2014 IEEE Symposium on Security and Privacy

When HTTPS Meets CDN: A Case of Authentication in Delegated Service

Jinjin Liang*†‡, Jian Jiang*†‡§, Haixin Duan†‡¶, Kang Li‖, Tao Wan **, Jianping Wu*†‡,
*Department of Computer Science and Technology, Tsinghua University
†Institute for Network Science and Cyberspace, Tsinghua University
‡Tsinghua National Laboratory for Information Science and Technology
§University of California, Berkeley ¶International Computer Science Institute
‖Department of Computer Science, University of Georgia **Huawei Canada

***Abstract*—Content Delivery Network (CDN) and Hypertext Transfer Protocol Secure (HTTPS) are two popular but independent web technologies, each of which has been well studied individually and independently. This paper provides a systematic study on how these two work together. We examined 20 popular CDN providers and 10,721 of their customer web sites using HTTPS. Our study reveals various problems with the current HTTPS practice adopted by CDN providers, such as widespread use of invalid certificates, private key sharing, neglected revocation of stale certificates, and insecure back-end communication. While some of those problems are operational issues only, others are rooted in the fundamental semantic conflict between the end-to-end nature of HTTPS and the man-in-the-middle nature of CDN involving multiple parties in a delegated service. To address the delegation problem when HTTPS meets CDN, we proposed and implemented a lightweight solution based on DANE (DNS-based Authentication of Named Entities), an emerging IETF protocol complementing the current Web PKI model. Our implementation demonstrates that it is feasible for HTTPS to work with CDN securely and efficiently. This paper intends to provide a context for future discussion within security and CDN community on more preferable solutions.**

I. Introduction

Content Delivery Networks (CDNs) are widely deployed to improve the performance, scalability and security of web sites. They were originally used to reduce the latency of web access by redirecting the user to a surrogate server (or cache server) close to the user, as well as to lighten the load of original web servers. In recent years, CDN providers also start to offer DDoS mitigation services by hiding the original web site and distributing the load of attack traffic to multiple surrogate servers. By deploying web application firewalls on cache servers, CDNs can also filter intrusions against original servers.

With CDNs, web access terminates at one of the surrogate servers distributed across the Internet, returning cached content. However, this "man-in-the-middle (MITM)" model introduces additional complexity in other techniques that were designed for end-to-end communication. HTTPS (or HTTP over TLS) is one such end-to-end protocol, which establishes encrypted tunnels to deliver sensitive information between clients and web servers. Web server operators can obtain certificates from a Certificate Authority (CA), which is trusted by both the server and client browsers. On accessing an HTTPS enabled web site, a client can validate the server's identity by verifying the server's certificate (e.g. whether it is issued by a trusted CA, and whether the server domain name matches the information listed in the certificate).

However, when a CDN (the "man-in-the-middle") is used, the CDN server cuts in the middle of HTTPS communications, and splits HTTPS into two parts: the front-end communication between end-user and CDN surrogate server, and the back-end communication between CDN surrogate server and original web server. In this case, the trust model and the establishment of the secure tunnel between two parties (a client and a web server) now involve three parties. While the back-end interaction is similar to original HTTPS, the front-end communication becomes complicated. Because adding an additional party in the HTTPS communication not only requires changes to the setup of the secure tunnel (such as using a different certificate), but also requires additional user awareness and delegation control, none of which need to be considered in the pure two party end-to-end HTTPS model. Specifically, when the owner of a web site delegates his authentication information of HTTPS to some CDN providers, there should be a mechanism that informs end-users of the delegation. Moreover, the web site owner should be able to efficiently and independently revoke his/her delegation from a CDN provider at his/her own will (without the need of an approval from the current CDN provider, *e.g.* in the case of changing CDN providers).

This paper studies the current practices of using HTTPS with CDNs. For the front-end communication, we investigated 20 popular CDN providers and 10,721 of their customer web sites. These web sites enable HTTPS access and use CDN through DNS based request-routing, which is a dominant mechanism to adopt CDN service in the Internet. Among these 10,721 web sites using HTTPS with CDNs, we observed that 15% of them raised alerts of invalid certificates, which broke the trust model of HTTPS. For those without certificate warnings, we observed that they used two types of certificates: *Custom Certificate* and *Shared Certificate*.

A *Custom Certificate* requires web site owners to upload

DOI 10.1109/SP.2014.12

their certificates and private keys to CDN providers. Essentially, sharing private keys between web sites and CDN providers violates the fundamental setting of public key cryptography. Practically, the owners of the original web sites are exposed to more security risks by sharing private keys with CDN providers since CDN providers may distribute this sensitive information to all their nodes across the Internet. Moreover, web sites cannot revoke their delegations from CDN providers independently and efficiently.

In the case of *Shared Certificate*, the CDN relies on a partner CA to issue a certificate valid for multiple domain names. To ensure web clients receiving a valid certificate, the CDN provider adds the customer's domain name into the Subject Alternative Name (SAN) extension [1] of his certificate. However, the proof of delegation, expressed by the shared certificate only, is not complete (see §IV-A2), which results in the loss of the functionality of HTTPS in displaying proper security indicators to end-users. For example, assume the web site owner has applied for an EV (Extended Validation) certificate to enhance the web site's assurance level, then he will have no way to show it to web site's users, but to share a low level DV (Domain Validated) certificate indicator belonging to his CDN provider. What's more, our experience shows that there exists a problem of delegating revocation in this mechanism as well.

For the back-end communication, we measured the behavior of five CDN providers and found that they were all far from perfect. Two of them used HTTP rather than HTTPS for back-end communication. The other three, although they used HTTPS, did not perform proper authentication when establishing the secure channel, and thus were vulnerable to MITM attack.

To address the challenges of deploying HTTPS with CDN, we first examine a potential solution using an existing technique called name constraint certificate. In this approach, the web site owner plays the role of subordinate CA to issue certificates to CDN providers, constrained to the owner's domain. Although this solution is theoretically feasible and without any protocol modification, we consider that it is not practical in deployment for the following three reasons. First, we found a vulnerability in some popular web browsers that could be used to bypass name constraints easily. Second, the approach poses heavy overhead on web site owners because of the need of running a subordinate CA. Further, commercial CAs are unlikely motivated to allow their customers being subordinate CAs because of heavy vetting and auditing responsibilities.

We then propose another solution by extending an emerging technique called DANE. In this solution, the web site's owner could show his delegation explicitly with his TLSA records which associate both the web site's and the CDN provider's certificates. And thus the end-user can verify the identities of both the original web site and the CDN provider, as well as the delegation between them. Our analysis and implementation show that this solution could address the problem of HTTPS in CDN effectively.

In summary, we make the following contributions in this paper:

- analysis on the problems and challenges for deploying HTTPS in CDN;
- measurements to investigate current techniques for HTTPS in CDN providers , identifying their defects and practice issues;
- the discovery and experiment on the problem of X.509 certificate name constraints for HTTPS usage;
- a lightweight and flexible DANE-based solution that addresses HTTPS authentication problem in CDN environment.

The remainder of the paper is organized as follows. We review the background of CDN and HTTPS in Section II. Section III explains the problems and challenges of composing these two together. Section IV measures the status quo and analyzes the defects. We then present why we consider the name constraint certificate as a practically questionable solution in Section V and propose a new solution based on DANE in Section VI. We also give further discussions on the problem in Section VII. Section VIII describes related work and Section IX gives a conclusion.

II. Background

A. CDN

Overview. A CDN is a distributed infrastructure that efficiently delivers web-related content to end-users. Originally, CDN service was used to reduce the latency of accessing the web site for users as well as lighten the load on web site's origin server. Recently, CDN providers also offer new security services for web sites, such as DDoS protection and Web Application Firewall (WAF).

A CDN is usually composed of a large number of surrogate servers distributed all around the world. If a web site uses the CDN service, a subset of the surrogate servers in the CDN will replicate that web site's content, either by pull or by push method. When users access the web site, they will be directed to the CDN and finally get the content from a nearby surrogate server rather than the web site's origin server.

Request-routing Mechanism. Request-routing techniques are the key component for CDN services since they are responsible of directing user requests from the original web site to the CDN and further to the appropriate surrogates, according to various policies and metrics. Many request-routing techniques are introduced in [2], but in this paper, we only focus on the three most common techniques: URL rewriting, CNAME and domain hosting.

- **URL Rewriting.** URL rewriting modifies the URL of specific content (*e.g.* images, css, scripts) in the origin web site. Thus when users access the web site and load

the content with modified URL, they will switch to visit the CDN to get the content.

- **CNAME.** CNAME (canonical name) is a type of DNS record that links a domain name to another name. By using CNAME records, the web site owner could point his domain name to a CDN's domain name as an alias, so that when users visit the web site, they will be eventually redirected to the CDN's domain name through DNS resolution, which is translated to IP addresses of some surrogates according to the CDN's policy at last.
- **Domain Hosting.** Domain hosting means a web site uses CDN's DNS server as the authoritative name server for its domain. Thus the resolution of the web site's domain name is controlled by the CDN provider, who directly points the web site's domain name to the IP addresses of its surrogate servers.

