The Inconvenient Truth about Web Certificates

Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler and Jean-Pierre Hubaux

Abstract HTTPS is the de facto standard for securing Internet communications. Although it is widely deployed, the security provided with HTTPS in practice is dubious. HTTPS may fail to provide security for multiple reasons, mostly due to certificate-based authentication failures. Given the importance of HTTPS, we investigate the current scale and practices of HTTPS and certificate-based deployment. We provide a large-scale empirical analysis that considers the top one million most popular websites. Our results show that very few websites implement certificate-based authentication properly. In most cases, domain mismatches between certificates and websites are observed. We study the economic, legal and social aspects of the problem. We identify causes and implications of the profit-oriented attitude of CAs and show how the current economic model leads to the distribution of cheap certificates for cheap security. Finally, we suggest possible changes to improve certificate-based authentication.

Nevena Vratonjic
School of Computer and Communication Sciences, EPFL, Switzerland, e-mail: nevena.vratonjic@epfl.ch

Julien Freudiger
School of Computer and Communication Sciences, EPFL, Switzerland, e-mail: julien.freudiger@epfl.ch

Vincent Bindschaedler
School of Computer and Communication Sciences, EPFL, Switzerland, e-mail: vincent.bindschaedler@epfl.ch

Jean-Pierre Hubaux
School of Computer and Communication Sciences, EPFL, Switzerland, e-mail: jean-pierre.hubaux@epfl.ch

1 Introduction

HyperText Transfer Protocol Secure (HTTPS) is a key factor of the growth of the Internet ecosystem. It is the de facto standard used to guarantee security of Internet communications such as e-banking, e-commerce and Web-based email. HTTPS notably provides authentication, integrity and confidentiality of communications, thus preventing unauthorized viewing of exchanged information. The security of HTTPS communications is increasingly relevant, given the popularity of Web services where users reveal private information.

Yet, in practice the provided security is dubious and HTTPS may not achieve the intended objectives for multiple reasons. In most of the cases, it is due to certificate-based authentication failures typically caused by one of the following four problems. First, certification authorities may fail to implement certificate-based authentication properly [20, 38]. Second, websites may not deploy digital certificates in the correct way [11]. Third, users frequently do not attempt or are not able to verify the status of HTTPS connections [26, 27, 29, 36, 41]. Lastly, Web browsers may fail to meaningfully convey security threats to users [25, 39].

In order to implement HTTPS and certificate-based authentication, website administrators need a public/private key pair and a matching digital certificate [9]. The digital certificate authenticates the entity owning a specific website and the associated public key. $X.509$ certificates are standard on the Web and assume a hierarchical system of certificate authorities (CAs) issuing and signing certificates. Certificates notably contain information about the issuer (a CA), the certificate owner, the public key, the validity period, and the hostname (website). Website administrators can purchase trusted certificates from root CAs. The list of trusted CAs on top of the CA hierarchy (called root CAs) is usually pre-installed in Web browsers and varies from one Web browser to the next. If a website owns a certificate signed by a root CA, then a chain of trust is established and Web browsers can authenticate the website [9].

In cases of authentication failures, communication is vulnerable to man-in-the-middle attacks. Not only are sophisticated active attacks (e.g., session hijacking) possible, but also attacks such as *phishing* [32] and *typosquatting* [35] where a malicious party may impersonate a legitimate entity. These attack scenarios are more realistic because they do not require the attacker to modify users' communication on-the-fly, but rather to simply obtain a valid certificate for the relevant domains [6]. For example, an adversary may obtain a certificate for a domain name that is similar to the domain name of a legitimate entity (e.g., *paypaal.com* for the legitimate domain name *paypal.com*) and rely on typosquatting attacks (i.e., users accidentally mistyping the domain name in the URL) for users to initiate communication with the adversary. In these scenarios, consumers are frequently not aware that they are under attack as browser indicators of a secure connection are present and there are no security warnings. Thus, users may reveal sensitive information (e.g., a credit card number) to the adversary.

Compromise of HTTPS communications may have severe consequences for both users and Web service providers. Therefore, it is important to assess the scale of

HTTPS' current deployment and evaluate the security it provides. In particular, it is crucial to investigate deployment practices of certificate-based authentication. We seek answers to the following research questions:

Q1: How much is HTTPS currently deployed?
Q2: What are the problems with current deployment of HTTPS and certificate-based authentication?
Q3: What are the reasons that led to these problems?

In this paper, we report the results of a large-scale empirical analysis of the use of HTTPS and certificate-based authentication, that considers the top one million websites.

Our results show that one-third of the websites can be browsed with HTTPS. Only 22.6% of websites with username and password fields implement user login via HTTPS. In other words, for 77.4% of websites users' credentials can be compromised because login pages are not securely implemented. We believe that for most websites the complexity and cost in operating HTTPS might deter administrators from implementing HTTPS.

More importantly, only 16.0% of the websites implementing HTTPS carry out certificate-based authentication properly, i.e., using trusted, unexpired certificates with valid signatures, deployed on proper domains. For most of the websites (82.4%), authentication failures are in most cases due to domain mismatch, i.e., the domain name that certificate is issued for does not match the domain name it is deployed for. Other authentication failures are caused by untrusted certificates, expired certificates and broken chains of trust. Untrusted certificates are certificates whose chain of trust does not originate at one of the root CAs trusted by Web browsers. This is the case with *self-signed certificates* which website administrators often produce, by signing certificates themselves, in order to avoid costs of purchasing certificates from CAs.

The results imply that website administrators either lack the know-how or the incentives to properly deploy certificates. To avoid domain mismatch warnings, websites need a different certificate for each subdomain or a wildcard certificate (that matches any subdomain). Obtaining such certificates from trusted CAs is expensive. Further, website administrators that deploy self-signed certificates might lack incentive to take the additional overhead of managing multiple certificates, because Web browsers do not trust self-signed certificates and will anyhow display security warnings to users.

Websites are not the only culprits as malpractices of CAs also contribute to weak certificate-based authentication. CAs sometimes do not follow rigorous procedures when issuing certificates and distribute *domain-validated only* certificates that do not provide trust in the identity of certificates' owners. These certificates are less costly, thus website administrators are tempted to choose such options.

Our results help to understand the modes of intervention to properly achieve the security promised by HTTPS. In particular, we need to rethink the economic incentives behind the certificate-based authentication system. Further solution approaches may utilize means of engineering (e.g., introducing a third-party that pro-

vides records of websites that deploy certificates properly, similarly to the *Google Certificate Catalog* project [15]), policy change (e.g., shifting the liability from users to the stakeholders), usability (e.g., preventing users to access websites that implement certificate-based authentication improperly) and reputation (e.g., maintaining public records on security (mal)practices of CAs or websites administrators).

The rest of the paper is organized as follows. In Section 2, we detail HTTPS underpinnings and provide related work on Web authentication including attacks and countermeasures. We explain the methodology used for data collection and processing in Section 3. The properties of the collected data are assessed in Section 4 and the main results of our study are presented in Section 5. We discuss possible causes of current status of affairs in Section 6 and conclude in Section 7.

2 Background and Related Work

Netscape Corporation introduced the Secure Socket Layer (SSL) protocol to secure Internet communications [2], later standardized by the Internet Engineering Task Force (IETF) as Transport Layer Security (TLS) [4]. HTTPS combines the Hypertext Transfer Protocol (HTTP) with SSL/TLS to securely transport HTTP over insecure networks.

A key part of HTTPS is authentication of Web servers. The authentication process is based on $X.509$ certificates and takes place when an HTTPS connection is initiated between a client and a server. We detail how $X.509$ certificates work and review the research literature identifying $X.509$ vulnerabilities and improvements.

Users can trigger HTTPS communications by using the *https://* prefix in URLs. Web browsers then initiate HTTPS connections by connecting on port 443 of Web servers [5]. If Web servers support HTTPS, they respond to the client by sending their digital certificate.

A digital certificate is an electronic document that binds a public key with an identity by relying on a digital signature. In a typical public key infrastructure (PKI), a trusted certificate authority (CA) generates the signature. A certificate allows third-parties to verify that a public key belongs to an individual, and thus to authenticate this individual. X.509 certificates include [9]:

- Version: X.509 version number.
- Serial Number: Uniquely identifies each certificate.
- Signature Algorithm: Algorithm used by issuer to generate digital signature and parameters associated with the algorithm.
- Issuer: Entity that issued the certificate (i.e., CA)
- Validity period: Date certificate is first valid from (Not Before) and expiration date (Not After).
- Subject: Identified entity.
- Subject Public Key: The public key.
- Extensions: Key Usage (e.g., encipherment, signature, certificate signing).
- Signature: Certificate's signature.

In practice, website operators obtain certificates from CAs by sending certification requests that contain the website name, contact email address, and company information. CAs should perform a two-step validation [21, 22]: (i) verify that the applicant owns, or has legal right to use, the domain name featured in the application; (ii) verify that the applicant is a legitimate and legally accountable entity. If both verifications succeed, CAs are entitled to sign certification requests, thus producing *Organization Validated (OV) certificates*.

Web browsers verify certificates' authenticity by checking the validity of their digital signature and of their different fields. To check a digital signature, Web browsers need a second certificate that matches the identity of the Issuer. All Web browsers come with a built-in list of trusted root CAs. If browsers can verify the signature and trust the associated CA, then the certificate is trusted. Trust in a digital certificate is thus inherited from the entity that signed it and relies on the concept of *chain of trust* [9].

2.1 Certificate Verification Failure

Certificate verification can fail for the following reasons: i) the certificate has expired, ii) the domains certificate is valid for do not match the visited website, iii) the signature is not valid, or iv) the certificate issuer is untrusted. In the event of such failures, Web browsers usually warn users using pop-up windows. Users can either ignore such warnings and continue to the website, or decide not to proceed.

Firefox 4 redesigned its warnings and made them harder to skip, compared to Firefox 2. The goal is to encourage safe behavior from users [8]. In the example of Figure 1, a user gets a warning because the certificate is valid for domain *www.paypal.com* and he tried to connect to *paypal.com*. If the user wants to continue to the site, he must click on "I Understand the Risks" and then the "Add Exception" button. The intention is to discourage unexperienced users from proceeding while enabling advanced users to take appropriate security decisions.

2.2 Attacks

Previous work introduced several attacks on HTTPS.

2.2.1 Attacking Certificate Authentication Failures

Certificate authentication failures may lead to man-in-the-middle attacks. An adversary can replace an original certificate with a rogue certificate. If users systematically bypass security warnings, they will not notice the subterfuge and their communications will be hijacked.

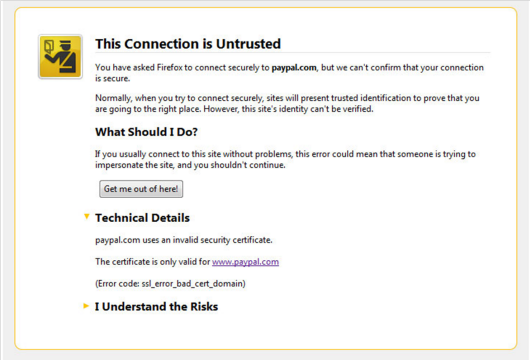


Fig. 1 Warning message for invalid certificates in Firefox.

