Trusted Principal-Hosted Certificate Revocation

Sufatrio[1] and Roland H.C. Yap[2]

[1] Temasek Laboratories, National University of Singapore, Singapore
tslsufa@nus.edu.sg
[2] School of Computing, National University of Singapore, Singapore
ryap@comp.nus.edu.sg

Abstract. Public Key Infrastructure is a key infrastructure for secure and trusted communication on the Internet. This paper revisits the problem of providing timely certificate revocation focusing on the needs of mobile devices. We survey existing schemes then present a new approach where the principal's server functions as the directory for *its own* revocation information. We evaluate the properties and trust requirements in this approach, and propose two new schemes, CREV-I and CREV-II, which meet the security requirements and performance goals. Evaluation of CREV shows it is more lightweight on the verifier and more scalable at the CA and the principals while providing near real-time revocation.

1 Introduction

Public Key Infrastructure (PKI) is a primary infrastructure relied on for disseminating trust as well as ensuring secure communications and transactions over the Internet. Certificate revocation, namely giving notice that a public and private key pair is no longer valid despite still being within its validity period, is a major challenge in PKI [1,2]. In X.509 based PKI [3,4], certificate revocation is conducted mainly by two standardized mechanisms, namely Certificate Revocation List (CRL) [4] and Online Certificate Status Protocol (OCSP) [5]. These two mechanisms, however, have their own shortcomings. CRL imposes a high bandwidth requirement on the verifier, and (generally) provides a low timeliness guarantee. It is common for a Certification Authority (CA) to update its CRL only *daily* (as suggested in [6]). OCSP offers a potentially real-time recency assurance, but puts a high online computational requirement on the CA.

This paper revisits the problem of providing timely and lightweight certificate revocation taking into account two important *emerging trends*. Firstly, the growth of Internet-connected mobile devices requires a revocation mechanism to be *lightweight on the verifier* due to the limited resources on the device. Secondly, the growth in volume and value of Internet transactions increases the need for revocation services with *higher* timeliness guarantees (i.e. much less than a day).

In this paper, we first survey various existing certificate revocation schemes and characterize them using our framework for classifying revocation schemes. We identify a new design choice which is not well-investigated to date, and then propose revocation schemes based on placing trust on a *principal* as a scalable distribution point for more timely revocation information. The rationale is that a

I. Wakeman et al. (Eds.): IFIPTM 2011, IFIP AICT 358, pp. 173–189, 2011.

verifier needs to place trust on the principal with respect to the ensuing communication/transactions anyway. Unlike existing schemes employing repository(ies) as revocation distribution points, having principal as a personal repository brings the following unique advantages to the parties involved:

- For each validity time interval, the CA needs to produce revocation information for a principal only *once*, and then sends/pushes the information to the principal. This significantly reduces the CA's workload as there are usually far fewer principals than verifiers. The CA's overhead thus becomes independent from the number of queries from verifiers.
- A principal stores its latest revocation information for its own verifiers to access and verify. Compared to a centralized repository approach, this offers more locality of processing, decentralization, and more balanced overheads, allowing more scalable revocation while meeting the security requirements.
- Verifiers do not need to contact the CA (or repository) to validate a certificate. Rather, they deal directly with the relevant principal. This will give more privacy protection than OCSP. Yet, the CA can still revoke a principal's certificate since all revocation information originates from the CA. When a principal cannot be contacted, other existing revocation mechanisms can still alternatively be used.

In addition, the setting also addresses the incentive problem on certificate revocation information management and dissemination [2].

We analyze our proposed schemes, CREV-I and CREV-II, to show that they meet the security and trust requirements as well as the performance goals. We then evaluate our schemes together with several well known existing schemes to show that our CREV schemes can provide near real-time timeliness (e.g. 10 minutes) while being lightweight on the verifier and more scalable than OCSP.

The rest of this paper is organized as follows. Section 2 presents a framework for analyzing revocation schemes. Section 3 describes our two CREV schemes, and analyzes their trust and security requirements. Section 4 compares the performance of CREV with several existing ones. Section 5 finally concludes.

2 A Framework for Classifying Revocation Schemes

Throughout this paper, we refer to the subject of a certificate as *principal*, the principal of an Extended Validation Certificate (EVC) as *EVCP*, and the party verifying certificates as *verifier*. *Certificate Status Information (CSI)* [7] denotes the CA's released information pertinent to the validity of a certificate. CSI thus encompasses CRL data, OCSP messages, and hash tokens in CRS/NOVOMODO [8,9]. *Certificate Management Ancillary Entity (CMAE)* denotes the designated repository from which a verifier may obtain the CSI. The *revocation latency* is the bound on the time between a CA making a revocation record and when that information becomes available to the verifiers. A low revocation latency means a high *revocation timeliness guarantee*. In a per-certificate revocation scheme, such as CRS/NOVOMODO, OCSP and ours, we assume that the CA will not unnecessarily delay publishing the CSI for a revocation event.