All of the three request-routing techniques have their own advantages and limitations. While URL rewriting technique offers fine grained redirection control for web sites, it requires content modifications in the origin web sites, which is tedious and error-prone; URL rewriting is also not applicable for DDoS or WAF protection, which usually require domain-level redirection. CNAME and domain hosting offer great convenience and flexibility that address the limitations of URL rewriting, however, they also lose URL rewriting's fine grained redirection control. Besides, CNAME could introduce additional overhead of DNS resolution.

B. HTTPS

Overview. HTTPS provides secure end-to-end communication channels between web servers and clients. In essence, HTTPS simply layers HTTP on the top of the Transport Layer Security (TLS) protocol, which provides a number of security primitives, such as authentication and encryption, against passive eavesdroppers and active attackers.

Concretely, HTTPS relies on the X.509 certificate and public key infrastructure (PKI) for server authentication. In the X.509 [3] system[1], a certificate is signed by a trusted certificate authority (CA) to bind a public key with a domain name. When accessing a server of a web site, a client first validates the certificate of the web site, and then uses the associated public key in the certificate to negotiate a session key with the server for further secure communications.

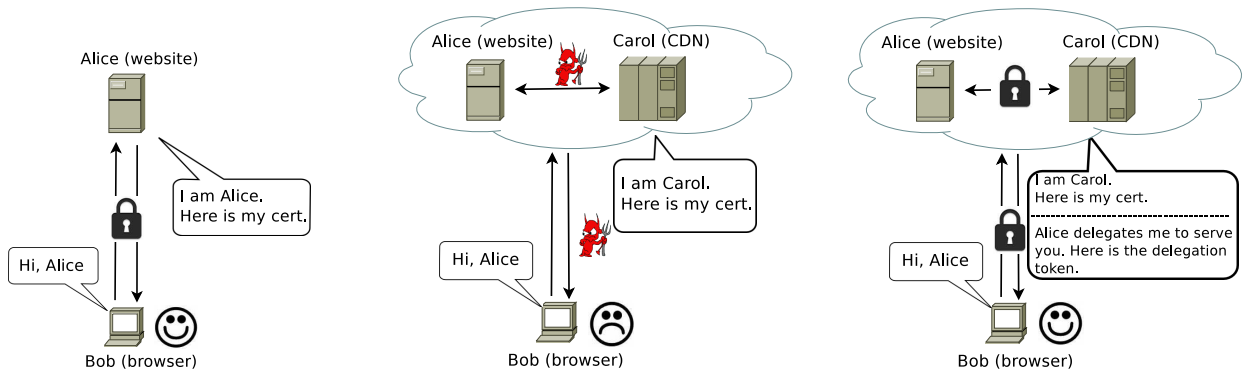
Certificate Validation. Modern web browsers perform certificate validation in three steps: chain validation, name validation and revocation check. If any step fails, browsers will show users various warnings to indicate potential risks of invalid certificates.

- **Chain Validation.** In current practice, a number of trusted root CAs are distributed with browsers or operating systems by default. Usually these root CAs will not directly issue server certificates, instead they delegate their signature ability to intermediate CAs that actually sign server certificates. Therefore, normally, a web server presents a complete certificate chain containing its certificate as well as all the intermediate CA certificates when performing a TLS/SSL handshake. A browser then verifies whether the certificates can form a complete chain by checking the signature and the valid period for each certificate, starting at the server certificate and ending at a trusted root CA.
- **Name Validation.** Apart from verifying the certificate chain, the browser also examines the domain name in the certificate to determine whether the certificate pertains to the current web site. The current practice utilizes two fields of a certificate to present its domain name: the Common Name (CN) field, and the Subject Alternative Name (SAN) extension which enables a certificate to include multiple domain names. For convenience, domain names in the certificate may use wildcards to cover all their subdomains (*e.g.* using `*.example.com` to represent all direct sub domains of `example.com`).
- **Revocation Checking.** In many cases, such as private key compromise, a CA needs to revoke an issued certificate before its expiration date. The key to certificate revocation is to publish revoked certificates in time so that a browser can recognize those certificates are invalid even though they pass the above validation. Currently two mechanisms have been widely adopted for certificate revocation: Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP).
  - **CRL.** A CRL contains a signed list containing serial numbers of certificates that are revoked by a CA. Browsers could fetch a current certificate's CRL from the `CRL Distribution Points` extension of a certificate to check its revocation status.

    A big problem of CRLs is the size. As the size of a CRL constantly increases, the overhead of distribution will eventually become unmanageable. Also because of the overhead, CRLs are not updated in a timely manner. Currently the publish periods can be one or more weeks.
  - **OCSP.** OCSP [1] is proposed as an alternative to CRL, which addresses the problems of CRLs by using a real-time protocol. Instead of downloading the whole CRL, a browser using OCSP queries an online server specified in the authority information access (AIA) extension of a certificate to check its revocation status.

    As a real-time protocol, the efficiency of OCSP depends on the capability of the OCSP servers

[1]In this paper, we refer by the X.509 system to the X.509 based public key infrastructure, standardized by the PKIX working group of the IETF, rather than the standards developed by the ITU-T.



(a) In HTTPS, the authentication between Alice and Bob is straightforward.

(b) Together with Carol, *i.e.* CDN provider, the process of secure browsing is split into two ends: front-end and back-end. Both of the two ends need to be protected with HTTPS and certificate authentication against attackers. Specifically, the authentication in the front-end, which involves three parties, cannot be directly addressed with standard HTTPS.

(c) To ensure secure web browsing in this setting, the back-end communication should first be protected with standard HTTPS. In addition, to solve the problem of front-end authentication, Carol should be able to show Bob a delegation token along with her certificate to convince Bob that she is delegated by Alice.

Figure 1. A conceptual view of the authentication problem and solution for composing HTTPS with CDN.

running by CAs. A recent study suggested that the OCSP servers were indeed overwhelmed and OCSP checking brought considerable latencies [4].

A few refinements on top of CRL and OCSP have also been proposed to reduce client-side overhead. Recently, Google developed a proprietary mechanism called CRLSet [5], which is deployed on its Chrome browser. In CRLSet, Google collects the updated CRLs from all CAs, and efficiently pushes the set of CRLs to clients with its global infrastructure. CRLset enables browsers to check CRLs locally and thus avoids network latencies. OCSP stapling [6] is another alternative approach to check the revocation status of certificates. It inserts the time-stamped OCSP response signed by OCSP server into the TLS/SSL handshake, hence clients could check the status of certificates without querying the OCSP server. Currently, OCSP stapling is supported by a number of vendors, such as OpenSSL, Firefox, Apache, and Nginx.

Certificate Types. Currently, commercial CAs provide three kinds of certificates for HTTPS communication: Domain Validated (DV), Organization Validated (OV) and Extended Validated (EV), which have successively higher levels of identity assurance because of the different requirements in identity verification. For issuing a DV certificate, a CA only validates the ownership of the domain name in the certificate request through simple channels such as E-mail. By contrast, OV certificate and EV certificate undergo more rigorous vetting. To issue such certificates, the CA is required to verify ownership of the domain name as well as the actual identity of the domain operator. What's more, while DV certificate usually only contains web site's domain name, OV certificate and EV certificate will also contain the identity information (*e.g.* organization name, country) of the web site.

The three types of certificates also have technical implications. In fact, the goals of HTTPS are not only to secure communication channel between browsers and web servers, but also to notify users to what extent their web surfings are assured by various browser indicators. Different certificate types play different technical roles in the latter part. Specifically, an EV certificate is different from the other two types in that it displays a more visible indicator in browser address bar to attest a highly-assured domain name and its associated web content.

III. When HTTPS Meets CDN: Problems and Challenges

While HTTPS provides server authentication and secure communication between user and web site[2], CDNs enable efficient content delivery. Both play important roles in today's web services. However, we observe that these two techniques cannot work together seamlessly.

Figure 1 depicts a conceptual view of how adoption of CDN changes secure web browsing with HTTPS dramatically. In Figure 1a, when a user accesses a web site (Alice) over HTTPS, the user's browser (Bob) starts by saying hello to Alice, and receives Alice's certificate after establishing a connection. Bob then happily believes the conversation is secure since Alice's certificate binds the connection with his initial hello message. However, upon adopting a CDN service provided by Carol (Figure 1b), the process is split into two ends: in front-end, Bob still starts the conversation

[2]We do not consider certificate-based client authentication of HTTPS with CDN in this paper.

with a hello message to Alice, yet eventually connects to Carol; in back-end, assuming a pull based strategy, Carol needs to fetch the content from Alice upon receiving request from Bob.

Under the scenario of Figure 1b, both of the front-end communication and back-end communication need to be protected by HTTPS with certificate authentication, in order to ensure secure web browsing against passive eavesdroppers and active attackers, as guaranteed in the original HTTPS communication in Figure 1a. However, this is not easy to achieve.

For the back-end communication, it is essential for Carol to authenticate Alice in order to detect impersonation attacks. A mutual authentication, though not necessary, could also help Alice to reject unsolicited requesters early. This is not considered technically challenging to implement with standard HTTPS. Yet, as we shall see in Section IV, this is not always the case in current practice.