2.2.2 Attacking Root CAs

Sogohian and Stamm [37] introduce the *compelled certificate creation attack* in which government agencies may compel a certificate authority to issue false certificates that can be used by intelligence agencies to covertly intercept and hijack secure communications. They note that too much trust is put in CAs and challenge the current trust system calling for a clean-slate design approach that notably reduces the number of entities that could violate users' trust.

2.2.3 Attacking Weak Certificate Validation

CAs do not systematically perform a proper two-step validation before issuing a certificate. Such weak validation affects the quality of certificates. For example, some CAs only verify that the applicant owns the domain name (step 1 of validation) and do not validate the identity of the applicant [21]. A challenge is emailed to the administrator appearing on the Domain Name Registrar, and if CAs receive an appropriate response, they issue the requested certificate. However, when purchasing a domain name, the identity of the claimed owner of the domain is not properly verified. Consequently, Domain Name Registrars are untrustworthy and should not be used as a basis for user authentication. Acknowledging this, CAs often use the term "Organization Not Validated" in the certificate. Unfortunately, such certificates bypass browser security warnings. This practice introduces the notion of *domain-validated only (DVO) certificate* that do not provide as much trust as *trusted OV certificate*.

Attackers can exploit the limitations of DVO certificates to their advantage. An adversary may register for the domain *bank-of-america.com* and obtain a corresponding DVO certificate.[1] By using an active redirection attack (e.g., DNS poisoning), or relying on typosquatting [35], users may connect to such fake websites.

[1] The legitimate domain is *bankofamerica.com*.

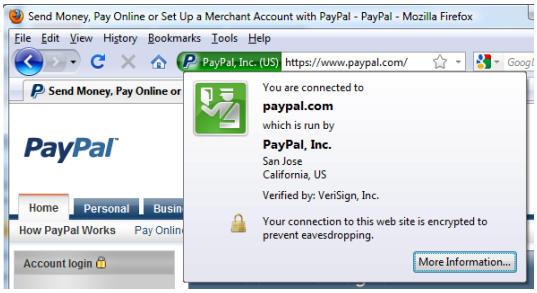


Fig. 2 User interface for EV SSL certificates in Firefox.

As Web browsers will not issue security warnings, the padlock will be displayed, and the URL will contain the bank's name, users may not realize they are on a phishing website. Most banking sites actually redirect their users from their main sites to e-banking URLs. Such URLs are sometimes long meaningless strings[2]. It is particularly hard for users to recognize a phishing URL from a legitimate one. These examples highlight the security risk associated with DVO certificates; they offer cheap untrustworthy authentication.

CAs have additionally introduced the concept of Extended Validation (EV) Certificates. To issue EV certificates, CAs use an audited and rigorous authentication method [12]. With EV certificate, Web browsers display an organization's name in green in the address bar as well as the name of the issuer (Figure 2). Together with the displayed colors, this makes it difficult for adversaries to hijack communications. For example, Firefox colors in green the address bar for a website with EV certificate and in blue for regular certificates. Unfortunately, this distinction is often unknown to regular users [25].

2.2.4 Attacking Cryptographic Primitives

Ahmad [23] discovered that the OpenSSL library used by several popular Linux distributions was generating weak cryptographic keys. Although the flaw was quickly fixed, SSL certificates created on computers running the flawed code are open to attacks on weak keys.

Stevens *et al.* [38] demonstrated a practical attack to create a rogue CA certificate, based on a collision with a regular end-user website certificate provided by a commercial CA. The attack relies on a refined chosen-prefix collision construction for MD5 and has since then discouraged the use of MD5 to generate signatures of certificates and encouraged adoption of SHA.

[2] E-banking URL of *ubs.com*:
`https://ebanking1.ubs.com/en/OGJNCMHIFJJEIBAKJBDHLMBJFELALLHGKIJDA`
`CFGIEDKHLBJCBPLHMOOKDAHFFKONKKKAMPMNAEDFPCIOENKBGNEGNBDKJNN6Aes21W`
`HTRFkGdlzvKKjjyZeB+GNeAGf-jzjgiO2LFw`

2.3 Proposed Countermeasures

In order to limit the effect of such attacks, multiple countermeasures were proposed.

2.3.1 Surveillance of Self-Signed Certificates

Wendlandt et al. [40] improve the Trust-On-First-Use (TOFU) model used for websites that rely on self-signed SSL certificates. Web browsers securely contact *notary* servers, who in turn independently contact the webserver and obtain its certificate. A man-in-the-middle attack can be detected by the fact that the attacker-supplied SSL certificate differ from those supplied by notary servers.

2.3.2 Improve Web Browsers' Interface

Jackson and Barth [31] propose to protect users who visit HTTPS protected websites, but who are vulnerable to man-in-the-middle attacks because they do not type in the *https://* component of the URL. Their system enables a website to hint to browsers that future visits should always occur via a HTTPS connection.

Herzberg and Jbara [30] help users detect spoofed websites by prominently displaying the name of the CA that provided the sites' certificate in Web browsers.

2.3.3 SSL Observatory

Recently, the SSL Observatory project [20] led by Eckersley and Burns investigated security practices of CAs and properties of digital certificates. This project is the first large scale empirical analysis of SSL certificates gathering a large number of certificates. Current results identify bad practices of CAs, such as issuing EV certificates non-compliant with the standard (e.g., issued for unqualified host names or improper key lengths) and having a high number of subordinate CAs. Eckersley and Burns suggest that Web browsers only need between 10 and 20 root CAs to use SSL with most websites, rather than the current long lists of CAs.

In comparison with the SSL observatory, we consider a different approach. First, while the SSL Observatory project analyzes root certificates and certificates that have a valid chain of trust, we investigate all trusted and self-signed certificates served by the top 1 million websites. Second, we collect certificates by crawling different domains whereas the SSL observatory project crawls the entire IP address space. The key difference is that we can check how certificates are used in practice by websites. For example, we can measure the relation between domains, their popularity, their category and the quality of certificate deployment. We can measure the exposure of a user browsing the Web to different types of authentication failures. The data collected by the SSL observatory enables to check the type of certification construction and properties but not how they are used in practice. In

other words, [20] gives an optimistic view of the current situation and our analysis complements their work.

3 Methodology

In this section, we describe the algorithms that are used for data collection and processing. We collect the data based on the HTTP and HTTPS connections established with Web servers of the most popular websites according to Alexa's ranking. In particular, we focus on understanding how certificates are deployed on these websites. To analyze the collected certificates we rely on OpenSSL [17] tools.

3.1 Algorithms for Data Collection

We conduct the survey on one million most popular websites (according to their Internet traffic), ranked by Alexa, a leading analytical firm that provides information on Internet traffic data [13]. This dataset imposes no limitations on websites' categories, countries, languages, or any other property. In order to determine if there is a significant difference in the results across different website categories, we additionally conduct the survey on 500 most popular websites from each of the Alexa's 16 categories: Adult, Arts, Business, Computers, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society and Sports.[3]

We crawl the websites from the list using a Python script whose pseudo-code is illustrated with Algorithms 1 and 2. For each *host* in the list, separately for HTTP and HTTPS, the script uses the retrieve function to initiate a connection and attempt to retrieve the content of the website. If redirections are encountered, they are followed unless the maximum of 8 redirections per host has been reached. Given that some websites are accessible only at *www.host*, the retrieve function performs forced redirection to *www.host* if the script was not automatically redirected and the DNS lookup for *host* failed. If the connection is successfully established and all redirections have been followed, the script saves the content, cookies, and URL of the final page. At the same time, it checks the content of the webpage for login forms by looking for type="password" in the HTML source. Login forms use this property to instruct browsers to hide the characters typed into the text box. Whenever an HTTPS connection can be established to the host, the script additionally saves the websites' certificates and records the cipher suite and version of TLS used throughout the connection (lines colored in blue). Because of redirections, it is possible that the script encounters more than one certificate per host. In such a case, it only saves the certificate associated with the final URL, i.e., the one following the last redirec-

[3] To illustrate how Alexa sorts websites into categories, we provide the list of top 5 websites per category in Appendix.

tion. The rationale behind this choice is that this is the certificate associated with the Web pages, users connecting to $https://host$ can actually browse.

Having collected this data, we proceed to the verification and analysis of each certificate. This step is performed off-line with a second Python script. The latter relies on OpenSSL to verify the validity of certificates' signatures and extract values of some of the fields.

Algorithm 1 HTTP data collection

```
for all host in list do
    retrieve(http://host)
    if success then
        store content and URL
        store cookies
        check for login
    else
        log connection failure
    end if
end for
```

Algorithm 2 HTTPS data collection

```
for all host in list do
    retrieve(https://host)
    if success then
        store content and URL
        store cookies
        check for login
        store certificate
        store cipher suite
        store HTTPS version
    else
        log connection failure
    end if
end for
```

3.2 Verifying X.509 Certificates

The verification process includes several steps, the first of which is building a certificate's chain of trust. For each certificate, the chain of trust is built starting from the certificate that is to be verified. Building each new level of the chain requires retrieving the certificate of the Issuer (i.e., the parent certificate) of the previous certificate. Typically, each certificate contains *CA Issuers' URI* which can be used to download its parent certificate. If any of the certificates in the chain cannot be retrieved, the verification process cannot proceed and the chain is *broken*. When a certificate is its own Issuer (i.e., the Subject and Issuer fields match), it is considered to be a *root certificate* and the chain is complete.

After successfully building the chain of certificates, the signatures in the chain should be verified. If all of the digital signatures can be verified according to their cryptographic signature algorithm, the certificate has a ***valid signature***. A certificate with valid signature is ***trusted*** if the issuer of the root certificate of the chain is trusted, otherwise it is ***untrusted***. To establish trust, we rely on a well-known list of trusted root certificates provided in the *ca-certificate 20090814-3* package of the Archlinux distribution. This package contains most of the root certificates provided in Mozilla [14] software products. Among untrusted certificates, we distinguish between ***self-signed*** certificates (whose chain contains only itself) and ***untrusted*** certificates (whose chain contains at least two certificates, but whose root certificate issuer is not in the list of trusted certificates). Privately-signed certificates are a par-

ticular case of untrusted certificates, which are often used in large companies, where a self-signed certificate is produced and trusted as a root certificate to sign other certificates (e.g., for email and Web servers).

Algorithm 3 Certificate verification

```
for all cert in downloaded certificates do
  current ← cert
  while current is not self-signed do
    if parent of current not available locally then
      try to retrieve parent
    end if
    if parent of current not available locally then
      return CHAIN BROKEN
    else
      current ← parent
    end if
  end while
  invoke openssl verify on cert
  if signature is valid then
    if parent of current is trusted then
      store "trusted"
    else if cert = parent of current then
      store "self-signed"
    else
      store "untrusted"
    end if
    invoke openssl x509 on cert
    store subject country, subject CN
    store Not before, Not after
    store Alternative DNS name
  else
    store "invalid signature"
  end if
end for
return SUCCESS
```

The actual verification performed by the script (for each certificate) uses OpenSSL *verify* tool [16]. The output of the tool is used to determine if the certificate signature is valid, and if so, whether the certificate is trusted, self-signed or untrusted (e.g. privately-signed). For each certificate that has a valid signature, we collect additional information. In particular, we extract the values of *Common Name* (CN) and *Country* from the Subject, and of the Not before and Not after fields. In addition, we extract *DNS name* entries from the *X509v3 Subject Alternative Name* extension, if it exists. Moreover, we obtain the root certificate of the chain and save the value of the Issuer field. Algorithm 3 illustrates the verification process.

