA extensions. For EV certificates, revocation checking is, in general, adequate, yet, for non-EV certificates, the certificate status verification is limited, ranging from "no check" (leading to browsers silently accepting TLS connections with websites presenting a revoked certificate) to checking only the status of the website certificate. The soft-fail approach is still the most widely encountered, instead of the more stringent hard-fail one.

From the user privacy point of view, CRL is better than OCSP, even though with OCSP stapling (adopted by almost 30% of the analyzed domains) a more privacy-preserving certificate revocation checking is achieved. Unfortunately, the OCSP Must-Staple, which provides privacy features for the hard-fail approach, has not been adopted by the web servers analyzed. We have also documented the support for the CT system in the most widely visited 1 Million web domains (listed in the Majestic list) in November 2022. Most of these domains included two or more (up to nine) SCTs. For the privacy part, more work should be done (as stated in [39]) to allow privacy-preserving SCT auditing, although some proposals already exist for this scope [57].

Finally, we note that other intermediate elements could exist between clients (web browsers) and the origin TLS servers, like proxies or edge servers. These components may alter the HTTPS trust model because they break the client-server TLS channel. They typically implement the so-called TLS "interception", which is increasingly used nowadays by anti-virus software, Content Delivery Networks, or enterprise proxy applications. Future work will address certificate validation in some common TLS interception elements. Moreover, we aim to integrate the analysis of X.509v3 certificates into the TLS-Monitor tool we have proposed in [58] to detect and counter possible TLS attacks.

ACKNOWLEDGMENT

The authors thank Corrado Vecchio and Matteo Simone, graduate students at Politecnico di Torino (Italy) for performing part of the tests described in this research. They also

thank the anonymous reviewers of IEEE Access journal for providing helpful comments that allowed them to improve this work.