The case of the front-end communication is rather complicated, as the conversation actually involves three parties, consequently the authentication cannot be directly addressed by standard HTTPS, which is a two-party protocol (without considering CA). As shown in Figure 1b, if the initial message sent by Bob and the certificate he received do not match, Bob will show user an invalid certificate warning, which undermines the effectiveness of HTTPS authentication from the user's point of view.

Essentially, the problem with front-end authentication is caused by Bob not knowing that *Carol is actually delegated to serve web content on Alice's behalf*. In fact, this problem can be regarded as a case of delegation in a distributed system, which has been generalized in previous literature [7], [8]. The key concepts of the proposed solutions are similar: a delegation token that explicitly expresses the path of delegation. However, the standard HTTPS cannot express such delegation token directly. Therefore, extra effort, as shown in Figure 1c, is needed to overcome this problem.

Previous researches have also suggested several security considerations in designing a delegation token scheme under various threat models. We summarize some of the suggestions that fit our case into the following three requirements:

1) **A delegation token must be unforgeable.** This is an essential requirement to counter active impersonation attacks. Only if a delegation token is verifiable and tamper-proof can a destination (in our case, Bob the browser) trust it in the process of authentication.
2) **Delegator should be able to issue and revoke the delegation token independently and efficiently.** The requirement of delegation revocation is also essential. Without guarantee of revocation, an attacker will still be able to perform impersonation attacks by intercepting and replaying stale delegation tokens. The requirement of delegation issuance comes from operational efficiency.
3) **A delegation token should include complete identification of delegator.** As we shall further discuss in Section IV, this requirement is also necessary to preserve the functionality of HTTPS certificate in displaying proper security indicator.

In the following sections, we shall use the above requirements to examine the defects of existing mechanisms, also to serve as guidelines for exploring new solutions.

IV. The Status Quo

In this section, we investigate how the potential problems discussed above emerge in current practice. At first, we look up the problems of the front-end authentication, which we regard as the most challenging part of composing HTTPS with CDN; then we turn to the back-end.

A. The Front-end

For the front-end authentication, the potential conflict between HTTPS and CDN is that HTTPS does end-to-end authentication between a user and a web site, while the interaction of CDN involves three parties: the user is redirected from the original web site to a surrogate server of CDN through one kind of request routing mechanisms. Thus, whether the problem occurs is determined by the request routing mechanisms. Recall that there are three common request routing mechanisms: URL rewriting, CNAME and domain hosting. Below we analyze each case under the scenario described in Figure 1:

- **HTTPS with URL Rewriting.** HTTPS works well in the URL rewriting case, because the domain name in a URL, serving as an identity, plays a key role in server authentication. If Alice modifies a URL, say `https://alice.com/foo.png`, to `https://alice.carol.com/foo.png`, it is analogous to an explicit message telling Bob that `foo.png` will be served by Carol, thus Bob will be happy with Carol's certificate.
- **HTTPS with CNAME.** HTTPS cannot work directly with CNAME based request routing. Because the redirection happens in DNS resolution, which is not recognized by browsers. In Figure 1, if Bob accesses `https://alice.com/foo.png`, and the domain name `alice.com` is CNAME-ed to `alice.carol.com`, Bob is reluctant to accept Carol's certificate since the domain name in Carol's certificate, say `carol.com`, does not match the original one `alice.com`, and he does not know the underlying CNAME process.
- **HTTPS with Domain Hosting.** Similar to the CNAME case, HTTPS also fails to work directly with domain hosting based request routing.

In summary, certificate name mismatch could occur when a web site enables HTTPS and uses a CDN with DNS based request routing, because the redirection in DNS is

Table I
SURVEY OF HTTPS SUPPORT BY CDN PROVIDERS

CDN Provider	Request-Routing Mechanism	HTTPS Support
Akamai	CNAME / Domain Hosting	Custom
Azure	CNAME	Not Support
Bitgravity	CNAME	Custom
Cachefly	CNAME	Custom
CDNetworks	CNAME	Custom / Shared
CDN77	CNAME	Custom
CDN.net	CNAME	Custom / Shared
Chinacache	CNAME	Custom
Chinanetcenter	CNAME	Custom / Shared
CloudFlare	CNAME / Domain Hosting	Custom / Shared
CloudFront	CNAME	Custom
Edgecast	CNAME	Custom / Shared
Fastly	CNAME	Custom / Shared
Highwinds	CNAME	Custom
Incapsula	CNAME	Custom / Shared
Internap	CNAME	Custom
KeyCDN	CNAME	Custom / Shared
Limelight	CNAME	Custom / Shared
NetDNA	CNAME	Custom / Shared
Squixa	CNAME	Custom / Shared

transparent in the authentication of HTTPS. As we have introduced in Section II, DNS based request routing has various advantages compared with URL rewriting; and it is indeed pervasive in practice. Therefore we believe this problem must be addressed for CDN providers to support HTTPS.

*1) **Survey**:*

By simply searching online, we find this problem has indeed raised many discussions. We also find that while some CDN providers, for example, Microsoft's Azure CDN, do not support HTTPS with DNS based request routing, many others do have this feature. For example, Amazon's CloudFront announced to support HTTPS with CNAME in June 2013. This preliminary information motivates us to conduct a survey to understand the current practice before considering possible solutions.

Methodology. We first aim to understand whether major CDN providers support HTTPS with CNAME or domain hosting, and if so, how they achieve this feature. We empirically investigate 20 well-known CDN providers (see Table I) by reading their technical specifications and contacting their customer services.

Our second goal is to learn the deployment status of HTTPS with DNS based request routing. For this purpose, we first probe domain names in Alexa's top 1 million sites. If a domain has a CNAME or NS names chaining to one of the CDN providers in Table I, we consider it a site deploying CDN by DNS based request routing, which we refer to as a *DNS-CDN-enabled site*. For each DNS-CDN-enabled site, we then access it with HTTPS and record the response.

Results. Table I shows the results of surveying HTTPS support in CDN providers, from which we see that 19 out of 20 investigated CDN providers support HTTPS with DNS based request routing (mostly CNAME). They develop two techniques called "Custom Certificate" and "Shared Certificate" to achieve this feature. We analyze these techniques in detail later.

Table II
HTTPS STATUS OF DNS-CDN-ENABLED SITES

HTTPS Status		# of web sites	%
Valid Cert	Custom Cert	2152	20.1%
	Shared Cert	1198	11.1%
Invalid Cert	Status 200	1637	15.3%
	Others	5734	53.5%
Total		10,721	100%

Table II presents the statistics of HTTPS status of DNS-CDN-enabled sites. In total, we observed 10,721 out of 14,199 DNS-CDN-enabled sites were reachable with HTTPS. 31.2% of all HTTPS reachable sites showed valid certificates. Among those sites, 64.2% (20.1% of all HTTPS reachable sites) used custom certificates; the rest used shared certificates. 68.8% of all HTTPS reachable sites showed invalid certificate warnings, among which only 22.2% (15.3% of all HTTPS reachable sites) ended up showing valid web pages (HTTP status code 200), others were either redirected back to HTTP (30x), or responded with errors (40x or 50x).

This survey is not comprehensive, however, we believe it is adequate to demonstrate how HTTPS has been deployed with DNS based request routing mechanisms of CDN currently. In particular, we observed 1,637 DNS-CDN-enabled sites accessible over HTTPS (reachable and responded with valid content), yet disturbed by invalid certificate warnings. Such cases might be caused by HTTPS-enabled web sites adopting ordinary deployments of CDN providers that support HTTPS problematically and result in the front-end authentication failure described in Section III.

*2) **Analysis of the Existent Mechanisms**:*

We learn from the survey that CDN providers have adopted so-called custom certificates and shared certificates to avoid the front-end authentication failure. However, our further study shows that both of these two techniques have their inherent shortcomings.

It is worth to note that the terms used by CDN providers are inconsistent and confusing; same term might even have different meanings. Nevertheless, we adopt these two commonly used terms consistently in this paper, as described below.

***Custom Certificate**:*

As shown in Figure 2, custom certificates work by having the CDN (Carol) requesting web site (Alice) to upload her certificate and private key. In this case, Alice issues delegation by explicitly copying her private key to Carol, then Carol simply announces the delegation by the fact that she holds Alice's private key which is used to establish HTTPS with Bob on Alice's behalf.

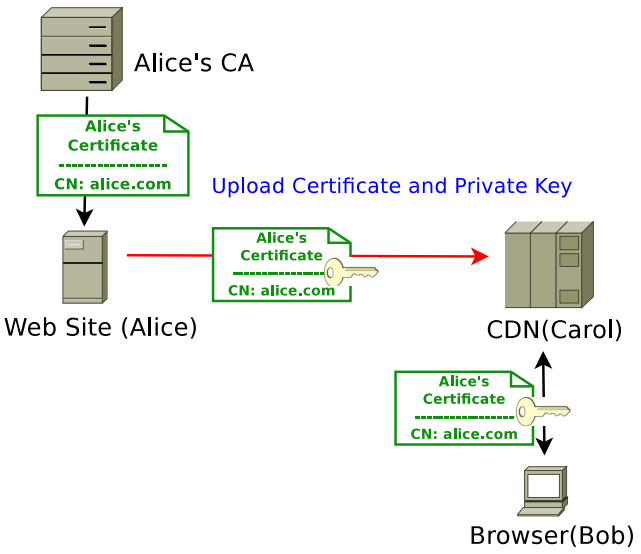


Figure 2. An illustration of custom certificate in CDN.

