On the Validation of Web X.509 Certificates by TLS Interception Products

Ahmad Samer Wazan, Romain Laborde, David W. Chadwick, Rémi Venant, Abdelmalek Benzekri, Eddie Billoir, and Omar Alfandi

Abstract—The Transport Layer Security (TLS) protocol aims to provide confidentiality and integrity of data. It is based on X.509 Certificates. Our previous research showed that popular Web Browsers exhibit non-standardized behaviour with respect to the certificate validation process [1]. This article extends that work by examining their handling of OCSP Stapling. We also examine several popular HTTPS interception products, including proxies and anti-virus tools, regarding their certificate validation processes. We analyse and compare their behaviour to that described in the relative standards.

Index Terms—Web PKI, X.509 certificate, certificate validation, OCSP stapling

1 Introduction

The TLS Protocol is designed to provide confidentiality and integrity of end-to-end communications [2]. However, the end-to-end protection provided by TLS is incompatible with other security products that need to retain visibility into network traffic, such as anti-virus tools. To retain visibility into network traffic, HTTPS interception products interpose in the middle of the communication between the client application and the web server and operate as a man-in-the-middle (MITM). As a consequence, these products transform the end-to-end communications into two TLS communications: Client<->HTTPS interception product and HTTPS interception product<->web server (Fig. 1).

Each HTTPS communication is based on X.509 Certificates. Therefore, the HTTPS interception product acts as a TLS server for the client application by presenting its own certificate. Similarly, the web server authenticates itself to the interception product by presenting its own certificate when the client initiates a TLS connection.

Different legitimate reasons necessitate the interception of HTTPS communications, such as enforcing a usage policy inside an enterprise (e.g., employees are not allowed to access streaming websites), Malware detection and Crypto compliance to use strong cipher suites, etc.

Theoretically, things are relatively simple. But in practice, things are much more complicated: the HTTPS interception product establishes the second HTTPS connection after the validation of the web server certificate. The client application has no visibility of the web server's identity; it delegates the server's certificate validation to the interception product, and simply accepts the decision of the HTTPS interception product.

Problems in the X.509 certificate validation process such as accepting revoked, untrusted or invalid certificates can cause dangerous consequences paving the way for attacks that may weaken the client's communications security.

In 2009, we highlighted the different behaviours of several web browsers (Internet Explorer (IE), Opera and Firefox) when validating certificates [8]. We explained that the reasons for these differences were either due to violation of the standards by the browsers, or ambiguity in the standards themselves.

In 2017, we performed an increased set of tests [1], and covered a greater number of web browsers (IE, Edge, Opera, Firefox, Safari and Chrome), as well as covering the newest standards. Our work described the quality of X.509 certificate validation implemented by these web browsers, as well as showing their evolution since 2009. Also, we produced new tests for analysing how web browsers implement the OCSP protocol.

In this paper, we complete our work by:

1. Applying the same set of tests to different HTTPS interception products. We tested anti-virus and proxy software in order to highlight their behaviours when they were confronted with chosen test values in specific certificate fields. The results were then analysed and compared to the expected behaviour described in the respective standards.
2. As in our 2017 study [1], we applied the same set of tests at two different dates between 2017 and 2019, in

- *A. S. Wazan and O. Alfandi are with the TI, Zayed University - Abu Dhabi Campus, Abu Dhabi, United Arab Emirates. E-mail: {ahmad.wazan, omar.alfandi}@zu.ac.ae.*
- *R. Laborde is with IT, University of Toulouse, Toulouse, France. E-mail: laborde@irit.fr.*
- *D. W. Chadwick is with the School of Computing, University of Kent, Canterbury, Kent, United Kingdom of Great Britain and Northern Ireland. E-mail: d.w.chadwick@kent.ac.uk.*
- *R. Venant and A. Benzekri are with IT, IRIT, Toulouse, Occitane, France. E-mail: {remi.venant, benzekri}@irit.fr.*
- *E. Billoir is with the IT, Paul Sabatier University, Toulouse, Occitane, France. E-mail: eddie.billoir@gmail.com.*

Manuscript received 28 Jan. 2020; revised 22 Apr. 2020; accepted 2 June 2020. Date of publication 8 June 2020; date of current version 17 Jan. 2022. (Corresponding author: Ahmad Samer Wazan) Digital Object Identifier no. 10.1109/TDSC.2020.3000595

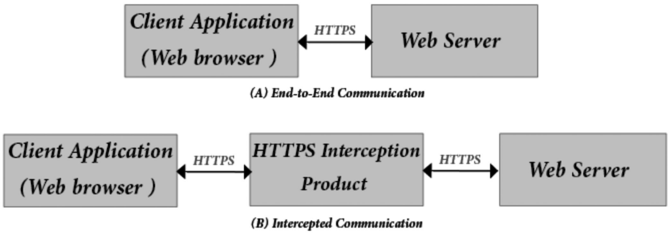


Fig. 1 The End-to-End HTTPS Communication / Intercepted HTTPS Communication.

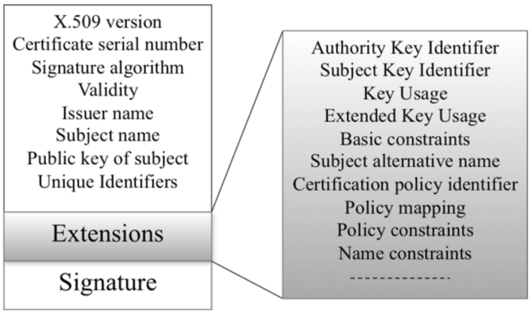


Fig. 2. Certificate contents (inspired by [9]).

order to show any evolution in the behaviour of the interception products. This helps to show whether they have improved their behaviour with regards to the related standards or not.

3. Analysing the behaviour of web browsers and web servers with regards to OCSP Stapling. We also show the results of a survey that we performed on two different dates between 2018 and 2019 in order to show the evolution of the deployment of OCSP Stapling.

Since 2014, several different works have been published about certificate validation errors detected in TLS clients. For example, Brubaker *et al* [14] proposed a tool that can generate a set of test certificates whose values are random. Detecting different behaviour between at least two TLS clients is considered an indication of a certificate validation error. Our work adopts a different test strategy in order to detect cases where *all* TLS clients behave badly with regards to a specific test certificate. In addition, our test certificates don't include random values; instead we present well-crafted test certificates for TLS clients. Carnavalet *et al* [7] have performed different tests on TLS interception products, but their tests covered mostly known issues in the TLS protocol (e.g., FREAK, CRIME, BEAST, etc.) rather than focusing on validation issues of X.509 certificates. Finally, both research studies didn't show the evolution of validation behaviour by TLS products and didn't handle the revocation-checking behaviour as we do in this work. The revocation problem is an important one because many of the studied TLS products can lead web users to accept revoked certificates. Our study shows, perhaps not unsurprisingly, that TLS interception products are not the only ones responsible for not correctly checking the revocation status of certificates, because even the major web browsers don't have a consistent approach to this. Worse still, we find that things are not necessarily improving over time. Finally, our study analyses the reasons behind certificate revocation failings and we give a suggestion for improving this in the conclusions.

The rest of this paper is structured as follows. Section 2 overviews the base set of standards related to X.509 certificates. Section 3 exposes and analyses the results of tests executed on six TLS interception products. In this section, we show how the behaviour of these products is inconsistent. Our study also shows the evolution of these products' behaviour between 2017 and 2019. In Section 3.3 we focus on the revocation problem and present the different revocation techniques. We show how web browsers and interception products implement them. We also show the deployment trend between 2018 and 2019 for the latest revocation checking technique, called OCSP Stapling. Finally, in Section 4 we discuss our findings and give our conclusions in Section 5.

2 What is an X.509 Certificate

The contents and processing of X.509 public key certificates (PKCs) are regulated through numerous standards documents, first officially described in the ISO/ITU-T X.509 standard [3]. X.509 provides the general framework for public key infrastructures (PKIs), the syntax of PKCs and revocation lists, and how PKCs can be extended (by literally anyone). Each standard certificate field has its own syntax and semantics as well as constraints on its possible values. In many cases a field can have different syntax choices. These fields provide information about the certificate version number, the subject of the certificate, the public key, the way the key can be used, and the certificate life cycle management process (Fig. 2).

Three kinds of field exist: mandatory fields, optional fields and extensions (which are all optional). When a field is mandatory, Certificate Authorities (CAs) must fill it and Relying Parties (RPs) must check it when validating certificates. Extensions can be marked as critical or not. If present and marked critical, the RP must obey its contents or reject the certificate. If marked not critical, the RP can ignore the extension if it does not recognize it, but must obey it otherwise i.e., it should not ignore a non-critical extension that it supports.

The complexity of the X.509 standard, in terms of fields that are mandatory, optional, choices, and extensions, means that it is almost impossible for two different implementers to produce fully interworking code. A PKC produced by one implementer cannot always be fully validated by another, and vice versa.

Consequently, the IETF PKIX group developed an X.509 standard profile (RFC 5280) to address the specific needs for using PKIs on the Internet. Especially, the profile eliminates most options, make choices where several are available, and specifies which extensions should be used. However, due to the large constituency of the IETF, many different authors proposed many different extensions and ways of using X.509 certificates, so that by now, over 50 PKIX specified RFCs exist. One can easily see why it is still not a trivial task to implement a fully conformant web browser.

Among all the certificate extensions defined in X.509 and RFC 5280, Internet applications (such as web browsers) must at least be able to recognize: basic constraints, certificate policies, policy constraints, subject alternative name,

TABLE 1
List Of Tested Products

Product	Version (Test Date)
Avast Antivirus Gratuit	17.4.2294 (2017), 19.5.2378 (2019)
Kaspersky Total security	17.0.0.611 (2017), 19.0.0.1088 (2019)
AVG Internet Security	17.4.3014 (2017), 19.5.3093 (2019)
ESET Internet Security	10.1.210.2 (2017), 12.1.34.0 (2019)
Squid	3.3.8 (2017), 4.6(2019)
Charles Web debugging Proxy	4.1.2 (2017), 4.2.8 (2019)
Mitmproxy	0.9.2 (2017), 4.0.4 (2019)
Telerik Fiddler	4.6.20171.14978 (2017), 5.0.20192.25091(2019)

key usage, name constraints, extended key usage and inhibit any-policy extensions; but do not need to recognize: authority and subject key identifiers, and policy mapping extensions [4].