We now specify the *security* and *performance* requirements of a revocation scheme. In this paper, we consider a certification system based on X.509 PKI model [3,4] where an issued certificate is valid *throughout* its issued lifetime unless otherwise revoked by means of the revocation mechanism in place. We simply refer to the CA as the issuer of a CSI. A certificate revocation scheme should provide the following security guarantees:

(Sec_1) **CSI availability:** makes a pertinent CSI available to each verifier that wishes to validate a not-yet-expired certificate issued by the CA.

(Sec_2) **Authenticity of CSI:** allows the verifier obtaining a CSI related to a certificate to verify (using strong cryptographic techniques) that the CSI originates from the CA and its integrity is preserved. The verifier must also be able to unambiguously determine the revocation status of the certificate.

(Sec_3) **Revocation timeliness:** specifies the validity period of the CSI, which must be less than a pre-determined maximum time bound.

(Sec_4) **Definition of revocation mechanism and CSI access point:** ensures that the certificate revocation mechanism(s) operated by the CA, together with the identity (including its accessible address on the network) of the access point where the CSI can be obtained, is clearly defined in the signed message portion of the certificate.

(Sec_5) **Assurance on certificate principal identity:** provides trustworthiness on the identity of the principal of a certificate (i.e. name, domain name). This is intended to protect against impersonation (e.g. phishing) attacks.

(Sec_6) **Privacy of verifiers:** ensures the privacy of verifiers validating a principal. Although the CA is trusted for managing certificates, the CA's knowledge on transactions involving a certificate can constitute a privacy leak.

In addition, the performance of the revocation scheme is critical in practice, especially on mobile devices. Thus we have the following performance goals:

($Perf_1$) **Scalability of the CA and CSI-Repository:** The bandwidth, storage and computational costs incurred on the CA and the CSI repository(ies) should scale well with the number of verifiers served.

($Perf_2$) **Performance constraints of the verifier:** Verifiers such as mobile devices impose special constraints in that they require revocation to be lightweight, i.e. low in computational, storage and bandwidth costs.

($Perf_3$) **Incentives for CSI dissemination:** There should be strong and clear economic incentives for the entities involved in the dissemination of CSI.

2.1 A Framework for Certificate Revocation Schemes

We characterize certificate revocation schemes based on the following four important design options:

(P_1) **Placement of directory:** The *directory* holding a copy of CSI issued by the CA, can be placed in either: the *CMAE* as a designated repository, the *verifier*, or the *principal*. Some schemes do not employ a directory, e.g. NOVOMODO [9] and OCSP where the CA is the Responder.

Table 1. Design choices in revocation schemes – options P_1 and P_2

Directory Placement	**CSI Coverage Scope**		
	All CA's Certs	**Subset of Certs**	**Individual Cert**
No directory (handled by CA)			NOVOMODO [9], OCSP [5], H-OCSP [10], MBS-OCSP [11]
CMAE	CRL [4], HS [12], CRT [13], 2-3 tree CRT [14]	Partitioned CRL, CSPR [15]	CRS [8]
Verifier	CPR [16], BCPR[17]		
Principal			***CREV-I, CREV-II***

(P_2) **Scope of released CSI:** A released CSI is verifiable to determine the revocation status(es) for either: *all* certificates, *a subset of* certificates, or *a single* certificate issued by the CA. CRL data gives the status of either all (as in standard CRL) or a subset of (as in Partitioned CRL or also known as CRL Distribution Points [4]) certificates. In contrast, the CSI in CRS, NOVOMODO or OCSP accounts for an individual certificate.

(P_3) **Positive or negative status representation:** The CA can state the status of certificates using either a *negative* (black-listing) approach, or *positive* approach where the status of each certificate is explicitly stated. Note the negative approach can only work when the scope of CSI is for all CA's certificates or a well-defined subset of CA's certificates.

(P_4) **Use of linked CSIs for subsequent status updates:** A CSI usually comes signed by the CA. The signed CSI may contain additional information aimed at allowing subsequent (periodic) release of more compact and unsigned CSI to update the status of certificate(s), e.g. the tip of a hash chain in CRS or the root and several interior node values of a Merkle Hash Tree in [10,11]. We characterize a revocation scheme based on *whether or not* it employs linked additional information for subsequent status updates.

We have purposely omitted several possible design options, such as periodic versus non-periodic CSI issuance, and pull versus push CSI access between the CA/CMAE and verifiers. This is because we assume standard X.509-based PKI practices as well as typical PKI-based applications. Thus, we focus on the information flow and representation of the CSI, which we believe are the main factors that characterize a revocation scheme.

