DCSP: Performant Certificate Revocation a DNS-based approach

Antonios A. Chariton
University of Crete, Greece
csd3235@csd.uoc.gr

Eirini Degkleri
FORTH-ICS, Greece
degleri@ics.forth.gr

Panagiotis Papadopoulos
FORTH-ICS, Greece
panpap@ics.forth.gr

Panagiotis Ilia
FORTH-ICS, Greece
pilia@ics.forth.gr

Evangelos P. Markatos
FORTH-ICS, Greece
markatos@ics.forth.gr

ABSTRACT

Trust in SSL-based communication on the Internet is provided by Certificate Authorities(CAs) in the form of signed certificates. Checking the validity of a certificate involves three steps: (*i*) checking its expiration date, (*ii*) verifying its signature, and (*iii*) making sure that it is not revoked. Currently, Certificate Revocation checks (i.e. step (*iii*) above) are done either via Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP) servers. Unfortunately, both current approaches tend to incur such a high overhead that several browsers (including almost all mobile ones) choose *not to check* certificate revocation status, thereby exposing their users to significant security risks.

To address this issue, we propose DCSP: a new low-latency approach that provides up-to-date and accurate certificate revocation information. DCSP capitalizes on the existing scalable and high-performance infrastructure of DNS. DCSP minimizes end user latency while, at the same time, requiring only a small number of cryptographic signatures by the CAs. Our design and initial performance results show that DCSP has the potential to perform an order of magnitude faster than the current state-of-the-art alternatives.

1. INTRODUCTION

Over the past few years, partly as a result of the Snowden revelations [12], an increasing number of web services are being served over a Secure Socket Layer (SSL): it is estimated that around 50% of the HTTP connections are being implemented over HTTPS [19]. Trust between the communicating endpoints of HTTPS is usually created through *Certificate Authorities* (CAs) which digitally *sign* and *issue* the public key of their clients in the form of a certificate. Thus, when a web client connects to a web server and receives the web server's certificate, it can validate that this particular public key indeed belongs to the web server.

In addition to *issuing* certificates, CAs may, at times, need to *revoke* certificates. If, for example, the private key of a web server is stolen, the issued certificates need to be revoked, and users need to be updated as soon as possible. The revocation of the certificate of a website is intended to convey a complete withdrawal of trust in the SSL certificate and thereby protect the website's users against fraud, eavesdropping, and theft. Nevertheless, the various implementations of certificate revocation have raised many concerns [13, 14]. For example, most of the contemporary browsers [1] allow the users of a website to continue using a revoked certificate for weeks, or even months. On April 2013 an intermediate certificate was revoked by RSA. This certificate was used to sign multiple SSL certificates of McAfee including one for its e-commerce website[2]. The revocation went almost unnoticed for more than a week [4].

One way for the web browsers to check the validity of the certificate they receive during an SSL negotiation, is to download the list of all revoked certificates: the Certificate Revocation List (CRL) from the CA. If they find the certificate in the CRL, obviously the certificate is revoked and should not be trusted anymore. This CRL-based mechanism is simple and effective. Unfortunately, at the same time, it may lead to very high network overhead (and associated latency) as the CRLs tend to be very large and grow even larger with time: several Megabytes long in some cases [20]. To reduce this overhead, the Online Certificate Status Protocol (OCSP) was proposed: CAs run OCSP servers who are able to respond in real-time to queries about the status of an *individual* certificate. In this approach, web browsers query OCSP servers about the validity of a *single* certificate. Although a significant improvement compared to CRLs, OCSP still imposes significant overhead: several hundreds of milliseconds per query [21]. Although OCSP servers are generally faster than CRLs, they tend to introduce a privacy concern: as a web browser repeatedly contacts an OCSP server to check the validity of the certificates it receives, it exposes part of the users' browsing history.