Starting from 2007, a new consortium of implementors of CAs, web browsers and OSs was established to improve the quality of certificate issuance and management, known as the CA/Browser Forum [13]. In the beginning, the consortium issued a set of guidelines for a new kind of certificate called an Extended Validation (EV) certificate. In 2011, CA/Browser forum issued “Baseline requirements” for any kind of public key certificate. Both standards are now globally adopted by the majority of CAs on the Internet.

Other important standards related to X.509 PKCs are:

- RFC 6125: this explains the rules that must be followed in representing and verifying the identity of servers identified in the PKCs,
- RFC 6960: this specifies the OCSP protocol used for checking a PKC’s status.
- RFC 5019: this addresses the scalability issues related to the deployment of OCSP servers in high-volume environments. It also specifies the rules to follow for caching OCSP responses.

Other standards will be mentioned in the rest of the paper at the appropriate point.

3 Analysis of TLS Interception Products’ Behavior

In this section, we provide the results of our tests with several HTTPS interception products when they validate web server certificates. We focus our tests on the fields related to the subject, the key usage and the certificate status.

Our objective is not to show the quality of certificate validation for every existing HTTPS interception product. Rather we selected a sample from the most popular ones. We didn’t limit the study to license-free products. We also tested trial versions of commercial products. Prior to testing, we reviewed the products’ specifications to ensure that they supported HTTPS interception, and we configured their settings to enable interception if a product did not do it by default. The list of tested products includes 4 anti-virus and 4 proxies (see Table 1).

With regards to the validation, we found three different strategies followed by HTTPS interception products:

- Full validation (fV): in this case, the product handles the validation of certificates itself. It shows personalized error messages that are different from those of the web browsers. Kaspersky, Mitm, Fiddler and Squid proxies are in this category.
- Delegated validation (dV): in this case, the interception product delegates the validation of certificates to the web browsers, except for revocation checking, which it performs itself. The product copies the certificate’s contents into a new PKC issued by itself and passes this to the browser for validation. The product is in effect becoming the issuing CA of all received certificates. Revocation checking cannot be delegated because the browsers receive the certificates generated by the interception product and not the original certificates generated by the web servers’ CAs. Avast, AVG and ESET fall into this category.
- Incorrect Validation (iV): In this case, the interception product delegates validation to the browsers, as in the dV case, but doesn’t handle the revocation checking itself. The Charles proxy falls into this category.

With regards to the products that fall under the fV category, we found three possible responses when the HTTPS interception product handles a certificate, denoted as follows:

- A: accept the certificate without any intervention by the user,
- W: warn the user about the existence of a problem by showing a warning message and asking him/her to make an accept/refuse decision,
- R: refuse the certificate and prohibit access to the web server without any intervention by the user.

To easily identify the evolution of TLS interception products’ behaviour compared to 2017, we use the symbol → to show any change in the product’s behavior with regards to a test case scenario. The result on the left side of the arrow represents the result obtained in 2017, and the result on the right side of the arrow represents the result obtained in 2019. In addition, we highlight the results that are not conformant to standards by colouring them in red. The evolution in behaviour of a TLS interpection product is considered as a regression when the table shows a change in the cell from a non-coloured result to a red coloured result (e.g., W→A). The evolution is considered as an improvement when there is change from a red-coloured result to a non-coloured result (e.g A →W).

3.1 TLS Certificate Subject

The TLS certificate subject represents the web server. The identity of the server may be either a Fully Qualified Domain Name (FQDN) or an IP address or both. FQDNs and IP addresses are different types of name (called name forms in the standards). A web server could hold many FQDNs that all point to the same IP address and conversely, one FQDN may point to different IP addresses.

3.1.1 What do the Standards State About the Subject?

The X.509 standard [3] states that the subject field identifies the entity associated with the public-key found in the subject public key field. An entity could have one or more

TABLE 2
Multiple FQDN Web Server Identities

	Antivirus								Proxies									
	Avast		Kaspersky		AVG		ESET		Squid		Charles		Mitm		Fiddler		Standards	
	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
i) SCN= sana1.fr, SAN-DNS=sana1dns.fr	dV	dV	W	A	dV	dV	dV	dV	W	A	iV	iV	W ➔ R	A	W	A	I	V
ii) SCN= null, SAN-DNS=sana1dns.fr	dV	dV	W	A	dV	dV	dV	dV	R	A	iV	iV	W ➔ R	A	W	A	I	V
iii) SCN= sana1.fr , no SAN-DNS	dV	dV	A	W	dV	dV	dV	dV	A ➔ W	R	iV	iV	A	W ➔ R	A	W	?	I
iv) SCN=null, no SAN-DNS	dV	dV	W	W	dV	dV	dV	dV	R	R	iV	iV	W ➔ R	W ➔ R	W	W	I	I
v) SCN=null, SAN-DNS=sana1.fr , sana1dns.fr	dV	dV	A	A	dV	dV	dV	dV	A	A	iV	iV	A	A	A	A	V	V

Where: ***S1*** *= sana1.fr,* ***S2*** *= sana1dns.fr,* ***dV*** *= delegated Validation,* ***iV*** *= incorrect Validation,* ***A*** *= Accept,* ***W*** *= Warn,* ***R*** *= Refuse,* ***I*** *= Invalid certificate w.r.t standard,* ***V*** *= valid certificate w.r.t standard.*

alternative names, of different types (or forms), held in the *subjectAltName* extension. According to the X.509 standard, an implementation that supports this extension is not required to process all the name types. If the extension is flagged critical, at least one of the name types that is present must be recognized and processed, otherwise the certificate must be considered invalid.

RFC 5280 states that the subject name may be carried in the subject field and/or the *subjectAltName* extension. If the subject naming information is present only in the *subjectAltName* extension, then the subject name should be empty and the *subjectAltName* extension must be critical. According to this statement a TLS certificate can hold multiple names in a combination of the Subject field (*commonName* (CN) component) and the Subject Alternative Name (*subjectAltName*) extension. These names must all refer to the same entity, although a browser need not recognize all the different name types.

According to the baseline requirements (BR) of the CA/Browser forum, CAs are discouraged from issuing certificates that have a commonName (CN) component in the subject field [16], however this is not prohibited. BR states that if the CN component is present, it MUST contain a single IP address or Fully-Qualified Domain Name that should be one of the values contained in the certificate's subjectAltName extension. However, the CA/Browser forum requires the presence of the subjectAltName extension in all certificates, and this may have a dNSName (i.e., DNS name) or an iPAddress value. Finally, since October 2016 the CA/Browser forum has prohibited the practice of inserting a reserved (private) IP address in the subjectAlternativeName extension or in the Subject commonName. This is because these addresses are typically local addresses, and consequently refer to thousands of internal servers, many of which are not be accessible from the Internet [25].

3.1.2 Tests and Results

The identity of a server could be represented by a FQDN value or by an IP address or both. We have performed experiments to test certificates holding the two types of name separately as well as both types together.

In the first set of experiments (Table 2), we tested how the HTTPS interception products reacted when the certificate contains zero, one or more FQDN names. We configured our web server to respond to requests sent to either sana1.fr or sana1dns.fr. As the names could be mentioned in either or both of the Subject Name - Common Name (SCN) and SubjectAltName - DNS Name (SAN-DNS) fields, we tested the following different combinations of names in our web server certificate:

i. SCN = sana1.fr, SAN-DNS = sana1dns.fr
ii. SCN = null, SAN-DNS = sana1dns.fr
iii. SCN = sana1.fr, no SAN-DNS field
iv. SCN = null, no SAN-DNS field
v. SCN = null, SAN-DNS = sana1.fr and sana1dns.fr.

For each combination, we recorded the reaction of each HTTPS interception product when accessing sana1.fr and sana1dns.fr. We also state whether the certificate is Valid (V) or Invalid (I) according to the X.509 standards.

In the second set of experiments (Table 3), our server was located at 192.168.133.149 and in some cases it was configured with the DNS name sanal.fr. We tested how the HTTPS interception products reacted when the certificate contains a combination of IP address and FQDN as follows:

i. SCN = null, no SAN-DNS field, SAN-IP = 192.168.133.149
ii. SCN = sana1.fr, no SAN-DNS field, SAN-IP = 192.168.133.149
iii. SCN = null, SAN-DNS = sana1.fr, SAN-IP = 192.168.133.149
iv. SCN = null, SAN-DNS = sana1.fr, no SAN-IP field
v. SCN = null, SAN-DNS = null, no SAN-IP field
vi. SCN = null, SAN-DNS = null, SAN-IP = 192.168.133.149

3.1.3 Analysis of the Results

The Primary objective of an X.509 PKC is to bind an identity to a public key. In the case of a web server, the identity is either a FQDN name or an IP address.