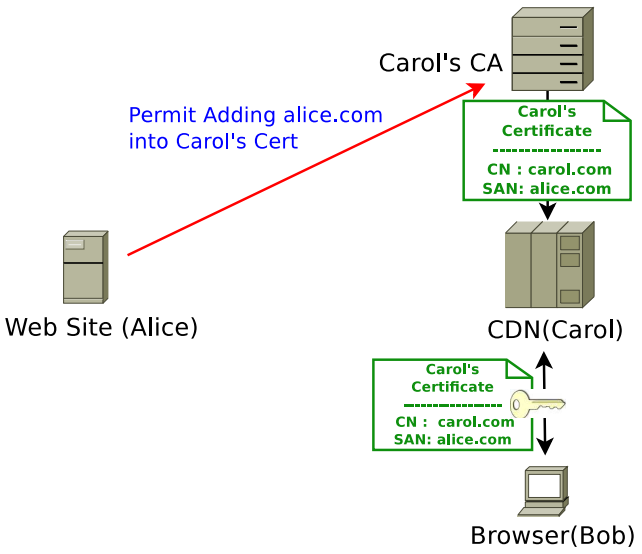


Figure 3. An illustration of shared certificate in CDN.

Shortcomings. While this approach does avoid warning of invalid certificate on browser side, we argue that it has two major shortcomings.

First, sharing private keys between a web site and CDN provider violates the fundamental setting of public key cryptography; practically it incurs additional security risks as the private key needs to be distributed to a number of surrogate servers, greatly increasing the attack surface. The fact that Alice deploys Carol's CDN service implies she trusts Carol to serve her web content honestly, rather than trusts Carol to protect her private key from being compromised. In general, we argue that *a technical scheme should not rely on sharing private key between different organizations in any circumstance*. After all, a private key is meant to be private.

In addition, a web site cannot revoke its delegation independently and efficiently. In Figure 2, since the delegation from Alice to Carol is issued by copying Alice's certificate and private key, to revoke it, Alice must request her CA to revoke the certificate. This might still be controllable by Alice, but not efficient. Further, Alice might still need her CA to sign a new certificate if she wants to keep using HTTPS, which should be true in most cases. The whole process of revocation could be highly expensive and time-consuming, especially when Alice holds an EV certificate, which requires a rigorous vetting process by her CA.

***Shared Certificate**:*

Shared certificates avoid warning of invalid certificate on browser side by taking advantage of the SAN extension of X.509v3 certificates. In Figure 3, when adopting Carol's CDN service, Alice issues delegation by allowing Carol's CA to issue Carol a new certificate ("`CN:carol.com`") that includes Alice's domain in its SAN extension ("`SAN:alice.com`"). Carol then uses the new certificate to communicate with Bob when Bob accesses `alice.com` yet is redirected to Carol through DNS based request routing.

Shortcomings. Although shared certificate could avoid the problem of sharing private key in custom certificate, it has its own problems. We consider its two major shortcomings as follows.

First, shared certificate could weaken the functionality of certificates as a security indicator. In Figure 3, suppose Alice has an EV certificate while Carol has an OV one, the user behind Bob (browser) would not be able to realize Alice is a highly assured web site. Because Bob could only see Carol's OV certificate which displays an ordinary HTTPS indicator, but would never know Alice's EV certificate that shows a more noticeable security indicator. This limitation can be reviewed under the frame of the three requirements proposed in Section III. As a delegation token, Carol's shared certificate does not satisfy the third requirement in that it only contains Alice's domain name rather than her complete identification, *i.e.* full information of Alice's certificate, which is required for displaying a correct indicator when showing user Alice's domain and web content.

Second, similar to custom certificate, a web site cannot issue and revoke its delegation independently and efficiently. In Figure 3, issuing delegation involves coordination of three parties: Alice, Carol and Carol's CA. Revoking delegation also involves these three parties and could be even uncontrollable: Alice might fail to revoke the certificate without Carol's agreement because it is actually issued by Carol's CA; however Carol may be disinterested in coordinating revocation because Alice is no longer her customer.

Case Study. We conducted a case study to further observe

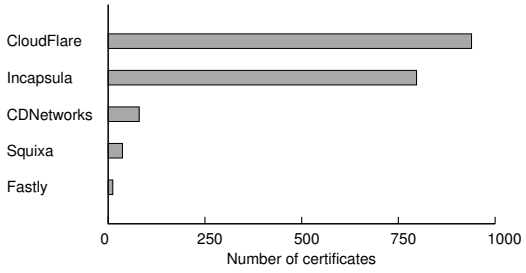


Figure 4. The number of shared certificates which are deprecated by CDN providers but not revoked by CAs.

how CDN providers operate shared certificates in practice. We first set up a web site with HTTPS enabled, then requested Incapsula, a CDN provider whose CA is GlobalSign, to serve our web site using HTTPS. After about half an hour we received an email from GlobalSign asking for permission to add our domain name into Incapsula's certificate. GlobalSign also depended on this step to verify our ownership of the domain name, since the email was received from the email account registered in the SOA record of our domain name. After replying with our permission, Incapsula deployed the new shared certificate into its surrogates in a few minutes, then our web site became accessible from Incapsula.

We then canceled Incapsula's service to observe the process of revocation. We observed that Incapsula deployed a new certificate on all of its surrogates in an hour, which excluded our domain from its SANs. However, we found that the abandoned shared certificate with our domain name as an SAN was not revoked by GlobalSign either through CRLs or its OCSP server even a month after we canceled Incapsula's service. We also tried to contact the customer service of GlobalSign several times for this issue, without success.

The case study shows that the process of issuing a new shared certificate is generally efficient with the help of some application layer utilities. However, the process of revocation is problematic. In our case study, Incapsula and GlobalSign seem to totally ignore the revocation of abandoned shared certificates, which exposes risk of impersonation: if the CDN provider is dishonest or some attackers manage to steal an abandoned shared certificate and the associated private key, they can launch a man-in-the-middle attack against any of the original web sites sharing that certificate.

Monitoring of Shared Certificates. We also launched a measurement to monitor the issuance and revocation of shared certificates. We monitored 1,198 sites that used shared certificates (Table II) to observe how frequently CDNs updated shared certificates on their surrogates. We also periodically requested the CRLs and the OCSP servers to check if the abandoned certificates were revoked by their CAs in a timely manner. Our measurement lasted for three months, during which we observed 1,865 updates for shared certificates, mainly resulted from customers joining or leaving CDN services; Figure 4 shows the number of updates for shared certificates observed from various CDNs. However, our measurement showed that none of the abandoned shared certificates were revoked by their CAs. This demonstrates that ignorance about revoking shared certificates is a common problem in current operations of CDNs and CAs.

Table III
THE DEPLOYMENT OF BACK-END AUTHENTICATION IN CDN PROVIDERS (TESTED IN NOV. 2013)

CDN Provider	Back-end Protocol	Certificate Validation
CDN77	HTTP	–
CDN.net	HTTP	–
CloudFlare	HTTP / HTTPS	No [3]
CloudFront	HTTP / HTTPS	Did not validate CN
Incapsula	HTTP / HTTPS	No

B. The Back-end

For protecting the back-end communication of CDN, as we state before, a standard HTTPS channel with server-side authentication is sufficient. This is not challenging from technical perspective, however, our investigation shows that the current practice is worrisome.

We manually tested five CDN providers that claim to support HTTPS communication. As presented in Table III, all of them were insecure. CDN77 and CDN.net did not even use HTTPS for back-end communication. CloudFlare and Incapsula did reach our site with HTTPS, but they did not seem to enable certificate authentication to web site's server as they failed to detect our MITM attacks using a self-signed certificate between CDN and our site. Although CloudFront verified whether the certificate presented by our site was signed by a trusted CA, it neglected to match the `CN` field with the domain name; thus we successfully launched a MITM attack using a CA-issued certificate.

As a surrogate of a (pull-based) CDN is essentially a reverse proxy with caching, and indeed some reverse proxy softwares have been recommended as open source CDN solutions, we therefore also look into these well-known open source reverse proxies. Surprisingly, several famous reverse proxies, such as Nginx, HAProxy and Varnish, do not support HTTPS as a back-end protocol.

Although our investigation on the back-end protocol of CDNs stops at a small scale, due to the limitation of resources, we believe the results are sufficient to demonstrate that although the back-end communication of CDN is technically easy to secure, it is actually problematic in the current practice and should be paid attention to by CDN providers.

[3]CloudFlare fixed the problem in Feb. 2014.

Reporting and Responses. CloudFlare enabled a feature called StrictSSL to support back-end certificate validation in Feb. 2014[9], after we reported the problem. They have also implemented back-end HTTPS and certificate validation for Nginx.

C. Summary

We have shown various defects of the current practice of composing HTTPS with CDN, some of which lead to risks of impersonation attacks. For the back-end, the problem is due to lack of awareness, which is fixable with operational efforts. However, for the front-end, the defects are mostly inherent. We therefore believe it is necessary to explore new techniques for the front-end authentication problem.