Table 1 characterizes numerous existing revocation schemes based on properties P_1 and P_2. We also make the following observations. For revocation schemes with the principal as the directory, a per-certificate CSI is generally preferred as a principal is usually concerned with its own verifiers. Per-certificate CSI is also usually better for performance. Revocation schemes with CSI for all certificates can be transformed into subset-based schemes as long as the certificate space can be partitioned clearly. Transformation into per-certificate (individual) CSI is possible only if a positive CSI representation is employed. Having either the

CA or CMAE(s) as the directory is simply a matter of repository delegation. However, for high timeliness or online revocation schemes (e.g. OCSP), a CMAE must have access to the revocation statuses hosted by the CA, and must be as trusted as the CA with respect to informing the certificate status.

2.2 Survey of Existing Certificate Revocation Schemes

CRL [4] is presently the most widely supported revocation scheme. However, it is also widely known to have serious shortcomings. Firstly, the CRL may eventually grow to a cumbersome size in very large PKIs [18]. Secondly, the downloaded CRLs may be mostly useless since more than 90% of the information can be irrelevant to the verifiers [18]. Lastly, CRL (generally) does not offer adequate timely revocation guarantees. There are several improvements on the basic CRL mechanism, such as CRL Distribution Points, Delta CRLs, and Indirect CRLs [4]. However, all these schemes still put the same requirement on the verifier to obtain a complete or subset of revocation list, including the unrelated entries.

OCSP [5] was proposed to provide more timely certificate status checking. It returns the status of an inquired certificate in an online fashion. However, it imposes high computational and network requirements on OCSP Responder [19]. Furthermore, there is a privacy problem since the Responder knows all verifiers dealing with a principal. Several modifications have been proposed on OCSP. Like our scheme (CREV-I), H-OCSP [10] and MBS-OCSP [11] employ a hash-chaining technique. However, they require the CA to cater for a *potentially large number of verifiers* while CREV-I reduces it to just the EVC principals. Thus, the CA's bandwidth and storage requirements in [10,11] remain high.

Certificate Revocation Status (CRS) [8] makes revocation more efficient by periodic release of compact hash-chain information. It assumes the use of a CMAE. The CA chooses a one-way hash function $H()$, a time interval period d, and the length of a hash chain ℓ. The lifetime of the certificate is: $d\,(\ell + 1)$. The CA then includes the following into a certificate: $HashAlgID$ (defining $H()$), d, ℓ, issue and expiration times, and two numbers Y and N for "valid" and "revoked" status respectively. The CA generates two secret random numbers Y_0 and N_0, with $Y = H^{\ell}(Y_0)$ and $N = H(N_0)$. On the i-th time interval after the certificate's issuance, the CA sends to the CMAE(s): a signed timestamped string containing all serial numbers of issued and not-yet-expired certificates; and V_i for each certificate, where: $V_i{=}H^{\ell-i}(Y_0)$ if a certificate is valid, or $V_i{=}N_0$ if it is revoked. The CMAE sends V_i in reply to a query from a verifier. The verifier checks the certificate status by comparing $H(V_i)$ with N, or $H^i(V_i)$ with Y. NOVOMODO [9] extended CRS using SHA-1 and avoids centralized CMAE(s).

Aiello et al. [12] proposed an improvement to CRS called "Hierarchical Scheme" (HS) to reduce the CA-to-CMAE communication. Yet, it can significantly increase the certificate size. Kocher [13] proposed Certificate Revocation Tree (CRT) employing Merkle Hash Tree (MHT). It however has a high computational cost to update the CRT. Naor and Nissim [14] extended the CRT by using a more suitable data structure, a 2-3 tree. In these three schemes, the data structures maintain the statuses of *all* certificates.

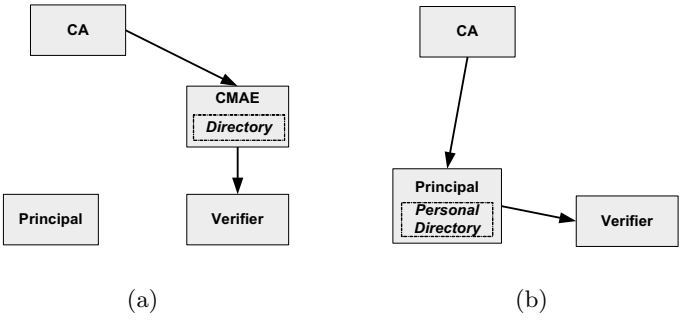

Fig. 1. Comparison of CSI communication flow: (a) typical revocation schemes (with CMAE); (b) Revocation setting with principal as the directory as in CREV schemes

Certificate Push Revocation (CPR) [16] and Beacon CPR (BCPR) [17] suggest placing the directory at the verifiers. Although the schemes may work for highly connected and high-bandwidth verifiers, it seems impractical otherwise and even less so for mobile verifiers. They also suggest placing the cache on the ISPs, but that begs a question of economic incentive for the ISPs to do so.