TABLE 3
IP Address Server and/or fqdn Identities

	Antivirus								Proxies									
	Avast		Kaspersky		AVG		ESET		Squid		Charles		Mitm		Fiddler		Standards	
	S1	IP	S1	IP	S1	IP	S1	IP	S1	IP	S1	IP	S1	IP	S1	IP	S1	IP
vi) SCN= null , SAN-IP=192.168.133.149	dV	dV	W	A	dV	dV	dV	dV	R	R	iV	iV	W → R	W → R	W	W → A	I	I
vii) SCN= sana1.fr ,SAN-IP=192.168.133.149	dV	dV	A	A	dV	dV	dV	dV	W	R	iV	iV	A	W → R	A	W → A	V	I
viii) SCN= null, SAN-IP=192.168.133.149 SAN-DNS=sana1.fr	dV	dV	A	A	dV	dV	dV	dV	A	R	iV	iV	W → A	W → R	A	W → A	V	I
ix) SCN= null ,No SAN-IP ,SAN-DNS=sana1.fr	dV	dV	A	W	dV	dV	dV	dV	A	R	iV	iV	A	W → R	A	W	V	I
x) SCN= null ,No SAN-IP ,SAN-DNS=null	dV	dV	W	W	dV	dV	dV	dV	R	R	iV	iV	W → R	W → R	W	W	I	I
xi) SCN= null , SAN-DNS=null , SAN-IP=192.168.133.149	dV	dV	W	A	dV	dV	dV	dV	R	R	iV	iV	W	W	W	W → A	I	I
xii) SCN= null , SAN-IP=141.115.26.43	dV	dV	? → W	? → A	dV	dV	dV	dV	? → R	? → R	iV	iV	? → R	? → R	? → R	? → A	I	V
xiii) SCN= dane.irit.fr ,SAN-IP=141.115.26.43	dV	dV	? → A	? → A	dV	dV	dV	dV	? → W	? → R	iV	iV	? → A	? → R	? → A	? → A	V	V
xiv) SCN= null ,SAN-IP=141.115.26.43 SAN-DNS=dane.irit.fr	dV	dV	? → A	? → A	dV	dV	dV	dV	? → A	? → R	iV	iV	? → A	? → R	? → A	? → A	V	V
xv) SCN= null , SAN-DNS =null, SAN-IP=141.115.26.43	dV	dV	? → W	? → A	dV	dV	dV	dV	? → R	? → R	iV	iV	? → R	? → R	? → W	? → A	I	V

Where: ***S1*** *= sana1.fr or dane.irit.fr,* ***IP*** *= 192.168.133.149 or 141.115.26.43,* ***Na*** *= not applicable,* ***dV*** *= delegated Validation,* ***iV*** *= incorrect Validation,* ***A*** *= Accept,* ***W*** *= Warn,* ***R*** *= Refuse,* ***I*** *= Invalid certificate w.r.t standard,* ***V*** *= valid certificate w.r.t standard,* ***? →*** *= means that we didn't make this specific test in 2017.*

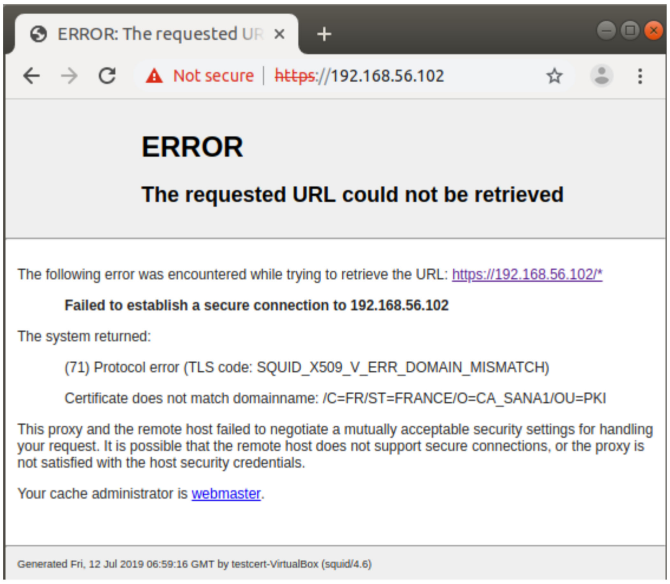
When the identity of the server is null (test iv, test x) the HTTPS interception product cannot authenticate the server, and therefore the TLS certificate is invalid. In 2017, the Squid proxy was the only product that refused the certificate. In 2019, the Mitm proxy has aligned its behaviour with Squid by rejecting this kind of certificate. All the other products under fV category keep the same behaviour as in 2017 by showing a warning to the user and asking him to take the right decision. Whether a certificate validation entity should immediately refuse an invalid certificate (R) or ask the user what to do (W) is partly a usability issue and partly a security issue. But it is not a standard's issue. The standards will only give guidance on whether a certificate is invalid or not, but will not advise a relying party what to do with it.

From a security perspective, if the HTTPS interception product cannot authenticate the web server, the certificate should be rejected (R). From a usability perspective the user could be given a choice (W), although in practice most users simply click OK to all the pop-up windows so invalid certificates end up being accepted.

In the case when the identity of the server is contained in only the SCN field without having the subjectAltName extension to hold the identity of the server (test iii), most of the interception products accept this certificate except the Squid Proxy. It is difficult to decide the validity of these certificates because on the one hand, the certificate is valid because the baseline requirements (BR) of CA/Browser forum don't prohibit the use of the SCN field to hold the identity of a server. On the other hand, the certificate is not valid because it doesn't have the subjectAltName extension, which is mandatory according to the BR requirements. This may explain the divergence in the behaviours of different products.

When the identity of the server is defined by the SAN-DNS field to hold its DNS name and the SAN-IP component to match its IP address (test viii), the Mitm proxy presented a warning message with the DNS name in 2017 although the certificate is valid. This behaviour was probably due to poor implementation of the Subject-Alt-Name Extension. In 2019, this behaviour was modified to accept the valid certificate.

When the identity of the server is defined by the SCN field to hold the DNS name and the SAN-IP component to match a reserved IP address (test vii), a diversity of behaviour is recorded with S1 access. The server's certificate is accepted by all the tested products of fV category except Squid, which presents a warning message. We obtained the same behaviour in 2017 and 2019. The related standards imply that such a certificate is valid because they don't prohibit the SCN field from containing a dNSName, and they mandate the existence of the subjectAltName extension, which is the case in our test. With an IP access, the same certificate is invalid because it contains a reserved IP address.


ERROR: The requested UR ×
Not secure | https://192.168.56.102

ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: https://192.168.56.102/*

Failed to establish a secure connection to 192.168.56.102

The system returned:

(71) Protocol error (TLS code: SQUID_X509_V_ERR_DOMAIN_MISMATCH)

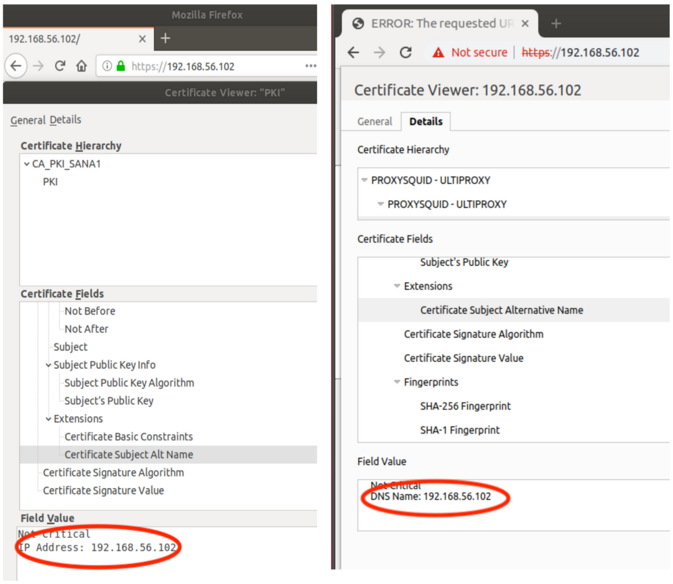
Certificate does not match domainname: /C=FR/ST=FRANCE/O=CA_SANA1/OU=PKI

This proxy and the remote host failed to negotiate a mutually acceptable security settings for handling your request. It is possible that the remote host does not support secure connections, or the proxy is not satisfied with the host security credentials.

Your cache administrator is webmaster.

Generated Fri, 12 Jul 2019 06:59:16 GMT by testcert-VirtualBox (squid/4.6)

Fig. 3. Error message from Squid Proxy.



```
Not Critical
IP Address: 192.168.56.102
```

Not Critical
DNS Name: 192.168.56.102

Fig. 4. Original server certificate and corresponding Squid generated certificate.

All the products under the fV category, except Squid and Mitm proxies, are not conformant to the standards as they accept this certificate with IP access (similarly, certificates in tests vi, vii, viii and xi are considered as invalid with a reserved IP address).

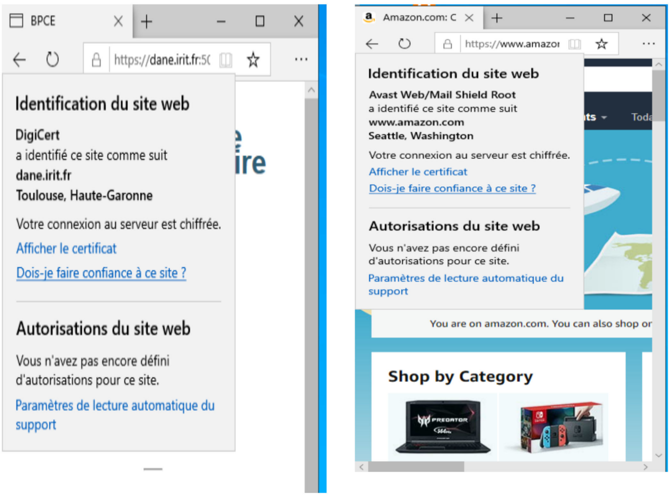
At first glance, the Mitm and Squid proxies look to be the only conformant products with regards to certificates with a reserved IP address. However, their behaviour can be interpreted differently i.e., they may not support SAN-IP. According to X.509 standard *"An implementation is not required to be able to process all name forms"*. So no HTTPS interception products have to support SAN-IP.

To understand the exact reason for the rejection of IP values by the Mitm and Squid proxies, we decided to add a new series of tests that include certificates with public IP addresses (xii, xiii, xiv, xv). The objective is to know whether the interception products give special treatment to reserved IP addresses or whether they support this kind of name form. The results of these tests show that all products show exactly the same behaviour whether the IP address is reserved or public.

With regards to the Mitm proxy, we have always obtained the same error message that indicates the absence of the SNI (Server Name Indication) extension in the TLS protocol, whenever a certificate has an IP value in its subjectAltName extension and we access the server using an IP address. SNI is a TLS extension in the Client Hello message sent by the client to inform the server which hostname it is attempting to access. The presence of SNI values in the new protocol of TLS is fundamental to proxies in order to be able to intercept TLS 1.3 communications [18]. Indeed, according to the specification of TLS 1.3 [2], the client hello message is the only message that will be sent in the clear.