REFERENCES

[1] L. M. Kohnfelder, "Towards a practical public-key cryptosystem," Doctoral dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1978.
[2] *Information Technology—Open Systems Interconnection—The Directory: Public Key and Attribute Certificate Frameworks*, Standard X.509-ISO/IEC 9594-8, 2000.
[3] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*, Standard RFC-8446, Aug. 2018.
[4] D. Berbecaru, A. Atzeni, M. De Benedictis, and P. Smiraglia, "Towards stronger data security in an eID management infrastructure," in *Proc. 25th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*, Mar. 2017, pp. 391–395, doi: 10.1109/PDP.2017.90.
[5] D. G. Berbecaru, A. Lioy, and C. Cameroni, "Providing login and Wi-Fi access services with the eIDAS network: A practical approach," *IEEE Access*, vol. 8, pp. 126186–126200, 2020, doi: 10.1109/ACCESS.2020.3007998.
[6] D. G. Berbecaru, A. Lioy, and C. Cameroni, "On enabling additional natural person and domain-specific attributes in the eIDAS network," *IEEE Access*, vol. 9, pp. 134096–134121, 2021, doi: 10.1109/ACCESS.2021.3115853.
[7] J. Hughes, S. Cantor, J. Hodges, F. Hirsch, and P. Mishra. (Mar. 2005). *Profiles for the OASIS security assertion markup language (SAML) V2.0*. OASIS Standard. Accessed: Jul. 23, 2023. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
[8] *OpenID Connect*. Accessed: Jul. 23, 2023. [Online]. Available: https://openid.net/connect/
[9] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *Proc. Conf. Internet Meas. Conf.*, Oct. 2013, pp. 291–304, doi: 10.1145/2504730.2504755.
[10] X. de Carné de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2016, pp. 1–17, doi: 10.14722/ndss.2016.23374.
[11] *Changes to Microsoft Edge Browser TLS Server Certificate Verificatio*. Accessed: Jul. 23, 2023. [Online]. Available: https://learn.microsoft.com/en-us/deployedge/microsoft-edge-security-cert-verification.
[12] *Microsoft Trusted Root Program*. Accessed: Jul. 23, 2023. [Online]. Available: https://learn.microsoft.com/en-us/security/trusted-root/program-requirements
[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Standard RFC-5280, May 2008.
[14] CA/Browser Forum. (Dec. 2022). *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates Version 1.8.6, 14*. Accessed: Jul. 23, 2023. [Online]. Available: https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.8.6.pdf
[15] A. Delignat-Lavaud, M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie, "Web PKI: Closing the gap between guidelines and practices," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2014, pp. 1–15. [Online]. Available: https://users.soe.ucsc.edu/~abadi/Papers/ndss14.pdf
[16] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP*, Standard RFC-6960, Jun. 2012.
[17] D. Berbecaru, M. M. Casalino, and A. Lioy, "FcgiOCSP: A scalable OCSP-based certificate validation system exploiting the FastCGI interface," *Softw.-Pract. Exper.*, vol. 43, no. 12, pp. 1489–1518, 2013, doi: 10.1002/spe.2148.
[18] *Google: CRLSets (The Chromium Projects)*. Accessed: Jul. 23, 2023. [Online]. Available: https://dev.chromium.org/Home/chromium-security/crlsets
[19] *What is Mozilla's OneCRL?* Accessed: Jul. 23, 2023. [Online]. Available: https://www.thesslstore.com/blog/what-is-onecrl/
[20] W. Wang, Y. Li, C. Wang, Y. Yan, J. Li, and D. Gu, "Re-check your certificates! Experiences and lessons learnt from real-world HTTPS certificate deployments," in *Proc. 15th Int. Conf. Netw. Syst. Secur.*, Tianjin, China, Oct. 2021, p. 1737, doi: 10.1007/978-3-030-92708-0_2.
[21] A. S. Wazan, R. Laborde, D. W. Chadwick, F. Barrere, and A. Benzekri, "TLS connection validation by web browsers: Why do web browsers still not agree?" in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2017, pp. 665–674, doi: 10.1109/COMPSAC.2017.240.
[22] T. Smith, L. Dickinson, and K. Seamons, "Let's revoke: Scalable global certificate revocation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2020, pp. 1–14, doi: 10.14722/ndss.2020.24084.
[23] T. Chung, J. Lok, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, J. Rula, N. Sullivan, and C. Wilson, "Is the web ready for OCSP must-staple?" in *Proc. Internet Meas. Conf.*, Boston, MA, USA, Oct. 2018, pp. 105–118, doi: 10.1145/3278532.3278543.
[24] M. Wbbeling. *Securing the TLS Ecosystem With Certificate Transparency A Curse and a Blessing*. Accessed: Jul. 23, 2023. [Online]. Available: https://www.admin-magazine.com/Archive/2020/60/Securing-the-TLS-ecosystem-with-Certificate-Transparency
[25] *Certificate Transparency Project*. Accessed: Jul. 23, 2023. [Online]. Available: https://certificate.transparency.dev/
[26] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27, Apr. 2016 on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)*. Accessed: 2023-07-23. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj
[27] D. Eastlake. *Transport Layer Security (TLS) Extensions: Extension Definitions*, Standard RFC-6066, Jan. 2011.
[28] CA/Browser Forum. (Aug. 2021). *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, Version 1.7.9, 16*. Accessed: Jul. 23, 2023. [Online]. Available: https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.7.9.pdf
[29] D. G. Berbecaru, "Dataset for an evaluation of X.509 certificate revocation and related privacy issues in the web PKI ecosystem," Politecnico di Torino, Italy, Tech. Rep., 2023. Accessed: Jul. 30, 2023, doi: 10.21227/1y3e-2f34.
[30] (Jul. 12, 2021). *Who Your Browser Trusts, and How to Control It*. Accessed: Jul. 23, 2023. [Online]. Available: https://expeditedsecurity.com/blog/control-the-ssl-cas-your-browser-trusts/
[31] A. Ray, *Cybersecurity for Connected Medical Devices*. Cambridge, MA, USA: Academic Press, 2022, ch. 7, pp. 217–262, doi: 10.1016/B978-0-12-818262-8.00007-3.
[32] R. Biddle, P. C. van Oorschot, A. S. Patrick, J. Sobey, and T. Whalen, "Browser interfaces and extended validation SSL certificates: An empirical study," in *Proc. ACM Workshop Cloud Comput. Secur.*, Chicago, IL, USA, Nov. 2009, p. 1930, doi: 10.1145/1655008.1655012.
[33] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna, "Cloud strife: Mitigating the security risks of domain-validated certificates," in *Proc. Appl. Netw. Res. Workshop*, Montreal, QC, Canada, Jul. 2018, pp. 1–16, doi: 10.1145/3232755.3232859.
[34] L. Schwittmann, M. Wander, and T. Weis, "Domain impersonation is feasible: A study of CA domain validation vulnerabilities," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Stockholm, Sweden, Jun. 2019, pp. 544–559, doi: 10.1109/EuroSP.2019.00046.
[35] *What is Extended Validation Certificate?* Accessed: Jul. 23, 2023. [Online]. Available: https://www.globalsign.com/en/ssl-information-center/what-is-an-extended-validation-certificate
[36] *What is Extended Validation (EV) SSL Certificate?* Accessed: Jul. 23, 2023. [Online]. Available: https://www.digicert.com/faq/public-trust-and-certificates/what-is-an-extended-validation-ev-ssl-certificate
[37] CAB (CA/Browser Forum). (Nov. 2022). *Guidelines for the Issuance and Management of Extended Validation Certificates, Version 1.8.0, 30*. Accessed: Jul. 23, 2023. [Online]. Available: https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-1.8.0.pdf
[38] B. Laurie, A. Langley, and E. Kasper. *Certificate Transparency*, Standard RFC-6962, IETF, Jun. 2013.
[39] S. Meiklejohn, J. DeBlasio, D. O'Brien, C. Thompson, K. Yeo, and E. Stark, "SoK: SCT auditing in certificate transparency," 2022, *arXiv:2203.01661*.
[40] M. Prince. (Oct. 29, 2012). *OCSP Stapling: How CloudFlare Just Made SSL 30% Faster*. Accessed: Jul. 23, 2023. [Online]. Available: https://blog.cloudflare.com/ocsp-stapling-how-cloudflare-just-made-ssl-30/
[41] Y. Pettersen. *The Transport Layer Security (TLS) Multiple Certificate Status Request Extension*, Standard RFC-6961, Jun. 2013.
[42] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's PKI," in *Proc. Internet Meas. Conf.*, Tokyo, Japan, Oct. 2015, pp. 183–196, doi: 10.1145/2815675.2815685.