Certificate Space Partitioning with Renewals (CSPR) [15] was proposed to reduce the high CA-to-CMAE communication cost such as in CRS. The CA divides its certificates into partitions, and signs the CSI for each partition which contains the *status bits* of all certificates in the partition. If there is no status change for the certificates in a partition, the CA renews the partition by releasing hash-chain information whose tip is embedded in the partition's CSI. CSPR employs *CMAEs* as directories, whereas CREV uses *principals* as directories.

3 CREV: Principal-Hosted Revocation

Using our framework, we have identified a new design option where the directory for *an individual* certificate is placed at the respective *principal's server*. The principal's server thus acts as a distribution point for its own CSI. This is advantageous as it allows for the co-location of web/transaction server with the corresponding CSI distribution point. Note that CREV schemes can co-exist with other schemes such as timely CRL. Hence, if an EVCP's server happens to be temporarily unavailable, the verifier can resort to other revocation mechanisms, such as downloading the CRL.

Figure 1 depicts the CSI communication flow between the parties involved. In this setting, we propose additional requirements for security and efficiency:

(*Req*$_1$) **Principal's domain and server:** A principal must have an exclusively controlled domain name and the associated publicly-accessible server.

(*Req*$_2$) **Per-certificate CSI with positive representation:** Since a principal provides a CSI access service only to its own verifiers, it requires a positive status representation to confirm the goodness ("*liveness*") of a certificate.

3.1 CREV Revocation Schemes

We propose two revocation schemes, called CREV-I and CREV-II, which make use of the principal-hosted revocation setting. CREV schemes leverage on the availability of the *Extended-Validation Certificates (EVCs)* from the CA/Browser Forum so as to ensure a verified domain name and the associated server (Req_1).

Two existing revocation schemes, CRS/NOVOMODO and OCSP, meet the Requirement Req_2. We enhance these two schemes, resulting in the two CREV schemes, to also achieve the previously stated Requirements Sec_1–Sec_6 and $Perf_1$–$Perf_3$. CREV-I takes advantage of available online schemes like OCSP. For each EVC, the CA produces *only one* OCSP Response per time interval, which is sent and subsequently hosted by the EVC principal's server. CREV-II improves CREV-I by making use of hash-chaining technique for lightweight subsequent updates. Unlike in CRS/NOVOMODO, CREV-II sets up a *session-based* hash-chaining service between the CA and an EVCP. Hence, it shortens the length of the hash chain for a faster verifier's operation and a significantly reduced CA's storage requirement.

To identify the availability of a CREV revocation scheme, we define the following extension to an EV Certificate:

$$\textit{CREV Extension} ::= \textit{CREV Scheme}, \textit{Transfer Mechanism} \quad (1)$$

CREV Scheme identifies the used CREV scheme, and *Transfer Mechanism* defines the CSI transfer mechanism, e.g. HTTP. The CSI for the EVC is accessible on the EVCP's server on the pre-determined URI (explained later).

With respect to Properties P_1–P_4, CREV realizes the following options:

- (P_1:) The directory is placed in the (server belonging to the) *principal*.
- (P_2:) The released CSI provides assurance for *a single certificate*.
- (P_3:) The status of a certificate is stated using a *positive* representation.
- (P_4:) CREV-I does not employ any linked CSI technique, whereas CREV-II makes use of a hash chain technique. We do not consider a scheme using MHT since it can potentially increase the size of the certificate.

3.2 CREV-I: Session-Based Online Status Scheme

CREV-I takes advantage of a CA's ability to support an online status notification service such as OCSP. Unlike OCSP, the CA in CREV-I produces *only one* OCSP Response for an EVCP per time interval (regardless of the number of verifiers). To support CREV-I, the CA includes the following in a certificate: (i) *CREV Extension* (Eq. 1); and (ii) *CREV-I Extension*, which contains: URI for CA's nonce (URI_{CA_nonce}) and URI for the latest OCSP Response (URI_{latest_OCSP}).

Similar to OCSP, the CA's availability for a request message might be attacked with denial of service by a flood of requests to the CA, including replaying previously valid request messages. To deal with this, we require every incoming *SubscriptionRequest* message (see CREV-I Step 1 below) to be signed by the EVCP. We also require the message to carry T, which is either: a timestamp of

the current time, or a CA's nonce accessible from URI_{CA_nonce}. This nonce is regularly updated by the CA at a pre-determinedly short time interval.