With regards to the Squid proxy, the rejection of certificates with IP addresses is due to a bad implementation. Every time a certificate is used with a URL containing an IP address, we obtain the same error message, which indicates a mismatch between the hostname of the server and the value contained in the subjectAltName (Fig. 3). By inspecting the certificates generated by the Squid proxy, we can see that these certificates don't match the certificates of the tested server. In fact, as Fig. 4 shows, the Squid proxy's certificate sets the IP value in the DNS field of the subjectAltName extension instead of setting it in the IP field of this extension. This explains why we always get the same error message for every certificate with an IP address value.

Another quite confusing behaviour is that of Avast and AVG with regards to public domains and public IPs. When we executed our tests with our public domain dane.irit.fr, we realized that Avast and AVG don't intercept the communication with our public domain. However, when we tested with another public domain such as www.amazon.com, Avast and AVG have intercepted the TLS communications with these websites (Fig. 5). More strangely, the decision to intercept a TLS communication by Avast and AVG changes according to the type of web browser used by the user. For example, when the web user uses the Edge browser, Avast and AVG intercept the communication with amazon.com.



Identification du site web

DigiCert
a identifié ce site comme suit
dane.irit.fr
Toulouse, Haute-Garonne

Votre connexion au serveur est chiffrée.

Afficher le certificat

Dois-je faire confiance à ce site ?

Autorisations du site web

Vous n'avez pas encore défini d'autorisations pour ce site.

Paramètres de lecture automatique du support

Identification du site web

Avast Web/Mail Shield Root
a identifié ce site comme suit
www.amazon.com
Seattle, Washington

Votre connexion au serveur est chiffrée.

Afficher le certificat

Dois-je faire confiance à ce site ?

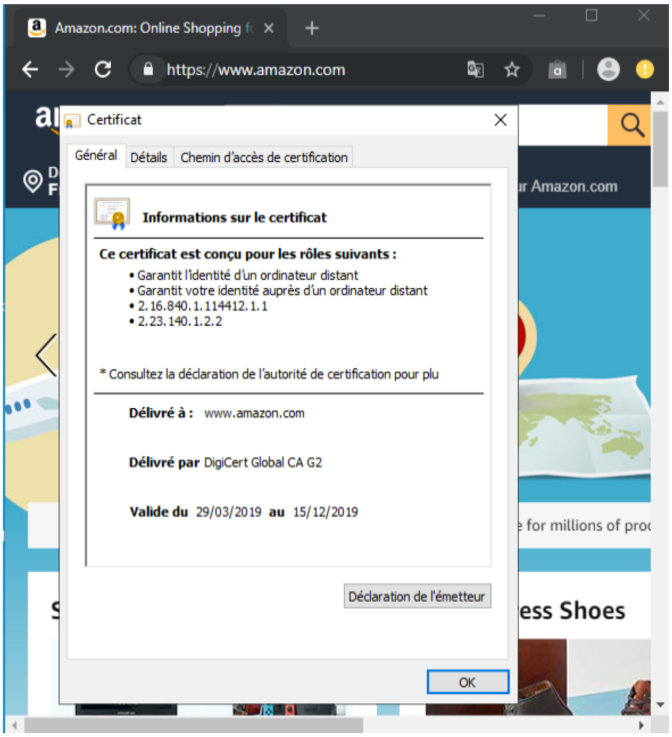
Autorisations du site web

Vous n'avez pas encore défini d'autorisations pour ce site.

Paramètres de lecture automatique du support

You are on amazon.com. You can also shop or

Fig. 5. This figure shows how Avast has intercepted the communicaiton with the domain amazon.com and not with dane.irit.fr.



Informations sur le certificat

Ce certificat est conçu pour les rôles suivants :

- Garantit l'identité d'un ordinateur distant
- Garantit votre identité auprès d'un ordinateur distant
- 2.16.840.1.114412.1.1
- 2.23.140.1.2.2

* Consultez la déclaration de l'autorité de certification pour plu

Délivré à : www.amazon.com

Délivré par DigiCert Global CA G2

Valide du 29/03/2019 **au** 15/12/2019

Fig. 6. This figure shows how Avast didn't intercept communication with amazon.com when the web user uses Chrome.

When the web user uses Firefox, Opera or Chrome, Avast and AVG don't intercept the communication with amazon.com (Fig. 6). Further research is needed to determine the interception strategy of Avast and AVG. Kaspersky and ESET have more consistent behaviour as they intercept all public domains and public IP servers.

3.2 Key Usage, Extended Key Usage

Key usage and extended key usage are used to determine the purpose of the public key contained in the certificate. A TLS server certificate could have a key usage extension or not.

3.2.1 What do the Standards State About the Key Usage and Extended Key Usage ?

The X.509 standard [3] states that if either the extended key usage or key usage extensions are recognized by the relying party then the certificate must be used only for one of the purposes stated in the both fields. The key usage and the extended key usage must be treated separately but they must have consistent values. If there is no purpose consistent with both fields, then the certificate shall not be used for any purpose [3].

RFC 5280 states that the key usage extension, when it appears, should be a critical extension. For a TLS certificate, RFC 5280 recommends that the key usage, when it is defined, should have the value of *"digital signature, key encipherment and/or key agreement"* and the consistent value of the extended key usage should be *"Server Authentication"*.

3.2.2 Tests and Results

The value needed in the key usage extension depends on the encryption algorithms used for generating the certificate's keys (RSA, DSA, DH, etc.) and on the cipher suite applied in the TLS communication between the HTTPS interception product and the web server. A cipher suite consists of a key exchange scheme, a signature algorithm, a block cipher algorithm, and a hashing algorithm for computing the authentication key. They're usually identified in a string [23] viz:

[SSL/TLS]_[key exchange]_[signature algorithm]_WITH_[block cipher]_[authentication hash]

We generated our test certificates using the RSA algorithm. In this case, two types of cipher-suites are possible:

- *TLS_ECDHE_RSA**:* in this case, the key exchange algorithm is ECDHE (Elliptc Curve with ephemeral Diffie-Hellman). This means that the RSA private key of the server's certificate will be used for signing the ECDHE public key and the associated parameters. The appropriate value of the key usage extension is digitalSignature.
- *TLS_RSA_*:* in this case the key exchange algorithm is RSA. This means that the HTTPS interception product will use the RSA public key of the server's certificate for encrypting the random value chosen by the client. The appropriate value of the key usage is keyEncipherment.

Since RSA keys can lead to different key usages, we first check the cipher suites agreed between our web server and the HTTPS interception product by looking at the Hello server message in the TLS protocol. Table 4 shows the cipher-suites chosen by the HTTPS interception product and the appropriate key usage value for each product.

We tested how the HTTPS interception product reacted when it validated a certificate, which conveyed an RSA public key and had a key usage value different from the correct value *digitalSignature* or *keyEncipherment* according to the cipher-suites used.

3.2.3 Analysis of the Results

Here, the diversity of the HTTPS interception products' behaviour is due to violations of the standards when the key usage and/or the extended key usage extension contain wrong values. Those certificates that should have been treated as invalid were treated as acceptable by most of the tested products.

In 2017, Squid accepted certificates when the key usage had wrong values of keyEncipherment instead of digitalSignature. For all others test cases, its behaviour was correct. Squid rejected invalid certificates without asking the user. In 2019, Squid gives the same behaviour with regards to the same wrong certificate. In addition, Squid now incorrectly accepts certificates that were correctly rejected in 2017, such as the certificate that has the wrong value of keyAgreement.

In 2017, all others products accepted certificates when the key usage had wrong values of dataEncipherment or keyAgreement instead of digitalSignature or keyEncipherment. In 2019, we obtain similar results, except for the Mitm proxy which changes its behaviour to reject invalid certificate whose key usage field is dataEncipherment.

In 2017, when the extended key usage had the wrong value of clientAuthentication instead of serverAuthentication,

TABLE 4
Chosen Cipher Suites

	Kaspersky	AVG & Avast	ESET	Squid	Charles	Mitm	Fiddler
Chosen cipher suite in 2017	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_128_SHA256
Valid key usage extension value for 2017 cipher suite	digitalSignature AA	keyEncipherment BB	keyEncipherment BB	digitalSignature AA	digitalSignature AA	digitalSignature AA	keyEncipherment BB
Chosen cipher suite in 2019	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
Valid key usage extension value for 2019 cipher suite	digitalSignature AA	digitalSignature AA	keyEncipherment BB	digitalSignature AA	digitalSignature AA	keyEncipherment BB	digitalSignature AA

where: cipher suite written in red to indicate the change of cipher suite for the same product between 2017 and 2019, ***AA*** *= refers to products that should have digitalSignature value in their key usage extension,* ***BB*** *= refers to products that should have keyEncipherment value in their key usage extension,* ***GCM*** *= Galois Counter Mode,* ***CBC*** *= Cipher Block Chaining.*

Kaspersky and Squid proxy rejected the certificate whilst Fiddler and Mitmproxy treated the certificate as valid and accepted access to the website. By 2019, the Fiddler and Mitm proxies had changed their behaviour to conform to the relevant standards.

It should be noted that in many cases, we have obtained the same results when presenting a test certificate to two different versions of the same product. For example, when presenting an invalid certificate to the Mitm product, the certificate was accepted in 2017 and 2019. However, in 2017 Mitm was using TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256 and in 2019 Mitm uses a different cipher suite. As a consequence, the (A)ccept behaviour in 2017 for a certificate that has KE in its Key usage was considered non-conformant to the standard, but in 2019 the same behaviour was considered conformant (A→A).

3.3 Revocation

Certificate revocation is one of the many challenges faced by PKIs. It is the action of declaring a certificate as invalid and no longer trusted before its scheduled expiration date.