Not before and Not after fields are used to compute the ***validity period*** of a certificate. If the current date is not within the validity period then the certificate is ***expired***.

Domains for which a certificate is valid are specified in the subject *common name* field or the *DNS name* field of the *X509v3 Subject Alternative Name* extension. According to RFC 2818 [5], if the *X509v3 Subject Alternative Name* extension exists and contains at least one field of type *DNS name*, it must be used as identity for the server. Otherwise, if no such field exists, the subject *CN* fields are used. Therefore, to verify if a certificate is deployed for a proper domain (i.e., if there is a domain match), we match the DNS names or subject CN fields against *host* for which the certificate is saved (after following all redirections). As there might be several candidates (several *DNS name* fields, or several subject *CN* fields), we match each candidate according to the rules given by RFC 2459 [3]. Namely, we attempt to match each candidate (using case-insensitive matching) to *host*, taking into account possible wildcards[4].

Based on the described comparison, there is a ***domain match*** if one of the following is true:

- *Host* and at least one of the candidate fields (case-insensitive) match exactly.
- The candidate field contains one or more wildcard (e.g. **.domain*) and *host* matches the regular expression given by the candidate field.

If a match is found, the certificate is said to have a ***valid domain*** for *host*, otherwise there is a ***domain mismatch***.

We also classify certificates as domain-validated only (DVO) certificates and extended validation (EV) certificates. Checking whether a given certificate is an EV certificate is easy: it suffices to look for the EV Object Identifiers (OID) of the root CA. If the OID appears in one of the certificate's policy fields, then the certificate provides extended validation. OIDs can be obtained directly from authorized CAs' certificate policy statements (CPS) that can usually be downloaded from CAs' websites.

Determining whether a certificate is a DVO certificate is more complicated, because different CAs tend to indicate that a certificate is DVO in different ways. Many of the DVO certificates contain *OU=Domain Control Validated* string in their subject field. However, not all of the certificates that contain this string in the subject field are DVO. Indeed, for some of the certificates that contain this specific string in their subject field, we found that the subject organization had been validated as well. Moreover, some DVO certificates do not contain this string, but *O=Persona Not Validated* string instead. However, as the number of root CA is (relatively) small and only a few of them signed a significant number of certificates, we examined a few certificates signed by each of the top CAs (in terms of the number of certificates signed) and looked for typical strings or indications that the certificate is DVO. Those strings (usually located in the subject field) are sometimes product names, such as *RapidSSL* or *QuickSSL*. In other cases, the presence of the string *OU=Domain Control Validated* in the subject field and having an organization field identical to the CN field, is an indicator that the certificate is DVO. Based on these observations, we design an algorithm that determines if a certificate is DVO.

[4] A wildcard "*" stands for at most one level of subdomain, i.e. **.domain.tld* matches *subdomain.domain.tld* but not *subsubdomain.subdomain.domain.tld*.

Summary of the certificate data set obtained in the survey is presented in Appendix (Figure 17).

4 Data Collected

We store all the collected data in a SQLite [18] database. The database and some examples queries are available at `http://icapeople.epfl.ch/freudiger/SSLSurvey`.

We create a list of unique hosts by merging the lists of top one million websites with 16 lists containing top 500 websites across categories. By including 787 hosts from the categories lists that were not in the top one million, we obtain a list of 1′000′787 unique hosts.

The script successfully established HTTP or HTTPS connections with 95.76% of unique hosts. Most connection failures were due to socket failures (connection timeout) or DNS failures (unable to resolve hostname). Other failures included redirections to invalid URLs or redirections to unknown protocols. ***We consider the 958'420 working hosts for our survey***.

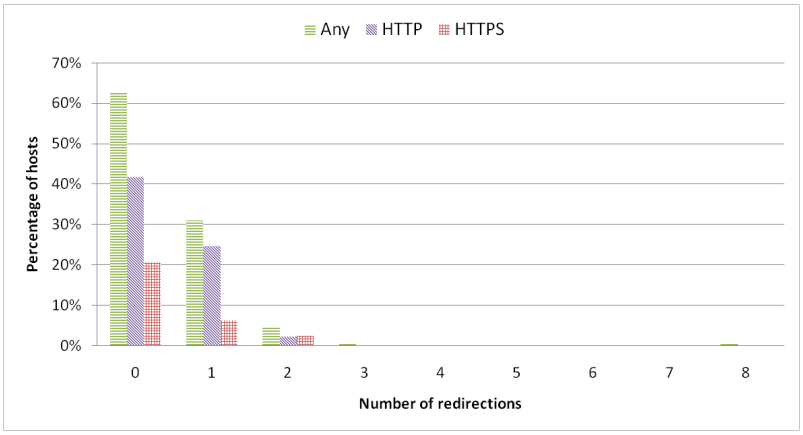


Fig. 3 Number of Redirections with HTTP and HTTPS. Most of the websites perform one or no redirection at all. Redirections occur more frequently when websites are browsed via HTTP than via HTTPS.

Based on the number of redirections (Figure 3) that we observed with HTTP or HTTPS, ***most websites perform one or no redirection at all***. We can also observe that ***redirections occur more often for websites browsed via HTTP***. The results also justify our decision to allow the data collection script to follow up to 8 redirections. For the few websites that had more than 8 redirections, the browser entered an infinite loop without reaching a final page. Thus, for proper hosts, up to 8 redirections were sufficient to successfully retrieve their content.

After following redirections, ***in most cases, the landing page belongs to the same domain or www subdomain*** (Figure 4) with both protocols. The script obtained 1′032′139 Web pages with HTTP and 339′693 Web pages with HTTPS.

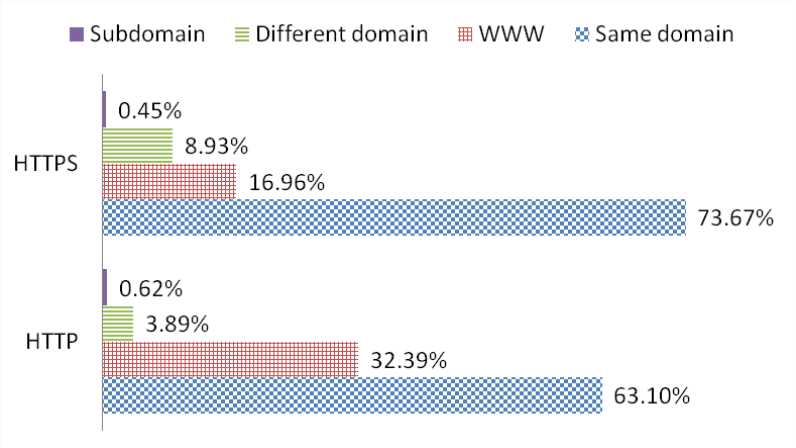


Fig. 4 Final domain after following redirections, compared to initial domain. Typically, the final page is in the initial domain or in the *www* subdomain with both HTTP and HTTPS.

5 Analysis

To answer our research questions, we generate different statistics on the usage of HTTPS based on the collected data. We run a number of SQL queries to obtain the following results.

5.1 HTTPS Deployment on the Web

According to Figure 5, ***more than half (65.3%) of the 1 million websites can be browsed only via HTTP***, whereas ***only one-third of websites can be browsed via HTTPS.*** Among websites that implement HTTPS, 0.99% can be browsed exclusively via HTTPS (do not respond to HTTP or redirect users from HTTP to HTTPS) and the remaining 33.7% support both HTTPS and HTTP.

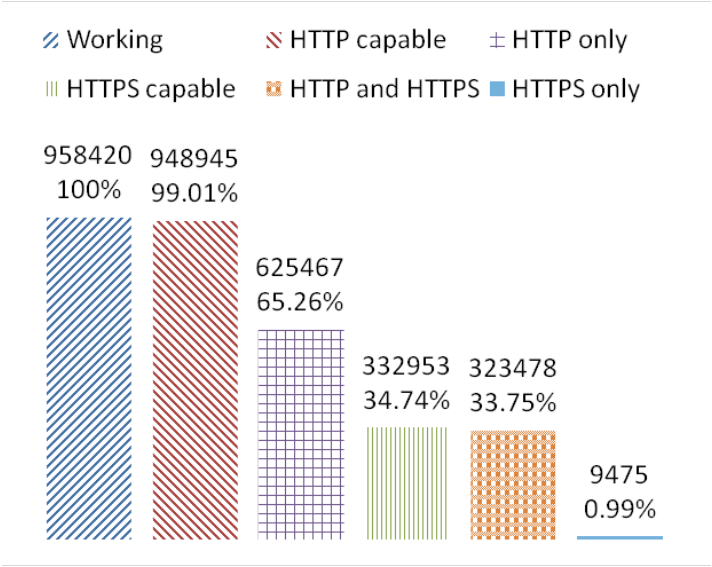


Fig. 5 HTTP vs HTTPS. About 65% of the websites can be browsed only via HTTP and one-third can be browsed via HTTPS.

5.1.1 HTTPS Across Website Categories

Given that the data set for each category contains 500 websites, we cannot draw strong conclusions about HTTPS deployment across categories. However, we still observe some trends: ***HTTPS is implemented most in categories Reference*** *(*33.75%***,*** ***Health*** *(*33.41%*)* ***and Business*** *(*31.12%*)* ***and least in categories Arts*** *(*17.67%*)* ***and Sports*** *(*20.21%*)*. Websites of universities belong to the *Reference* category and contribute to the high percentage of that category as most of them implement secure services, such as emails. In the *Health* category, websites may deal with sensitive medical data and we observe that a high percentage of them implements HTTPS. On the contrary, websites in categories *Sports* and *Arts* most likely do not need HTTPS, and we observe smaller deployment rate in those categories.

5.1.2 HTTP vs. HTTPS for Login Web Pages

We check whether websites that require users' login credentials (i.e., username and password) implement HTTPS. To do so, we searched for retrieved Web pages containing login and password fields. Surprisingly, ***only 22.6% of Web pages with password fields were implemented via HTTPS!*** In most cases, websites do not encrypt Web pages at all or use HTTPS encryption only partially, for parts of Web pages containing credentials. However, if the entire page is not transmitted over HTTPS, it can be compromised by man-in-the-middle attacks and lead to the compromise of

credentials. Therefore, 77.4% of websites put users' security at risk by communicating users' credentials in clear text or by encrypting only parts of Web pages. Such weak security practices may be due to trade-offs between security and performance, the lack of know-how or the burden to implement HTTPS.