Probably as a response to the very high overhead associated with certificate checking (especially in mobile devices), more and more web browsers tend *not* to check all the certificates they receive, but only a few certificates considered important. These web browsers seem to fall victims of an ob-

EUROSEC'16, April 18-21, 2016, London, United Kingdom
© 2016 ACM. ISBN 978-1-4503-4295-7/16/04...$15.00
DOI: http://dx.doi.org/10.1145/2905760.2905767

[1]For illustration purposes we describe the problem with the communication end-points being browsers and web servers. Obviously, the described problem, and the proposed solution apply to any SSL communication between a server and a user agent (e.g. e-mail, LDAP)
[2]www.mcafeestore.com

vious dilemma: `security or performance`? Their choices seem to imply that in order to get high levels of security we need to sacrifice performance and vice versa.

In this paper we advocate that this dilemma (`security or performance`) is a false dilemma. We advocate that it is possible to obtain *both* good performance *and* high levels of security at the same time. To do so, we capitalize on existing scalable infrastructures who have demonstrated that they can respond in a matter of milliseconds. Thus, (i) instead of talking to costly OCSP servers, we propose to talk to fast and scalable DNS servers; and (ii) instead of building expensive HTTP connections with OCSP, we propose to use fast UDP-based communication.

To summarize, we make the following main contributions:

1. We show how the DNS infrastructure can be used to store certificate revocation information.
2. We explain how attackers may try to use their time-old DNS spoofing approach to pollute certificate revocation information and show how we are able to detect and mitigate such attacks.
3. We reduce the amount of signatures needed by CAs while maintaining the same (or higher) level of authenticity in our responses.
4. We combine all mechanisms into DNS-based Certificate Status Protocol (DCSP): a lightweight UDP-based and DNS-supported certificate validation protocol that can be significantly faster than the existing approaches.

2. BACKGROUND

2.1 Certificates

Certificates are an important component of SSL, binding the identity of a subject to its public key. A typical digital certificate contains *(i)* a serial number that is used as a unique identifier of the certificate, *(ii)* the subject identified (e.g. the website), *(iii)* the signature algorithm used to create the signature, *(iv)* the issuer, which is the entity verified the subject's information and issued the certificate, *(v)* the actual signature to verify that it came from the issuer, *(vi)* the expiration date, *(vii)* the subject's public key, and finally *(viii)* other information like revocation information.

Most of the browser vendors including Firefox, Chrome, Internet Explorer, Safari etc., accept digital certificates created only by a small group of trusted Certificate Authorities (CAs) usually called *Root* CAs. Browser vendors accept an organization as a *Root* CA, only after out-of-band procedures that prove its trustworthiness. Once a browser vendor authorizes an organization as a Root CA, the latter is added to the list of trusted Root CAs in the browser library along with its root certificate. From this point onwards, all digital certificates signed by this Root CA are considered trusted.

Obviously, not *all* Certificate Authorities manage to become *Root* CAs. Fortunately, they can still be trusted if they carry a certificate signed by a *Root* CA. Once they get this certificate, they are considered trusted to issue certificates, even though they are not *Root* CAs. In this way trust is delegated from one CA to another, forming chains of trust.

2.2 Revocation

Although certificates should be valid until their expiration date, it is possible that they need to be revoked before then, if, for example, the private key of a web server is leaked, or even stolen. To check the revocation status of a certificate web browsers use two main approaches: (*i*) the Certificate Revocation List (CRL), and (*ii*) the Online Certificate Status Protocol (OCSP).

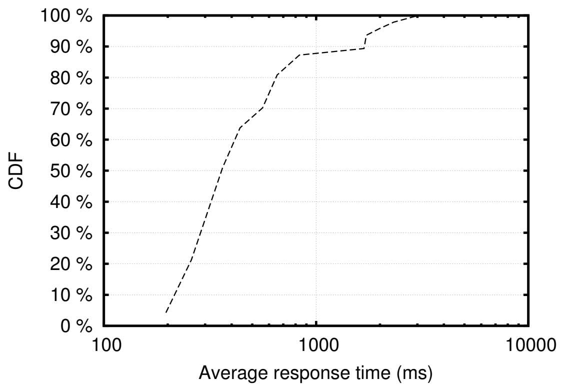

Figure 1: Distribution of the response times of the examined OCSP requests. Average response time: 628ms, median: 363 ms, 95th percentile: 1,986 ms.