The following protocol establishes a CSI subscription session between CA and EVCP. We assume that serial number (S_No) and CA's name (CA_ID) are sufficient to identify a unique certificate. The notation $\langle M \rangle_{K_A^{-1}}$ denotes a message M which is signed using the private key of principal A (i.e. K_A^{-1}); whereas $nonce_X$ denotes a nonce from X.

1. $EVCP \rightarrow CA$: "*Subscription Request*"$= \langle SubsReq, EVCP\_ID, CA\_ID, S\_No, T, nonce_{EVCP}, t_{EVCP}, d_{EVCP}, SigAlgID \rangle_{K_{EVCP}^{-1}}$.
   where: $SubsReq$ = header indicating a Subscription Request message;
   $EVCP\_ID$ = identity (i.e. domain name) of the EVCP;
   T = either a timestamp, or CA's nonce accessible on URI_{CA_nonce};
   t_{EVCP} = EVCP's proposed lifetime of the established session;
   d_{EVCP} = EVCP's proposed time interval between two OCSP Responses;
   $SigAlgID$ = identification for the signing algorithm.
2. CA : If T or K_{EVCP}^{-1} is incorrect, then abort.
3. $CA \rightarrow EVCP$: "*Subscription Reply*"$= \langle SubsReply, EstStatus, CA\_ID, EVCP\_ID, S\_No, nonce_{EVCP}, CertStatus, d_{CA}, SessStart, SessExpiry, nonce_{CA}, SigAlgID \rangle_{K_{CA}^{-1}}$.
   where: $SubsReply$ = header indicating a Subscription Reply message;
   $EstStatus$ = status indicator for a successful subscription establishment;
   $CertStatus$ = status of the EVC;
   d_{CA} = selected time interval between two consecutive OCSP Responses.
   The session's lifetime $(t_{CREV\_I}) = SessExpiry - SessStart$.
4. $EVCP$: if $nonce_{EVCP}$ or K_{CA}^{-1} incorrect,
   or $EstStatus$ is unsuccesful, then abort.
5. $EVCP \rightarrow CA$: "*Subscription ACK*"$= \langle SubsACK, EVCP\_ID, CA\_ID, S\_No, nonce_{CA}, SigAlgID \rangle_{K_{EVCP}^{-1}}$.
   where: $SubsACK$ = header indicating a Subscription ACK message.
6. $EVCP$: Establish an update association with CA.
7. CA : If $nonce_{CA}$ and K_{EVCP}^{-1} is incorrect, then abort.

After Step 7, the CA starts delivering (pushing) OCSP Response messages to the EVCP every d_{CA} time interval for ℓ_{CA} times, or until the certificate is revoked. Upon receipt of every OCSP Response from the CA, the EVCP puts it at URI_{latest_OCSP}. Note that the periodically released OCSP Responses contain no requester's nonce. As such, the EVCP and the verifier must verify the freshness of an OCSP Response by checking the included values of `thisUpdate` and `nextUpdate` to be current. The CA must properly set a Response's validity period (i.e. `nextUpdate - thisUpdate`) to d_{CA}.

3.3 CREV-II: Session-Based Hash-Chaining Scheme

Despite their benefits, CRS and NOVOMODO can incur significant overheads when the hash chains employed are rather long since the length of the hash chain

increases with timeliness. The verifiers need to perform more repeated hash computations, and the CA must allocate more storage for the hash chains. CREV-II improves CRS/NOVOMODO by setting up a *session-based* hash-chaining service between the CA and an EVCP. The idea is that each EVCP and the CA establish a secure session using the "*Session Establishment*" protocol given below. Thus, CREV-II employs a hash chain, but the length of the hash chain is much shorter. CREV-II also represents an improvement over CREV-I by allowing lightweight subsequent status updates using a (one-way) hash chain.

In a published certificate, the CA includes: (i) *CREV Extension* (Eq. 1); and (ii) *CREV-II Extension*, which contains: URI for the CA's nonce (URI_{CA_nonce}), URI for the *Session Reply* message ($URI_{hashchain\_session}$), and URI for the hash-chain token (URI_{hash_token}). The following protocol is executed by the CA and EVCP to establish a hash chain session on a valid EVC.