5.1.3 HTTPS Cipher Suites

The majority ($\sim$ 70%) of websites use DHE-RSA-AES256-SHA cipher suite. DHE denotes ephemeral Diffie-Hellman, where the Diffie-Hellman parameters are signed by a signature-capable certificate, itself signed by a CA. The signing algorithm used by the server is RSA, specified after the DHE component of the cipher suite name. The cipher used is AES with 256 bit keys. The last field notifies the message authentication code (MAC) used, in this case SHA that stands for a modified version of SHA-1. It is a good news that a majority of websites use this cipher suite, because it is in the top of the list of cipher suites recommended and preferred by major software companies (e.g., Mozilla). Most websites use 256 bits ($\sim$ 76%) or 128 bits ($\sim$ 22%). Surprisingly, there are some ($\sim$ 50) websites that still use 40 or 56 bit keys.

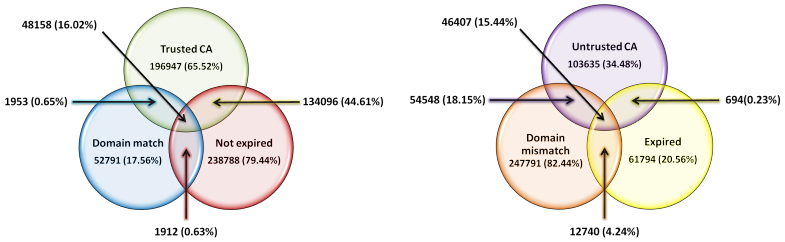
Nevertheless, our findings show that ***good cipher suites are selected***. It means that the potentially weak part of establishing a secure HTTPS connection is server authentication.

5.2 Authentication Failures

Authentication failures are the major cause of improper implementation of HTTPS in practice. Besides malicious behavior, TLS-based authentication can fail for several reasons:

- **Broken chain of trust:** If a signature in the chain of trust cannot be verified, the chain of trust is broken.
- **Untrusted root certificate:** Trusted root certificates are self-signed certificates of CAs. Any other self-signed certificate is untrusted. In general, any certificate is untrusted if it is signed by an entity whose certificate is not among the trusted root certificate. Users must manually check whether they trust the Issuer of certificates untrusted by Web browsers.
- **Expired certificate:** Certificate validity period is defined using Not Before and Not After markups. Certificate validity varies from a few months to a few years, as agreed with CAs. Standards require that Web browsers check certificate validity periods and issue a warning to users in case of expiration. Certificate signatures can be verified even after a certificate expires because signature verification only guarantees the integrity of the certificate's content.
- **Domain mismatch:** Certificates apply to hosts identified in the Subject markup using the common name (CN) tag (e.g., *CN=www.epfl.ch*) or to the DNS name

specified in the Alternative Name Extension. If the host does not match exactly the name specified in the CN field or the DNS name of a certificate, Web browsers issue a domain mismatch warning. If another host is located at *login.epfl.ch*, then another certificate is required to identify this other host or the website can use a *wildcard certificate* (**.epfl.ch*) that is valid for any subdomain of the host.



(a) Authentication Success - no browser warnings.

(b) Authentication Failure - browser warnings.

Fig. 6 Web browser authentication outcomes for websites that implement HTTPS and whose certificate signatures can be verified. Certificates of only 16.02% (48′158) of those websites allow for a correct authentication. When authentication fails, in 82.44% of the cases it is due to a domain mismatch.

Each problem occurs in our dataset and multiple combinations of problematic scenarios exist. Firstly, among 330′037 downloaded certificates, the signature of 300′582 could be properly verified. Our analysis is thus based on those certificates with valid signatures. Surprisingly, we observe (Figure 6(a)) that ***only 16.02% of all certificates with valid signatures allow for a correct authentication***, i.e., would not cause Web browsers to pop-up security warnings to users and HTTPS connection will be established transparently. It is only a minority (48′158) of all tested websites that enable proper Web authentication. The ***domain mismatch failure is clearly the main cause of problems*** (Figure 6(b)). It accounts for 82.44% of failures, followed by untrusted, expiration date and broken chain failures. These results show that website operators fail to understand the domain to which acquired certificates apply to or do not wish to bear the cost of handling multiple certificates for one website.

5.3 Certificate Reuse Across Multiple Domains

While looking for an explanation for the high number of domain mismatch failures, we noticed that a high number of the same certificates (both trusted and self-signed) appear for a number of different domains. With the exception of a few wildcard certificates that can be valid for multiple domains, other certificates are usually valid for a single domain and when deployed on other domains will cause a domain mismatch failure. Figure 7 shows the distribution of unique certificates that

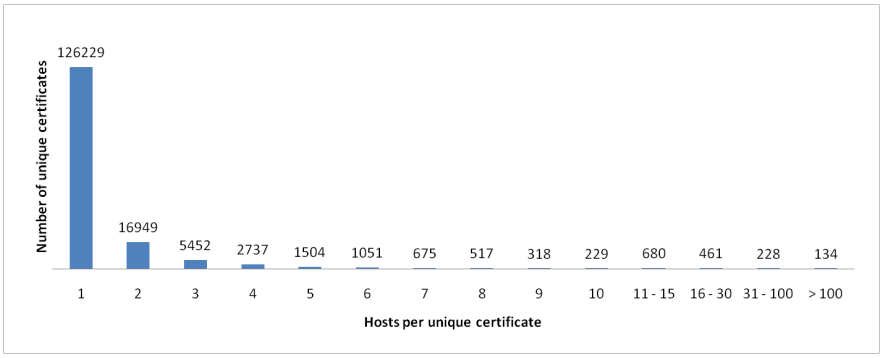


Fig. 7 Reusing certificates across multiple domains. A high number of certificates (both trusted and self-signed) that are issued for a single domain appear across a number of different domains. Deployment on those other (invalid) domains causes a domain mismatch authentication failure.

appear across different hosts. ***Among the 330'037 collected certificates, there are 157'166 (47.6%) unique certificates, 126'229 of which appear each on only one host***. The same certificate sometimes appears on more than 10′000 different domains! We find that there are 24 unique certificates that are reused across at least 500 domains each. In other words, 52′142 (26.5%) of the hosts that have a trusted certificate with valid signatures, have certificates that are reused across at least 500 domains. 20 of those certificates are certificates of Internet hosting providers (accounting for 46′648 hosts).

Typically, with virtual hosting (when many websites are hosted at same IP address) hosting providers serve the same certificate for all of the hosted websites. During the establishment of a TLS connection, the server does not know which website the client is requesting, because this information is part of the application layer protocol. Thus, the practice of hosting servers is to provide a default certificate, which is the behavior we observe. Table 1 shows a few examples with the number of hosts for which the certificate of a hosting provider is served and the domains for which the certificate is valid. In most of the cases, hosted websites do not belong to subdomains of hosting providers and rather have a completely different domain name, which causes domain mismatch warnings. Even though technically those websites are hosted at the provider's servers, the authenticity of those business should not be vouched for by the provider. Hosted websites should irrespectively obtain valid certificates for their domains from CAs and hosting providers should implement Server Name Indication (SNI), an extension of TLS which aims at solving this problem [7]. The main idea is that the client provides the domain name of the requested website during the TLS negotiation phase, thereby allowing the server to serve an appropriate certificate. Nowadays, SNI is supported by most of Web browsers and Web servers. However, even if a client does not support SNI, servers should not serve default certificates that do not match domains of hosted websites, but rather refuse such connections.

Table 1 Certificate reuse due to Internet hosting.

Certificate Validity Domain	Number of hosts
***.bluehost.com**	10'075
***.hostgator.com**	9'148
***.hostmonster.com**	4'954
***.wordpress.com**	4'668
***.websitewelcome.com**	2'912
***.justhost.com**	2'908

A website often simply "borrows", i.e., uses a certificate of another website. If a certificate appears on a smaller number of domains, it might also be that the same administrator is in charge of these domains and then uses a single certificate for all of them. In either case, such certificate deployment is a bad practice.

5.4 Properties of Self-Signed Certificates

We investigate the differences in the deployment of trusted and self-signed certificates. ***Among certificates with valid signatures, 65.6% are trusted (signed by trusted CAs) and the remaining 34.4% are self-signed*** (Figure 6(a).

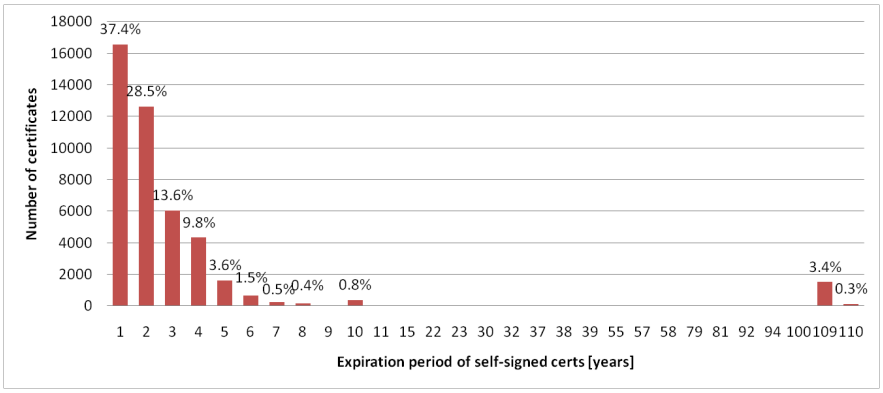


Fig. 8 Distribution of the expiration periods (in years) of self-signed certificates. In addition to being untrusted, most of the self-signed certificates are also expired (45.54%) and have a domain mismatch (97.48%). Even though self-signed certificates have almost no cost and are easy to generate, they are not maintained properly.

We observe that with self-signed certificates, in addition to being untrusted, at least one other authentication problem likely occurs (e.g., expired or domain mismatch). As self-signed certificates are free and easy to generate, it is to be expected

that self-signed certificates are up-to-date and that they match domains they are used for. Our results show the opposite. We observe that ***almost half of the self-signed certificates are already expired***. Some certificates expired a long time ago (e.g., 100 years).[5] Distribution of the time validity periods of the non-expired self-signed certificates is presented in Figure 8: most of the self-signed certificates are valid for one or two years. We also notice a number of certificates that have a validity of 100 years.

Interestingly, ***97.48% of the self-signed certificates have an invalid domain***. This shows that website administrators either do not know how to properly manage certificates or simply do not care what kind of warnings are displayed to users, as there will be one for a self-signed certificate anyway (due to the lack of trust in certificates' issuer). It is unclear whether users would trust self-signed certificates more if other fields (e.g., validity and domain) are correct, or whether it does not make a difference.

5.5 Properties of Trusted Certificates

In the following, we consider only trusted certificates with valid signatures. We observe that ***among trusted certificates with valid signatures, only 7% are expired, but 74.5% have a domain mismatch***.

5.5.1 Domain Matching for Trusted Certificates

By comparing domains certificates are deployed for (i.e., *host*) with domains certificates are valid for (i.e., common names (CN) and DNS names in the subject alternative name extension fields of X.509 certificates), we observe the following cases (Figure 9(a)):

No mismatch: *Host* matches one of the domains certificate is valid for.

Lack subdomain redirection: The certificate is valid for *subdomain.host* and deployed on *host*. Automatic redirection from *host* to *subdomain.host* would resolve the domain mismatch problem in this case.