2.2.1 *Certificate Revocation List*

The CRL schema consists of lists enumerating the revoked certificates issued by each CA. Web browsers should periodically download these lists. The CRL works as follows: when a browser connects to a web server and receives a chain of certificates, the browser checks whether these certificates are included in CRLs. If any of these certificates is included, the browser knows that this certificate is not valid anymore. Although simple, CRLs may grow very large in size imposing a high network overhead. To make matters worse, since CRLs are downloaded only periodically the information they contain may be stale between periods.

2.2.2 *Online Certificate Status Protocol*

To reduce the network overhead of downloading large CRLs, and improve the *freshness* of provided information, the OCSP protocol was proposed [18]. When a browser receives a certificate chain, it queries an OCSP server (namely *OCSP responder*) for the status of each certificate in the chain. Each OCSP responder checks its always up-to-date revocation list and responds back with the status of the certificate: "good", "revoked", or "unknown".

Although OCSP is more accurate than CRLs, it may impose a very high latency overhead to each and every HTTPS request. Indeed, in order to check the validity of a certificate, CRLs make an inexpensive *local* access to the downloaded CRL list, while the OCSP protocol needs to make an expensive *remote* request to the OCSP server.

In order to estimate this latency overhead of OCSP we conducted an experiment by querying OCSP responders from a list of 53 well known CAs. We grouped the response times in bins of 100ms width and in Figure 1 we present the distribution of the average response time of each one of them. The average response time is about 628 ms, while the 95th percentile reaches up to 1,986 ms. We believe that such a high latency adds a very high overhead to the time needed for establishing each new HTTPS connection, which can adversely impact the end users' quality of experience.

To reduce the overhead associated with OCSP requests, a recent version of OCSP proposes that a web server should obtain an OSCP response and provide it to the web clients along with its certificate (OCSP stapling). To ensure that the web server does not provide a *fake* or *replayed* OCSP response, this response is timestamped and signed by the CA. Unfortunately, such responses suffer from the same "information staleness" problem of the CRLs, since they are timestamped and signed "periodically", at the times they are obtained by the web server.

2.2.3 Modern Browsers

Caught in the turmoil of all the competing approaches, most of the modern browsers, sadly, *disable* OCSP checking by default. Recent studies show that several Desktop browsers have poor certificate validation mechanisms and Mobile browsers uniformly *never* check certificate revocation status "likely driven by the higher cost (in terms of latency and power) of obtaining revocation information on a mobile device" [15]. To remedy this situation, in this paper we propose DCSP: a low latency approach to check certificate revocation status based on lightweight (UDP) protocols.

3. THREAT MODEL

Recent security incidents (i.e. the Debian vulnerability [3], the Heartbleed bug [1]) provided opportunities to expose the poor implementation of the certificate revocation in modern browsers. The inadequacy of current revocation mechanisms leave users vulnerable to man-in-the-middle (MITM) and impersonation attacks. An adversary having in its possession the private key of a website, is able to trick browsers into connecting to a fraudulent website, to eavesdrop all the communication between a web client and the legitimate web server, and to modify the exchanged messages. There were several incidents reported where the Heartbleed bug was used for compromising secure websites like Canada's tax agency and popular UK web forum Mumsnet [11].

In this work, we consider that an adversary achieves to get the private key of a legitimate website. We assume that the adversary is able to monitor the traffic between the communicating entities, tamper any transactions and disrupt the proper functionality of the network by injecting bogus DNS records. Furthermore, we consider that the web server administrator becomes aware of the key leakage, and requires revocation of this particular certificate by the CA. In this paper we study how a CA can correctly revoke a certificate and how the web browsers can check the validity of the certificate they receive in a fast and reliable way.

4. DESIGN

In this section we describe the architecture of DCSP, the process followed for setting up and updating certificate revocation information in the DNS system, and the certificate status verification mechanism.

4.1 The DCSP Algorithm

4.1.1 The DNS as a fast cache

DCSP uses the DNS system to store certificate revocation information. That is, when a certificate is revoked, the Certificate Authority publishes this information in the DNS system. When a web browser wants to check whether a certificate has been revoked, it just queries the DNS system to find information about the certificate[3]. To make sure that the information returned by the DNS servers is authentic, the CAs sign all revocation status information they inject into the DNS system. Although this algorithm seems simple, it unfortunately suffers from a well-known replay attack. Assume, for example, that an attacker has managed to control the DNS information seen by a victim (e.g. by performing Man-in-the-Middle attack, DNS poisoning etc.) After the certificate is revoked, the attacker, who as we have assumed manages to control the DNS information seen by the victim, may fool a victim browser into believing that the revoked certificate is still valid by providing (replaying) a previous DNS response - of a time when the certificate was *indeed* valid; a time *before* the revocation of the certificate.

4.1.2 Adding Epochs

One way to mitigate this attack is to divide the time in *epochs* and timestamp (and sign) the validity information of each certificate. This information (timestamped and signed) will be valid for only one epoch. If the attacker tries to replay valid information from a previous epoch, the web browser (who expects a timestamped and signed reply) will realize that this information is from a previous epoch: it is stale - out-of-date - invalid. The introduction of *epochs* significantly mitigates the replay attacks: attackers cannot just take old validity information and replay it to victim's browsers - this would not work anymore. Of course, besides the significant improvement of the original algorithm, is still vulnerable to replay attacks *within* an epoch. However, by fine-tuning the duration of an epoch, the window of vulnerability can be made arbitrarily small.

Unfortunately, reducing this window of vulnerability by aggressively decreasing the duration of an epoch does not come without a cost: at each and every epoch, each and every valid certificate has to be timestamped and signed. Depending on the Certificate Authority, this process can be expensive and/or require a large amount of time to happen.

4.1.3 Collective Records

To minimize this overhead we introduce the notion of *collective records*, which reduce the number of signatures that need to be done in each epoch. These collective records contain a set of domains along with *some* validity information about the certificates of these domains. To be precise, each collective record contains (i) the names of its domains, and (ii) the latest version number of each domain's revocation list. Collective records are timestamped (once per epoch) and signed by the CA. Ordinary (not collective) records contain information about a domain, its certificates, and the last version number of each domain's revocation list. Although ordinary records are signed, they are *not* timestamped, and thus, they do not need to be signed at each and every epoch - they just need to be signed once: at the time of their creation. The DCSP now works as follows:

- When a certificate of a domain D is revoked, the CA adds an individual record about domain D in the DNS system. The record contains a list of all revoked (but not expired) certificates of domain D along with the latest version number of this revocation list. To preserve authenticity, all records are signed by the CA.

[3]The exact way of *how* is this information stored into the DNS system is explained subsequently.

- At each epoch, and for each collective record, the CA updates the collective record with the most recent information. This record is timestamped and signed.

When charged with the task of checking the validity information of a certificate of a domain D, web browsers bring from DNS (i) the individual record of domain D and (ii) the collective record of domain D. If the timestamp of the collective record does not match the current epoch, a DNS attack is probably going on. Otherwise, if the version numbers of the two records do not match, a DNS attack is going on. If both are not the case, the revocation information for the status of the certificate is valid and can be used.

4.2 Certificate Format

In order for our system to work, we add a custom certificate extension to an existing certificate that is marked as non-critical, the "DNS Revocation Domain". The information can also be present in the "Certificate Authority Information Access" as a new method, under the name "DCSP". The URI of this entry is set to a valid DNS domain which will be queried for answers using the DNS protocol. We will refer to this value as *Revocation Domain*.

4.3 DNS Record Format

The Certificate Authority is responsible for adding the DNS records to the authoritative name servers for the Revocation Domain. Accordingly, the CA divides the total amount of domain names into small groups. This can be done arbitrarily by the CA and groups can be of any size. We have performed some measurements on how the size of these groups affects the performance of browsers, as can be seen in Figure 2. Additionally, the CA must perform at least as many cryptographic signs per day as the total number of groups available. Each group must be given a unique name that can be used as a subdomain, for example "group-00001". The size of the group is dynamic and can easily change at any time, without impacting old certificates.

4.3.1 Collective Record

After the grouping has been performed, the CA generates a new domain in the form of "group-name.revocation.ca-name.com". Currently there is no technical limitation in our system, even if arbitrary domains are used for each group, but a more elegant and scalable approach is recommended here. This domain name is then included in the certificate as the Revocation Domain.

The CA then proceeds to add a TXT record for each domain[4] in this group, following the format below:

$$\underline{DOMAIN} - \underline{VERSION} - (\underline{POSITION}/\underline{TOTAL})$$

In the $\underline{DOMAIN}$ variable, the top level domain of the site is included. The content of the $\underline{VERSION}$ variable is the latest revision of this domain's records. For the $\underline{POSITION}$ and $\underline{TOTAL}$ variables we include the domain name's serial number within the particular group and the group size respectively. An example record for the first revision of the domain `example.com`, which is the second one in this 100-domain sized group, can be seen below:

$$example.com - 1 - (2/100)$$

[4] In case of a certificate with an IP Address, its "in-addr.arpa" format can be used.

In addition to that, two extra TXT records must be set for the same domain. These two records include the Certificate Authority Signature of the above TXT records, which the CA must update daily as well as the date of the signing. The Private Key used for performing the signatures must be the one that signed the original domain certificate, i.e. the last intermediate certificate. The process of generating the signature consists of concatenating the content of all TXT records above, ordered by their third part, then appending the date record, and finally hashing the result using a Cryptographic Hash function (e.g., SHA-256). The output of this operation is converted to Base-32 [9] and included as follows:

$$SIG - \underline{BASE_{32}SIGNATURE}$$

The date record has the following format, where the variable $\underline{SIGNATUREDATE}$ contains the date these records were updated, for example "20160101". This date must be treated as a UTC date to avoid problems with timezones.

$$DATE - \underline{SIGNATUREDATE}$$

4.3.2 Individual Record

When a catalog for each domain name and its latest revision number has been established, the CA needs to add the individual TXT records for each domain inside that group. These records are added in a domain of the following form:

$$\underline{domain.name}.\underline{groupname}.revocation.ca.com$$

Hence, for the domain `example.com` in group `iana`, the records will be available in: `example.com.iana.revocation.ca.com`. If there is no revoked certificate for this domain name, the word $\underline{NONE}$ followed by the revision number and the string "(1/1)" is included, as can be seen in the following:

$$\underline{NONE} - \underline{REV} - (1/1)$$

Of course, the $\underline{REV}$ variable is the Revision Number, an always increasing positive integer. If there are revoked non-expired certificates, then one record is added for each revoked certificate. These records have the following format:

$$\underline{CERTID} - \underline{REV} - (\underline{POSITION}/\underline{TOTAL})$$

The $\underline{CERTID}$ variable contains the SHA-1 fingerprint of the revoked certificate. The hashing algorithm used here can change, but has been selected since most existing software calculates certificate fingerprints using it. The $\underline{REV}$ part is the same as the $\underline{NONE}$ record's. The $\underline{POSITION}$ and $\underline{TOTAL}$ parts include the position of this certificate and the total amount of revoked certificates in this domain.

In addition, two extra TXT records are added. The first one is the "DOMAIN" record that has the following format:

$$DOMAIN - \underline{DOMAINNAME}$$

Here, the variable $\underline{DOMAINNAME}$ includes the domain name, just as it appears in the subdomain of the Revocation Domain, for example `example.com`. The final field is the "SIG" record, which is formatted as below:

$$SIG - \underline{BASE_{32}SIGNATURE}$$

The content of $\underline{BASE_{32}SIGNATURE}$ is the Base-32 encoding of the Cryptographic Signature. This signature is generated by appending all records including a revision number, ordered by their final part, the "DOMAIN" record and then, signing the SHA-256 hash of that string.

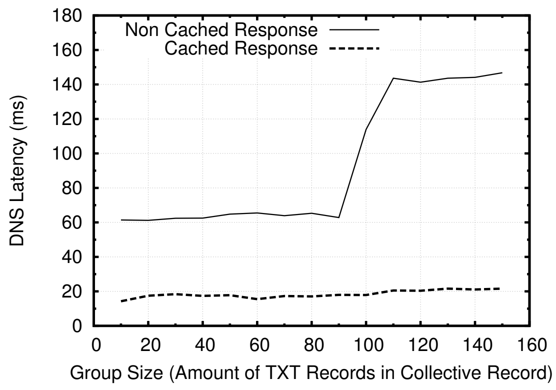

Figure 2: Average query time for various group sizes. For small group sizes, it is around 60 ms, and around 140 ms for the larger ones. The rapid increase after 90 records is due to the switch from UDP to TCP. When the record is cached, it is around 20 ms.

Every time a new certificate is revoked or a revoked certificate expires, the CA must increase the Revision Number in the TXT records that refer to this particular domain and then sign the result again.

4.4 Validation of Certificates

During the initiation of a HTTPS connection, the web server provides a chain of certificates to the web client, which allows it to verify that the web server is indeed, who it claims to be. This chain consists of the leaf certificate that corresponds to the particular domain the client has requested, the root certificate and all the intermediate ones. Accordingly, each certificate in the chain, starting from the root towards the leaf, verifies the validity of the next one, and eventually, the last intermediate certificate verifies the leaf certificate that corresponds to requested domain.

4.4.1 Leaf Certificates

After receiving a certificate chain, the web browser verifies that the leaf certificate is valid and non-expired. After that initial validation, the browser must also ensure that this certificate has not been revoked. In our system, it reads the Revocation Domain from the provided certificate and then asynchronously performs two DNS queries, one to the Revocation Domain and one to a subdomain of it, the connection's top-level domain, asking for their TXT records. For example, a connection to `www.example.com` with a Revocation Domain of `iana.revocation.ca.com` will query the DNS server for the domains `iana.revocation.ca.com` and `www.example.com.iana.revocation.ca.com`.

As soon as the two responses arrive, the Individual Record is checked first. If the fingerprint of the certificate matches one of the fingerprints that are included in the individual record, and the signature is valid, the browser knows that this certificate has been revoked. If not, it proceeds to check if the revision of this record is indeed the latest revision for this domain, using the Collective Record. If the revision numbers do not match or the signature of the Collective Record is invalid, or the timestamp of the Collective Record is over 24 hours old, the browser knows that an attack is taking place and handles the situation accordingly.

4.4.2 Certificate Chains

According to the design of our system, the CA DNS server can provide information on all revoked, non-expired certificates for a particual domain name, as well as proper metadata for the client to ensure that the DNS response contains the latest version of this domain's revocation list. The above method covers the revocation of all leaf certificates for the individual websites. For the intermediate certificates, a different approach is followed.

Each non-root, non-leaf certificate of the chain is verified using a small CRL that can be downloaded daily. Recent studies have shown that the number of intermediate certificates is small [5], [15] and that they aren't revoked as often as leaf certificates [15]. Google Chrome and the Chromium browser already follow a similar mechanism, CRLSet [2].

5. DISCUSSION

5.1 Privacy of Requests

The current server-to-client, query-based model of OCSP certificate revocation mechanism clearly exposes the browsing history of the end users to the Certificate Authorities jeopardizing the users' privacy. Each query of the user's browser reveals the domain that the user is ready to visit allowing the CAs through their OCSP responders, to reconstruct parts of the users' browsing history.

OSCP-stapling has been proposed for solving this problem, but it remains entirely on the intention of the website's administrator to periodically retrieve a fresh OCSP response and staple it to the website's certificate. According to the latest report [10], as of November 2015, only 21% of the Alexa top million domains that support HTTPS employ OCSP-stapling.

DCSP is able to ensure that users' browsing history will not be revealed to any third entity except from the DNS, which already is aware of this information, after providing the IP resolution of the particular domain. Therefore, our system is able to provide similar privacy guarantees to OCSP stapling, but requiring much less signatures from the CAs.

6. EVALUATION

Adding many TXT fields in a collective record may seem that it will increase the latency of the end users when accessing DNS to check validity of recent certificates. To evaluate the impact that TXT records have in DNS latency we set up an experiment where a client repeatedly queries a collective record. The client is located in Frankfurt, while the authoritative server is located in London. For the DNS resolver we use `4.2.2.1`, the well-known Level 3 resolver. We vary the number of TXT fields in the collective record from 10 to 150 and we measure the latency to retrieve the DNS record. In Figure 2 we plot the average response times and as we see, for group sizes up to 90 TXT records the latency is practically constant, around 60 ms. For more than 100 TXT records the latency reaches 140 ms because the DNS protocol switches from UDP to TCP incurring extra RTT due to the TCP handshake. Fortunately, when the requested DNS records are cached the latency drops down to around 20 ms.

Overall we see that the average response time of DCSP is in the area of a few tens of milliseconds, which is an order of magnitude improvement compared to the respective response time of OCSP.

7. RELATED WORK

There are several approaches proposed as an alternative to the existing revocation mechanisms. However, most of them focus on the factor of trust and how this will get more distributed, replacing the traditional CA model with a different more decentralized one. As a consequence, all of these approaches fail to deal with the actual issue of SSL in a world that time matters: latency.

One of them is the Perspectives project [22], which has inspired the Convergence [6, 16] strategy for replacing SSL Certificate Authorities. It employs a crowd-sourcing network of "notary servers", which regularly monitor the websites to build a global database of the certificates used by each site. These servers can be maintained by anyone e.g., the EFF, public organizations, private companies, Google, a university or even a group of friends. This way, the users are free to pick the entity of their trust and query the validity of the certificates. There are cases where users might need to have more than one notary server to vouch for the certificate, in order to have a valid response. This operation, although provides a certain level of redundancy, imposes a significant amount of latency to the user's new browsing session. To make matters worse, besides the distributed nature and the provided trust agility of the project, up to now there is a very limited number of well-maintained notary servers [17].

Another approach aiming to replace the CAs is the DNS-based Authentication of Named Entities (DANE) [8]. DANE, similar to DCSP, also leverages DNS infrastructure to authenticate SSL certificates. DANE introduces a new type of DNS record, named TLSA, in which stores the *whole* certificate of a domain and uses DNSSEC to validate its integrity. According to a recent study [23] the adoption of DANE in the wild is yet far too low; from 485,000 collected signed zones, there were only 997 TLSA names. To make matters worse, among these TLSA records there were 7-13% which were invalid, mismatching server's certificates over the time of the measurement. In addition, 33% of the TLSA responses were larger than 1500 Bytes, which results to IP fragmentation, imposing additional latency to the query procedure and leaving the protocol vulnerable to fragmentation attacks [7]. DCSP does not abolish the role of CAs, on the contrary, it is the CAs who are responsible of maintaining the DNS records, guaranteeing this way their validity.

The table below shows the benefits and drawbacks of the approaches so far:

Method	Privacy	Low Number of Signatures	Low Latency	Freshness of Information	Placement of Trust
OCSP	✗	✗	✓	✓	CA
OCSP Stapling	✓	≈	✓	≈	CA
CRLs	✓	✓	≈	≈	CA
DANE	✓	-	✓	✓	Admin
DCSP	✓	✓	✓✓	✓	CA

8. CONCLUSION

In this paper we study the existing certificate revocation approaches as well as their relative shortcomings, and propose DCSP: a new lightweight DNS-based certificate revocation protocol.

DCSP leverages DNS as a fast cache to distribute revocation information. Using UDP-based communication and the widespread adoption of DNS, DCSP manages to enable browsers query certificate revocation information much faster than current state-of-the-art approaches. Initial performance results show that DCSP has the potential to perform an order of magnitude faster than OCSP. By introducing a grouping approach which we call "collective records", and capitalizing on multiple DNS TXT fields to implement it, we significantly reduce the amount of signatures required by the Certificate Authorities: a significant improvement compared to previous approaches, such as OCSP stapling.

9. ACKNOWLEDGMENTS

This work was supported by the project GCC, funded by the Prevention of and Fight against Crime Programme of the European Commission – Directorate-General Home Affairs under Grant Agreement HOME/2011/ISEC/AG/INT/40000 02166 and the FP7 Marie-Curie iSocial ITN, funded by the EC under Grant Agreement No 31680.

10. REFERENCES

[1] Codenomicon. The heartbleed bug. http://heartbleed.com.

[2] G. R. Corporation. An evaluation of the effectiveness of chrome's crlsets. https://www.grc.com/revocation/crlsets.htm.

[3] Debian. Debian security advisory. https://www.debian.org/security/2008/dsa-1571.

[4] R. Duncan. How certificate revocation (doesn't) work in practice. http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html.

[5] Electronic Frontier Foundation. SSL Observatory. https://www.eff.org/observatory.

[6] D. Goodin. Qualys endorses alternative to crappy ssl system. http://www.theregister.co.uk/2011/09/30/qualys_endorses_convergence/.

[7] A. Herzberg and H. Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *IEEE Conference on Communications and Network Security.*

[8] P. Hoffman and J. Schlyter. The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA. https://tools.ietf.org/html/rfc6698, 2012.

[9] S. Josefsson. The base16, base32, and base64 data encodings. https://tools.ietf.org/html/rfc4648.

[10] H. Kario. SSL/TLS scan results. https://securitypitfalls.wordpress.com/category/scan-results/.

[11] L. Kelion. Heartbleed hacks hit mumsnet and canada's tax agency. http://www.bbc.com/news/technology-27028101.

[12] S. Landau. Making sense from snowden: What's significant in the nsa surveillance revelations. *IEEE Security & Privacy*, 2013.

[13] A. Langley. No, don't enable revocation checking. https://www.imperialviolet.org/2014/04/19/revchecking.html.

[14] A. Langley. Revocation doesn't work. https://www.imperialviolet.org/2011/03/18/revocation.html.

[15] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. An end-to-end measurement of certificate revocation in the web's pki. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, IMC '15.

[16] M. Marlinspike. Convergence. http://www.convergence.io.

[17] M. Marlinspike. Notaries. https://github.com/moxie0/Convergence/wiki/Notaries.

[18] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. https://tools.ietf.org/html/rfc6960, 1999.

[19] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste. The Cost of the "S" in HTTPS. In *Proceedings of the 10th ACM CoNEXT*, 2014.

[20] Netcraft. CRL sites ordered by average body size. http://uptime.netcraft.com/perf/reports/performance/CRL.

[21] Netcraft. Total http time of ocsp sites. http://uptime.netcraft.com/perf/reports/performance/OCSP.

[22] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving ssh-style host authentication with multi-path probing. In *USENIX 2008 Annual Technical Conference.*

[23] L. Zhu, D. Wessels, A. Mankin, and J. Heidemann. Measuring dane tlsa deployment. In *Traffic Monitoring and Analysis*, Lecture Notes in Computer Science. 2015.