1. $EVCP \rightarrow CA$: "*Session Request*"= $\langle SessReq,\ EVCP\_ID,\ CA\_ID,\ S\_No,\ T,\ nonce_{EVCP},\ SigAlgID \rangle_{K^{-1}_{EVCP}}$.
   where: $SessReq$ = header indicating a Session Request message;
   T = timestamp or a CA's nonce accessible at URI_{CA_nonce}.
2. CA : If T or K^{-1}_{EVCP} is incorrect, then abort.
3. $CA \rightarrow EVCP$: "*Session Reply*"= $\langle SessReply,\ ReplyStatus,\ CA\_ID,\ EVCP\_ID,\ nonce_{EVCP},\ S\_No,\ CertStatus,\ HashAlgID,\ d, Y, N, SessStart, SessExpiry,\ nonce_{CA},\ SigAlgID \rangle_{K^{-1}_{CA}}$.
   where: $SessReply$ = header indicating a Session Reply message;
   $ReplyStatus$ = status indicator for a successful session establishment;
   $HashAlgID$, d, Y, N = hash chain parameters in CRS (see Section 2.2);
   $SessStart$ and $SessExpiry$ = the start and end times of the session.
4. $EVCP$: If $nonce_{EVCP}$ or K^{-1}_{CA} incorrect, or $ReplyStatus$ is unsuccesful, then abort.
5. $EVCP \rightarrow CA$: "*Session ACK*"= $\langle SessACK, EVCP\_ID, CA\_ID, nonce_{CA},\ S\_No,\ SigAlgID \rangle_{K^{-1}_{EVCP}}$.
   where: $SessACK$ = header indicating a Session ACK message.
6. $EVCP$: Put *Session Reply* from Step 3 at $URI_{hashchain\_session}$;
   Establish an association for hash chain updates with CA.
7. CA : If $nonce_{CA}$ and K^{-1}_{EVCP} is incorrect, then abort.
   Start providing timely hash-chain token updates until the session expires, or the EV certificate is revoked.

The established session is good for time interval $t_{CREV\_II} = SessExpiry - SessStart$. For a valid certificate, on the i-th time interval (for $1 \leq i \leq \ell_{CREV\_II}$) after $SessStart$, the CA releases the latest hash chain token (V_i) to the EVCP as in CRS/NOVOMODO. The EVCP always puts the most recent V_i that it receives from the CA at URI_{hash_token}.

The verifier obtains the *Session Reply* from the specified $URI_{hashchain\_session}$, and V_i from URI_{hash_token}. It validates K^{-1}_{CA} in the *Session Reply*, and determines that the EVC is still valid if $H^i(V_i) = Y$, or that the EVC is revoked if $H(V_i) = N$. Note that the length of the hash chain in a CREV-II session

is $\ell_{CREV\_II} = \frac{t_{CREV\_II}}{d} - 1$, which is much smaller than CRS/NOVOMODO. Thus, the workload for the verifier's repeated hash operations is much reduced. The CA also stores and keep tracks of a much shorter (session-wide) hash chain.

3.4 Evaluating CREV on the Requirements

We now evaluate how the CREV schemes satisfy the trust and security requirements Sec_1–Sec_6:

- (Sec_1:) By definition, each EVC is required to have the associated publicly-available server. This server is expected to be continuously up and running in order to provide related online service(s). Given a reliable communication channel between the CA and this server, the CSI is thus available for access.
- (Sec_2:) The OCSP Response in CREV-I is always signed by the CA. In CREV-II, the periodically released hash tokens are unsigned. However, due to the signed hash chain tip and the use of one-way hash chain, a verifier can easily verify as to whether a hash token originates from the CA.
- (Sec_3:) The validity of an OCSP Response in CREV-I can be determined by checking the values of `thisUpdate` and `nextUpdate` (as two fields in the standard OCSP Response). The timeliness guarantee of hash token updates in CREV-II can be seen from the *Session Reply* message signed by the CA.
- (Sec_4:) The employed revocation scheme can be identified from the *CREV Extension*, in which the EVC principal serves as the CSI access point.
- (Sec_5:) Our use of EV certificates provides a strong protection on both the principal's name and server against impersonation attacks.
- (Sec_6:) Since a verifier obtain the CSI of an EVCP directly from its server, with whom the verifier will potentially conduct online transactions afterwards, CREV schemes do not reveal the status queries to any other third party including the CA. The EVCP may find out the (addresses of) verifiers who obtains its CSI but do not proceed with any online transactions afterwards. However, the information obtained by the EVCP in this case is negative in nature, e.g. who did not (proceed to) use the provided service. There is still less information about the verifiers than the CA (which is an external party with respect to the transactions) can obtain in OCSP.

Now we analyze how CREV addresses the performance requirements:

- ($Perf_1$:) In CREV, the EVCP servers are self-managed (i.e. independent from the CA), and deal only with *their own* respective verifiers. Thus, the workload of CSI access is now distributed to the EVCP's servers, which are expected to have sufficient capacity to meet the respective transaction volumes. As such, potential bottlenecks (such as the CA in OCSP) is avoided.
- ($Perf_2$:) Due to our use of per-certificate CSI, the verifier's bandwidth requirement is low. Only one signature verification operation by the verifier is needed in CREV-I. CREV-II makes use of a session-based hash chaining in order to reduce the verifier's required repeated hash operations.
- ($Perf_3$:) An EVCP's server has a strong incentive to provide reliable and timely CSI access service for its *own* transactions. Also notice that the CA

provides the service to EVCPs. Hence, CREV also allows for a workable economic model in which the CA charges EVCPs, rather than the verifiers, for the rendered CSI services. It also has economic advantages to schemes employing the verifier's ISP as directory [16,17]. CREV thus addresses the incentive problem on CSI management and dissemination [2].