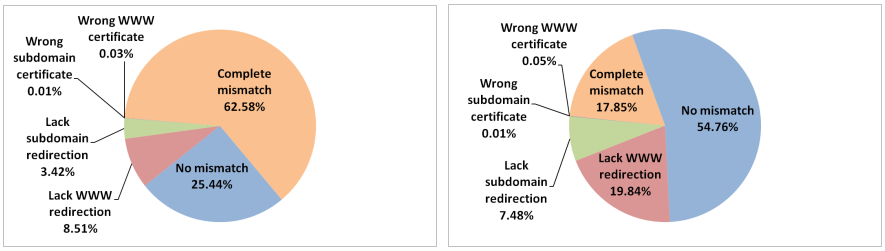
Lack www redirection: The certificate is valid for *www.host* and deployed on *host*. Automatic redirection from *host* to *www.host* would resolve the domain mismatch problem in this case. This case is a specific instance of the previous case and we look into it separately.

Wrong subdomain certificate: The certificate is valid for *host* and deployed on *subdomain.host*. To resolve the domain mismatch problem in this case website administrator has to obtain a certificate valid for *subdomain.host*.

[5] Expiration periods are computed with respect to February 2010.

Wrong www certificate: The certificate is valid for *host* and deployed on *www.host*. To resolve the domain mismatch problem in this case website administrator has to obtain a certificate valid for *www.host*. Again, this is a specific instance of the previous case.

Complete mismatch: (i) The *host* does not match the domains certificate is valid for, (ii) the *host* is not a subdomain of the domains certificate is valid for, or (iii) the domains certificate is valid for are not subdomains of *host*.



(a) Domain matching for trusted certificates with valid signatures.

(b) Domain matching for unique, trusted certificates with valid signatures.

Fig. 9 A majority of trusted certificates are deployed for non-matching domains. Partially, domain mismatch happens because of certificate reuse across different domains (e.g., due to Internet hosting). After excluding reused certificates, the major problem that causes domain mismatch is deployment of certificates issued for *subdomain.host* on *host* domains. Simply by automatically redirecting to *subdomain.host*, about 27% of the websites would avoid security warnings being displayed to users when vising their websites.

From the results in Figure 9(a) we observe that ***trusted certificates are mostly (62.58%) deployed for domains that are completely different from the domains certificates are valid for***. For 11.93% of the websites with trusted certificates, the domain mismatch problem could be easily solved with automatic redirection: to *subdomain.host* or *www.host*.

Because we have seen that certificates are often reused (mostly due to hosting providers) we narrow our analysis to unique certificates only and, as expected, results are better. ***Domain mismatches happen for 45.24% of the unique trusted certificates with valid signatures*** (Figure 9(b)). The number of complete mismatches is thus drastically reduced from 62.58% to 17.85%. A possible interpretation for the remaining complete mismatches is that online businesses and major companies require at least one certificate and understand that the certificate has to be up-to-date and timely renewed, for the purposes of its online transactions or simply for a good reputation. However, as most certificates are valid for a single domain (with the exception of rarely used wildcard certificates), websites need to obtain multiple certificates for multiple domains. This cost is most likely too high, and website administrators rather deploy the same trusted valid certificate across different domains. A very common case is that websites obtain certificates for *subdomain.host*

and use it for *host* domain as well. In these situations, browsers also issue security warnings due to domain mismatch. This problem can be solved if websites automatically redirect to *subdomain.host* when visiting *host*. ***With automatic redirection to subdomain.host, about 27.32% of websites with trusted certificates would avoid domain mismatch warnings*** (Figure 9(b)). In particular, redirecting to *www.host* would resolve domain mismatch problem for about 20% of the websites. In a small percentage of cases (0.06%), websites have certificates that are valid for *host* and it is used on *subdomain.host*.

5.5.2 Validity Period of Trusted Certificates

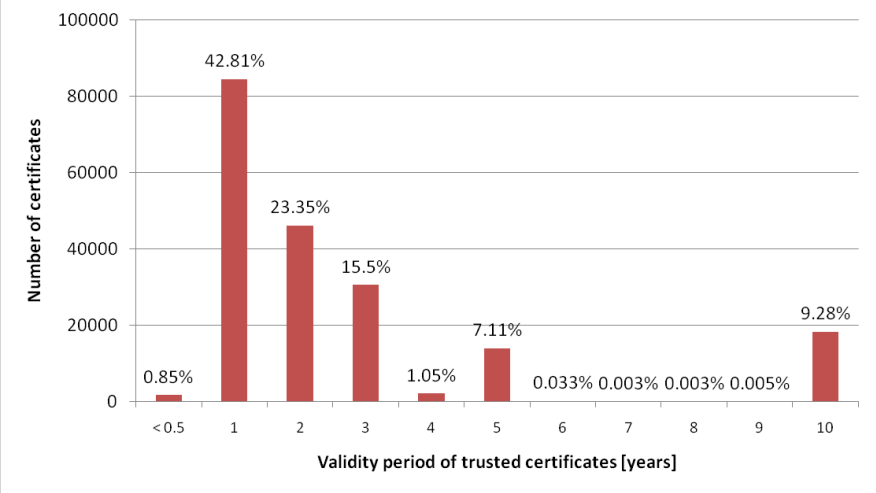


Fig. 10 Distribution of validity periods (in years) of trusted valid certificates. Almost half of the certificates are issued for one year, indicating that it might be too costly for businesses to pay for certificates valid for several years or that they do not favor long term investment. It might also be due to unwillingness of CAs to trust websites for too long, as it limits the risk of bad publicity in case a malicious websites is actually issued a certificate.

Figure 10 shows the validity time distribution of trusted certificates. We notice that almost ***half of the trusted certificates have a validity of 1 year***. Typically, CAs offer certificates for periods of 1, 2 and 3 years. Similarly as for obtaining certificates for multiple domains, it seems that it is too costly to obtain certificates for more than one year. We found a surprising number (almost 10%) of certificates that have a validity of 10 years or more. However, it appears that all of those certificates are DVO and the price of such 10-year DVO certificates is approximately the price of a properly validated 1-year OV certificate. CAs have incentives to issue short term

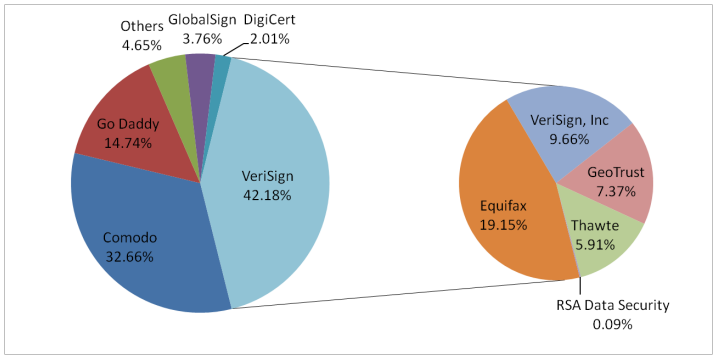


Fig. 11 CA root certificates. VeriSign has the largest share of the market, followed by Comodo. The certificates issued by GeoTrust, Thawte and Equifax are counted as VeriSign certificates as these CAs were acquired by VeriSign.

certificates in order to minimize the risk of being associated and vouching for an organization that might turn out to be compromised.

5.6 (Mal)practices of CAs

We looked into how many certificates were issued by each CA (Figure 11) and the common (mal)practices of CAs when issuing certificates. Notably, we focus on investigating whether CAs issue: (i) domain-validated only certificates (ii) certificates based on MD5 hash-functions and (iii) certificates with keys of inappropriate length with respect to their time validity.

VeriSign, together with its acquired CAs (Equifax, Thawte and GeoTrust), has the largest part of the market, issuing (42.2%) of the certificates, followed by Comodo with 32.7% of the certificates (Figure 11).

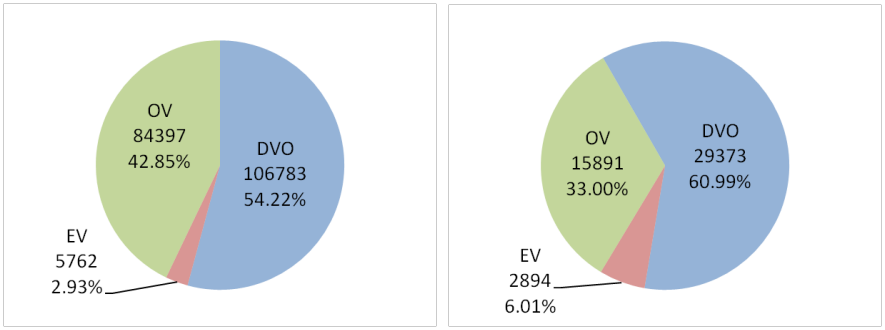
5.6.1 DVO, OV and EV Certificates

We investigate the usage of DVO, OV and EV certificates. Bad news is that 54.2% of trusted certificates with valid signatures are only domain-validated (Figure 12(a)). In other words, ***half of the certificates issued by CAs are issued without properly verifying the identity of certificates' owners***. As previously discussed, these certificates do not guarantee trust and do not provide the security that users expect. In addition, there are no explicit security warnings to notify users about the difference in provided security.

Results from Figure 12(b) show that among the small number (48′158) of valid certificates, ***users should not trust about 61% of them as the legitimacy of the organizations behind these certificates was not properly verified by CAs***.

Only about 3% (5′762) of trusted certificates with valid signatures are EV (Figure 12(a)). But ***only 2'894 EV certificates are actually not expired and valid for the requested domain*** (Figure 12(b)). OV certificates are traditional SSL certificates that are issued by CAs after the proper two-step validation, but not following special EV recommendations. OV certificates can as well authenticate the organization owning the certificate.

Essentially, ***18'785 websites have valid certificates that can prove the identity of the organization owning a certificate (either with EV or OV certificates)***.



(a) Trusted certificates with valid signatures.

(b) Certificates that allow for successful authentication by Web browsers.

Fig. 12 Types of certificates: EV, OV and DVO. A small number of websites have certificates (EV or OV) that provide the trust in the identity of the organization owning a certificate. About 61% of the certificates trusted by Web browsers do not guarantee the legitimacy of the owner, i.e., are DVO.

5.6.2 Certificates Using MD5

To sign a certificate, CAs first produce the hash of the certificate (typically with MD5 or SHA-1 hashing functions) and then encrypt the hash with their private keys. MD5 is not a collision resistant hashing function as it has been shown that it is possible to create two files that share the same MD5 checksum and consequently, to fake SSL certificates [38]. After the discovery of this attack, VeriSign announced [10] that it immediately discontinued the use of flawed MD5 cryptographic function for digital signatures, while offering a free transition for customers to move to certificates using the SHA-1 algorithm. Unfortunately, we found that certificates with MD5 are still in use. In our study, we found 2071 **trusted, not expired certificates that use MD5 and are all issued by Equifax** (belonging to VeriSign). Some certificates are valid until year 2014. Perhaps, some of these websites are not willing to go through the hassle of obtaining new certificates and decide to keep potentially vulnerable certificates. Nevertheless, CAs should not allow for such websites that expose customers to serious security threats.