The security of the PKI depends on our ability to revoke lost, stolen, or compromised certificates from circulation as quickly as possible. This is usually done by asking the relying party to check the certificate's status before accepting it. Consequently, CAs must make all revocation information available. They can revoke a PKC by publishing its serial number in a Certificate Revocation List (CRL) that can be downloaded from a repository. However, CRLs might become very large, resulting in an unacceptable latency. The second approach is the Online Certificate Status Protocol (OCSP). A CA can indicate that a PKC has been revoked by running an OCSP server to which a client submits an OCSP request. The server responds with the status of the PKC.

OCSP has some limitations, such as the privacy of clients, as it gives the OCSP server a lot of information about which PKCs are being used where. Another severe problem is the availability of the OCSP server for under resourced CA infrastructures. High traffic websites can result in a large number of requests being sent to the OCSP servers. As a consequence, some clients are not able to make contact and obtain an OCSP response, and so no PKC revocation information is delivered.

In this case and from a security point of view, such a PKC should be considered as invalid. However, practically all clients implement OCSP in *soft fail* mode, meaning that if the client receives no positive response (good or revoked), then the PKC will be considered as good and the client will allow access to the associated web content. This problem makes the whole OCSP concept vulnerable: if an attacker tries to use a revoked certificate it can simply block connections to the OCSP server (e.g., a DDOS attack).

CrlDistributionPoints (CDP) and AuthorityInfoAccess (AIA) extensions are used to hold the CRL and the OCSP indicators respectively in a PKC. They tell the RP where it can fetch CRLs or OCSP responses from, respectively.

OCSP Stapling and Must-Staple are new alternatives to OCSP for checking a PKC's status. For this reason, we start by giving a brief description of them. OCSP Stapling is described in RFC 6066 (for checking the status for server certificates) and RFC 6961 (for checking the status of every certificate on the chain). Must Staple is described in RFC 7633.

OCSP Stapling eliminates the need for the client to request an OCSP response directly from the CA's server. As shown in Fig. 7, the web server makes the OCSP request and then caches the response. This allows the web server to staple the OCSP response within the TLS handshake via the Certificate Status Request extension.

This approach offers three main advantages. First, it reduces the costs for CAs because the number of OCSP request is significantly reduced, coming only from web sites. Secondly, it improves the privacy of clients because CAs cannot identify the web sites that users are visiting and thirdly, it improves the performance of clients as a second connection to an OCSP server does not need to be established. However, this approach does not resolve the problem of single point of failure and DDOS attacks. An attacker can still attack the OCSP servers of a CA to prevent web servers from fetching new OCSP responses. As a consequence, access to these web sites would be authorized with a soft-fail policy.

TABLE 5
Key Usage Test

Key Usage	Standard Validity	Avast	Kaspersky	AVG	ESET	Squid	Charles	Mitm	Fiddler
key usage extension = KA No Extended key usage extension	I	dV	A	dV	dV	R ➔ A	iV	A	A
key usage extension = DE No Extended key usage extension	I	dV	A	dV	dV	R	iV	A ➔ R	A
key usage extension = KE No Extended key usage extension	I (AA Products) V(BB Products)	dV	A	dV	dV	A	iV	A ➔ A	A ➔ A
key usage extension = DS No Extended key usage extension	I (BB Products) V(AA Products)	dV	A	dV	dV	A	iV	A ➔ A	A ➔ A
key usage extension = KA key Extended usage extension =SA	I	dV	A	dV	dV	R ➔ A	iV	A	A
key usage extension = DE key Extended usage extension =SA	I	dV	A	dV	dV	R	iV	A ➔ R	A
key usage extension = KE key Extended usage extension =SA	I (AA Products) V(BB Products)	dV	A	dV	dV	A	iV	A ➔ A	A ➔ A
key usage extension = DS key Extended usage extension =SA	I (BB Products) V(AA Products)	dV	A	dV	dV	A	iV	A ➔ A	A ➔ A
No key usage extension key Extended usage extension =CA	I	dV	W	dV	dV	R	iV	A ➔ R	A ➔ W
key usage extension = DS key Extended usage extension =CA	I	dV	W	dV	dV	R	iV	A ➔ R	A ➔ W

Where: ***CA*** *= clientAuthentication,* ***DE*** *= dataEncipherment,* ***DS*** *= digitalSignature,* ***KA*** *= keyAgreement,* ***KE*** *= keyEncipherment,* ***dV*** *= delegated Validation,* ***iV*** *= incorrect Validation,* ***SA*** *= serverAuthentication,* ***I*** *= invalid w.r.t standard,* ***V*** *= valid w.r.t standard,* ***A*** *= Accept,* ***W*** *= Warn,* ***R*** *= Refuse,* ***I (AA Products)*** *= means the certificate is invalid for AA products,* ***V(AA Products)*** *= means the certificate is valid for AA products.*

A new certificate extension called Must Staple has been defined to require OCSP Stapling in the TLS handshake. CAs issue certificates to web servers with this new extension, and this requires the web server to send a cached OCSP response along with its server certificate to the client (RP). Clients should ensure this stapled OCSP response is present otherwise they should hard-fail the TLS connection.

3.3.1 What do the Standards State About the CRL Distribution Points, Authority Info Access and The TLS Status_Request Extensions?

The X.509 standard states that the CDP extension can be, at the option of the certificate issuer, critical or not; but it recommends it to be non-critical for interoperability reasons. When it is a critical extension, a certificate-using system shall not use the certificate without first retrieving and checking the certificate against the downloaded CRL [3]. However, when the extension is not critical a certificate-using system can use the certificate only if the revocation checking is not required by a local policy or it is accomplished by other means [3].

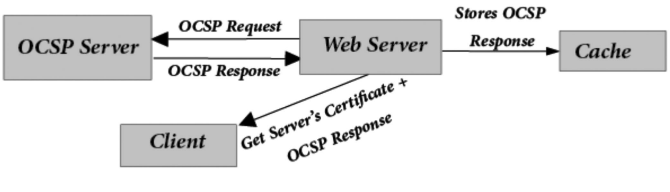


Fig. 7. The OCSP Stapling Approach.

According to RFC 5280, the CDP and AIA extensions should be non-critical extensions, but it recommends supporting these extensions by CAs and applications [4].

Starting from 1 January 2013, the CA/Browser forum imposed the use of the OCSP protocol. However, the CA/Browser forum also allowed the use of other checking methods such as CRLs and OCSP stapling. With regards to OCSP stapling, BR states *"If the Subscriber Certificate is for a high-traffic FQDN, the CA MAY rely on stapling, in accordance with [RFC4366], to distribute its OCSP responses. In this case, the CA SHALL ensure that the Subscriber "staples" the OCSP response for the Certificate in its TLS handshake. The CA SHALL enforce this requirement on the Subscriber either contractually, through the Subscriber Agreement or Terms of Use, or by technical review measures implemented by the CA."*. Note that RFC4366 has been obsoleted by RFC 6066. According to RFCs 6961, 6066 and 7633, the only TLS feature extensions that are relevant to the revocation status are the Certificate Status Request extension (status_request) and the Multiple Certificate Status Extension (status_request_v2). These extensions should not be marked critical. Marking the TLS feature extension critical breaks backward compatibility and is not recommended unless this is the desired behavior.

3.3.2 OCSP-Stapling Survey

In 2013, Netcraft [5] performed a survey that indicated that around 22 percent of certificates were served with a stapled OCSP response. However, to the best of our knowledge

TABLE 6
Error Types

	SSL Error	Rejected	Unreachable
2018	26%	27%	46%
2019	39%	24%	37%

there has been no survey since then. So we decided to undertake our own to see whether there has been any change since 2013.

a) Dataset and methodology

We implemented a Java program in order to detect whether a web server supports OCSP Stapling or not. The program doesn't check all available web domains. Instead it checks the top one million websites that we obtained from alexa.com. For each domain, the program establishes a TLS connection and notifies the server that an OCSP Stapling response is needed (by adding the TLS certificate status request extension during the Handshake phase). Our objective is to know whether the server supports: OCSP-Stapling, Must-Staple via a certificate extension or Must-Staple via the HTTP header.

We ran our program two times: the first time was on March 3, 2018 and the second time was on May 28, 2019. This can be useful to understand the evolution of OCSP-Stapling adoption. Our program ran on a 32-cores architecture (2 Intel Xeon processors with 64GB RAM). It used all the cores to run the tests. The total duration of the 2018 survey to process all the 999,950 websites was 24 hours, 29 minutes and 15 seconds whereas the total duration of the 2019 survey was 22 hours, 59 minutes and 54 seconds.

b) Survey Results

In 2018, our program was able to test only 735,320 web domains from the 999,950 websites whereas in 2019, our program was able to test 828,777 web domains from the 999,950 websites. We were not able to test all domains because of different types of errors. For example, some errors were due to bad configuration of TLS. Table 6 shows the division of error types in the 2018 and 2019 surveys. *Unreachable* means the server did not answer any request (either from OpenSSL or the HTTP client). This means that either the website is no longer online, or the server does not listen to port 443. *Rejected* means the server answered back, but rejected the TCP connection and sent an ICMP message. Finally, *TLS Error* means that the server answered back on port 443 but the TLS handshake failed. This study does not detail the reasons for the handshake failure, but for instance, some of the servers were serving HTTP rather than HTTPS on port 443, whilst others were using a deprecated version of SSL, and others provided invalid certificates, etc.

Of the 735,320 tested web domains in 2018, we found that only 141,541 (19.25 percent) supported OCSP-Stapling. However, in 2019 we found that 221733 of the 828777 tested web domains (26.75 percent) supported OCSP-Stapling.

Table 7 shows that the proportion of OCSP-Stapling servers that support Must-Staple (by adding the Must-Staple extension into their X.509 PKC) was tiny in 2018, but no website is supporting the Must-Staple option in 2019.

The third columns of Table 7 shows that the proportion of OCSP-Stapling servers that support the Must-Staple HTTP Header is even smaller (only 0.00007 percent), but we got the same results in 2018 and 2019.

Our analysis shows a significant rise in the use of OCSP-Stapling by Cloudflare (Content Delivery Network provider) where in 2018 only 23 percent of Cloudflare responses were supporting OCSP-Stapling, while in 2019 >80 percent of Cloudflare responses supported OCSP-stapling.