We remark that the overhead of session management is incurred only between the CA and EVCPs. Nevertheless, there are far fewer EVCPS than verifiers. Thus, we expect the overhead to be manageable. Later, we show that the CA's cryptographic overheads of CREV schemes are much lower than OCSP.

We project that the CREV schemes are practical even with near a real-time timeliness guarantee from 10 to 1 minute. A 10-minute guarantee may already be considered short-lived in many environments. A 1-minute guarantee could be even considered as being "indistinguishable" from a real-time service due to potential clock time differences among the entities [9]. The recency requirements in CREV are set by the CA and EVCP, and not the verifier. But, with a high timeliness guarantee, a verifier receives greater assurance.

4 A Performance Evaluation of CREV

We evaluate CREV schemes with others using a simple analytical model to measure the costs with a single-CA system during the *steady-state condition*, i.e. when certificate expiration is balanced by the new certificates added. This approach is commonly adopted by many papers [20,21]. It allows us to focus on the stable behavior of the revocation schemes and omit external variable factors. Since we consider a certification system with a uniform certificate lifetime β, the steady-state condition thus takes place after β days from the *first* certificate generation in the system, i.e. the time interval $(\beta, +\infty)$.

The commonly made assumptions when analyzing revocation schemes under the steady state, which we also make, are below. The total number of principals and the corresponding valid certificates (N) is constant. Certificates have the same lifetime (of β days), where b percent of N certificates are revoked within β days. The revocation of a certificate occurs at the half of its lifetime. The time interval between two successive CRL releases (Δt) is constant, which is the same as the time interval between two successive certificate generations (ΔX). Certificate issuance takes place at a constant rate (of $\frac{N \cdot \Delta t}{\beta \cdot 1{,}440}$ per issuance). (We use time units in minutes, and 1,440 = the number of minutes in a day). When a certificate is revoked, a new certificate is issued immediately to replace it.

Using a model where $\Delta X = \Delta t$, we thus consider the situation where the certificate issuance and the CRL release take place continuously. Table 2 summarizes the notation and parameter values used in our analytical evaluation.

Performance Comparison Metrics. We use the following notation to denote various costs incurred in a particular revocation scheme: Ovh_A= computation time needed by entity A (in seconds), Bw_{A-B} = network bandwidth needed from entity A to B (in MB), and $Stor_A$ = storage needed on A (in MB). The entities involved are: CA, Ver (indicating a verifier), $CMAE$, and $EVCP$.

Table 2. Notation and parameter values used in the performance model

Symbol	Description	Unit	Value(s)	
N	Number of valid (non-revoked) and not-yet-expired certificates	-	100,000	
β	Issued lifetime of a certificate	days	365	
b	Percentage of certificates revoked	-	0.1 = 10%	
δ	Revocation latency	mins	60	10
$\Delta X = \delta$	Time interval between two successive certificate generations	mins	60	10
$\Delta t = \delta$	Time interval between two successive CRL releases	mins	60	10
$d = \delta$	Time interval for periodic hash-token release	mins	60	10
t_{CREV_I}, t_{CREV_II}	Session lifetime in CREV-I and CREV-II	mins	720	180
U	Total number of CRL and hash-token releases per day	-	$1,440/\delta$	
V	Number of verifiers	-	30,000,000	
Q_{V_daily}	Average daily queries needed by a verifier	-	30	
$Q_{V\_issued\_daily}$	Average daily queries *issued* by a verifier	-	$min(U, Q_{V\_daily})$	
Q_{daily}	Total average daily queries received by CA/CMAE	-	$V \cdot Q_{V\_issued\_daily}$	
$Q_{per\_cert\_daily}$	Average daily queries on a certificate	-	Q_{daily}/N	

To compare different schemes, we use the following cost metrics:

1. Certificate creation cost: Ovh_{CA}.
2. Update costs: Ovh_{CA}, Ovh_{CMAE} (Ovh_{EVCP}) and $Bw_{CA-CMAE}$ ($Bw_{CA-EVCP}$).
3. Query costs: Ovh_{CA}, Ovh_{CMAE} (Ovh_{EVCP}), $Bw_{CMAE-Ver}$ ($Bw_{EVCP-Ver}$) and Ovh_{Ver}.
4. Storage requirement at one point in time: $Stor_{CA}$ and $Stor_{CMAE}$ ($Stor_{EVCP}$).
5. Timeliness: the revocation latency (δ), which also represents the *window of vulnerability* of the revocation scheme.