5.6.3 Certificate Public Key Length wrt. Expiration Date

CAs may issue certificates with keys of inappropriate length with respect to their time validity. We extract the expiration date (Not After field) and key length from certificates and we represent them in Figure 13. The size of a bubble in the graph corresponds to the number of data points that have the same value and the center of the bubble to the (Expiration year, Key length) point. We also plot the recommended (optimistic) key length that is considered to be secure in a given point in time [34]. Data points (centers of bubbles) that are above the recommended curve are acceptable and represent well chosen keys. Data points that are below the curve are badly chosen and are considered to be vulnerable at the point in time they are used. In aggregate, ***about a half (97'436) of the trusted certificates have inappropriate key length with respect to their time validity***. Ideally, these certificates should not be used and CAs should rigorously follow the recommendations about the key length.

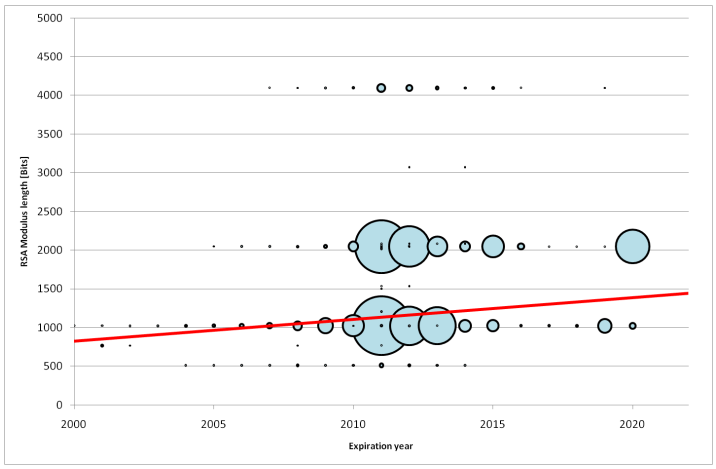


Fig. 13 Appropriateness of the key length wrt. expiration time. Data point (Expiration Year, Key Length) is represented with the center of a bubble and the size of the bubble represent a number of data points with the same value. Data points above the recommended key length curve (linear) are well chosen, the ones below are not considered to be secure at the time they are used. About half of the trusted certificates have inappropriate key length with respect to their time validity.

5.7 Correlation of the Authentication Failure Rate with Other Parameters

To better understand the underlying reasons for the observed certificate deployment, we correlate the authentication failure rate with other parameters such as issuing CAs, subjects' countries, website categories and rank.

5.7.1 Authentication Failure Rate wrt. CAs

Since CAs are only responsible for issuing certificates, not for managing how they are deployed, it might not be fair to correlate authentication success rate to certificates' issuing CAs. Given that the authentication success rate mostly depends on whether a certificate is deployed on a matching domain, it is a responsibility of the organizations who purchased the certificates to properly maintain them and make sure that they allow proper authentication. Nevertheless, it is interesting to compare authentication success rate that is achieved with certificates issued by different CAs (Figure 14). We limit our results to those CAs for which we collected at least 4′000 trusted valid certificates.

We observe that certificates issued by GoDaddy, GlobalSign and VeriSign obtain a higher authentication success compared to others. Interestingly, certificates that are signed by root certificates belonging to smaller and perhaps less famous CAs (Equifax, Thawte and UserTrust)[6] have a smaller success rate.

There are different hypotheses to explain this. GlobalSign, GoDaddy and VeriSign are well-established and trusted CAs with major clients. Their certificates typically have a larger price than competitors. Hence, only resourceful companies may afford to purchase such certificates and these organizations may care more about properly deploying certificates in order to provide good security. On the contrary, less security-conscious website administrators may opt for inexpensive and easier to obtain certificates, that are typically issued by other CAs. Given their lack of incentives, it follows that they might not bother deploying certificates properly. Another possibility is that GlobalSign, GoDaddy and VeriSign only issue certificates after a proper two-step validation process or that they make sure that their customers know how to properly deploy certificates.

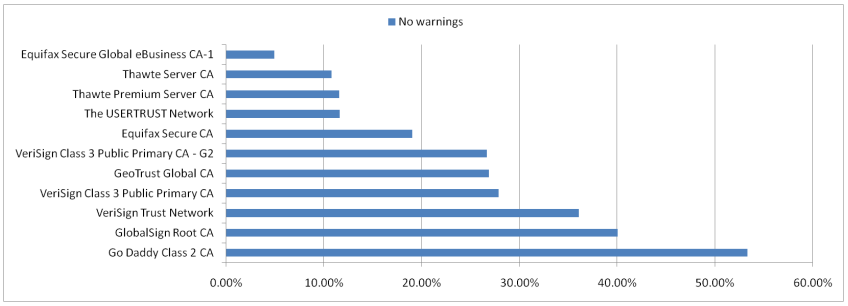
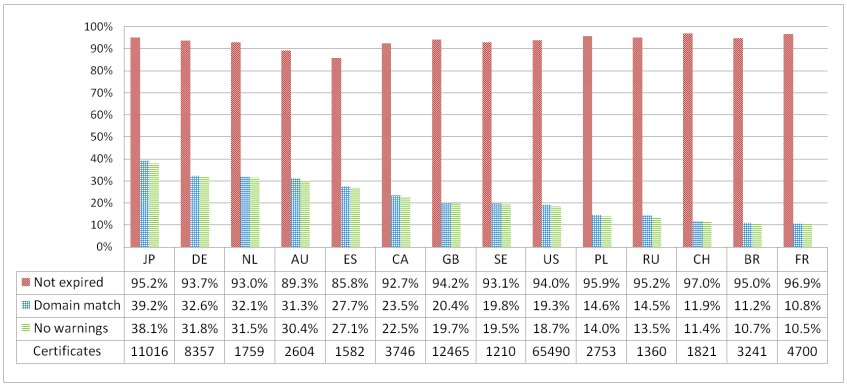


Fig. 14 Authentication success rate across CAs. Certificates issued by GlobalSign, GoDaddy and VeriSign achieve higher authentication success rate. Either they help their clients manage certificates properly or their customers are more security conscious and resourceful and take better care of their certificates.

[6] Even though some CAs (e.g., Equifax and Thawte) were acquired by VeriSign, we refer to them as separate CAs as they offer different products and services and have different policies.



	JP	DE	NL	AU	ES	CA	GB	SE	US	PL	RU	CH	BR	FR
Not expired	95.2%	93.7%	93.0%	89.3%	85.8%	92.7%	94.2%	93.1%	94.0%	95.9%	95.2%	97.0%	95.0%	96.9%
Domain match	39.2%	32.6%	32.1%	31.3%	27.7%	23.5%	20.4%	19.8%	19.3%	14.6%	14.5%	11.9%	11.2%	10.8%
No warnings	38.1%	31.8%	31.5%	30.4%	27.1%	22.5%	19.7%	19.5%	18.7%	14.0%	13.5%	11.4%	10.7%	10.5%
Certificates	11016	8357	1759	2604	1582	3746	12465	1210	65490	2753	1360	1821	3241	4700

Fig. 15 Certificate validity across countries. Organizations from Japan, Germany and Netherlands have the best, whereas France, Brazil and Switzerland have the poorest practices in deploying certificates. The major reason for authentication failure is due to domain mismatch, as most of the certificates are not expired.

5.7.2 Authentication Failure Rate wrt. Countries

We investigate whether organizations from different countries differ in the way they deploy certificates. In Figure 15, we show properties of trusted certificates with valid signatures for organizations across several countries. We consider countries for which we observed more than $1'000$ certificates. We compute the statistics based on the total number of trusted valid certificates we have collected for each country (the last row in Fig. 15). The results confirm that the major reason for authentication failure is due to domain mismatch, as most of the certificates are not expired. Therefore, the total percentage of certificates that do not cause any certificate warnings is dictated by the certificates being properly deployed for the domain they are issued for. We observe that organizations from Japan are most successful in the proper certificate deployment, having successful authentication with 38.1% of certificates. Second best are organizations from Germany with 31.8% of their certificates leading to successful authentication, followed by Netherland with 31.5%. The US is in the middle, having a percentage 18.7% that is closer to the average number observed across the top 1 million websites (16.02%). Poorest deployment practices are in France, Brazil and Switzerland. The major factor for a low authentication success rate among Swiss websites is due to the fact that many of them are hosted by an Internet hosting provider that serves its certificate for each hosted website.

5.7.3 Authentication Failure Rate wrt. Website Categories

If we look at the authentication success across different categories of websites, firstly we observe that websites from Computer category have a remarkably high percentage 70.25%. Typically sites of technological companies belong to this cate-

gory and it seems that they have a good know-how and understand the relevance of properly deploying certificates. Reference, Regional and expectedly Business category are also significantly better than the average with more than 40%. It is understandable as Reference sites include University sites, Business websites have e-commerce services and Regional include tech companies such as Google, Yahoo, and Apple. Sports, News, Home and Adults category have the lowest number.

Table 2 Certificate deployment across website categories.

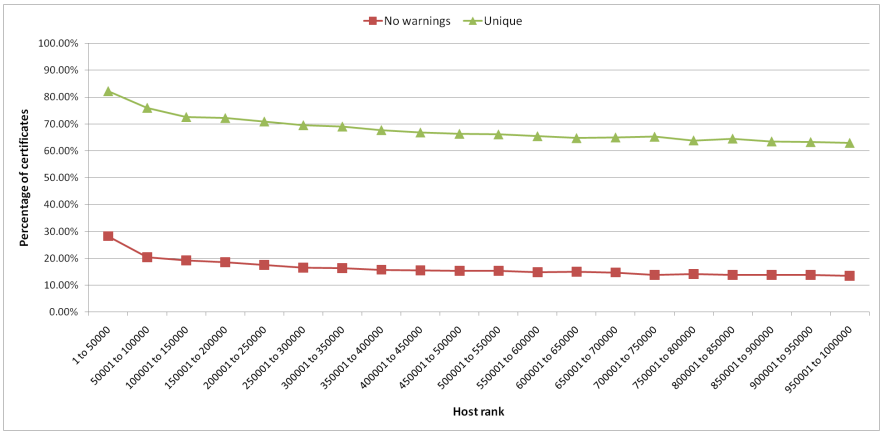
Category	Total	Trusted	No Warnings
Computers	121	109 (90.08%)	85 (70.25%)
Reference	133	116 (87.22%)	70 (52.63%)
Business	130	122 (93.85%)	57 (43.85%)
Regional	99	93 (93.94%)	43 (43.43%)
Shopping	129	126 (97.67%)	50 (38.76%)
Recreation	129	105 (81.39%)	45 (34.88%)
Kids and Teens	87	71 (81.60%)	29 (33.33%)
Games	113	87 (76.99%)	35 (30.97%)
Society	126	97 (76.98%)	39 (30.95%)
Arts	75	50 (66.67%)	23 (30.67%)
Science	131	101 (77.09%)	40 (30.53%)
Health	146	115 (78.77%)	41 (28.08%)
Adult	100	61 (61.0%)	26 (26.0%)
Home	103	73 (70.87%)	26 (25.24%)
News	85	64 (75.29%)	18 (21.18%)
Sports	93	71 (76.34%)	13 (13.9%)