V. Name Constraint Certificate: A Questionable Solution

In seeking new directions to address the problem of the front-end authentication, we first look at techniques within the current frame of the X.509 system. We recognize that a special extension of the X.509 certificate, namely the name constraints extension, is potentially applicable to address the problem. However, we further realize that its practical feasibility is questionable after detailed investigations.

A. Basic Idea

Back to the custom certificate scenario illustrated in Figure 2, if Alice can issue a new certificate with all necessary information to Carol, instead of giving her own certificate, our major concern of sharing private key can be avoided. In fact, in the X.509 system, Alice's CA could issue Alice a signing certificate (a certificate with "`BasicConstraints=CA:True`"), so that Alice becomes an intermediate CA who can issue new certificates to Carol. The problem is that simply doing so allows Alice to sign valid certificates for any domain to anyone, which raises serious security concerns. X.509 system has addressed this issue by a special certificate extension called *name constraints* [3]. Essentially, the name constraints extension restricts a signing certificate only being able to issue certificates with a certain space of identities.

Conceptually it is straightforward to apply these features of the X.509 system to solve this problem. As shown in Figure 5, Alice first needs to apply for a subordinate CA certificate with name space being restricted to `alice.com`. When she adopts Carol's CDN service, she issues a new certificate to Carol stating that `alice.com` has been delegated to Carol, which is further shown to Bob as proof of delegation when Bob tries to access `alice.com` yet connects to Carol. Alice can also revoke the delegation independently with standard certificate revocation techniques such as CRL and OCSP.

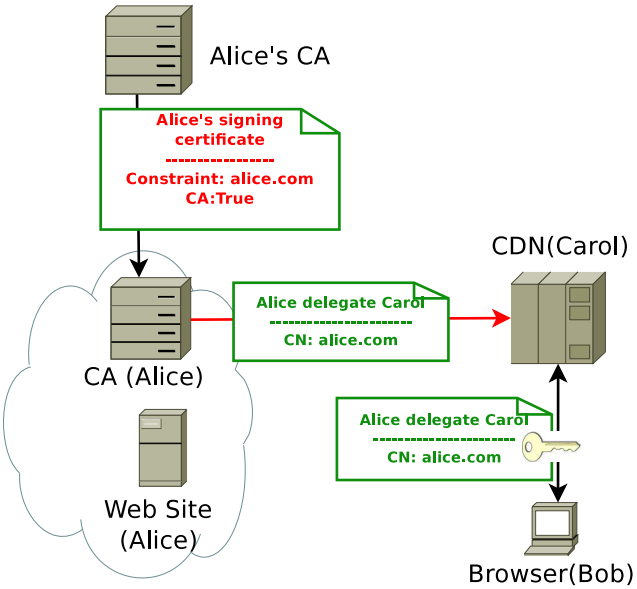


Figure 5. Leveraging name constraint certificate to support HTTPS in CDN.

B. Discussions on Impracticality

This approach is seemingly attractive as it fulfills all requirements of this case based on existent standards. However, we doubt its practical feasibility after careful investigations and considerations.

*1) **Improper Enforcement**:*

This approach works only if browsers and other client softwares correctly enforce the name constraints of a signing certificate, otherwise it could undermine the trust model of the X.509 PKI. However, our investigation reveals pitfalls in the specifications and implementations of the name constraints extension, as presented below.

Name Structure of the X.509 Certificate. The entire identity of an X.509 certificate consists of the "Distinguished Name (DN)" which is one field of its "Subject", and all names in its SAN extension if such an extension is presented.

The "DN" field is further composed by a number of attributes such as "Common Name (CN)", "Organization (O)", "Country (C)". The SAN extension could be filled with one or more names with various types, including email address, DNS name, IP address, directory name and uniform resource identifier.

The Name Constraints Extension. The name constraints extension in a signing certificate of an intermediate CA describes one or more rules that restrict the name space of certificates issued by the CA. Different components of certificate identity have different name constraint syntax and matching rules. For example, a certificate with DN "`C=Internet,O=FTP`" does not match a DN constraint "`C=Internet,O=WWW`" since the former does not con-

tain the latter. As another example, a DNS name constraint "example.com" only matches sub domains of "example.com". A name constraint can further be specified as permitted or excluded. For a responsible intermediate CA, a certificate request is permitted to be signed only if all names in its identity do not match the excluded constraints and match the permitted ones.

Pitfalls in Current Practice. In a certificate used for web, the only meaningful parts of its identity are the domain names, which are either presented as CN attributes of the DN field, or presented as DNS names in the SAN extension.

On validating the name of a certificate, following the standard [10], a browser first checks the SAN extension if it is present. If the domain name of the current URL appears in the SAN extension, the validation succeeds. Without seeing a SAN extension, the browser further checks whether the domain name is present in the DN field as a CN attribute.

Considering the name constraints extension, the browser should further check whether the identity of the certificate passes the name constraint rules. However, we find that not all the browsers have implemented this feature. As shown in Table IV(a), on MAC OS, all investigated browsers except FireFox do not implement name constraints checking. This is because Security Framework API, the TLS/SSL library on MAC OS does not support this feature while the NSS library used by FireFox does.

In fact, when applying to web, the standard name constraints checking is not secure. A dishonest intermediate CA, who is restricted by a name constraints extension, still can issue certificates with arbitrary domains, and fool browsers to accept. The reason is that in a certificate used for web, the domain name can be presented in the DN field as a CN attribute. However, DN field is only examined by DN constraints according to [3], which only checks if the former literally contains the latter. In other words, even if a CN has a value in the form of domain name, it would not be checked against a DNS name constraint at all. Such examination cannot prevent arbitrary domains from being included in additional CN attributes. For example, if a CA, who is restricted by a DN constraint "C=Internet,O=WWW,CN=example.com" and a DNS name constraint "example.com", issues a certificate with a DN "C=Internet, O=WWW, CN=example.com, CN=google.com" but without SAN extension, the certificate will be accepted as valid for google.com by a browser who merely follows the standard. Because the DN field is legitimate to the DN constraint; moreover, the DNS name constraint will not be examined since there is no SAN extension.

To prevent this problem, a browser should apply DNS name constraint matching rule on CN attributes as well, which is beyond the standard. We found this issue has been briefly discussed in IETF mailing list [11]. Our investigation reveals that Chrome and Opera on Linux still fail to do so (see Table IV(b)), which means their name constraints checking can be bypassed by the above trick.

Table IV
THE IMPLEMENTATIONS OF NAME CONSTRAINTS CHECKING IN VARIOUS BROWSERS.

Operating System	Browsers				
	IE	Firefox	Chrome	Safari	Opera
Windows	✓	✓	✓	✓	✓
Mac OS	N/A	✓	×	×	×
Linux	N/A	✓	✓	N/A	✓

(a) Support of distinguished name constraints on the Subject field and DNS name constraints on the SAN extension.

Operating System	Browsers				
	IE	Firefox	Chrome	Safari	Opera
Windows	✓	✓	✓	✓	✓
Mac OS	N/A	✓	×	×	×
Linux	N/A	✓	×	N/A	×

(b) Support of DNS name constraints on the common name attribute.

*2) **High Operational Overhead**:*

Even assuming perfect enforcement, a large majority of web sites probably could not afford, nor have the technical capability, to become subordinate CAs. It is complicated and costly to operate a CA, due to the extensive security requirements on certificate issuance imposed by standard and industrial bodies such as ESTI [12] and CA/Browser Forum [13]. To meet those requirements, significant investment and technical skills are required in CA's infrastructure and operation.

*3) **Lack of Incentive**:*

Even if all web sites could afford becoming CAs, current root CAs (or their subordinates) are unlikely motivated to issue them intermediate CA certificates due to high overhead from vetting of future subordinate CAs (such as auditing their security and policy conformance). This vetting process is commonly mandated by browser vendors [14], in order for a root CA's public key certificate to be included as trust anchor in their browsers.

*4) **Evidence of Rare Adoption**:*

We searched through the ICSI Notary certificate database [15], which had collected about 1.5 million HTTPS certificates in the Internet, but found that none of these certificates contained a name constraint extension. This evidence demonstrates that although name constraint certificate is an existent technique in standard, it is rarely, if ever, adopted in practice.

*5) **Summary**:*

Based on the above discussion, we do not believe that name constraint certificates could become a practical solution to the front-end communication problem in HTTPS over CDN.

VI. DANE with Delegation Semantics: A Long Term Solution

In this section, we propose a new solution for the front-end authentication problem. The approach is based on a slight extension of DANE [16], a protocol currently being standardized by the IETF. Although this approach is not immediately deployable because of its dependencies on DANE and DNSSEC, we believe it has potential as a long term solution, once DANE becomes a common practice.

A. Overview of DANE

The purpose of DANE is to provide an alternative or complementary trust model of TLS/SSL to address some weaknesses of existent mechanisms. Regarding HTTPS, the X.509 PKI based trust model has two main weaknesses. First, the trust is all or nothing: there is no practical way to prevent any trusted CA from issuing a valid certificate for any domain. Hence any compromised or dishonest CA could threaten the whole Internet. Second, X.509 PKI cannot verify self-signed certificates. This prevents free and ubiquitous deployment of HTTPS without commercial CAs.