3.3.3 Tests and Results

We performed two sets of experiments. The first set related to OCSP-Stapling support in web browsers. We checked if the OCSP Stapling and Must Staple approaches were supported, and if they were automatically implemented or not. In 2017, we tested the popular web browsers: Internet Explorer 11(IE11), Firefox 52 (FF52), Opera 44 (OP44), Microsoft Edge 25 (ED25), Google Chrome 57 (GC57). In 2019, we tested Firefox 68 (FF68), Chrome 75 (GC75), Edge 44 (ED44), Opera 62 (OP62) and Internet Explorer 11 (IE11). The first step was to enable and configure OCSP Stapling in our Apache web server, which has been supported since Apache HTTPD Server 2.3.3+.

In tests i) and ii) in Table 8, we show the reaction of web browsers when the server's stapled OCSP response indicates that the server's certificate status is good or revoked, and in test iii) when the stapled OCSP response is not present. We obtained exactly the same results in 2017 and in 2019. Table 8 shows the browsers that support Must Staple. The results that we obtain in test iii in Table 8 are the same whether Must Staple is activated or not.

In the second set of experiments, we tested the HTTPS interception products. We first determined the revocation methods supported by each product, and whether they were automatically configured or not (Table 9 i-iv)). We then show the reaction of the HTTPS interception product when:

- the CRL is not retrievable or the OCSP server is down (Table 9 v) & vi));
- the HTTP methods supported to fetch an OCSP response (Table 9 vii));
- the OCSP stapled response indicates that the server's certificate is revoked (Table 9 viii));
- there is no stapled OCSP response, with OCSP Stapling and Must Staple (Table 9 ix) & x)).

3.3.4 Analysis of the Results

The overall TLS system suffers from two major problems: the first problem is related to the trustworthiness of CAs [19], [20], the second problem is with regards circumstances under which clients should check whether server certificates are revoked or not.

The success or failure of the OCSP Stapling check depends on the implementations of both the web browser and the web server. The web browser only obtains an OCSP stapled response from the web server if the browser asks for it and the server supports it. If either the browser or the web server do not support OCSP Stapling, then OCSP Stapling is

TABLE 7
OCSP Staple and Must-Staple Support

	OCSP Stapling support	Must-Staple certificate extension support	Must-Staple HTTP Header Support
2018	19%	0,04% (58 websites)	1 website
2019	27%	0%	1 website

TABLE 8
Web Browser's Revocation Tests

	OP44, OP62	FF52, FF68	GC57, GC75	IE11	ED25, ED44
i) Certificate is good in the stapled OCSP Response	A	A	A	A	A
ii) Certificate is revoked in the stapled OCSP Response	R	R	R	R	R
iii) There is no stapled OCSP Response available (try later)	A	R	A	A	A

Where: ***A*** *= Accept,* ***W*** *= Warn,* ***R*** *= Refuse.*

not used and certificate validity status checking will automatically revert to the other revocation approaches supported by the browser. Unfortunately, most of the HTTP interception products do not support OCSP Stapling (Table 9 iii)) and none support Must Staple, even though many web sites are taking advantage of OCSP Stapling. Our own analysis shows a rise of OCSP-Stapling adoption. In 2019 26,25 percent of tested websites support OCSP stapling whereas in 2018 19.25 percent of tested websites support OCSP-stapling. However, the support of the Must Staple extension in their PKCs has totally disappeared in 2019.

For the tested web browsers, all of them support OCSP Stapling but not the Must Staple extension (Table 10). OCSP Stapling is configured and enabled by default in all of them. If a stapled OCSP response (good or revoked) is present in the TLS handshake message, all the web browsers behave correctly. They will refuse the TLS connection with a revoked certificate and accept it with a valid one (Table 8 i) & ii)). However, when no OCSP response is stapled (Table 8 iii), whether to check for OCSP Stapling is set to mandatory or optional, all the web browsers except Firefox treat the certificate as valid.

The differences in the behaviours between web browsers is however conformant to the standard RFC6961, which states that "*If the OCSP response received from the server does not result in a definite "good" or "revoked" status, it is inconclusive. A TLS client in such a case MAY check the validity of the server certificate through other means, e.g., by directly querying the certificate issuer. If such processing still results in an inconclusive response, then the application using the TLS connection will have to decide whether to close the connection or not. Note that this problem cannot be decided by the generic TLS client code without information from the application. If the application doesn't provide any such information, then the client MUST abort the connection, since the server certificate has not been sufficiently validated.* "

The acceptance of certificates with unknown revocation status is due to the preferred soft-fail policy of browsers. The reasoning behind this is that the lack of an OCSP response could be due as much to a network error or mal configuration as to malicious activity. Also, according to Adam Langley from Google, web browsers apply this policy because they consider that hard-failing raises a different security issue by creating a single point of failure paving the way for effective DDOS attacks.

Firefox's behaviour is explained by its support for OCSP Must Staple by default. This provides stronger revocation

TABLE 9
HTTPS Interception Products Revocation Tests

	Avast	Kaspersky	AVG	ESET	Squid	Charles	Mitm	fiddler
i) CRL checking	Automatic	Not supported	Automatic	Automatic	Not supported	Not supported	Not supported	Not supported
ii) OCSP checking	Not Supported	Automatic	Not supported	Automatic	Not supported	Not supported	Not supported	Configurable
iii) OCSP Stapling checking	Automatic	Not supported	Automatic	Not supported	Not supported	Not supported	Not supported	Not supported
iv) OCSP Must Staple checking	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Where: Configurable means that the HTTPS interception product checks the certificate status after setting, but by default it does not do it. Automatic means that the product checks the certificate status automatically.								
v) CRL is not retrieved	R➔A	NA	R➔A	R➔A	NA	NA	NA	NA
vi) OCSP server is down	NA	A	NA	A	NA	NA	NA	R➔W
vii) OCSP request HTTP methods	NA	GET only	NA	GET only	NA	NA	NA	GET only
Where: GET only means that the HTTPS interception product supports only the GET method, if it fails the HTTPS interception product will not send a POST request. NA means not applicable								
viii) Certificate is revoked in the stapled OCSP Response	R	NA	R	NA	NA	NA	NA	NA
ix) No stapled OCSP response with OCSP Stapling	R➔A	NA	R➔A	NA	NA	NA	NA	NA
x) No stapled OCSP response with OCSP Stapling and Must Staple	NA	NA	NA	NA	NA	NA	NA	NA

Where: ***NA*** *= not applicable,* ***A*** *= Accept,* ***W*** *= Warn,* ***R*** *= Refuse.*

TABLE 10
Revocation Approaches Supported by Each Web Browser

	CRLs	OCSP	OCSP Stapling	OCSP Must Stapling
GC57,GC75	NS	NS	S	NS
IE11	S	S	S	NS
ED25,ED44	S	S	S	NS
OP44,OP62	NS	NS	S	NS
FF52,FF68	NS	S	S	S

***Where:** **NS** = means NOT supported, **S** = means supported.*

checking with its requirement to ensure a stapled OCSP response is in the TLS handshake. No other browsers currently support this. Table 10 shows which revocation approaches are supported by each web browser.

Chrome supports OCSP Stapling in addition to its own CRLSets method of checking for a revoked certificate [21]. The basic idea of CRLSets is that Google merges all the CRLs of all the existing CAs and reduces the obtained list by removing PKCs that it considers unimportant. The result is a minimal CRL list that is periodically pushed to Google Chrome. OneCRL is a similar method used by Firefox [22].

In 2017, Mozilla announced that Firefox will disable OCSP checking for Domain validated (DV) and Organization validated (OV) certificates because of performance concerns, but it will continue to fetch OCSP response for extended validated (EV) certificates. However, in 2019 we found that Mozilla Firefox still supports OCSP checking. We believe that keeping support for OCSP is vital for Internet security because as we said earlier only 26.25 percent (according to our 2019 survey) of websites currently support OCSP-Stapling.

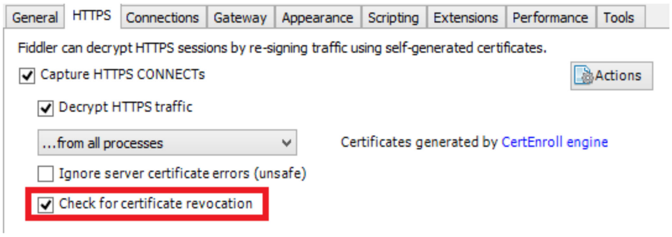
As noted earlier, web servers should implement OCSP Stapling correctly. For some servers, extra-configuration is required by the website administrator to enable it correctly. This is not the case for CRLs and OCSP, which only involve the CA and the browser.

It is reasonable to also check the implementation of OCSP Stapling in web servers. According to Google developer Ryan Sleevi [6], there are several requirements for a proper OCSP Stapling implementation.

First, the implementation of OCSP Stapling should be 'on' by default without the intervention of the website administrator. Apache is not compliant with this requirement; enabling OCSP stapling is only supported in Apache2.4+ by the addition of specific configuration directives, which can be a complex and delicate task in a shared system.

Secondly, the web server should support a long-lived Stapling cache. This means that any restarting of the web server should not remove any OCSP responses previously obtained. An OCSP response should be cached until either the server gets a new one or it expires. For Apache, cached OCSP responses do not persist across server restarts, because they are only kept in a short-lived memory cache. We noted also that Apache fetches its OCSP responses during the handshakes of the first connections instead of doing it on start-up. Thus an extra latency is recorded in this case.

Thirdly, the web server should avoid a situation where it is unable to send out a valid OCSP response. Therefore, it should refresh an old response in sufficient time before its expiration. It is preferable to start to fetch a response halfway through its validity period i.e., "not Before + (not After - not Before) / 2" in order to handle non-deterministic situations ("try later" or "internal error"). Moreover, the web server should never throw away a valid response until it has a newer one. Apache does not do this. If the OCSP server is unavailable, and Apache is unable to renew the OCSP response, it still throws away the existing valid response, meaning it cannot then send out a stapled OCSP response. Many problems have been reported regarding this behavior [24].