5.7.4 Authentication Failure Rate wrt. Websites Ranks

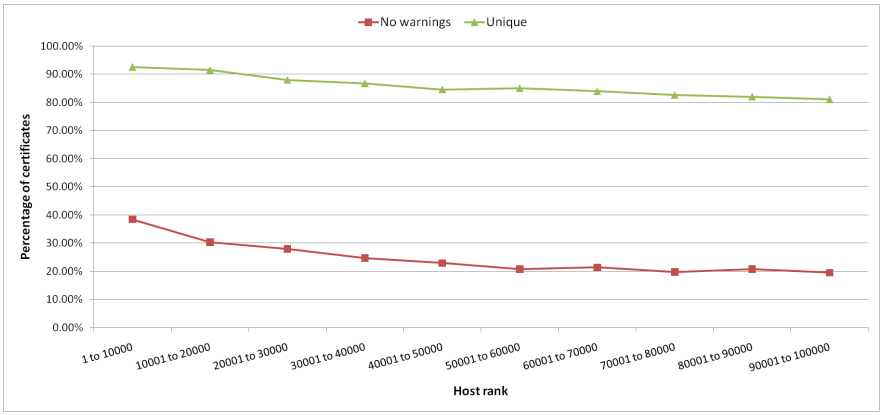
We looked at how the authentication success changes with respect to websites' rank. We divide the ranked 1 million websites into bins of 50′000 websites each, and compute the number of certificates found among those 50′000 websites that allow for a proper authentication and the number of unique certificates (the corresponding two plots in Figure 16(a)). The number of certificates with a certain property is expressed in percentages with respect to the total number of certificates found in the corresponding bin. We observe that the authentication success is significantly better for the first 50′000 websites and then it decreases for lower ranks. This is expected as popular websites generate more revenue from users' traffic and thus may afford better security practices (or perhaps because better security practices attract more users to these websites). We provide in Appendix a few examples of well ranked websites that suffer from authentication failures.

Given that certificate reuse across domains contributes to domain mismatch and leads to authentication failure, we also found the number of unique certificates. One may notice a strong correlation between the shapes of the two curves, authenti-

cation success and unique certificates, which might confirm that indeed certificate reuse across domains is a significant contributor to authentication failure. Since we observe higher dynamics for the highest ranks, we zoom into the highest 100′000 ranked websites (Figure 16(b)). We draw the same conclusions as for 1 million websites and observe correlations between all the rank, the authentication success rate and the usage of unique certificates.



(a) Top 1 million websites.



(b) Top 100′000 websites.

Fig. 16 Certificate deployment properties vs website rank. It appears that the proper certificate deployment, in terms of authentication success and using unique certificates, is correlated to the rank. Higher ranked websites have better practices in implementing certificates properly.

6 Discussion

We outline and interpret most interesting results of Section 5 where we obtained several weaknesses of certificate-based authentication leading to security failures. Economic, legal and social reasons may explain these issues.

6.1 Failures

Out of top one million websites, about 35% can be browsed via HTTPS. Unfortunately, most of them poorly implement certificate-based authentication and generate authentication problems. *Only about* 48′158 *websites (*16.02*% of the ones with verifiable certificate signatures) have valid certificates*, i.e., certificates issued by trusted CAs, not expired, deployed on domains they are issued for, and with verifiable signatures.

Successful authentication does not necessarily mean that users may trust authenticated websites. CAs increasingly issue domain-validated only certificates, for which they only verify that the applying entity has registered for the requested domain. Such validation process may not guarantee the legitimacy of certificates and lead to man-in-the-middle attacks (putting users' security at risk). Consequently, users should not systematically trust all websites that their browsers trust. Our results show that 61% of valid certificates are DVO. This reduces the number of websites that users can fully trust to 18′785. Essentially, *only* 5.7*% of the websites that implement HTTPS properly implement certificate-based authentication* and enable users to securely establish HTTPS connections.

6.1.1 Economics

Our investigations showed many domain mismatches were due to improper handling of certificates by websites. Several reasons can explain this, mostly boiling down to *misaligned incentives*. Websites typically offer several services on different subdomains and should obtain a certificate for each of them. This is complex, as it requires technical understanding, and expensive, as it requires obtaining several certificates. Hence, website operators often prefer to reuse a single-domain certificate across a number of (sub)domains, leading to domain mismatches. For example, if people lacking technical know-how (e.g., business managers) were responsible for obtaining certificates, they may focus on cost reduction, whereas people with technical know-how (e.g., engineers) may not invest sufficient time to carefully design certificate-based authentication systems. Certificate management is a cost and does not directly generate revenue compared to other services. Hence, *most website operators have an incentive to obtain cheap certificates*.

CAs are also culprits for authentication failures. CAs' business model depends on the price and the number of certificates sold. From an economic point of view, *CAs*

have an incentive to distribute as many certificates as possible in order to increase profit. CAs segment their market and apply differentiate pricing. Consequently, they created different forms of certificates: Domain-validated only certificates, EV certificates and regular certificates.

In our results, we observed that *most website operators choose cheap certificates leading to cheap Web authentication*. Domain-validated only certificates are popular amongst small websites because they are easy and fast to obtain. They require minimum effort from CAs. Several CAs even offer free trials where websites can be certified for free for short periods of time. EV certificates differ from regular certificates and domain-validated only certificates in that they require rigorous verifications. They are a preferred option for large websites dealing with important user information. Information asymmetry plays a large role in pushing cheap certificates. As website operators cannot tell the difference between good and bad security (i.e., market for lemons), they might as well take the cheaper option, thus pushing race to the bottom price.

A positive result is the low number of expired certificates we observed. This is probably because CAs strongly encourage renewal to increase revenue. This shows that *CAs could provide incentives to push proper adoption of certificates*. Yet, most trusted certificates were not deployed properly showing that CAs do not make that investment.

6.1.2 Liability

Liability should be assigned to the party that can best manage risk. Unfortunately, *most CAs transfer their liability onto their customers*. This reduces their risk and involvement in providing security. For example, Verisign License agreement version 5 [1] states that Verisign "SHALL NOT BE LIABLE FOR (I) ANY LOSS OF PROFIT, BUSINESS, CONTRACTS, REVENUE OR ANTICIPATED SAVINGS, OR (II) ANY INDIRECT OR CONSEQUENTIAL LOSS". It caps to $5000 for total liability for damages sustained. This exhibits a serious flaw of the system: although CAs distribute an essential element of Web security, they do not take responsibility for it.

In this three-body problem, CAs pass their liability onto websites that in turn transfer it to users through their website policies. This tendency to push liability to others is another example of misaligned incentives reinforced by information asymmetry. CAs are not encouraged to protect users and rather focus on risk-reducing strategies. This problem appears in other situations, such as security economics of banking [24]. It may be difficult to expect full liability from CAs but a reasonable involvement could dramatically improve the current situation.

Lack of liability is known by economists to generate a moral-hazard effect [24]: As CAs know that customers cannot complain, they tend to be careless in the distribution of certificates, leading to erroneously distributed certificates such as [19].

6.1.3 Reputation

Man-in-the-middle attacks caused by the use of weak certificates can harm websites' and CAs' reputation. Hence, CAs should have an incentive to provide good security. Our results show that *well-established CAs tend to properly issue certificates and rely on a number of less prominent subsidiaries to issue certificates far less rigorously*. This helps preserve their reputation, while not missing good business opportunities. In addition, our results show that CAs tend to provide short-lived certificates, e.g., for one year. This limits the risk of bad publicity in case a malicious website is actually authenticated.

For websites, we observe that mostly large corporations get EV certificates in order to limit risk for their customers. Even if they could afford the cost of a MitM attacks, they wish to protect their own reputation and provide good security. *Most less exposed websites select domain-validated only certificates.* In other words, they are fine with cheaper certificates. This may be because website administrators underestimate the value of the data they handle and wish only to reduce security costs. In addition, peer influence from other websites adopting similar weak security practices, may encourage websites administrators to choose domain-validated only certificates.

6.1.4 Usability

For most users, security is secondary as they seek offered services. The variety of options of certificate-based authentication (e.g., domain validated, EV certificates, self-signed certificates and notion of certificates) actually makes it difficult for users to understand the system. Users may misinterpret security warnings as annoyances that prevent them from using Web services. Bad certificate management leads to more security warnings. *The more interruptions users experience, the more they learn to ignore security warnings*. This is counter-productive. Regardless of how compelling, or difficult to ignore SSL warnings are, users may think they are of little consequence because they also see them at legitimate websites [27]. A recent study about SSL warnings' effectiveness shows that users' attitudes and beliefs about SSL warnings are likely to undermine certificates' effectiveness [39] and it suggests to avoid warnings altogether and make security decisions on behalf of users.

Finally, *Web browsers have little incentive to limit access to websites* whose certificates are issued by untrusted CAs and thus stop users from accessing websites they could access from other browsers. Firefox currently tries to discourage users from communicating to websites with self-signed certificates by showing users complex warnings. Such approach spurred agitated debates on the usability of Firefox for Web authentication. In addition, we have seen that unfortunately there are situations (with domain-validated certificates) where users cannot entirely rely on browsers to help them decide whom to trust.

6.2 Countermeasures

We observe that proper incentives to secure the certificate-based authentication systems are missing. The current deployment of digital certificates, mostly based on self-regulation, is moving towards a business model that does not put the emphasis on security and needs a change. We suggest multiple regulation options to modify incentives and improve the situation.

- **New Third-Parties:** An independent third-party could change the current equilibrium of the system. This third-party could be managed by users with an open website (e.g., wiki), by an association of CAs or by Web browsers directly. Basically, such third-party could interfere with the current free-market approach to introduce information related to performances of CAs, and steer the system in a better direction.
  This independent third-party could provide transparency by providing information similar to our results about security performances of CAs (Fig. 14). This may stimulate competition among CAs to provide better security. CAs would actually have to worry about how certificates are used by websites. Similarly, it could agree with a small set of trusted root CAs, more transparent, hierarchical and localized. Finally, it could also monitor how well websites use certificates and rate websites based on the security they provide.
  Users could also run themselves a third-party to form groups of users sharing information with each other. This could reduce the problem of asymmetric information.
- **New Policies:** Changing legal aspects is a difficult and slow process, but may be very effective. It is important that CAs take responsibility for certificate-based authentication. They should be liable for the security of the system as responsibility should follow those that earn revenue. In order to tackle the asymmetric information problem, previous work suggests the use of certification schemes in order to guarantee the quality of provided certificates [33]. Such certificates could be operated by governments (e.g., Orange book) or commercial companies (e.g., Common criteria). However, regulation is costly. One-model-fits-all approach is hard to put in place, especially for smaller companies [28].
  Another option is to force websites to be responsible for properly implementing certificate-based authentication. However, websites are customers of the system and it is difficult to blame them for not understanding how to invest money in security.
  Finally, Web browsers could pressure CAs in order to improve the quality of CAs' practices. For example, Web browsers could have the policy to trust only the top performing root CAs in terms of provided security.
  In general, even though websites generate most authentication failures, we believe that policies should focus on certification authorities and Web browsers.