DANE mitigates these two weaknesses by providing a way to securely bind a domain name and a certificate. The binding is implemented by adding the certificate as one of the domain's DNS records named TLSA records, which is further secured by DNSSEC. The binding enhances the original authentication of HTTPS in web, in that it gives a web site the ability to pin its certificate. Based on this information, a browser is able to reject a mechanically valid yet impersonating certificate, or accept a self-signed certificate. Specifically, DANE defines four use cases [4]:

- **CA Constraints.** The CA constraints case refers to a web site adding its CA's certificate as its TLSA record, which prevents browsers from accepting certificates issued by unauthorized CAs.
- **Service Certificate Constraints.** The service certificate constraints case is much stricter than the CA constraints case in terms of certificate pinning. In this case, a certificate can be trusted only if it passes the X.509 PKI validation and is presented as a TLSA record.
- **Trust Anchor Assertion.** This case is similar to the CA constraints case, except that a web site can choose an unofficial CA, i.e. the CA's certificate can be out of the commercial CAs of the X.509 PKI.
- **Domain-Issued Certificate.** This case is similar to the service certificate constraints case, except that the certificate presented in TLSA records can be self-signed.

Adopting DANE requires the deployment of DNSSEC, as well as change of certificate validation process on TLS/SSL clients; both of them need tremendous efforts. Nevertheless, DNSSEC and DANE have been well recognized as substantial steps to make the whole Internet more secure; the community is making great efforts to promote the deployment of these two techniques.

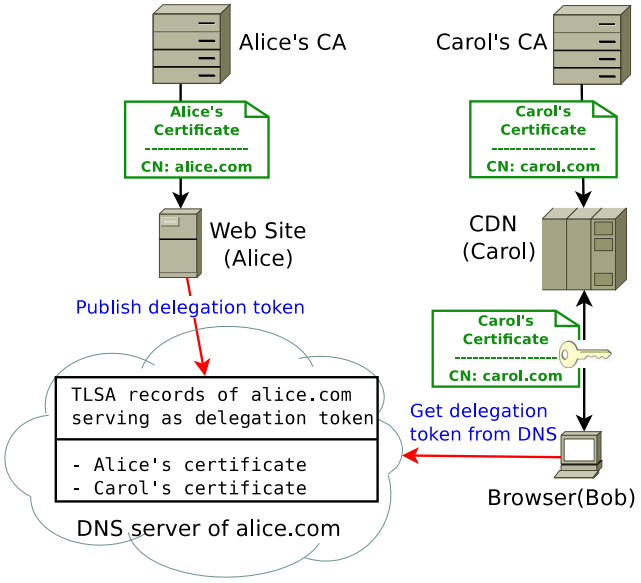


Figure 6. Extending DANE to support HTTPS in CDN.

B. Basic Idea

Actually, some use cases of DANE can be directly applied on the front-end authentication problem. By utilizing DANE, Alice can bind Carol's certificate with her domain name, which could help Bob to recognize the delegation relationship between Alice and Carol. However, we do not consider this as an acceptable solution. Our major concern is that, similar to shared certificate, Bob cannot obtain Alice's original certificate, thus not be able to display proper security indicator to users.

We observe that a simple extension of DANE can overcome the above drawback. As illustrated in Figure 6, to issue a delegation, Alice adds both of her certificate and Carol's certificate as her TLSA records. When Bob connects to Carol and receives her certificate, he further issues a DNS query to request Alice's TLSA records. After receiving the response, Bob not only recognize the delegation from Alice to Carol by seeing Carol's certificate appear as Alice's TLSA record, but is also able to obtain Alice's certificate which is presented in the response as well.

In essence, our proposal broadens the semantics of DANE by not only binding a name with a certificate, but also expressing delegation relationship between entities.

C. Analysis

Deployability. Assuming DANE has been well supported, we believe this approach is quite acceptable from the practitioners' perspective. First, this approach merely needs to extend the semantics of a few bytes in the current TLSA

data format (we omit the details for clarity); in terms of implementation, it only needs to slightly modify the validation process on client side compared to DANE. Second, the operations of delegation issuance and revocation are also convenient and efficient: Alice only needs to add or remove Carol's certificate from her TLSA records, which is simple and fully controllable by herself.

Security. In this approach, the unforgeability of delegation token, *i.e.* the TLSA records of Alice, is guaranteed by DNSSEC. Alice does not need to share her certificate's private key with Carol. And as the revocation of delegation is fully controlled by Alice, there are also no risks of impersonation attacks caused by insufficient revocation. Further, since the delegation token contains Alice's certificate, Bob is able to show user a correct security indicator.

One potential risk is replay attack: suppose Alice changes her CDN provider and removes Carol's certificate from her TLSA records, replaying the stale TLSA records could still convince Bob to believe Carol is a valid delegatee from Alice. Since this problem is inherent in DNSSEC and could be mitigated by expiration time in DNSSEC signatures, we believe it is acceptable in practice.

It is worth mentioning that the authentication of this approach is different from that of the original authentication of HTTPS in terms of source of trust. In the former case, the trust comes from Alice's DNSSEC key which signs the delegation token, while in the latter case, the trust comes from the private key of Alice's certificate. Although this difference is conceptually fundamental in the sense that Alice now needs to protect two keys rather than one to prevent key-compromised impersonation attacks, we argue that the actual impact is insignificant. First, both of the certificate key and the DNSSEC key are highly critical and they must be carefully protected. Moreover, this difference is actually inherent in DANE. In the frame of DANE, to some degree, the DNSSEC key is even more important than the certificate key. Because once the private key of DNSSEC is compromised, the attacker could claim any other "valid" certificate from DANE to bypass the protection of original certificate validation. Therefore we believe the increased risk of protecting both of the certificate key and the DNSSEC key is tolerable from a practical perspective.

D. Implementation

We have implemented a proof of concept (PoC) of our proposal as a FireFox extension[4], which is a slight revision of another Firefox extension demonstrating DANE [17]. We modify the DANE Firefox extension to allow for the validation of the delegation path among the certificates returned from the web channel (HTTPS) and those from the DNS channel (DANE). Figure 7 illustrates how a browser with our extension interacts with a CDN provider and the original web site. Compared to the standard web browsing, it adds an extra network round-trip to fetch TLSA records as well as a local validation process. It also changes how a browser displays security indicator to users. Note that the extra round trip of the DNS lookup for TLSA records can be avoided if we had chosen to modify the browser to do either TLSA query in parallel with the A query or DNS prefetching.

[4]Our PoC and the demo site are available at https://github.com/cdnsec.

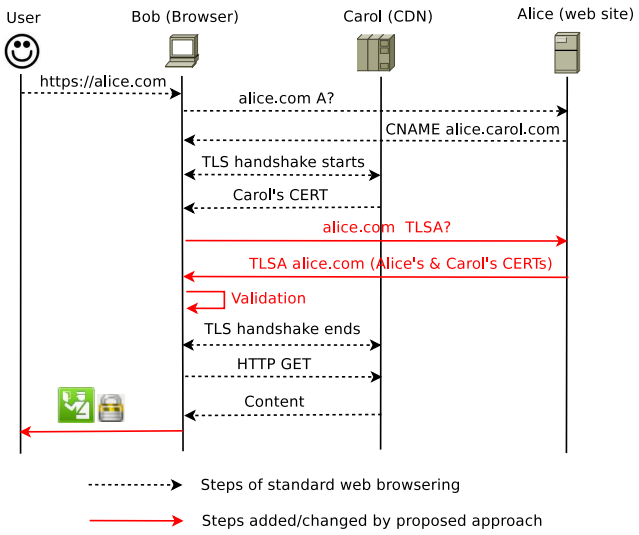


Figure 7. The interaction of proposed approach for the front-end authentication that composes HTTPS with CDN.

With our PoC, we can now demonstrate the two main properties of our proposal: 1) a web site using CDN service can provide seamless HTTPS experience to end-users and show its certificate to them; and 2) a web site can effectively and independently revoke its HTTPS delegation to a CDN provider without requiring any cooperation from the CDN provider or the CA. We first setup a web site supporting DANE, and obtained a certificate from a CA. We then applied for a CDN service for our web site, and added the certificates of our CDN provider and our own to our TLSA records. Note that we neither upload our certificate to our CDN provider, nor apply for the use of a shared certificate provided by the CDN provider. Without our PoC, a user visiting our web site via CDN will be alerted of invalid certificate. With our PoC, a user is not given any warning of certificate errors. Further, the user can click our modified Firefox indicator to obtain information about our original certificate and the delegation path. We then removed the CDN providers certificate from our TLSA records to revoke the delegation. After that, when users access our web site via this CDN provider, they will be alerted of certificate errors.