Fiddler can decrypt HTTPS sessions by re-signing traffic using self-generated certificates.

☑ Capture HTTPS CONNECTs

☑ Decrypt HTTPS traffic

...from all processes

Certificates generated by CertEnroll engine

□ Ignore server certificate errors (unsafe)

☑ Check for certificate revocation

Fig. 8. Revocation setting for fiddler proxy.

These requirements seem rather basic, but they necessitate the re-engineering of Apache's OCSP Stapling implementation in order to make it more robust and reliable.

All the HTTPS interception products provide less than optimum support for revocation checking, despite its critical importance for securing the integrity of the Internet's PKIs. All the anti-virus products support at least one automatic revocation method, whilst none of the proxies do.

Maintaining a revocation service (either CRLs or OCSP) is a requirement for CAs. The standards also recommend, but do not mandate, that relying parties should ensure that certificates are not revoked before they rely on them. When the AIA and CDP extensions are present and understood by the relying parties, they are required to process them.

Mitmproxy, Squid and Charles do not fetch any revocation information to check before accepting a certificate, which means they do not understand these extensions. All certificates are treated as valid by them, even after being revoked. For Squid, the web administrator has the possibility to implement revocation via a specific command.

The fiddler proxy has implemented OCSP checking but automatic certificate status verification is not enabled by default. This may allow the use of revoked certificates without the users being aware of it. Fig. 8 shows the extra option for fiddler to ensure revocation checking.

Three HTTPS interception products support CRL checking (Avast, AVG, ESET). When they cannot fetch the CRL (Table 9 v)), all of them were fully conformant in 2017 as none accepted the PKC. However, in 2019, all of them changed their behaviours to accept the certificate when the CRL was not retrievable.

Three products support OCSP (Kaspersky, ESET and fiddler). RFC 6960 states *"the OCSP client suspends acceptance of the certificate in question until the responder provides a response"*. In the 'OCSP down' test (Table 9 vi)), the responses provided by Kaspersky and ESET are not compliant to the standard, only fiddler was rejecting the certificate in 2017 but changed this behaviour to show a warning message in 2019.

RFC 6960 requires OCSP requests to be sent using either the GET or POST methods. The three HTTPS interception products respect this issue. However, some OCSP servers may support only one OCSP request method (POST or GET). Our OCSP test server only supports the POST method. Our tests (Table 9 viii)) show that all tested products support only the GET method. RFC 6960 should clarify this issue by mandating OCSP servers to support both methods.

Between 2017 and 2019, we didn't find any evolution with regards to the support of OCSP stapling by the interception products. Indeed, only two HTTPS interception products support OCSP Stapling (Avast, AVG). Both products do not trust a revoked PKC that appears in an OCSP stapled handshake. However, when the OCSP stapled response is absent, both interception products were rejecting the PKC in 2017 but accepting it in 2019. In other word, both products hard-failed the connection when no OCSP stapled response was available in 2017 but preferred the soft-fail policy in 2019.

4 Discussion

In 2009 we were among the first to raise the problem of X.509 certificate validation in browsers [8]. Almost ten years later, whilst many of the original issues have been resolved, others still persist and new ones have been introduced.

Since 2007, the web PKI industry has made a lot of positive advancements by improving the quality of certificate issuance. Today, commercial CAs have less freedom than before, and their certificate issuing processes are regulated by a set of standards issued by the CA/Browser forum [13], [16]. One of the important newer mechanisms for monitoring a CA's performance is Certificate Transparency (CT), which was introduced by Google in 2012 [12], [17]. This system was conceived as a result of several attacks against the TLS ecosystem, including the issuing of fraudulent Google certificates. The root causes of these attacks were either a CA's negligence or an abuse of the trust placed in the CA. CT can be seen as a global public log to which all CAs are forced to record all their issued certificates. In this way, any fraudulent certificate can be detected and removed very quickly. The CT log is hosted on different synchronized servers. In the end, it is planned that all web browsers will verify all received server certificates against this log and will block connections if a server's certificate is not present in the CT log.

Thus, we may conclude that the term trusted third party (TTP) that has historically been given to CAs, is no longer valid because we usually don't have to monitor people that we trust. Ronald Regan's famous phrase '*Trust but verify*' is more appropriate to CAs today. Several different stories show how CAs are not considered to be TTPs anymore. For example, Symantec, one of the largest CAs in the world, has decided to sell its SSL unit to Digicert after a dispute with Google [10] who detected that Symantec didn't respect the requirements of the CA/Browser forum [13].

However, end-to-end security doesn't necessitate controlling only the issuing process, but also the validation process. Our different studies from 2009 until now show the inconsistencies and dangerous behaviors in the validation processes of different types of PKI client (web browsers and HTTPS interception products). This is why we proposed in our previous research work to introduce a new entity, the trust broker, into the X.509 trust model [19], [20]. The role of the trust broker is to help web users decide whether X.509 certificates are trustworthy and valid.

By inspecting the current practices regarding certificate status verification, we notice that key players in the web PKI industry are leaning towards the abandonment of OCSP, with OCSP Stapling being only deployed by 26.25 percent of web servers. Surprisingly, results obtained by our survey are slightly better than those obtained in 2018 (19.25 percent). This can be accounted for by the increasing support of OCSP-Stapling by Cloudflare. Indeed, more than 80 percent of HTTPs requests with Cloudflare get an OCSP-Stapling Response, whereas in 2018 only around 23 percent of HTTPS requests got an OCSP stapling response. In addition, the market share of Cloudflare has grown between 2018 and 2019. In 2018, Cloudflare was serving 11.8 percent of the tested web domains whereas in 2019 it served 15.2 percent of the tested web domains. It is of note that some high profile web sites such as google.com and youtube.com are not using OCSP-Stapling, even though their certificates are not EV ones (which ironically means that Google Chrome may not issue OCSP requests to verify their status). Finally, we detected in 2018 some inconsistencies in the same organization; for example www.yahoo.com is applying OCSP-stapling whereas yahoo.co.jp is not applying it.

Another interesting fact from our 2019 survey shows that the adoption of the Must-Staple option by adding the Must-Staple extension into their X.509 PKC has totally disappeared and that only one website in our survey deploys the Must-Staple mechanism using the HTTP header. We think that the Must-Staple proposal was not adopted because either the web server must ask the CA to issue a certificate that includes this optional feature, or because the HTTP header solution is insecure and suffers from the first-visit problem. A better solution might be to create a platform that allows websites to tell web browsers whether they prefer the Soft-fail or Hard-fail policy. A similar kind of platform already exists for the HTTP Strict Transport Security mechanism (HSTS). Google has created the platform https://hstspreload.org that allows the administrators of websites to declare whether they want to serve their contents exclusively via HTTPS. Today, all major web browsers share this list. Similarly, website administrators could use this kind of platform to inform web browsers whether they should apply a hard-fail or soft-fail policy.

Our 2017 study [1] showed some other validation issues related to web browsers. For example, Safari 10 always gave a warning message regardless of the seriousness of the certificate validation error. Even when a server's certificate was revoked, the web user had the possibility of proceeding to the web site.

The current study has shown that the certificate validation performed by HTTPS interception products are even worse than that performed by web browsers. The results confirmed our expectations and show that these products present inconsistent behaviors. The difficulty with TLS interception products is that they have to combine the security measures of Web browsers and servers. However, as we saw in our earlier studies, web browsers handle certificate validation subject inconsistently.

This situation should not be an excuse for TLS interception products because they failed doing very basic validation. Adam Langley from Google describes how the proxies' misbehaviors have delayed the deployment of TLS 1.3 by one year [11]. He added in his article one sentence that chimes with our work *"I'll briefly mention the fact that HTTPS proxies aren't always so great at performing cryptographic checks. (We recently notified a major proxy vendor that their product didn't appear to validate certificates at all. We were informed that they can validate certificates; it's just disabled by default. It's unclear what fraction of their customers are aware of that.)"* [11].

5 Conclusions

The objective of our work is to show why the validation of X.509 certificates is a complex issue. Specifically, we presented two important findings:

1. The validation of X.509 certificates is highly neglected by all types of TLS interception products. When they do not verify whether servers' certificates are revoked or not can be very dangerous for web users. Our study tested the same products over a period of two years and showed that no significant improvements had been made by these products during this time.
2. None of the existing revocation checking techniques work consistently or effectively today, for different reasons. Even the latest technique, OCSP-Stapling, is far from being applied everywhere (only around 27 percent of Web servers in 2019). Unfortunately, this technique depends on the web server's administrator, who either may not be aware of it or does not have the motivation to deploy it.

The reasons behind the inconsistent behaviour are: (1) the standards are complex or vague or allow different implementations for different contexts of use, (2) there are a multitude of standards (~50) that handle validation and revocation, (3) the products are perhaps more concerned about their performance than the security of their web users (e.g., by removing OCSP checking), and (4) the absence of a viable technique for addressing validation failures.

The validation situation today resembles the certificate generation situation in the mid-2000s where the procedures with regard to X.509 certificate generation were not of good quality. This situation has largely improved after the intervention of the CA/Browser forum, which published a set of minimum requirements for EV and DV certificates. However, our work shows that web PKI still has some way to go before it reaches a consistent and effective approach to the validation of X.509 certificates. Clarifying existing PKI standards or introducing new PKI standards won't necessary solve this problem. This is because the PKI standards in general only give guidance on whether a certificate is invalid or not, but do not mandate what an RP should do with it. This problem should be somewhat easier to solve in web browsers than in interception products because the major web browser vendors coordinate through the CA/Browser forum. But clearly this is not effective today. So we propose one possible solution. Today, the validation APIs that do exist are quite complex and ask for different parameters that are not always understood by software developers. Publishing a standard PKI validation API, with clearly defined parameters, that all browsers and interception products implement, would go some way towards solving this problem. Supplementing this with a conformance test suite would also help product developers ensure that their products conform to the API.