7 Conclusion

We crawled the top one million popular websites and investigated how they deploy certificate-based authentication. Our results show that nearly one-third of websites can be browsed with HTTPS, but only 18′785 (5.7%) of them properly implement certificate-based authentication. In other words, only 5.7% of the websites that implement HTTPS, do so without causing security warnings in Web browsers and with providing trust in the identities of certificates' owners.

We discuss multiple reasons that may have led to the failure of the current model for Web security. We argue that the current free market approach, where utility-optimizing entities try to maximize profits at minimum cost, is the root of the problem. We can compare the current situation to a *market for lemons*: information asymmetry occurs because CAs know more about certificates than websites and users. Hence, most website administrators acquire cheap domain-validated only certificates and poorly implement them on their servers. Only a fraction of elite website administrators achieves high security by obtaining EV certificates and installing them properly. We also observe strategic behavior from CAs that rely on subsidiaries to sell less trustworthy certificates and maximize profits. This situation is not satisfactory as it affects the global security of the Internet ecosystem. We believe that the right incentives are not in place and suggest multiple policy changes to solve this issue. Notably, we suggest to make CAs liable for the proper use of certificates, web browsers to trust only top performing CAs, and the creation of an open-source community checking on root CAs.

Acknowledgements We would like to thank Jens Grossklags for his valuable insights and feedback.

Appendix

Table 3 Top 5 websites by categories

	Rank (within category)				
Category	1	2	3	4	5
Adult	LiveJasmin.com livejasmin.com	Youporn youporn.com	XNXX Galleries xnxx.com	Adult Friend Finder adultfriendfinder.com	Streamate.com streamate.com
Arts	Facebook facebook.com	YouTube - Broadcast yourself youtube.com	The Internet Movie Database imdb.com	BBC Online bbc.co.uk	CNN Interactive cnn.com
Business	PayPal paypal.com	Yahoo Finance finance.yahoo.com	ESPN espn.go.com	Alibaba.com alibaba.com	EzineArticles.com ezinearticles.com
Computers	Google google.com	Facebook facebook.com	YouTube - Broadcast yourself youtube.com	Yahoo! yahoo.com	Gmail mail.google.com
Games	IGN ign.com	GameSpot gamespot.com	Pogo.com pogo.com	MiniClip.com miniclip.com	Yahoo! Games games.yahoo.com
Health	National Institutes of Health (NIH) nih.gov	WebMD webmd.com	PubMed ncbi.nlm.nih.gov/pubmed/	Mercola mercola.com	Focus on Digestion medicinenet.com
Home	Yahoo Finance finance.yahoo.com	eHow ehow.com	Yelp yelp.com	Open DNS opendns.com	Google Product Search google.com/products
Kids and Teens	W3 Schools w3schools.com	Thesaurus.com thesaurus.reference.com	GameSpot gamespot.com	Weebly weebly.com	Universal Currency Converter xe.com/ucc/
News	Yahoo News news.yahoo.com	BBC Online bbc.co.uk	CNN Interactive cnn.com	The New York Times nytimes.com	BBC News bbc.co.uk/news/
Recreation	Metacafe metacafe.com	TripAdvisor tripadvisor.com	Booking.com booking.com	Expedia.com expedia.com	XE.com xe.com
Reference	Yahoo! Answers answers.yahoo.com	Google Maps maps.google.com	StumbleUpon stumbleupon.com	Stack Overflow stackoverflow.com	WikiAnswers - Q&A wiki wiki.answers.com
Regional	Google google.com	Yahoo! yahoo.com	Google India google.co.in	Amazon.com amazon.com	Google UK google.co.uk
Science	Google Translate translate.google.com	NCBI ncbi.nlm.nih.gov	CNET News.com news.cnet.com	Urban Dictionary urbandictionary.com	Time and Date timeanddate.com
Shopping	Amazon.com amazon.com	eBay ebay.com	Netflix netflix.com	Amazon.co.uk amazon.co.uk	Wal-Mart Online walmart.com
Society	Digg digg.com	deviantART deviantart.com	OMG omg.yahoo.com	hi5 hi5.com	Yahoo! Shine shine.yahoo.com
Sports	AOL aol.com	ESPN espn.go.com	Yahoo Sports sports.yahoo.com	NBA.com nba.com	Yahoo! Sports: NBA sports.yahoo.com/nba/

Table 4 Top websites' implementation failures

Rank	Host	Cause of failure
31	fc2.com	Domain mismatch (CN=fc2server.com)
269	techcrunch.com	Domain mismatch (CN=*.wordpress.com, wordpress.com)
322	nfl.com	Domain mismatch (CN=a248.e.akamai.net, *.akamaihd.net)
336	stackoverflow.com	Domain mismatch (CN=stackauth.com, *.stackauth.com)
377	39.net	Self-signed & Domain mismatch (CN=cms.39.net)
394	www.informer.com	Expiration

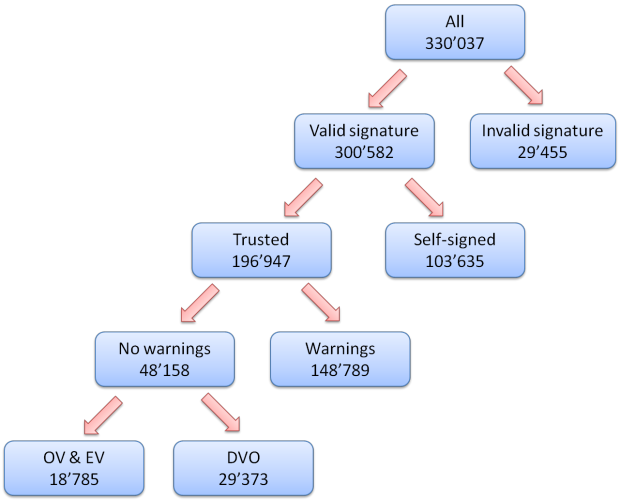


Fig. 17 Data set of certificates used in the survey.

References

1. VeriSign Inc. URL http://www.verisign.com/ssl/buy-ssl-certificates/secure-site-services/index.html
2. The SSL Protocol, Version 3.0 (1996). URL http://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00
3. Internet X.509 Public Key Infrastructure Certificate and CRL Profile (1999). URL http://www.ietf.org/rfc/rfc2459.txt
4. The TLS Protocol, Version 1.0 (1999). URL http://tools.ietf.org/html/rfc2246
5. HTTP Over TLS (2000). URL http://tools.ietf.org/html/rfc2818
6. Cardholders targetted by Phishing attack using visa-secure.com (2004). URL http://news.netcraft.com/archives/2004/10/08/cardholders_targetted_by_phishing_attack_using_visasecurecom.html
7. Transport Layer Security (TLS) Extensions (2006). URL http://tools.ietf.org/html/rfc4366
8. Has Firefox 3 certificate handling become too scary? (2008). URL http://www.betanews.com/article/Has-Firefox-3-certificate-handling-become-too-scary/1219180509
9. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (2008). URL http://tools.ietf.org/html/rfc5280
10. Tim Callan's SSL Blog, MD5 attack resolved (2008). URL https://blogs.verisign.com/ssl-blog/2008/12/on_md5_vulnerabilities_and_mit.php
11. EV and SSL Certificate Trends For The Top 100 Retailers (2010). URL http://www.lexiconn.com/blog/2010/09/ev-ssl-top-100-retailers/
12. Guidelines For The Issuance And Management Of Extended Validation Certificates (2010). URL http://www.cabforum.org/Guidelines_v1_3.pdf
13. Alexa the Web Information Company (2011). URL http://www.alexa.com/
14. Home of the Mozilla Project (2011). URL http://www.mozilla.org/

15. Improving SSL certificate security (2011). URL http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html
16. OpenSSL: Documents, verify(1) (2011). URL http://www.openssl.org/docs/apps/verify.html
17. OpenSSL: The Open Source toolkit for SSL/TLS (2011). URL http://www.openssl.org/
18. SQLite Home Page (2011). URL http://www.sqlite.org/
19. SSL Certificate for Mozilla.com Issued Without Validation (2011). URL http://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html
20. The EFF SSL Observatory — Electronic Frontier Foundation (2011). URL http://www.eff.org/observatory
21. Trusted Certificates vs. Browser Recognized Certificates (2011). URL http://www.instantssl.com/ssl-certificate-support/guides/ssl-certificate-validation.html
22. What are the types of SSL Certificates? (2011). URL http://www.globalsign.com/ssl-information-center/what-are-the-types-of-ssl-certificate.html
23. Ahmad, D.: Two Years of Broken Crypto: Debian's Dress Rehearsal for a Global PKI Compromise. IEEE Security and Privacy **6**, 70–73 (2008)
24. Anderson, R., Moore, T.: Information security economics–and beyond. CRYPTO pp. 68–91 (2007)
25. Biddle, R., Oorschot, P.C.V., Sobey, J., Whalen, T., Patrick, A.S.: Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study. In: CCSW'09: Proceedings of the 2009 ACM workshop on Cloud computing security (2009)
26. Dhamija, R., Tygar, J.D., Hearst, M.A.: Why phishing works. In: Computer Human Interaction, pp. 581–590 (2006)
27. Downs, J.S., Holbrook, M.B., Cranor, L.F.: Decision strategies and susceptibility to phishing. In: Symposium On Usable Privacy and Security, pp. 79–90 (2006)
28. Ghose, A., Rajan, U.: The economic impact of regulatory information disclosure on information security investments, competition, and social welfare. In: WEIS (2006)
29. Good, N., Dhamija, R., Grossklags, J., Thaw, D., Aronowitz, S., Mulligan, D.K., Konstan, J.A.: Stopping spyware at the gate: a user study of privacy, notice and spyware. In: Symposium On Usable Privacy and Security, pp. 43–52 (2005)
30. Herzberg, A., Jbara, A.: Security and identification indicators for browsers against spoofing and phishing attacks. ACM Trans. Internet Technol. **8**, 16:1–16:36 (2008)
31. Jackson, C., Barth, A.: Forcehttps: Protecting high-security web sites from network attacks. In: WWW (2008)
32. Jakobsson, M., Myers, S.: Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. Wiley-Interscience (2006)
33. Landwehr, C.: Improving information flow in the information security market. Economics of Information Security pp. 155–163 (2004)
34. Lenstra, A.K.: Key length (2004)
35. Moore, T., Edelman, B.: Measuring the Perpetrators and Funders of Typosquatting. Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010)
36. Schechter, S.E., Dhamija, R., Ozment, A., Fischer, I.: The Emperor's New Security Indicators. In: IEEE Symposium on Security and Privacy, pp. 51–65 (2007)
37. Sogohian, C., Stamm, S.: Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. In: HotPETs (2010)
38. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., Weger, B.: Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In: Cryptology Conference on Advances in Cryptology, pp. 55–69 (2009)
39. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In: USENIX Security Symposium, pp. 399–416. USENIX Association (2009)

40. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing . In: USENIX Annual Technical Conference (Usenix ATC) (2008)
41. Whalen, T., Inkpen, K.M.: Gathering evidence: use of visual security cues in Web browsers. In: Proceedings of Graphics Interface 2005, GI '05, pp. 137–144. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (2005)