

Certificate Revocation Guard (CRG): An Efficient Mechanism for Checking Certificate Revocation

Qinwen Hu

Department of Computer Science
The University of Auckland
Auckland 1142
New Zealand

Email: qhu009@aucklanduni.ac.nz

Muhammad Rizwan Asghar

Department of Computer Science
The University of Auckland
Auckland 1142
New Zealand

Email: r.asghar@auckland.ac.nz

Nevil Brownlee

Department of Computer Science
The University of Auckland
Auckland 1142
New Zealand

Email: n.brownlee@auckland.ac.nz

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 Communicating parties exchange and validate these certificates, the validation fails if a certificate has been revoked. In this paper we propose the Certificate Revocation Guard (CRG) to efficiently check certificate revocation while minimising bandwidth, latency and storage overheads. CRG is based on OCSP, which caches the status of certificates locally. CRG could be installed on the user's machine, at the organisational proxy or even at the ISP level. 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 certificate revocation.

I. 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 for providing encryption for end-to-end communication over the Internet. There is 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. If not revoked properly, a leaf certificate can 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 domains. Therefore, it is essential to check certificate revocation during the certificate validation process.

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 a network round-trip.

Most recent studies [1], [2] have shown that many web browsers do not use the recommended methods for checking certificate revocation. The main reason behind that could be bandwidth overhead, network latency, storage overhead 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 on the effectiveness of certificate revocation and associated privacy concerns. In [3], Topalovic *et al.* suggested to have short-lived certificates. However, this design results in high bandwidth overheads due to the hassle of fetching new certificates valid only for a short span of time. Kocher [4] 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 [5]–[9] require changes in the legacy infrastructure.

In this paper 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. 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 at the ISP level.

II. PROPOSED SOLUTION

Certificate revocation is not widely used today because the existing methods fail to meet acceptable privacy and performance requirements. In this paper we propose CRG, that aims at addressing problems associated with CRL and OCSP methods. As we can see in Figure 1, certificate issuance (Step I), CRL publication (Step II) and SSL/TLS flow (Steps 1 and 2) stay the same. However, CRG intercepts the Certificate message of the SSL/TLS protocol (Step 3) and checks 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 been revoked then CRG has to communicate this

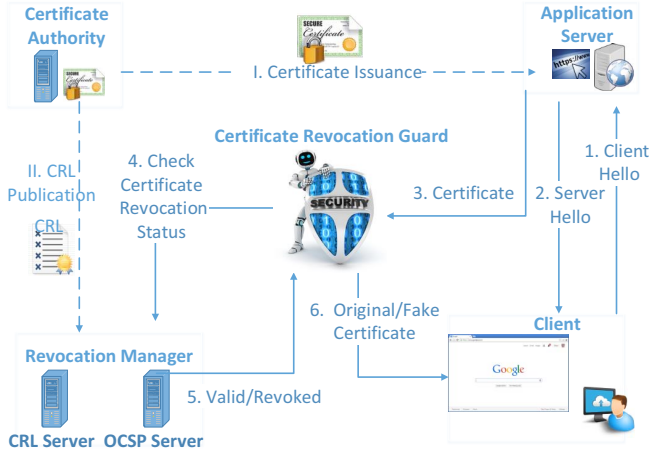


Fig. 1. Basic overview of our proposed approach in which CRG intercepts the SSL/TLS Certificate message (Step 3) and consults the certificate revocation status with the revocation manager (Step 4). CRG caches the response (Step 5) and, depending on the valid or revoked certificate, sends either the original or fake certificate (Step 6) to the client, respectively.

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 2 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. 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. This check is performed based on the certificate’s serial number, which is extracted by the certificate checker. 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. 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 resized once it approaches the configured size limit.

Certificate Validator. The goal of the certificate validator is to check the certificate revocation by contacting the revocation manager. The certificate checker queries 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.

III. PERFORMANCE 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,

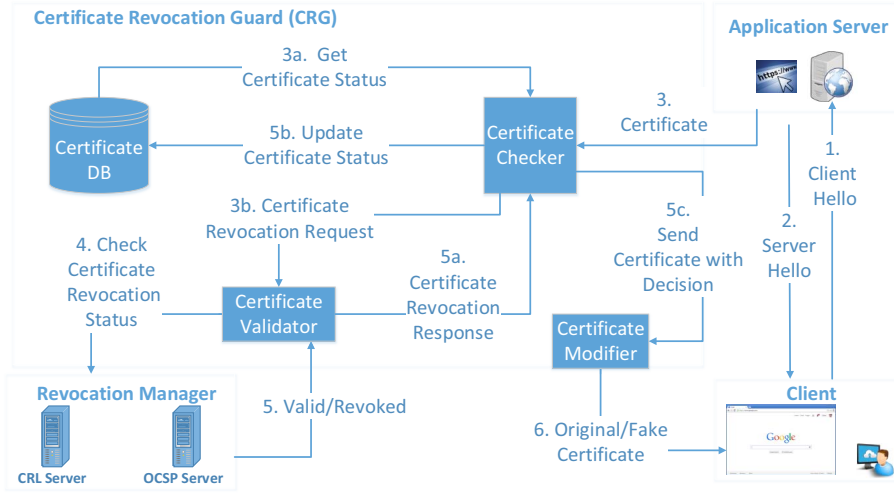


Fig. 2. A detailed view of CRG: in case of a new certificate, the Certificate Checker intercepts the SSL/TLS Certificate message (Step 3), consults the Certificate DB (Step 3a) before forwarding the Certificate Revocation Request to the Certificate Validator (Step 3b). The Certificate Validator requests 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).

we analyse network latency as well as bandwidth and storage overheads. Our results show that CRG outperforms both CRL and OCSP methods.