For future work, we would like to test the validation of PKCs by Content Delivery Network (CDN) providers such as Cloudflare [15]. Today many websites mandate CDN providers cache their contents so that end users can fetch web pages from the CDN infrastructure instead of directly from the websites. This offers a lot of advantages to websites such as saving bandwidth costs, security protection, service availability, etc. CDN providers act in this case as online proxy servers that can intercept all the traffic of end users. End users can not figure out whether the web pages were brought from the CDN infrastructure or from the original web server. It would be interesting to make a detailed study about this kind of online proxy to show how they handle the validation of the servers' certificates, and whether they still deliver content from web sites with revoked certificates.

References

[1] 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.*, 2017, pp. 665–674. doi: 10.1109/COMPSAC.2017.240.

[2] E. Rescorla Mozilla. The Transport Layer Security (TLS) Protocol Version 1.3, Aug. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446

[3] ITU-T Recommendation X.509 | ISO/IEC 9594-8: 2016. *Information Technology Open Systems Interconnection- The Diretory: Public-key and Attribute Certificate Frameworks*.

[4] Cooper, NIST, Santesson, Microsoft, Farrell, Trinity College Dublin, Boeyen, Entrust, Housley, Vigil Security, Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, May 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280

[5] Statistics of Netscraft, 2013. Accessed: Apr. 18, 2020. [Online]. Available: https://news.netcraft.com/archives/2013/07/19/microsoft-achieves-world-domination-in-ocsp-stapling.html.

[6] R. Sleevi, OCSP Stapling, 2015. Accessed: Apr. 18, 2020, [Online]. Available: https://gist.github.com/sleevi/5efe9ef98961ecfb4da8

[7] X. de Carné de Carnavalet, and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–17.

[8] A. S. Wazan, R. Laborde, D. W. Chadwick, F. Barrere, and A. Benzekri, "Which Web browsers process SSL certificates in a standardized way?" in *Emerging Challenges for Secur., Privacy and Trust*, 2009. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-01244-0_38#citeas

[9] A. Jøsang, I. G. Pedersen, and D. Povey, "PKI seeks a trusting relationship," in *Proc. Inf. Secur. Privacy*, 2000, pp. 191–205.

[10] Chrome's Plan to Distrust Symantec Certificates, Accessed: Apr. 18, 2020, Sep. 11, 2017. [Online]. Available: https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html,

[11] Adam Langley. TLS 1.3 and Proxies, Accessed: Apr. 18, 2020, 10 Mar 2018. [Online]. Available: https://www.imperialviolet.org/2018/03/10/tls13.html

[12] Certificate Transparency Project, 2015. Accessed: Apr. 18, 2020. [Online]. Available: https://www.certificate-transparency.org

[13] CA/Browser forum website, 2005. Accessed: Apr. 18, 2020. [Online]. Available: https://cabforum.org

[14] 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*, 2014, pp. 114–129.

[15] End-to-end HTTPS with Cloudflare - Part 3: SSL options, Accessed: Apr. 18, 2020. [Online]. Available: https://support.cloudflare.com/hc/en-us/articles/200170416-End-to-end-HTTPS-with-Cloudflare-Part-3-SSL-options

[16] CA/Browser forum, Baseline Requirements, v. 1.5.5, 2017. Accessed: Apr. 18, 2020. [Online]. Available: https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.5.5.pdf

[17] E. Stark *et al.*, "Does certificate transparency break the Web? Measuring adoption and errorrate," in *Proc. 40th IEEE Symp. Security Privacy*, 2019.

[18] F. Andreasen., N. Cam-Winget, E. Wang, Cisco Systems. TLS 1.3 Impact on network-based security, internet-draft, Dec. 29, 2018. [Online]. Available: https://tools.ietf.org/id/draft-camwinget-tls-use-cases-03.html

[19] A. S. Wazan *et al.*, "Trust Management for public key infrastructures: implementing the X.509 trust broker," in *Security and Communication Networks*, Hoboken, NJ, USA: Wiley, vol. 2017, 2017.

[20] A. S. Wazan, R. Laborde, F. Barrère, and A. Benzekri, "A formal model of trust for calculating the quality of X.509 certificate," in *Security and Communication Networks*, Hoboken, NJ, USA: Wiley, vol. 4, no. 6, pp. 651–665, Jun. 2011.

[21] CRLSets, 2012. Accessed: Apr. 18, 2020. [Online]. Available: https://dev.chromium.org/Home/chromium-security/crlsets

[22] OneCRL, 2015. Accessed: Apr. 18, 2020. [Online]. Available: https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/

[23] Adam Bard. On secure SSL: The least every developer should know, 2013. Accessed: Apr. 18, 2020. [Online]. Available: https://adambard.com/blog/the-new-ssl-basics/

[24] 2014. Accessed: Apr. 18, 2020[Online]. Available: https://bz.apache.org/bugzilla/show_bug.cgi?id=57121.

[25] 2012. Accessed: Apr. 18, 2020. https://cabforum.org/wp-content/uploads/Guidance-Deprecated-Internal-Names.pdf

Ahmad Samer Wazan is an associate professor with Zayed University. Between 2014 and 2019, He was an associate professor at the University of Toulouse and a member of the SIERA research group at IRIT Laboratory. His research topics include trust management, PKIs, access control, OS security and security requirement engineering. Between 2007 and 2011, he led a research project that defined a new trust model for X.509 standard, by adding a new entity called Trust Broker. This is now included in the 2016 edition of the X.509 standard. He has also participated with other researchers from UK and France in the implementation of the first proof of concept verifiable credential system. Very recently, he conducted a new research project that proposed a new command called *sr* (switch role) that intends to replace the command *sudo* in Linux environment . For more information please visit https://github.com/SamerW/RootAsRole

Romain Laborde received the PhD degree in computer science from University Paul Sabatier, in 2005, and the 'Habilitation à Diriger des Recherches', in 2016. He is an associate professor with the University of Toulouse (Paul Sabatier-IUT 'A'), France since 2006. He is also a member of the Institut de Recherche en Informatique de Toulouse. He is the IUT A Computer Science Department chief, since July 2019. Previously, he was a research associate with the Information Systems Security Group with the Computer Science Department, University of Kent at Canterbury, UK. His research focuses on security management applied to security requirements, network security configuration, identity and access management and privacy. He participated in the research projects in the context of aeronautic, IoT, SME, etc.

David Chadwick received the BSc and PhD degrees. He was a professor of Information Systems Security at the University of Kent for 15 years, and is currently an emeritus professor and CEO of Verifiable Credentials Ltd. He has published widely, with m 200 publications in books, international journals, conferences, and workshops. He is the BSI lead representative to ISO/ITU-T X.500 standards meetings, and was intimately involved in the design and standardisation of X.509 Public Key Infrastructures and Privilege Management Infrastructures (PMIs). He is an invited expert to the W3C Verifiable Credentials Working Group and a co-author of its Verifiable Credentials Data Model Recommendation.

Rémi Venant received the PhD degree. He is an associate professor at Le Mans University (France), and a member of the Laboratoire d'Informatique de l'Université du Mans, and a member of the Institut de Recherche en Informatique de Toulouse. His research mainly focuses on technology enhanced learning, within the field of learning analytics, artificial intelligence for education and learning environment design. He is also involved in security management research activities that address identity and access management or privacy issues.

Abdelmalek Benzekri is a full professor with University Paul Sabatier (UPS) for 20 years where he is affiliated to the computer science department of the IUT A – University Institute of Technology. His research activities are conducted at IRIT, where he leads the SIERA team and its 17 full-staff positions. His focus is on systems and networks management and specifically on information security management. After being director of a master's degree on information systems architecture, he recently became the head of the Computer and Networks Department of the UPSSITECH school of engineers, and launched, in 2016 a new master curriculum dedicated to Information Systems and Networks Security where her introduced an Identity and access management lecture for years addressing Federated Identity Management authentication and authorisation frameworks but also more recent user-centric as well as self-sovereign blockchain based technologies. He has supervised around 20 PhD theses and published more than a hundred scientific articles. Currently, he works in information systems infrastructure security – dynamic trans-organizational access control policies for virtual organization formations – and since 2016 he is formally in charge of security research policies at IRIT. The results of his research are exploited and disseminated mainly in the context of the aeronautical supply chain and also in a fortnight European and National research projects. He is member of the Regional PRISSM cybersecurity think-tank for the development of the security industry. He is participating to CyberSecurity4Europe, one of the four H2020 pilot projects launched recently by the Commission to reinforce the EU's cybersecurity capacity coping with the security issues related to GDPR, PSD2, eIDAS and the upcoming ePrivacy regulation. He will be involved in demonstrating e-Banking and privacy-preserving IAM scenarios.

Eddie Billoir received the high school degree with honors in digital professional, the IT DUT degree, and the graduate degree in electronics , he is working toward the bachelor's degree in telecommunication systems & IT networks at Toulouse Paul Sabatier. He is doing this year in apprenticeship as system security tester at Atos Company.

Omar Alfandi received the MSc degree in telecommunication engineering from the University of Technology Kaiserslautern, Germany, in 2005, and the doctoral degree (Dr. rer. nat.) in computer engineering and telematics from the Georg-August-University of Goettingen, Germany, in 2009. He is an associate professor at the College of Technological Innovation at Zayed University. Between 2009 and 2011, he enjoyed a post-doctoral fellowship at Telematics Research Group and he founded a Research and Education Sensor Lab where he is currently as Lab advisor. Before that he carried his Doctoral Research as part of an Industry, Academia and Research centers collaboration European Union (EU) project. He was working package leader of EU DAIDALOS II in the 6th framework project. He published numerous articles on Authentication Framework for 4G Communication Systems, Future Internet and Trust and Reputation Systems in Mobile ad hoc and Sensor Networks. He is the co-founder and co-director of the SMART (Sensors and Mobile Applications Research and Education) Lab at CTI. His current research activities are directed toward Internet of Things (IoT), security in next generation networks, smart technologies, security engineering, mobile and wireless communications. In August 2015 he was appointed as the assistant dean for Abu Dhabi Campus.