E. Discussions on Potential Overhead

Without considering the local process changes, compared to the standard web browsing, the overhead of the proposed

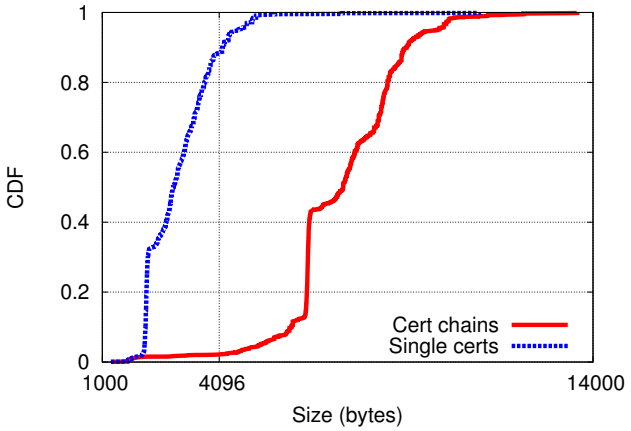


Figure 8. Comparison of the size of collected single certificates and certificate chains.

approach is mainly the potential latency brought by the extra DNS round-trip. However, as we state before, the potential latency is highly related to various implementation strategies, which makes a comprehensive evaluation a difficult work. Therefore we consider a direct latency measurement as future work. Here instead of comparing with the standard web browsing, we narrow our discussions on the difference between our proposal and DANE.

Although our proposal does not use more network round-trips than DANE, the DNS conversation in our proposal is still heavier than the original DANE because web site needs to transmit more certificates in a DNS reply. More specifically, in our proposal, to help Bob validate Alice's certificate without other efforts, the DNS conversation should bring back the whole certificate chain of Alice. To explore the possible overhead of transmitting a complete chain rather than a single certificate in a DNS conversation, we collect available certificate chains when conducting the measurements in section IV-A1. Figure 8 plots the CDF of the sizes of collected certificate chains, along with the CDF of the sizes of single certificates. The most useful information in Figure 8 is that 97.86% of all certificate chains exceed 4,096 bytes, while in single certificates the ratio is 11.68%. This means that in most cases, the DNS conversation in our proposal will first try UDP then turn to TCP because the response exceeds the maximum length of 4,096 bytes allowed by UDP currently, which will cause more network latencies.

To mitigate this issue, we recommend OS vendors as well as browser vendors to support issuing DNS query over TCP directly. We believe this could be a common requirement in the DNSSEC era. After all, even if not considering our proposal, a considerable portion of DNS conversations in DANE could still face the first-UDP-then-TCP issue as our data in Figure 8 shows that 11.68% of single certificates have already exceeded the 4,096 byte limitation.

F. Summary and Future Work

Our proposal is lightweight in itself, and is incrementally deployable. For example, a web site can immediately improve its security by publishing its delegation token in DNSSEC and encouraging visitors to use our browser plugin. While our proposal cannot be deployed immediately on a large scale due to its dependence on DNSSEC and DANE, we believe it is a valuable long term solution, since both DNSSEC and DANE have attracted significant interest and deployment effort from the Internet community. In fact, our proposal is another example of how DNSSEC and its applications such as DANE can help bootstrap trust in Internet services.

Another note is that recently due to the needs of cooperations between CDNs and ISPs, industry vendors have proposed cascading CDN service [18], [19], *i.e.* more delegation layers between multiple CDN providers. Our solution can be easily extended to support this scenario, *e.g.* using more DNS queries to follow the possible certificate delegation path step by step. We leave this for future work.

VII. Discussions

Other Possible Solutions and Comparisons. We are aware of a few other techniques that are possibly applicable to compose HTTPS with CDN.

Proxy certificate [20] is conceptually similar to the name constraint certificate therefore has similar practical issues.

WASP [21] turns TLS/SSL handshake into a three-party protocol. In WASP, CDN relays TLS/SSL authentication to web site and then receives TLS/SSL master secret to accomplish session key negotiation with browser, so that it is able for the browser to show the certificate of the web site meanwhile avoids sharing private key. Comparing with our DANE-based solution, WASP does not require client-side changes, which makes it relatively easy to deploy and probably favored by the industry. However, WASP still needs heavy change on server-side. Further, WASP could greatly weaken the performance improvement and DDoS protection of adopting CDN as it requires web site to be involved in every HTTPS connection. In addition, it is unclear how WASP could be extended to support cascading CDN service.

Tight Coupling of HTTPS. From an architectural perspective, to some extent, the front-end authentication problem is caused by the tight coupling of HTTPS. In HTTPS, the authentications of the transport layer protocol (TLS) and the application-layer protocol (HTTP) are tightly coupled in that they share same identity (certificate) and same validation process of the identity. This is why CDN, which essentially is a transport layer man-in-the-middle, breaks the application layer authentication, instead of being transparent to upper layers. From this point of view, although the proposed DANE-based approach does not completely decouple the

application layer authentication from the transport layer authentication as they still share same identity, it does loosen the coupling in that it provides a different identity validation process for application layer authentication.

Boundary of Trust. We have revealed various practical defects of the current HTTPS practice of CDN providers. These defects also reflect a fundamental, yet often misunderstood security concept: the boundary of trust. In this case, the misunderstood trust boundaries behind some defects might be simply caused by technical unawareness. The insecure back-end communication is such an example, as everyone will agree that the network between such "back-end" communication is apparently untrusted. However, the trust boundaries behind some others are more subtle: we have trusted a CDN to deliver our content, shall we trust it not to abuse our identities or private keys, without (or with) guarantee of revocation? For a theoretical problem, the answer for such question should apparently be no. However, in a practical scenario like this one, it is often unclear. Nevertheless, we believe we should be conservative in considering the trust boundaries of our Internet systems, especially in the current context of the pervasive state-level Internet surveillance, demonstrated by the recent events of NSA leaks from Edward Snowden.

VIII. Related Work

Delegation and Multi-party Web Protocols. The most challenging problem studied in this paper is a special case of delegated authentication. The generalized frame of delegation has been recognized by Sollins [7], referred to as cascaded authentication. Gasser and McDermott further give a detailed study on delegation in a distributed system under the context of access control; they also first considered the revocation of delegation. Except the case of composing HTTPS with CDN, several multiple party web protocols can also be regarded as delegation protocols. For example, in the scenario of the OAuth [22] protocol, a user (resource owner) delegates a web server (consumer) to access his resources on another web server (service provider). In fact, if we broaden the concept of delegation, many multiple party protocols in web can be viewed in this way. To some degree, the process of Single Sign On (SSO) is also a process of delegation. In the SSO protocols such as CAS [23], SAML [24], and OpenID [25], a service provider (SP) delegates an identity provider (IdP) to authenticate a user. In e-commerce system, the process of Cashier-as-a-Service [26] based checkout is also a form of delegation in that an online merchant (*e.g.* , Amazon) delegates an online cashier (*e.g.* , Paypal) to charge its users. Protocols involving multiple parties are much more complicated than two party protocols. The point of view of delegation is useful in clarifying the relationships of involved parties from the complex protocol interactions.

Server Authentication on the Web. This paper studies a case of authentication, in which we explore how to avoid improper security indicator of HTTPS in presence of CDN. The main purpose of HTTPS certificate warning and other HTTPS security indicators are to help users identify MITM attack or fake sites, *i.e.* phishing sites. Countering phishing sites is a rather complicated issue because indeed the victim in phishing attacks is human rather than machine [27] [28]. While some studies have showed that the current security indicators of HTTPS are not efficient in preventing phishing sites for various reasons [29] [30]. Others have tried to re-design the indicators [30]. In addition to HTTPS, researchers have invented several authentication schemes to further help users properly identify web sites. SiteKey [31] is a technique adopted by BankOfAmerica, which employs a user-specific icon to enhance server authentication. PwdHash [32] and BeamAuth [33] prevent users from leaking their credentials to phishing sites by enhancing the authentication with specially crafted second factors.

TLS/SSL Trust Model. To some degree, the problem studied in this paper occurs because the trust model of HTTPS, *i.e.* the X.509 PKI system lacks the ability to express delegation relationship between certificates. Besides the X.509 PKI system, and the DANE protocol which we have introduced, some other trust models of TLS/SSL have been applied. Another common application of TLS, the secure shell (SSH), adopts a trust model named Trust-on-first-use [34], which is essentially a historical behavior based trust. The web-of-trust model [35] is flexible and potentially able to express the semantics of delegation, however, it is not being applied with TLS/SSL. Research on TLS/SSL trust model mainly focuses on the problems of the X.509 system. Clark *et al.* give a comprehensive review on this topic [36]. Perspectives [37] and Convergence [38] are two proposals trying to bring the historical behavior based trust to web. Certificate Transparency [39] provides an open platform to monitor and audit TLS/SSL certificates, which helps to address the weaknesses of the X.509 system. The Chrome browser implements certificate pinning which associates HTTPS web sites with a group of expected certificates.

Study of Certificate. Though not our direct motivation, our study reveals some pitfalls of certificate processing in various browsers, in the course of which we strongly feel that certificate is highly complex that could be misunderstood, and further be misused in many ways. Indeed, this topic has attracted many efforts in the past few years. A number of measurements have been conducted to investigate the current state of TLS/SSL certificates in the Internet [40], [41], [42], [43], [44], [45]. The certificates are collected either through scanning the entire IPv4 address space, or probing the Alexa's Top 1m sites, or through passively monitoring. In [41] and [42], the authors show the statistics of the collected certificates and reveal their existing problems. Akhawe *et al.* aim at understanding the TLS/SSL errors for certificates on the web service and also present some practical recommendations based on their analysis [43].