[43] P. Hallam-Baker. *X509v3 Transport Layer Security (TLS) Feature Extension*, Standard RFC-7633, Oct. 2015.
[44] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, "Mission accomplished: HTTPS security after diginotar," in *Proc. Internet Meas. Conf.*, London, U.K., Nov. 2017, pp. 325–340, doi: 10.1145/3131365.3131401.
[45] B. Laurie, E. Messeri, and R. Stradling. *Certificate Transparency Version 2.0*, Standard RFC-9162, Dec. 2021.
[46] 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. ACM SIGCOMM Conf. Internet Meas. Conf.*, Berlin, Germany, Nov. 2011, Art. no. 427444, doi: 10.1145/2068816.2068856.
[47] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating SSL certificates in non-browser software," in *Proc. ACM CSS*, Raleigh, NC, USA, Oct. 2012, pp. 38–49, doi: 10.1145/2382196.2382204.
[48] 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 *Proc. ACM Internet Meas. Conf.*, Santa Monica, CA, USA, Nov. 2016, pp. 527–541, doi: 10.1145/2987443.2987454.
[49] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2014, pp. 114–129, doi: 10.1109/SP.2014.15.
[50] *Net Market Share: Browser Market Share*. Accessed: Feb. 3, 2022. [Online]. Available: https://netmarketshare.com/browser-market-share.aspx
[51] *OpenSSL Library*, Accessed: Jul. 23, 2023. [Online]. Available: www.openssl.org
[52] *Network Security Services (NSS)*. Accessed: Jul. 23, 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS?retiredLocale=it
[53] *Chromium*. Accessed: Jul. 23, 2023. [Online]. Available: https://www.chromium.org/developers/design-documents/network-stack#TOC-SSL-TLS
[54] *The Majestic Million*. Accessed: Jul. 23, 2023. [Online]. Available: https://majestic.com/reports/majestic-million
[55] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez, "A long way to the top: Significance, structure, and stability of internet top lists," in *Proc. Internet Meas. Conf.*, Boston, MA, USA, Oct. 2018, Art. no. 478493, doi: 10.1145/3278532.3278574.
[56] A. S. Wazan, R. Laborde, D. W. Chadwick, R. Venant, A. Benzekri, E. Billoir, and O. Alfandi, "On the validation of web X.509 certificates by TLS interception products," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 227–242, Jan. 2022, doi: 10.1109/TDSC.2020.3000595.
[57] D. Kales, O. Omolola, and S. Ramacher, "Revisiting user privacy for certificate transparency," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Stockholm, Sweden, Jun. 2019, pp. 432–447, doi: 10.1109/EuroSP.2019.00039.
[58] D. G. Berbecaru and G. Petraglia, "TLS-monitor: A monitor for TLS attacks," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2023, pp. 1–6, doi: 10.1109/CCNC51644.2023.10059989.

DIANA GRATIELA BERBECARU (Member, IEEE) received the M.Sc. degree in computer science from the University of Craiova, Romania, and the Ph.D. degree in computer engineering from Politecnico di Torino, Italy. She is currently an Assistant Professor with the Department of Control and Computer Engineering, Politecnico di Torino. Her current research interests include authentication, identity and access management, data privacy, the IoT security, and trusted computing. She is a member of the TORSEC Cybersecurity Research Group.

ANTONIO LIOY received the M.Sc. degree (summa cum laude) in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Torino. He is currently a Full Professor with Politecnico di Torino, where he leads the TORSEC Cybersecurity Research Group. His current research interests include network security, policy-based system protection, trusted computing, and electronic identity.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement