Checking Certificate Revocation Efficiently using Certificate Revocation Guard*

Qinwen Hu, Muhammad Rizwan Asghar, Nevil Brownlee

School of Computer Science, The University of Auckland, New Zealand

Abstract

In the Public Key Infrastructure (PKI) model, digital certificates play a vital role in securing online communication. Communicating parties exchange and validate these certificates and the validation should fail if the certificate has been revoked. However, some existing studies [1, 2] raise an alarm as the certificate revocation check is skipped in the existing PKI model for a number of reasons including network latency overheads, bandwidth costs, storage costs and privacy issues. In this article, we propose a Certificate Revocation Guard (CRG) to efficiently check certificate revocation while minimising bandwidth, latency and storage overheads. CRG is based on OCSP, which caches the revocation status of certificates locally, thus strengthening user privacy for subsequent requests. CRG is a plug and play component that could be installed on the user's machine, at the organisational proxy or even in the ISP network. Compared to a naive approach (where a client checks the revocation status of all certificates in the chain on every request), CRG decreases the bandwidth overheads and network latencies by 95%. Using CRG incurs 69% lower storage overheads compared to the CRL method. Our results demonstrate the effectiveness of our approach to improve the certificate revocation process.

1. Introduction

Online transactions are becoming more ubiquitous nowadays. In today's Public Key Infrastructure (PKI), the Secure Socket Layer (SSL) or Transport Layer Security (TLS) is the most widely deployed protocol for securing online communication. It offers mutual authentication and establishes a secure channel that provides encryption for end-to-end communication over the Internet. There are a wide range of services that use SSL/TLS, such as secure web browsing (*i.e.,* HTTPS), secure virtual communication (*i.e.,* VPNs) and secure file transfer (*i.e.,* SFTP).

The SSL/TLS protocol is based on certificate validation, which takes place before establishing the secure channel. This validation fails if any certificate in the certificate chain (involving root, intermediate and leaf) is revoked, which could be due to compromised or stolen private keys or fraudulent issuance [4, 5]. There are serious security consequences if the revocation process is incomplete or poorly-implemented, for instance a leaf certificate could enable attackers to easily eavesdrop the communication until the certificate is expired. The situation is even worse when it comes to an intermediate (or root) certificate because it allows attackers to issue 'valid' certificates for any domain. Therefore, it is essential to check certificate revocation during the certificate validation process.

In 2014, more than 80,000 certificates were affected by the Heartbleed bug [6]. Since then, certificate revocation checking has become more pressing to eliminate the possibility of establishing a secure channel using a revoked certificate, thus preventing any man-in-the-middle attack. There are two commonly used methods to check certificate revocation, namely Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP). The CRL method allows clients to fetch a list of revoked certificates while the OCSP method responds with the status of a certificate when requested. There is also an extension of OCSP, called OCSP Stapling, that allows clients to pull the revocation status from the application server instead of the OCSP server, thus reducing network latency by saving an extra round-trip.

Most recent studies [1, 2] have examined the revocation checking in all major browsers. They find that many web browsers check certificate revocation with the pre-defined conditions. Liu *et al.* [2] show that CRLSet[1] contains revoked certificates is surprisingly low (0.35%) from their experiments. Google Chrome uses a CRLSet instead of the recommended methods for checking certificate revocation. Firefox only checks leaf and Extended Validation (EV)[2] certificates that contain the OCSP responders. Some desktop browsers bypass the leaf certificate revocation checking if the revocation information cannot be found in an issued certificate. Worse, mobile browsers do not perform certificate revocation checking. The main reason behind that could be extra network latency, bandwidth costs, stor-

*Part of the material has appeared in the proceedings of the 41st Annual IEEE Conference on Local Computer Networks (LCN) held in November 2016 under the title "Certificate Revocation Guard (CRG): An Efficient Mechanism for Checking Certificate Revocation" by Qinwen Hu, Muhammad Rizwan Asghar, and Nevil Brownlee [3].

Email addresses: qhu009@aucklanduni.ac.nz (Qinwen Hu), r.asghar@auckland.ac.nz (Muhammad Rizwan Asghar), n.brownlee@auckland.ac.nz (Nevil Brownlee)

[1] CRLSet contains a list of revoked certificates. Typically, CRLSet is made public. Through a public URL, CRLSet could be fetched periodically by Chrome.

[2] Extended validation is a mechanism for CAs to assert that the identity verification process has followed a set of established criteria.

age overheads, or even privacy risks. Unfortunately, bypassing the certificate revocation check can result in session hijacking and unauthorised issuance of new certificates.

Over the last two decades, there has been a lot of work [7, 8, 9, 10, 11, 12, 13] on the effectiveness of certificate revocation and associated privacy concerns. In [7], Topalovic *et al.* suggested using short-lived certificates. However, their design results in high bandwidth overheads due to the hassle of fetching new certificates valid only for a short span of time. Kocher [8] proposed a tree-based approach for checking certificate revocation, requiring high computational cost at the client end. There are also some other solutions, but most of those solutions [9, 10, 11, 12, 13] required changes in the legacy infrastructure.

In this article, we propose a new certificate revocation method: Certificate Revocation Guard (CRG). CRG is based on OCSP and caches the certificate revocation status locally. It takes advantages of both OCSP and CRL methods. It saves bandwidth cost, network latency and storage cost when compared with OCSP and CRL. Due to local caching, CRG improves user privacy by not sending every OCSP request to the revocation manager. We emphasise that CRG does not require any changes in the legacy infrastructure and can be installed on the user's machine, at the organisational gateway/proxy or even in the ISP network.

This article extends our initial work [3]. Compared with our 4-page conference paper [3], this article provides more details on performance issues in existing revocation mechanisms and explains the factors that cause users not to use the existing revocation mechanisms. Further, our previous work only considered the feasibility of using CRG from the performance perspective. Moreover, we provide a complexity analysis of CRG. In this work, we have put more considerations on how to deploy this CRG in various environments (such as from the client end, the middlebox, and the service provider end), as well as the privacy and security challenges we may face and how to properly mitigate these challenges. Also, a comprehensive comparison between our solution and the existing techniques has been discussed. We analysed the advantages and disadvantages of using each solution in terms of performance and deployment costs. In what follows, we highlight some new contributions in this article.

- We discuss the feasibility of deploying CRG in the current infrastructure. The advantages and disadvantages of each deployment scenario are listed, and feasible solutions are proposed.
- We explain privacy and security concerns. For instance, CRG does not interact with the revocation manager for every request. After making the first request to the revocation manager, the OCSP response is cached for a certain period in which it remains valid. Consequently, CRG will not disclose user activities to third parties. Besides, we describe how CRG could defend against the most common attacks. Our security analysis shows that the proposed approach is secure.
- We provide a comprehensive comparison between CRG and the existing techniques. Based on the efficiency, privacy, and some other parameters, we find CRG is more effective than other solutions and easier to deploy as well as it does not require changes in the existing infrastructure.

The remainder of this article is organised as follows. In Section 2, we provide some background and explain the limitation of using existing methods. Our new solution, CRG, is explained in Section 3. Section 4 analyses performance and complexity of CRG. Section 5 discusses privacy and security concerns. Different deployment scenarios and performance improvements are covered in Section 6. In Section 7, we review related work and provide a comprehensive comparison between our solution and existing solutions based on network resource consumption, privacy, and deployment cost. Finally, in Section 8, we conclude this article and provide research directions for future work.

2. Background

There are a number of studies, such as [14] and [15], showing that many applications (such as online banking apps, VPNs, and web browsers) use SSL/TLS for establishing a secure channel to exchange sensitive information. The main goal of this secure channel is to offer mutual authentication and preserve integrity and confidentiality of exchanged information. Basically, SSL/TLS allows clients to verify identity of the target server before establishing any private communication. SSL/TLS falls under PKI, which we briefly review in this section. In particular, we describe a digital certificate, which is the core of PKI. We also explain how a digital certificate is issued, verified and revoked.

Digital Certificate. A digital certificate is used as a key component that binds a subject's ID to its public key. It consists of a number of fields including, but not limited to:

- the subject's name;
- the subject's public key;
- the certificate issuer, *a.k.a.* the Certificate Authority (CA), an authorised entity that issues digital certificates;
- a serial number that uniquely identifies the digital certificate;
- validity period indicating start and expiration dates of the certificate;
- a certificate chain, building a chain of certificates up to the root CA, which is a trust anchor[3] in the PKI model; typically, there are three types of digital certificates: a leaf, an intermediate and a root CA, where latter two types are part of the certificate chain; and information about the authority that manages revocation information.

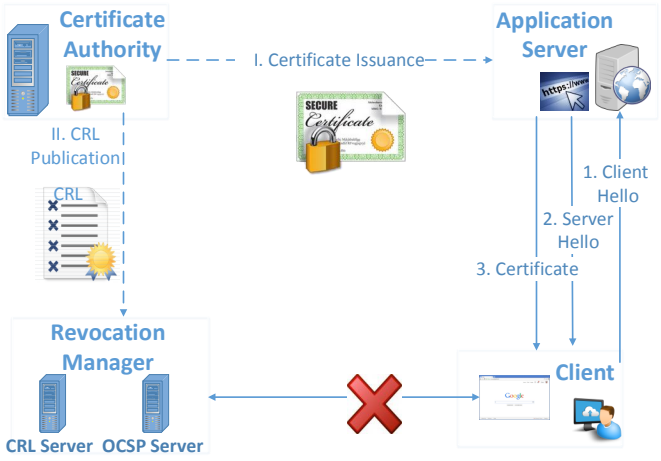

Figure 1: A basic overview of the PKI model indicating certificate issuance and revocation as well as showing SSL/TLS certificate exchange and validation.

Certificate validation is an essential part of SSL/TLS. While validating a certificate, it is also important to check certificate revocation. Basically, a certificate might need to be revoked before its expiry if the private key corresponding to the certificate is compromised or stolen. If not revoked properly, leaf and intermediate certificates with compromised/stolen private keys can result in session hijacking and unauthorised issuance of new certificates, respectively.

There are two commonly used methods for disseminating information about certificate revocation: CRL and OCSP. Ideally, as part of certificate validation, the certificate revocation status must be checked.

Figure 1 illustrates the model and visualises the problem. As we can see in the figure, a CA issues a certificate to an Application Server (Step I). If requested, the CA can revoke an issued certificate and publishes a CRL (Step II), which is managed by the Revocation Manager. In our model, we assume that the revocation manager manages both a CRL server and an OCSP server.

In Figure 1, dotted lines represent the certificate issuance and revocation workflow while solid lines show the sequence of establishing an SSL/TLS connection. In SSL/TLS, a Client (a web browser) sends a Client Hello message (Step 1) to request a new SSL/TLS connection. The application server responds with a Server Hello message (Step 2) followed by a server Certificate (Step 3). After receiving a certificate, the client validates the certificate. However, as reported by some studies [2, 16], some clients miss the certificate revocation check as we can see in Figure 1.

In the following, we describe both certificate revocation methods and discuss advantages and disadvantages of each one.

2.1. Certificate Revocation Lists (CRLs)

A CRL contains a list of revoked certificates [17]. Each CRL entry represents a serial number, a revocation timestamp and a revocation reason (optional) for the revoked certificate. In this method, CAs embed the CRL's URL in each released certificate, which can be downloaded by the client. The CRL file helps the client to check whether the current certificate's serial number is listed in that CRL. Every CRL has an expiration date that allows the client to cache the CRL file until it expires. This method requires CAs to periodically re-issue a new CRL, even if it is not updated. If the cached CRL expires, the clients must download an updated CRL file. The major limitations of this method include high bandwidth, network latency and storage overheads for the client.

2.2. Online Certificate Status Protocol (OCSP)

As an alternative solution to CRLs, OCSP [18] reduces bandwidth and storage overheads by allowing clients to query an OCSP server for the revocation status of a single certificate. Similar to CRLs, the OCSP server's URL can be found in the issued certificates. The client sends an HTTP GET request to the OCSP server for querying the status of the given certificate. There are three different types of OCSP status responses: *good*, *reject* and *unknown*, indicating that a certificate is not revoked, it is revoked or the OCSP server does not know about the certificate being requested. This method has its own drawbacks. The client queries the certificate revocation status every time instead of caching the list of revoked certificates, thus increasing the latency as well as privacy risks [7].

Using OCSP, the privacy of users could be compromised in a sense that it requires the client to contact an OCSP server for each SSL connection. To solve the privacy issue, there is also an extension of OCSP that allows clients to get the certificate status directly from the application server. This new method is called OCSP Stapling [19] in which the application server queries the OCSP server and caches the status of its own certificate. The OCSP stapling method solves the privacy issue in OCSP and reduces the latency. However, this extension is not supported by all the servers. Second, this extension still increases some bandwidth and latency because an application server has to send back the OCSP status to the client, which reports the certificate revocation status. Besides, for long distance TCP sessions, the session latency increased if the sessions need multiple round trips.

3. Proposed Solution

In this article, we propose CRG that aims at addressing problems associated with CRL and OCSP methods. The basic workflow of CRG is outlined in Figure 2. As we can see, both certificate issuance (Step I) and CRL publication (Step II) stay the same. Besides, there is no change in the SSL/TLS flow. That is, we do not require any modification in the SSL/TLS protocol so steps 1, 2 and 3 are the same as already illustrated in Figure 1. However, CRG intercepts the Certificate message of the SSL/TLS protocol (Step 3) and consults its status with the Revocation Manager (Step 4), which returns either valid or revoked (Step 5). If the status is valid, CRG sends (Step 6) the original certificate it received in Step 3. If the certificate has

[3] A list of trust anchors is shipped with Operating Systems (OS).

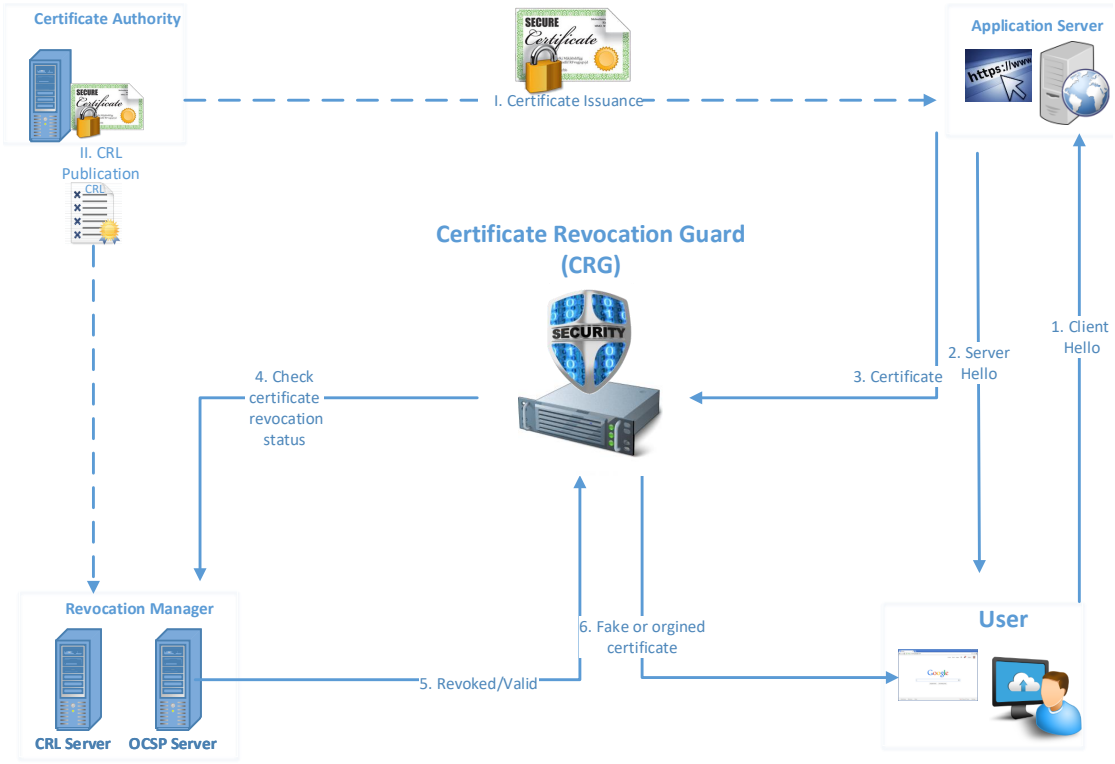

Figure 2: Basic overview of our proposed approach in which CRG intercepts the SSL/TLS Certificate message (Step 3) and consults the certificate revocation status with the revocation manager (Step 4). CRG caches the response (Step 5) and, depending on the valid or revoked certificate, sends either the original or a fake certificate (Step 6) to the client, respectively.

been revoked then CRG has to communicate this information to the client. It is important to mention that we do not modify the underlying protocol between the client and the application server. Instead, we generate a fake certificate to indicate that the certificate is not valid anymore. In other words, the fake certificate is such that its validation on the client will fail.

Certificate Revocation Guard (CRG). Figure 3 provides a detailed view of CRG. As we can see, CRG is decomposed into four main components: Certificate Checker, Certificate DB, Certificate Validator and Certificate Modifier. From the deployment point of view, CRG could be installed on the user's machine, at the organisational proxy or even at the ISP. Each deployment choice has possible trade-offs that are discussed in detail in Section 6. The functionality of each entity is described below:

Certificate Checker. We define the certificate checker as a core component in our design. It is an interface between the application server and rest of the components in CRG. It is responsible for intercepting the SSL/TLS certificate messages. After intercepting a certificate message, for each certificate in the certificate chain (except root certificates), it checks whether the certificate is valid or revoked. One challenge of using the certificate's serial number as the search key is that the serial number may not be unique. To achieve this, CRG uses the certificate's serial number and the certificate's issuer information for classifying the different certificates with identical serial numbers.

For each certificate, there are two possibilities: either the certificate is an existing one, meaning its status was already cached recently, or it is a new certificate (or an existing one with expired caching time, which is explained later). The former case saves CRG bandwidth and reduces the latency. Therefore, it first checks with the certificate DB. In the latter case, it has to request the certificate validator for checking the revocation status. In either case, it sends the certificate chain together with a revocation decision to the certificate modifier. The decision is considered *yes* if the status of all the certificates in the certificate chain is valid. Otherwise, the decision will be *no*.

Certificate DB. The certificate DB creates and manages the revocation information for each requested certificate. For each certificate entry, the database maintains the following fields: the certificate's serial number, the revocation status, the next update time when the revocation status needs to be checked again, the certificate type, frequency of request (indicating how many times a certificate has been requested) and the certificate issuer as depicted in Table 1. We assume that the certificate DB is populated based on requests by the certificate checker. From the configuration point of view, CRG is flexible enough. Specifically, it is possible to allocate a certificate DB of a certain size. In that case, the certificate DB has to be downsized once it approaches the configured size limit. To this end, we have different choices. One straightforward choice is to remove entries with a lower frequency of requests, because they are less likely to be requested again. We can also filter based on cer-

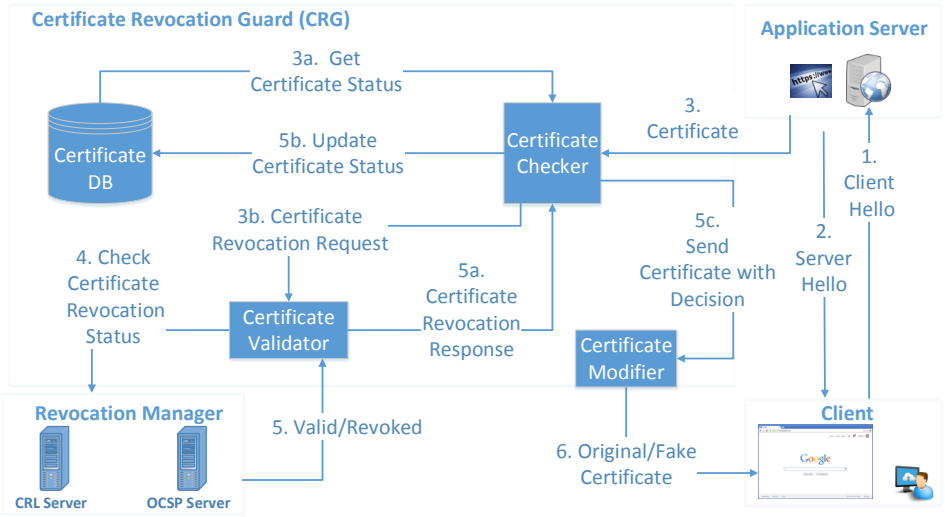

Figure 3: A detailed view of CRG: in case of a new certificate, the Certificate Checker intercepts the SSL/TLS Certificate message (Step 3), consults the Certificate DB (Step 3a) before forwarding the Certificate Revocation Request to the Certificate Validator (Step 3b). The Certificate Validator requests the certificate status (Step 4), gets the status (Step 5) and sends the response to the Certificate Checker (Step 5a). The Certificate Checker updates the Certificate DB and forwards the decision together with the certificate (Step 5c) to the Certificate Modifier, which can send the original or faked certificate to the client (Step 6).

Table 1: An example of a database with cached certificates, each one identified with a serial number, stored with its revocation status (valid or revoked), next update time when the Revocation Manager needs to be contacted again, certificate type (leaf or intermediate), frequency of request and certificate issuer.

Serial Number	**Revocation Status**	**Next Update Time**	**Certificate Type**	**Frequency of Request**	**Certificate Issuer**
123456789	Valid	25-04-2016	Leaf	50	GeoTrust Global CA
345678912	Revoked	28-04-2016	Leaf	40	GeoTrust Global CA
781234567	Valid	30-04-2016	Intermediate	120	GeoTrust Root CA
116890045	Revoked	29-04-2016	Intermediate	100	Baltimore CyberTrust Root
...	...	...	...	...	...

tificate type, next update time, revocation status or certificate issuer. In case of filtering based on certificate type, we can remove entries with leaf certificates assuming intermediate certificates will be requested more frequently. Another potentially attractive choice is removing all entries where next update time has been reached (or is about to be reached). Likewise, we can choose either valid or revoked ones. The certificate issuer could be a more personalised choice that takes into account what are typical issuers in the user's vicinity because they have the likelihood to be requested the most. Alternatively, we can consider refreshing the certificate DB based on most recently used or least recently used in which case we also need to record the time of the request. Without loss of generality, we can specify a policy that could be based on a set of possible designed choices optionally with complex conditions.

Certificate Validator. The goal of the certificate validator is to check the certificate revocation by contacting the revocation manager. The certificate checker requests the certificate validator by providing information about the authority that manages revocation information, *i.e.,* the revocation manager. After receiving the request, the certificate validator connects with the revocation manager and gets the certificate status, which is either *valid* or *revoked*. It is important to distinguish between two authorities that manage revocation information: a CRL server and an OCSP server. A certificate provides information about at least one of them. Typically, a certificate includes URLs of both servers. Using a CRL server may provide more information than required because the certificate validator will get a list of all the revoked certificates, incurring high bandwidth and storage costs but being more privacy-preserving and efficient from the point of view of caching the complete list for subsequent requests. On the other hand, if we contact the OCSP server, we can lower storage and bandwidth overheads at the cost of increased latencies for subsequent requests. However, OCSP is less privacy-preserving as the revocation manager would be able to profile behaviour of the client. For efficiency reasons, we use the OCSP method, but we cache the response in the certificate DB so that we can minimise potential privacy issues as well as decrease the latency and bandwidth overheads.

Certificate Modifier. From the usability point of view, it is a common practice to hide the lock icon in the address bar and show a security warning to the user if the certificate validation fails. Taking inspiration from this approach, CRG issues a fake certificate if the certificate has been revoked. The fake certificate is specifically designed to signal the client that the certificate has been revoked. A fake certificate can easily be made from the original one by modifying the certificate expiry. Alternatively, we can change other fields (such as signatures) in the certificate that can result in unsuccessful validation. However, the aforementioned approaches of generating fake certificates will result in high latency due to modifications in original certificates. We can follow a radically different approach by pre-computing a set of fake certificates that could be used by the certificate modifier. If the original certificate is valid, the certificate modifier does not take any action and forwards the original certificate to the client.

CRG in Action. In the following, we discuss three cases indicating three situations: when CRG receives a new certificate, a cached certificate with the next update time reached and a cached certificate without the next update time reached. The two former ones represent the most complex case because in both situations CRG has to contact the revocation manager.

Case 1) In case of a new certificate, after receiving the certificate (Step 3), the certificate checker gets the status update from the certificate DB (Step 3a). Since the certificate is new, there will be no entry in the database. The certificate checker has to contact the certificate validator (Step 3b). The certificate validator checks the status of this new certificate (Step 4) and gets the response (Step 5), which it forwards back to the certificate checker (Step 5a). To cache the status, the certificate checker updates the database (Step 5b) by inserting a new entry for this new certificate and initialises the frequency of request field by 1. The certificate checker also sends the decision to the certificate modifier (Step 5c). Note that both Steps 5b and 5c can run in parallel. However, for reducing the latency, we recommend Step 5c should take precedence. Based on the decision made by the certificate checker, the certificate modifier finally communicates back to the client (Step 6).

Case 2) If there is a cached certificate with the next update time reached, like the case of a new certificate, CRG performs the same steps (*i.e.,* Steps 3b, 4, 5, 5a and 6) except Step 5b. For updating status of a certificate having an entry in the database with the next time reached, CRG updates its revocation status and next update time as well as incrementing the frequency of request by one.

Case 3) If the certificate is a cached one without the next update time reached, after receiving the certificate (Step 3), the certificate checker gets the status update from the certificate DB (Step 3a). Since there is already an entry in the table, it increments the frequency of request by one. Next, it directly sends the response to the certificate modifier (Step 5c), which ultimately forwards the certificate to the client (Step 6).

Extending CRG. In general, CRG is able to efficiently deal with certificate revocation. CRG relies on the assumption that status of a certificate will stay the same until the next update time. The next update time is part of the OCSP or CRL response. However, in practice, the certificate can get revoked before this time. In such cases, the basic version of CRG will not be so effective. To tackle those cases, we propose to extend CRG. The idea is to subscribe for instant revocation of certificates whenever a new certificate gets cached. This will require the revocation manager to introduce an additional server that is responsible for handling subscriptions. This additional server will update all subscribers with information about certificates that get revoked before the next update time. The certificate validator on the CRG end can be responsible for dealing with such subscriptions. Since our design is flexible, we allow users to configure whether this extension of CRG with the subscription mode should be on or off. Like the policy for downsizing the certificate DB, users can specify a policy for narrowing down a set of certificates which they can subscribe for.

4. Performance and Complexity Analysis

In this section, we provide a comparative analysis of overheads incurred by existing methods (including CRL and OCSP) and our proposed method, *i.e.,* CRG. Specifically, we analyse network latency as well as bandwidth and storage overheads. Further, we analyse the communication complexity of the network latency. Our results show that CRG outperforms both CRL and OCSP methods.

4.1. Experimental Environment

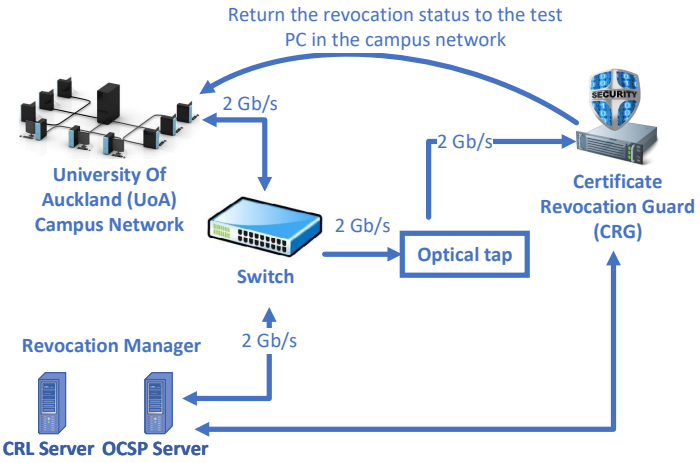

Figure 4: Measuring bandwidth consumption and latency between the existing revocation checking mechanisms and CRG. CRG on the mirror host can monitor all incoming and outgoing traffic from the UoA campus network. Here, CRG can return the revocation status to the test PCs in the UoA campus network.

Figure 4 illustrates the experimental environment in detail. Basically, we run CRG out of our main network (see Figure 4). CRG receives traffic from optical taps and operates only on the duplicated traffic. This design allows us to collect the performance results from the existing revocation process as well as the results from CRG. For instance, to calculate the latency improvement, we used the test PC from the UoA campus network to query a certificate revocation status from a target OCSP server. We start the timer when the request is sent, and we stop the timer once we receive the reply. The latency is the time the user takes to receive the revocation response. Meanwhile, if the CRG has a record of the requested certificate in the local cache, it returns its response immediately to the test PC. As a result, the test PC will receive two certificate revocation status responses – one from the OCSP server and another one from CRG. Next, we analyse the arrival time of the two responses and determine which scheme takes the least time. Such strategies are used to compare the bandwidth consumption between the existing revocation mechanisms and CRG.

4.2. Data Collection

For our experiments, we have collected live Internet traffic (*i.e.,* HTTPS) at the gateway of The University of Auckland, New Zealand. To collect the data, we wrote a Python script that

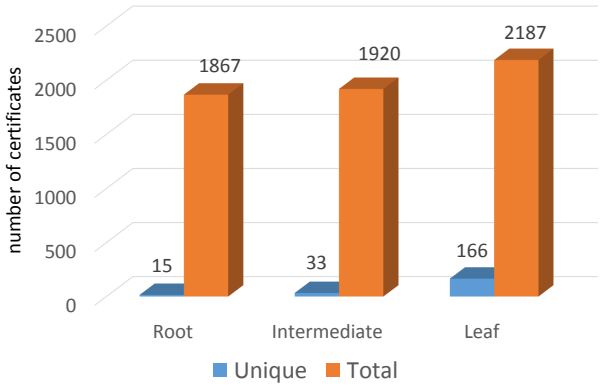

Figure 5: A summary of data we used for our experiments: we distinguish three types of certificates, namely root, intermediate and leaf. For each type of certificate, we show how many unique as well as total requests we observed.

uses the PFRing[4] capture mechanism to minimise the packet drop rate in a 10 Gb/s high-speed network. The script captured and recorded the first three messages (*i.e.,* Client Hello, Server Hello and Certificate) of SSL/TLS flows. After the data collection, we launched our benchmarking experiments. We used a Windows 7 machine having a Core i5 2.2 GHz processor with 8 GB memory to launch three different test scenarios. In OCSP test cases, we used the Openssl[5] library to send OCSP queries for each certificate. We captured and analysed the operation overhead through Wireshark reports. On the other hand, we tested the CRL approach in different browsers to analyse the same number of certificates, the results demonstrated the bandwidth overhead and the local storage requirement for downloading all the CRL files. We then used our CRG approach to process the same trace file on the same machine.

To extract certificate messages from the trace file, we used TShark[6]. Each certificate message in the SSL/TLS protocol represents a chain of certificates involving root, intermediate and leaf certificates.

Our results show that there are 2200 SSL/TLS certificate messages in the dataset we had recorded, as described above. Figure 5 illustrates breakdowns of the root, intermediate and leaf certificates in all the requests. In total, there are 1867 root, 1920 intermediate and 2187 leaf certificates. In our dataset, we discovered that the unique root, intermediate and leaf certificates are 15, 33 and 166, respectively.

In our experiments, we do not consider checking revocation status of root certificates, assuming they are taken care by OS or browsers. Without loss of generality, root certificates can also be handled by our approach.

[4] http://www.ntop.org/products/packet-capture/pf_ring/

[5] https://www.openssl.org/

[6] https://www.wireshark.org/docs/man-pages/tshark.html

4.3. Ethical Considerations

Our research involved collecting network traffic, of which some would have been initiated by persons using computers on the UoA network or accessing UoA network. The only information collected by our research was the first three messages of the SSL exchange between client and server. The ethics approval must be sought from the UoA Human Participants Ethics Committee (UAHPEC) for projects involving human participants. "A human participant is a person with whom there is some intervention or interaction that would not be occurring, or would be occurring in some other fashion, but for the research, or as a result of the research." As the research collection was passive, there was no interaction with human participants. The Oxford English Dictionary (OED) defines intervention as "the action of intervening, 'stepping in', or interfering in any affair, so as to affect its course or issue." As our research collection did not change or affect any data, our research collection was not an intervention, as per aforementioned definition. As the research collection was not an interaction or an intervention with a human participant, it did not require approval by UAHPEC. On the privacy issue the Privacy Act 1993 defines personal information as "information about an identifiable person". The research collection of network traffic only collected metadata concerning the SSL protocol and the associated certificates. It did not collect data about the resources (*e.g.,* web sites) being accessed nor did it attempt to decrypt the content of SSL traffic. Therefore, the research information collected does not meet the definition of personal information. Even if the information collected was classed as personal information Principle 2 of Section 6 of the Privacy Act provides researchers with the ability to collect personal information from sources other than directly from the individual where it "is not reasonably practicable in the circumstances" or "that the information will not be used in a form in which the individual concerned is identified; or will be used for statistical or research purposes and will not be published in a form that could reasonably be expected to identify the individual concerned." As it is not practicable to get this information from individuals and is being used for research purposes and will not be published in a form that could possibly identify any individuals then the collection from the network does not breach the collection/consent principle of the Privacy Act 1993. Finally, the collection was performed under the authority of UoA staff.

4.4. Bandwidth Cost

Bandwidth refers to the data throughput capacity for each certificate's revocation checking. In case of CRG and CRL, after checking certificate revocation, the client caches the revocation status locally for a certain time. This caching saves bandwidth cost because the client can locally check the revocation status of certificates that are repeated. However, OCSP requests the OCSP server to check the revocation status of each certificate even if the certificate is repeated. For measuring bandwidth overhead, we use the dataset described in Section 4.2 and calculate the accumulative bandwidth cost of all the requests. The comparison of bandwidth has been shown in Figure 6. Our

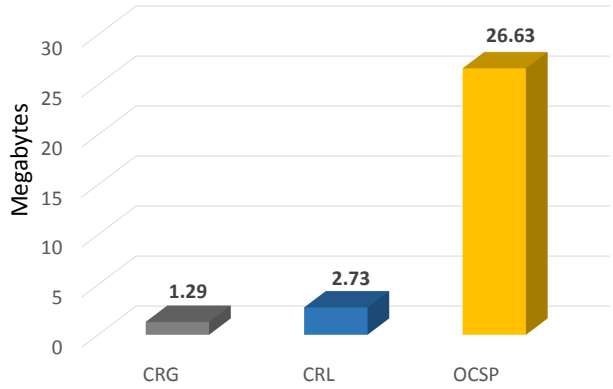

Figure 6: Comparison of bandwidth overheads of two existing methods with CRG. CRG saves over 95% bandwidth cost when compared with OCSP.

dataset shows that OCSP requires 850 bytes for checking the revocation status. Since CRG is built on OCSP, it also requires 850 bytes for checking the revocation status and the same caching cost, which is explained in Section 4.6. CRL uses more bandwidth for fetching the CRL file, the median value of bandwidth usage is 1800 bytes, which is more than double than that of CRG. However, the size of a CRL response is dependent on the number of certificates to be revoked. Therefore, a CRL response can be large, while an OCSP response is always the same size. If we compare CRG with OCSP, we save over 95% bandwidth cost.

4.5. Network Latency

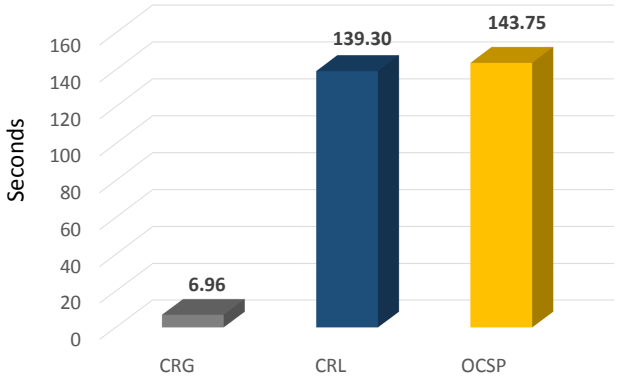

Figure 7: Comparison of network latency of two existing methods with CRG. As comapred to OCSP, CRG improves latency by 95%.

Network latency is the time between sending the revocation status request and receiving the response. In our experiments, we calculated the round trip time between each certificate request and response. From the experiment results, we observed a network latency from 20 milliseconds (ms) to 2 seconds. The average latency for OCSP and CRG is 30 ms. It may take longer

for CRL depending on the file size. The median value for latency in case of CRL is 700 ms. Similar to bandwidth cost, we use the dataset described in Section 4.2 and calculate the accumulative network latency of all the requests. In our experiment, CRG is placed close to the gateways of our campus's access network. This approach provides two benefits: it captures client's connections to any TLS servers, and the round trip time is minimized by reducing the communication path.

We present results of network latency in Figure 7. Similarly to the bandwidth comparison, CRG incurs a low latency overhead because it processes 95% of the requests locally. In contrast, OCSP incurs the most latency overhead, as it interacts with the OCSP server for each request.

4.6. Storage Cost

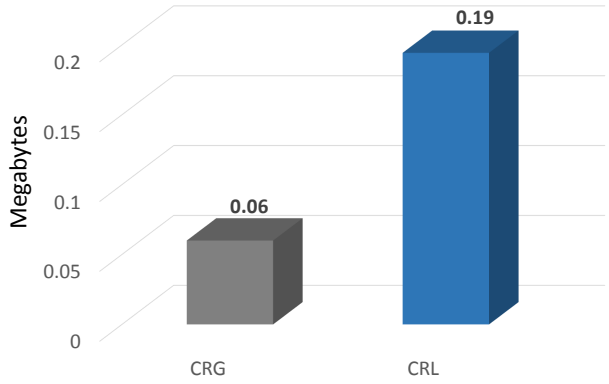

Figure 8: Comparison of local desk storage between CRG and CRL. CRG saves storage space by 69%. OCSP is not included because it does not cache the certificate revocation status locally.

Since we cache the status of revoked certificates, that requires some storage. In a CRL file, there is a single entry for each revoked certificate. The file size determines the total number of entries. This is similar to the certificate DB in CRG. On average, each certificate entry (already explained in Section 3) occupies 300 bytes in CRG. In contrast, most of the CRLs are of 1000 bytes. The difference is caused by the inconsistent CRL file size. Typically, the file includes all the revoked certificates from the same CAs. Gibson Research Corporation generated a report in 2013 [20], they observed the smallest CRL file is 236 bytes and the largest is 28,198,758 bytes. Two years later, Liu *et al.* observed that the CRL file size can be up to 51 kB [2]. Conversely, each entry in CRG only contains revocation status of the requested certificates. Figure 8 demonstrates the storage cost for saving revocation status of approximately 200 certificates using CRG and CRL. OCSP is not considered because it does not cache the revocation status. As compared to CRL, CRG saves storage cost by 69%.

4.7. Communication Complexity

We consider the case where a client sends a certificate revocation request to an OCSP server. The latency (L) for packet transmission can be expressed as:

$$L = D_{(src,dest)}/v + L/b + H * T_{router} + T_{contention}$$

where $D_{(src,dest)}$ is the distance from the source node to the destination node, v is a transmission medium speed. For instance, the multimode fibre optic cable speed and the transmitting distance limits are 100 Mb/s for distance up to 2 km (100BASE-FX), Gb/s up to 1000m, and 10 G/s up to 550 m. L is packet size, b is channel bandwidth, H is hop count, T_{router} is router delay, and $T_{contention}$ is the delay due to network contention. Based on this equation, we can see that many factors can affect the network delay; for example, the distance between source and destination, the longer distance takes more time to process the packet. Moreover, the number of routers between the source and destination also has an impact on latency because each router requires time to process the packets that pass through it. Besides, the signal speed in transmission media also affects network latency, such as a fibre optic cable has a lower latency than a copper cable via a long-distance connection. Furthermore, we need to consider the time it takes the client to establish a connection with the OCSP server. To estimate the communication overhead, we installed CRG on the gateway of the campus network. The distance between the test PC to the gateway is less than 2KM; the number of hops is 1. Based on our trace file, we find the connection between the test PC and OCSP server is very fast, so we removed $T_{contention}$ when we calculate the network delay. The advantage of this is that we can reduce the delay caused by the longer distance, or the delay caused by having to go through multiple routers, and the transmission speed decreases due to the congestion on the transmission path. In this study, we did not consider the energy consumption when using a smartphone to communicate CRG versus the power cost via an OCSP server. However, we found some works [21, 22] on this aspect to help us in the future for measuring the power consumption of CRG and how to optimise CRG from the energy efficiency perspective.

5. Privacy and Security Analysis

CRG is a novel solution that caches the certificate revocation status. CRG aims at ensuring that Internet users do not use certificates that are no longer trusted. Therefore, CRG is likely to be targeted by attackers, such as hackers can compromise data integrity, data confidentiality, and CRG availability. In this section, we will discuss the possible security and privacy issues that CRG may face as well as providing the relevant solutions to address each security or privacy risk.

Minimising Privacy Loss. Privacy is an issue when we use OCSP because a user's browsing behaviour can be deduced by analysing OCSP requests. In CRG, we do not interact with the revocation manager for every request. After making the first request to the revocation manager, the OCSP response is cached for a certain period in which it remains valid, *i.e.,* the next update time. The next request to the revocation manager is sent

when the next update time is reached or a new certificate is observed. This design significantly improves the user behaviour visible to the revocation manager.

Database Security. In our current solution, a certificate revocation status is temporarily stored in the local database, but with the increasing use of certificate revocation status, the local storage method will bring many security risks, such as data integrity and Single Point of Failure (SPOF). To address those concerns, we reviewed the existing studies [23, 24] and found the blockchain-based solutions. Blockchain emerges as a robust solution and provides compelling properties about data integrity as well as eliminating the SPOF issue. For example, each certificate revocation status will be saved into a block, when a block is part of the chain. All participants in this blockchain network have agreed on its content. Hence, all saved certificate revocation statuses are practically non-repudiable, transparent, and persistent. As a result, it is infeasible for anyone to modify or delete the existing certificate revocation record from the blockchain network. There is no central DB for managing the data storage and verifying the revocation status. Therefore, the SPOF issues can be eliminated by using this decentralised infrastructure.

Denial of Service (DoS). DoS attacks can be mounted to overload the certificate DB or by sending fake requests in an attempt to exhaust the processing resources. For the former case, users can set up a size limitation for the certificate DB. When the certificate DB size limit is reached, CRG will stop caching OCSP responses, however the revocation checking will still work. In this way, we can minimise the risk of DoS and other attacks (such as the memory overloading attack, which could easily be mounted when the certificate DB size is small and an attacker is able to send a large number of requests). In the latter case, if a user sends a large number of revocation checking requests for different certificates during a short period of time, CRG will add the user's IP address into a blacklisting. CRG could be configured to ignore traffic from that IP or reject that traffic.

Communication Intercepting and Hijacking Attacks. As illustrated in Figure 3, we introduced a certificate DB in CRG to query and update the certificate revocation status. If attackers are able to monitor the communication with this certificate DB, they can launch a man-in-the-middle attack by intercepting messages and modifying the revocation status in Step 3a and Step 5b, respectively. In order to ensure the confidentiality and integrity of each request, we use authenticated encryption (using symmetric keys) to build a secure channel between CRG and the external DB.

If attackers manage to have malicious access to CRG, in particular to the communication channel between the certificate modifier and the client, they can hijack and modify the certificate response in Step 6 of Figure 3. Here, we can distinguish two scenarios in the context of modifying the response in Step 6. First, an attacker can modify a fake certificate with a legitimate one. This scenario brings down the security to that of the traditional infrastructure, where no revocation check is performed. In the second scenario, we assume that an attacker modifies a legitimate certificate with a fake one. We can overcome the issue in such a scenario by setting up an entity that should be responsible for keeping track of the count of fake certificates sent to the client. A System Administrator could be alerted if the count of fake certificates is more than a certain limit. Devising a sophisticated mechanism for dealing with such a scenario is part of our future work.

6. Discussion

Our comparative analysis provides useful insights into the performance improvements of using CRG. We highlight a positive linear relationship between the CRG performance cost and the number of requests sent to OCSP servers. Firefox proposes an OCSP deployment scenario [25] that uses a shared cache of OCSP responses for accessing the same websites. However, this type of solution can only be used for a particular browser. Our approach is generic enough to cache OCSP responses at the system, router, gateway, proxy, or ISP level. We next discuss possible design choices to deploy CRG and highlight some performance achievements.

6.1. CRG Deployment Scenarios

From the deployment point of view, it is possible to install CRG on the user's devices or to consider an on-path deployment strategy. CRG needs to receive each SSL/TLS flow between clients and servers for managing the certificate revocation status. In the following, we discuss possible choices to deploy CRG.

Client Side Deployment. CRG can be installed on either powerful personal computers or mobile devices. Our findings confirm that fetching revocation status can be an expensive operation for clients, particularly those devices that have limited resources. For example, mobile devices with a limited storage space or power, and PCs with a high network latency.

There is a trade-off between limited storage usage and low network capacity. For the former case, clients have to set a limited size on CRG for configuring the certificate DB and clear it when it is nearly full. In the latter case, clients can increase the storage usage for caching more OCSP responses. From the communication point of view,all the processes will be handled locally by CRG on the device. Generally, there is no communication cost for checking the cached certificates.

Router Deployment. Caching OCSP responses on the router closer to the target networks reduces the latency because the latency depends largely on how far the client is away from the OCSP servers. The drawback of doing this is the storage size because existing routers use most of their memory to maintain routing information, so they may not have sufficient space available for a CRG database. If we want to deploy CRG on the existing routers, we have to consider the certificate DB size. However, placing CRG on the router brings many benefits including improving global availability, reducing bandwidth and no additional upgrading requirements on the client side.

Gateway/Proxy Deployment. CRG can be located at the gateway or proxy level. It would help to improve user experience for fetching the certificate revocation status with a lower bandwidth cost and a lower latency. From the management point of view, this deployment option reduces an operation cost for making any changes in CRG compared to the aforementioned ones. Network administrators only need to modify CRG at the gateway/proxy instead of changing all the routers or all the client devices in a target network.

ISP Level Deployment. There is no additional requirement for placing CRG at the ISP level. The only concern is the storage size because the number of cached certificates can become very high by monitoring all certificates from the subscribing networks. Clients have to determine the most effective size of their certificate DB. The database size is the main requirement when deploying CRG in different locations. We suggest clients configure storage size as high as possible in any location to fully exploit potential benefits of CRG.

6.2. Increased Performance Gain

Due to our disk storage limitation, we collected live traffic only for half an hour. Based on our sample file, we noticed 95% of requests were using cached certificates. That is why, CRG is more efficient than the other two schemes (CRL and OCSP), and the results showed that CRG improved the bandwidth and latency by 95% when performing the certificate revocation check. Besides, we found 65% local desk storage improvement when comparing CRG and CRL. Therefore, if we extend the measurement time to a week (which is a typical time when the caching status expires) or more, we believe more cached certificates will be observed, thus enabling increased performance gain. Consequently, it will generate better performance results.

7. Related Work and Comparative Analysis

In last two decades, there has been much work in the area of certificate revocation for improving the existing revocation solutions [7, 8, 13, 33] and measuring certificate revocation behaviour [1, 2, 16]. In the following, we will point out the differences between our approach and the existing solutions.

7.1. Improving CRL and OCSP Methods

In [34], Myers investigated options that can be used to address some revocation issues. Some of Myers' suggestions have already been taken into account by existing techniques, such as OCSP. The study by Topalovic *et al.* [7] also builds on top of Myers' suggestion of implementing short-lived certificates. Furthermore, Myers suggested fetching the results from the online relying party. Alternatively, the relying party could ascertain a certificate's reliability using cached data. There are two drawbacks of using Myers' approach: the cost of deploying the short-lived certificate and how can we trust a certificate status from an online relying party. In contrast, We proposed CRG that validates a certificate revocation status from a certificate issuer and utilise the existing certificate revocation processing to reduce deployment costs. The similar deployment issues have been observed from [35, 32], Schulman *et al.* [35] presented RevCast in 2014; their approach transmits the revocation status over existing FM radio rather than the traditional Internet links. As a result, end users will receive the revocation status through an embedded FM RDS receiver, or via a proximal FM RDS receiver. Two years later, Chariton *et al.* [32] suggested that CA pushes a revoked certificate information to the DNS system. Therefore, a web browser can query the DNS system to find revocation information about the certificate. However, Both solutions require changes to our existing revocation infrastructure.

A study by Millen and Wright [36] is closer to our work. They addressed the trade-off between the time of the certificate is revoked and the time of the update. They proposed a depender list that includes a set of subscribing parties. Those subscribing parties receive notifications immediately after a certificate is revoked, rather than delaying a periodic schedule. Like their work, by extending the basic CRG, we introduce subscribing clients and publishing servers. However, the client only subscribes to each certificate she requests (and caches), instead of a CRL file. Another similar approach has been made by Szalachowski *et al.* [31] in 2016. They introduced a new framework that use middle-boxes to store the revocation information. Besides, they take advantage of content-delivery networks to receive the new revocations from the subscribing CA servers. This solution, however, is still insecure and inefficient, it deals with lack of trust in the dissemination system, as well as the deployment costs. Unlike their solution, CRG uses the existing revocation process, it queries the trusted CA and extracts the revocation status from each response. Furthermore, CRG does not require any additional cost for updating the existing infrastructure.

In [33], Russell *et al.* pointed out the performance problem on mobile digital units for checking certificate revocation. They proposed a new hash-based authentication solution for checking a fraudulent revocation status. We cannot compare their solution directly with our work because our study focuses on the revocation efficiency instead of checking for fraudulent revocation responses.

In [13], Gutmann studied some workarounds for reducing the CRL bandwidth overhead, such as by using different expiry times, assigning long-term or short-term CRLs. However, these solutions do not solve storage limitations or revocation lookup complexity.

In [8], Kocher proposed a Certificate Revocation Tree (CRT) solution. The main idea is to find an efficient and more scalable method for distributing revocation information. Kocher designed a tree issuer for compiling revocation information, a confirmation issuer to publish the elements from CRT and a receiver to fetch the certificates. Compared to our work, CRT introduces a high computational load because it has to compute hashes for checking the certificate status.

In [7], Topalovic *et al.* discovered some certificate security breaches of certificate authorities imposed by OCSP. Consequently, some browsers, such as Google Chrome, permanently disable OCSP and take direct ownership over certificate revoca-

Table 2: Comparison of CRG with existing solutions based on the following characteristics: latency (*i.e.,* the time taken to check revocation status); bandwidth (*i.e.,* the communication overhead to check revocation status); storage (*i.e.,* the amount of storage required to store revocation information); privacy (*i.e.,* hiding user behaviour from third parties); deployable (*i.e.,* ideally requiring no changes to existing infrastructure); generic (*i.e.,* whether the solution is generic for all the clients or specific to a particular client *i.e.,* browser in most cases).

Revocation Solutions	**Low Latency**	**Low Bandwidth**	**Low Storage**	**Privacy**	**Deployable**	**Generic**
Short-lived certificates [7]	No	No	No	Yes	No	Yes
Certificate revocation tree [8]	Yes	Yes	Yes	No	No	Yes
Short-term CRLs [13]	No	No	No	Yes	No	Yes
CRL [17]	Yes	No	No	Yes	Yes	Yes
OCSP [18]	No	No	Yes	No	Yes	Yes
OCSP Stapling [19]	No	No	Yes	Yes	No	Yes
CCSP [26]	Yes	Yes	Yes	Yes	No	Yes
CRLite [27]	Yes	Yes	Yes	Yes	No	Yes
Revocation transparency [28]	No	No	Yes	No	No	Yes
CRLSet [29]	No	No	No	Yes	No	No
OneCRL [30]	No	No	Yes	Yes	No	No
RITM [31]	Yes	Yes	Yes	Yes	No	No
DCSP [32]	Yes	No	Yes	Yes	No	Yes
CRG (Our proposed solution)	Yes	Yes	Yes	Yes	Yes	Yes

tion. To reduce this risk, they developed a new prototype for automatically releasing short-lived certificates. For the client side, they developed a Chrome plug-in for receiving update notices from the subscribing servers. Their study addressed potential security issues in certificate validation and revocation while our solution aims at improving the revocation performance.

Some recent studies suggest modifications in existing systems. In [28] has explored extending certificate transparency [37] approach, where Revocation Transparency (RT) uses public logs to ensure that users and servers can obtain the recent revocation status from the log. However, the solution does not scale up well regarding performance, because the client still retrieves a revocation status for each observed certificate.

Additionally, Google and Mozilla proposed their own revocation mechanisms: namely CRLSet [29] and OneCRL [30], respectively. Both solutions require the clients to pull the revocation information from the pre-defined the servers. The drawback of these solutions is performance. That is, the clients have a difficulty to handle revocation information of a large number of certificates when there are limited processing, bandwidth, and storage capabilities.

Another study [26], Chariton *et al.* introduced a new approach that aggregates several certificates' revocation information into a single OCSP response. This OCSP response is designed as a bitmap, where each single bit represents revocation status of a certificate. If the certificate is revoked, the bit will be set as "1" and "0" otherwise. Their results show that the new approach reduces space requirements and the number of signing operations performed by the OCSP server. Larisch *et al.* [27] design a system (CRLite) to push all valid certificate revocations to browsers. CRLite contains two components: a server-side and a client-side; the former component aggregates all valid certificate revocation information, while the latter one will fetch the information to check the observed certificate's revocation status. CRLite shows a significant improvement regarding reducing latency and bandwidth and ensuring privacy. However, deploying these approaches requires changes in the existing infrastructure, which is the main drawback. In contrast, we can deploy CRG into the existing infrastructure without any modification.

There are also other studies that have been conducted [9, 10, 11, 12]. However, solutions proposed in those studies are not widely deployed for various reasons. For instance, most solutions either change or replace the existing CRL and OCSP methods, while our method does not require any changes in the legacy infrastructure. Overall, CRG adds another layer on top of existing methods for improving the revocation checking. Table 2 compares CRG with existing revocation solutions regarding efficiency (including latency, bandwidth, and storage), privacy, deployability, and applicability (*i.e.,* generic or specific).

7.2. Measuring Certificate Revocation

Most recently, Zhu *et al.* [1] measured the latency of OCSP queries. Their results show that the average latency for each OCSP lookup is 20 ms, but we observed it closer to 35 ms. The difference could be caused by the location of the OCSP server. In other words, the Round-Trip Time (RTT) plays a significant role in calculating the latency. Liu *et al.* [2] observed revocation checking behaviour by analysing similarities and differences between web browsers and OS as well as Google's certificate revocation infrastructure. Zhang *et al.* [16] studied how many certificates are reissued or revoked after detection of the Heartbleed OpenSSL bug.

Note that some measurement studies are not directly linked to our work, but they exposed the existing revocation issues and explained how this impacts the existing revocation methods. These studies provide a complete view of certificate revocation in today's PKI. In short, all the certificate revocation methods suffer from at least one of the following issues: privacy invasive, high latency, high bandwidth and storage overheads.

7.3. Browser's "Fail Soft" Policy

Some articles [38, 20] pointed out that many browsers set the default security settings not checking the revocation status.

Samoshkin *et al.* called it a "fail soft" policy, meaning that browsers treat no reply to an OCSP request as a good reply. Recent studies also detected similar issues. Liu *et al.* [2] observed that mobile browsers disabled certificate revocation checking, because of the latency and power issues. To address this issue, in this article we propose CRG: a lightweight solution to improve certificate revocation performance without changing the existing infrastructure.

8. Conclusions and Future Work

Certificate revocation checking is an essential part of certificate validation. There are existing methods to check certificate revocation; however, they suffer from high bandwidth overhead, network latency or storage overhead. As an alternative solution, this article has presented CRG to reduce bandwidth overhead, network latency and storage overhead when compared with existing methods. Furthermore, CRG ensures users' privacy by minimising interaction with OCSP servers.

For future work, we suggest further investigating the CRG extension to proactively handle certificates that are revoked before the next revocation update is due. We also plan to devise a sophisticated mechanism for dealing with scenarios when CRG gets compromised.

References

[1] L. Zhu, J. Amann, J. S. Heidemann, Measuring the latency and pervasiveness of TLS certificate revocation, in: Passive and Active Measurement - 17th International Conference, PAM 2016, Heraklion, Greece, March 31 - April 1, 2016. Proceedings, 2016, pp. 16–29.

[2] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, 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, ACM, New York, NY, USA, 2015, pp. 183–196.

[3] Q. Hu, M. R. Asghar, N. Brownlee, Certificate revocation guard (CRG): an efficient mechanism for checking certificate revocation, in: Local Computer Networks (LCN), 2016 IEEE 41st Conference on, IEEE, 2016, pp. 527–530.

[4] H. Adkins, An update on attempted man-in-the-middle attacks, Google Online Security Blog.

[5] W. Andrew, Distrusting WoSign and StartCom certificates, Google Online Security Blog.

[6] P. Mutton, Certificate revocation: Why browsers remain affected by Heartbleed (2014).
URL https://bit.ly/2S8yGTv

[7] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, D. Boneh, Towards short-lived certificates, Web 2.0 Security and Privacy.

[8] P. C. Kocher, On certificate revocation and validation, in: Financial Cryptography, Springer, 1998, pp. 172–177.

[9] E. N. Ed, W. Aiello, S. Lodha, R. Ostrovsky, Fast digital identity revocation, in: in 18th Annual International Cryptology Conference (CRYPTO98), Springer-Verlag, 1998, pp. 137–152.

[10] C. A. Gunter, T. Jim, Generalized certificate revocation, in: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM, 2000, pp. 316–329.

[11] A. Levi, Ç. K. Koç, Reducing certificate revocation cost using NPKI, IEEE Journal on Selected Areas in Communications 18 (2000) 561–570.

[12] P. Szalachowski, L. Chuat, A. Perrig, PKI Safety Net (PKISN): Addressing the too-big-to-be-revoked problem of the TLS ecosystem, arXiv preprint arXiv:1601.03874.

[13] P. Gutmann, PKI: it's not dead, just resting, Computer 35 (8) (2002) 41–49.

[14] Y.-S. Jeong, S.-S. Shin, An efficient authentication scheme to protect user privacy in seamless big data services, Wireless Personal Communications 86 (1) (2015) 7–19.

[15] M. Conti, L. V. Mancini, R. Spolaor, N. V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Transactions on Information Forensics and Security 11 (1) (2016) 114–125.

[16] L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, C. Wilson, Analysis of SSL certificate reissues and revocations in the wake of Heartbleed, in: Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, ACM, New York, NY, USA, 2014, pp. 489–502.

[17] R. Housley, W. Polk, W. Ford, D. Solo, Internet X. 509 Public Key Infrastructure certificate and Certificate Revocation List (CRL) profile, IETF RFC 3280.

[18] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, IETF RFC 6961.

[19] Y. Pettersen, The Transport Layer Security (TLS) multiple certificate status request extension, IETF RFC 6961.

[20] Gibson Research Corporation, Security certificate revocation awareness the case for OCSP Must-Staple (2013).
URL https://www.grc.com/revocation/ocsp-must-staple.htm

[21] R. McGeer, P. Mahadevan, S. Banerjee, On the complexity of power minimization schemes in data center networks, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, IEEE, 2010, pp. 1–5.

[22] L. Corral, A. B. Georgiev, A. Sillitti, G. Succi, A method for characterizing energy consumption in android smartphones, in: 2013 2nd international workshop on green and sustainable software (GREENS), IEEE, 2013, pp. 38–45.

[23] Y. Zhang, R. Deng, X. Liu, D. Zheng, Outsourcing service fair payment based on blockchain and its applications in cloud computing, IEEE Transactions on Services Computing.

[24] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, V. Sassone, Blockchain-based database to ensure data integrity in cloud computing environments.

[25] Mozilla, CA:OCSP-TrustedResponder (2010).
URL https://wiki.mozilla.org/CA:OCSP-TrustedResponder

[26] A. A. Chariton, E. Degkleri, P. Papadopoulos, P. Ilia, E. P. Markatos, CCSP: A compressed certificate status protocol, in: INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, IEEE, 2017, pp. 1–9.

[27] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, C. Wilson, CRLite: A scalable system for pushing all TLS revocations to all browsers, in: Security and Privacy (SP), 2017 IEEE Symposium on, IEEE, 2017, pp. 539–556.

[28] B. Laurie, E. Kasper, Revocation transparency, Google Research.

[29] A. Langley, Revocation checking and Chrome's CRL, ImperialViolet (blog).

[30] M. Goodwin, Mozilla security blog: Revoking intermediate certificates: Introducing OneCRL (2015).

[31] P. Szalachowski, L. Chuat, T. Lee, A. Perrig, RITM: Revocation in the Middle, in: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Vol. 00, 2016, pp. 189–200. doi:10.1109/ICDCS.2016.91.

[32] A. A. Chariton, E. Degkleri, P. Papadopoulos, P. Ilia, E. P. Markatos, DCSP: Performant certificate revocation a DNS-based approach, in: Proceedings of the 9th European Workshop on System Security, EuroSec '16, ACM, New York, NY, USA, 2016, pp. 1:1–1:6. doi:10.1145/2905760.2905767.

[33] S. Russell, Fast checking of individual certificate revocation on small systems, in: Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual, 1999, pp. 249–255.

[34] M. Myers, Revocation: Options and challenges, in: Financial Cryptography, Springer, 1998, pp. 165–171.

[35] A. Schulman, D. Levin, N. Spring, Revcast: Fast, private certificate revocation over FM radio, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, ACM, New York, NY, USA, 2014, pp. 799–810. doi:10.1145/2660267.

2660376.
[36] J. Millen, R. Wright, Certificate revocation the responsible way, in: Computer Security, Dependability and Assurance: From Needs to Solutions, 1998. Proceedings, 1998, pp. 196–203.
[37] B. Laurie, A. Langley, E. Kasper, Certificate transparency, IETF RFC 6962.
[38] A. Samoshkin, SSL certificate revocation and how it is broken in practice (2018).
URL https://bit.ly/2DFTXvs