A. Data Collection

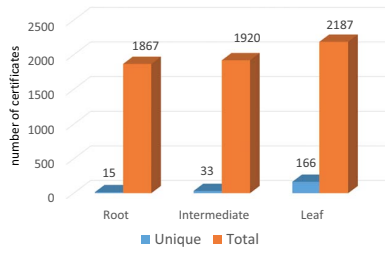


Fig. 3. A summary of data we used for our experiments: we distinguish three types of certificates, namely root, intermediate and leaf. For each type of certificate, we show how many unique as well as total requests we received.

For our experiments we have collected live Internet traffic (*i.e.*, HTTPS). To collect the data, we have written a Python script of nearly 300 lines of code. We run this script on a monitoring PC at the gateway of The University of Auckland, New Zealand. Using our Python script, we captured and recorded the first three messages (*i.e.*, Client Hello, Server Hello and Certificate) of SSL/TLS flows. The monitoring machine is running Ubuntu on an Intel 6 Xeon E5-2430 2.2 GHz with 32 GB memory. Since the gateway typically processes a huge amount of data, we were able to collect 1.2 GB data within half an hour. Due to local disk storage limitation, we could not store more data. After the data collection, for running our benchmarking experiments, we used a Windows 7 machine having a Core i5 2.2 GHz processor with 8 GB memory. To extract certificate messages from the trace file, we used TShark¹. Each certificate message in the SSL/TLS protocol

¹<https://www.wireshark.org/docs/man-pages/tshark.html>

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 used for our experiments. Figure 3 illustrates breakdowns of root, intermediate and leaf certificates in all the requests. In total, there are 1867 root, 1920 intermediate and 2187 leaf certificates. We discovered that root, intermediate and leaf certificates are repeated on average around 124, 58 and 13 times, but they only appear uniquely 15, 33 and 166 times respectively in our dataset.

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

B. Bandwidth Cost

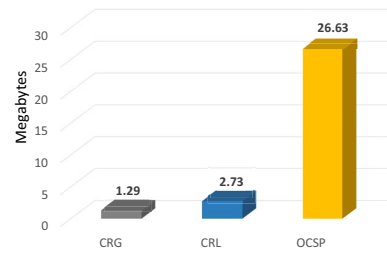


Fig. 4. Comparison of bandwidth overheads of two existing methods with CRG. CRG saves over 95% bandwidth cost when compared with OCSP.

Bandwidth refers to the data throughput capacity needed 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

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 III-A and calculate the accumulative bandwidth cost of all the requests. The comparison of bandwidth has been shown in Figure 4. Our 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 III-D. 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.

C. Network Latency

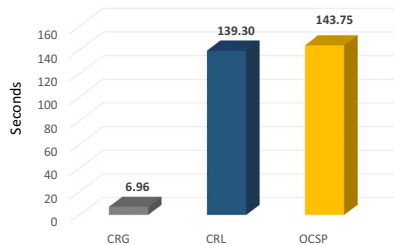


Fig. 5. Comparison of network latency of two existing methods with CRG. As compared to OCSP, CRG improves latency by 95%.

Network latency is the time between sending the revocation status request and receiving the response. In our experiments, we observed a network latency from 20 milliseconds 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 III-A and calculate the accumulative network latency of all the requests. We present results of network latency in Figure 5. Like bandwidth comparison, CRG incurs a low latency overhead because it processes 95% requests locally. In contrast, OCSP incurs the most latency overhead, because it interacts with the OCSP server for each request.

D. Storage Cost

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 II) 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. 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 6 demonstrates the storage cost for saving

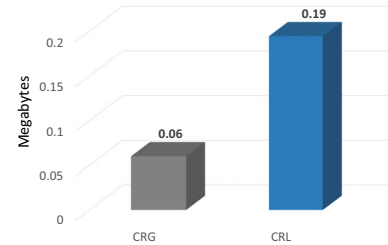


Fig. 6. Comparison of local disk storage between CRG and CRL. CRG saves storage space by 69%. OCSP is not included because it does not cache the certificate revocation status locally.

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%.

IV. 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 paper 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.

REFERENCES

- [1] L. Zhu, J. Amann, and 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. Misllove, 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*, ser. IMC '15. New York, NY, USA: ACM, 2015, pp. 183–196.
- [3] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh, "Towards short-lived certificates," *Web 2.0 Security and Privacy*, 2012.
- [4] P. C. Kocher, "On certificate revocation and validation," in *Financial Cryptography*. Springer, 1998, pp. 172–177.
- [5] E. N. Ed, W. Aiello, S. Lodha, and R. Ostrovsky, "Fast digital identity revocation," in *18th Annual International Cryptology Conference (CRYPTO98)*. Springer-Verlag, 1998, pp. 137–152.
- [6] C. A. Gunter and T. Jim, "Generalized certificate revocation," in *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 2000, pp. 316–329.
- [7] A. Levi and Ç. K. Koç, "Reducing certificate revocation cost using NPKI," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 561–570, 2000.
- [8] P. Szalachowski, L. Chuat, and A. Perrig, "PKI Safety Net (PKISN): Addressing the Too-Big-to-Be-Revoked Problem of the TLS Ecosystem," *arXiv preprint arXiv:1601.03874*, 2016.
- [9] P. Gutmann, "PKI: it's not dead, just resting," *Computer*, vol. 35, no. 8, pp. 41–49, August 2002.