Amann *et al.* analyze the certificate trust relationships in SSL ecosystem and expose their surprising dynamics [44]. Delignat-Lavaud *et al.* try to assess the adoption of the X.509 PKI guidelines in current practice [45].

Certificate revocation is a major concern in our study. Recently Topalovic *et al.* point out some problems with certificate revocation in OCSP and propose to use short-lived certificates to mitigate the problem of certificate revocation [46]. Delignat-Lavaud *et al.* also discover certificate revocation issues for CDN services as we do individually [45].

Other Security Problems Brought by CDN. This paper studies an authentication problem brought by CDN. In the model of this paper, web sites trust the CDN to be honest when their contract is valid. Others might consider not trusting the CDN. Lesniewski-Laas *et al.* propose SSL splitting to protect the integrity of data served by untrusted proxies [47]. Michalakis *et al.* investigate the problems of content integrity in peer-to-peer CDNs, where not all the replicas are trusted [48]. They also present a system called *Repeat and Compare* to ensure the content integrity in untrusted peer-to-peer CDNs.

IX. Conclusion

The authentication problem of composing HTTPS with CDN is much more complicated than it seems to be. CDNs, transparent to end-users in most cases, introduce complexity to the end-to-end communication.

We give a systematic investigation on the current practices of composing HTTPS with CDN, which includes 20 leading CDN providers and 10,721 of their customer web sites. Our study finds that the status quo is far from satisfactory. Varieties of problems exist in HTTPS deployment of those popular CDN providers, including widespread use of invalid certificates, private key sharing, neglected revocation of stale certificates, insecure back-end communication with customers' original web sites and so on, ranging from operational level to mechanism design level.

In seeking new solutions, we first examine a potential solution using existent technique called name constraint certificate. However, we consider it an impractical approach for various reasons. Then we propose a solution based on DANE, an emerging technique being standardized by the IETF. Our implementation shows that the DANE-based solution could compose HTTPS with CDN securely and effectively.

We expect our work to raise the awareness of this emerging problem in the community. In the short term, we expect the vendors take operational efforts to address some of the defects of the current practices. In addition, we hope our work can set off further discussion among practitioners and researchers on more preferable solutions.

Acknowledgements

We would especially like to thank Bernhard Amann for his valuable suggestions and assistance with ICSI Notary. We are also grateful to receive constructive comments from anonymous reviewers, and from Tao Wei, Nick Sullivan, Joseph Bonneau, David Lee, Moritz Steiner and Jianwei Zhuge. This work is supported by the National Natural Science Foundation of China (Grant No. 61161140454). Kang Li's research on this work is partially supported by the US National Science Foundation CISE grants 1127195, 1318881, and a gift grant from Intel Corp.

References

[1] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "RFC 2560, X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP," *Internet Engineering Task Force RFC*, 1999.

[2] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms," *Internet Engineering Task Force RFC*, vol. 3568, 2003.

[3] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," *Internet Engineering Task Force RFC*, 2008.

[4] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh, "The Case for Prefetching and Prevalidating TLS Server Certificates," in *Proceedings of the 19th Network and Distributed System Security Symposium*, 2012.

[5] "Revocation Checking and Chrome's CRL," 2012, https://www.imperialviolet.org/2012/02/05/crlsets.html.

[6] "OCSP Stapling." [Online]. Available: http://en.wikipedia.org/wiki/OCSP_stapling

[7] K. R. Sollins, "Cascaded Authentication," in *Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on*. IEEE, 1988, pp. 156–163.

[8] M. Gasser and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*. IEEE, 1990, pp. 20–30.

[9] "Introducing Strict SSL: Protecting Against a Man-in-the-Middle Attack on Origin Traffic." [Online]. Available: http://blog.cloudflare.com/introducing-strict-ssl-protecting-against-a-man-in-the-middle-attack-on-origin-traffic

[10] E. Rescorla, "RFC 2818: HTTP over TLS," *Internet Engineering Task Force: http://www. ietf. org*, 2000.

[11] "[pkix] Name Constraints on Domain Name in Common Name." [Online]. Available: http://www.ietf.org/mail-archive/web/pkix/current/msg27619.html

[12] ETSI, "Policy Requirements for Certification Authorities Issuing Public Key Certificates." [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102000_102099/102042/01.01.01_60/ts_102042v010101p.pdf

[13] C. Forum, "Baseline Requirements for the Issuance and management of Publicly-Trusted Certificates." [Online]. Available: https://www.cabforum.org/Baseline_Requirements_V1.pdf

[14] "SubordinateCA Checklist." [Online]. Available: https://wiki.mozilla.org/CA:SubordinateCA_checklist

[15] I. C. S. Institute, "The ICSI Certificate Notary," 2012. [Online]. Available: http://notary.icsi.berkeley.edu

[16] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," RFC 6698, August, Tech. Rep., 2012.

[17] "Extended DNSSEC Validator." [Online]. Available: https://os3sec.org/

[18] B. Niven-Jenkins, F. L. Faucheur, and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement," *RFC6707, Sep*, 2012.

[19] G. Bertrand, E. Stephan, T. Burbridge, K. Eardley, Pand Ma, and G. Watson, "Use Cases for Content Delivery Network Interconnection," *RFC6770, Nov*, 2012.

[20] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson *et al.*, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile," RFC 3820 (Proposed Standard), Tech. Rep., 2004.

[21] N. Modadugu and E. jin Goh, "The Design and Implementation of WASP: A Wide-Area Secure Proxy," Technical Report, Stanford, Tech. Rep., 2002.

[22] M. Jones and D. Hardt, "RFC 6749: The OAuth 2.0 Authorization Framework: Bearer Token Usage," RFC 6750, October, Tech. Rep., 2012.

[23] "CAS." [Online]. Available: http://en.wikipedia.org/wiki/Central_Authentication_Service

[24] "SAML." [Online]. Available: http://saml.xml.org/

[25] "OpenID." [Online]. Available: http://openid.net/

[26] R. Wang, S. Chen, X. Wang, and S. Qadeer, "How to Shop for Free Online–Security Analysis of Cashier-as-a-Service Based Web Stores," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 465–480.

[27] R. Dhamija, J. D. Tygar, and M. Hearst, "Why Phishing Works," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 581–590.

[28] J. Hong, "The State of Phishing Attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[29] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness." in *USENIX Security Symposium*, 2009, pp. 399–416.

[30] R. Biddle, P. C. van Oorschot, A. S. Patrick, J. Sobey, and T. Whalen, "Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 19–30.

[31] "Sitekey." [Online]. Available: http://en.wikipedia.org/wiki/SiteKey

[32] B. Adida, "BeamAuth: Two-Factor Web Authentication with a Bookmark," in *Proceedings of the 14th ACM conference on Computer and communications security (CCS)*, 2007, pp. 48–57.

[33] D. Florêncio and C. Herley, "Password Rescue: A New Approach to Phishing Prevention," in *Proceedings of USENIX Workshop on Hot Topics in Security*, 2006.

[34] G. Toth and T. Vlieg, "Public Key Pinning for TLS Using a Trust on First Use Model," 2013.

[35] A. Abdul-Rahman, "The PGP Trust Model," in *EDI-Forum: the Journal of Electronic Commerce*, vol. 10, no. 3, 1997, pp. 27–31.

[36] J. Clark and P. van Oorschot, "SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements," in *2013 IEEE Symposium on Security and Privacy (SP)*, 2013, pp. 511–525.

[37] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing." in *USENIX Annual Technical Conference*, 2008, pp. 321–334.

[38] "Convergence." [Online]. Available: http://www.convergence.io/

[39] A. Langley, E. Kasper, and B. Laurie, "Certificate Transparency," 2013.

[40] "The Electronic Frontier Foundation." [Online]. Available: https://www.eff.org/observatory

[41] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL Landscape: A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 427–444.

[42] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux, "The Inconvenient Truth About Web Certificates," in *Economics of Information Security and Privacy III*, B. Schneier, Ed. Springer New York, 2013, pp. 79–117.

[43] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer, "Here's My Cert, So Trust Me, Maybe?: Understanding TLS Errors on the Web," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 59–70.

[44] B. Amann, R. Sommer, M. Vallentin, and S. Hall, "No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 179–188.

[45] A. Delignat-Lavaud, M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie, "Web PKI: Closing the Gap between Guidelines and Practices," in *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014.

[46] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh, "Towards Short-Lived Certificates," *Web 2.0 Security and Privacy*, 2012.

[47] C. Lesniewski-Laas and M. F. Kaashoek, "SSL Splitting: Securely Serving Data from Untrusted Caches," *Computer Networks*, vol. 48, no. 5, pp. 763–779, 2005, web Security.

[48] N. Michalakis, R. Soulé, and R. Grimm, "Ensuring Content Integrity for Untrusted Peer-to-Peer Content Distribution Networks," in *Proceedings of the 4th USENIX conference on Networked Systems Design and Implementation (NSDI)*, 2007.