For metrics (1) to (3), we measure the total *cost per day*. We use $D\_\langle Cost \rangle$ to denote *the daily cost* of a cost metric. In order to have a more compact notation, we abuse the notation slightly and write $\forall A$ for all instances of entity A, and $\exists A$ for a single instance of A. Thus, for example, $D\_Bw_{CMAE-\exists Ver}$ denotes the daily bandwidth cost between a CMAE and a single verifier, whereas $D\_Bw_{CMAE-\forall Ver}$ is the daily bandwidth needed by a CMAE to all the verifiers.

4.1 A Simple Performance Model for Revocation Schemes

We derive the costs for the following schemes using a simple analysis model: CRL (with one CMAE employed), OCSP (with the CA as the Responder), CRS (with one CMAE employed), and our two CREV schemes. For bandwidth calculation,

we do not consider the cost due to the underlying network transfer mechanism(s). We use L_{Msg} to denote the length of message portion Msg. Due to space reasons, we describe the costs compactly as the derivations are straightforward.

In our evaluation, we use revocation latency (δ) of 1 hour and also 10 minutes. The number of average daily queries *needed* by a verifier (Q_{V_daily}) is 30 (see Table 2). We assume that the queries from a verifier are issued throughout the day with uniformly distributed time intervals. Since a verifier can cache the revocation information as long as it is still valid, the number of daily queries *actually issued* by a verifier is: $Q_{V\_issued\_daily} = min(\frac{M}{\delta}, Q_{V\_daily})$, where $M = 1,440$ is the number of minutes in a day. The total number of daily queries issued by *all* verifiers is therefore: $Q_{daily} = V \cdot Q_{V\_issued\_daily}$. In all the schemes below, the *daily* cost of certificate creation is: $D\_Ovh_{CA} = (\frac{N}{\beta} + \frac{N \cdot b}{\beta}) \cdot C_{sign}$.

CRL (with a CMAE). The size of CRL is: $L_{CRL} = L_{CRL\_fields} + \lfloor Nb/2 \rfloor \cdot L_{CRL\_entry}$, where $L_{CRL\_fields} = 400$ bytes is the length of the CRL header and signature, and $L_{CRL\_entry} = 39$ bytes is the length of each entry in CRL [22]. With $U = \frac{M}{\delta}$ as the total number of CRL updates in a day, the daily update costs are: $D\_Bw_{CA-CMAE} = U \cdot L_{CRL}$, $D\_Ovh_{CA} = U \cdot C_{sign}$, and $D\_Ovh_{CMAE} = U \cdot C_{verify}$.

The daily query costs of CRL are: $D\_Bw_{CMAE-\exists Ver} = Q_{V\_issued\_daily} \cdot L_{CRL}$, $D\_Bw_{CMAE-\forall Ver} = Q_{daily} \cdot L_{CRL}$, $D\_Ovh_{CA} = 0$, $D\_Ovh_{CMAE} = 0$, and $D\_Ovh_{Ver} = Q_{V\_issued\_daily} \cdot C_{verify}$.

The storage requirements are: $Stor_{CA} = Stor_{CMAE} = L_{CRL}$. Finally, the revocation latency is δ minutes.

OCSP (with the CA as OCSP Responder). There is no update cost between the CA and CMAE, since no CMAE is involved. With $L_{OCSP\_Resp} = 459$ bytes as the length of OCSP Response [22], the daily query costs (due to status reply) are: $D\_Bw_{CA-\forall Ver} = Q_{daily} \cdot L_{OCSP\_Resp}$, $D\_Bw_{CA-\exists Ver} = Q_{V\_issued\_daily} \cdot L_{OCSP\_Resp}$, $D\_Ovh_{CA} = Q_{daily} \cdot C_{sign}$, and $D\_Ovh_{Ver} = Q_{V\_issued\_daily} \cdot C_{verify}$. The storage requirement is: $Stor_{CA} = 0$. The revocation latency of OCSP can be close to zero when desired.

CRS (with a CMAE). We set the time interval for periodic hash-token release $d = \delta$ minute(s). The length of the hash chain is thus: $\ell = \frac{\beta M}{\delta} - 1$. Here, we assume that the CA stores the whole hash chain for all the valid certificates in its storage. An amortization technique such as [23] can be used to reduce its storage requirements, but at the cost of additional online processing for the CA.

We use $L_{CRS\_fields} = 161$ bytes to denote the length of the CA's timestamp and signature, and $L_{S\_No} = 7$ bytes to denote the length of a certificate's serial number [22]. The bandwidth cost for a *single* CRS update between CA and CMAE is: $Bw_{CA-CMAE} = L_{CRS\_fields} + \big((N + \lfloor Nb/2 \rfloor) \cdot (L_{S\_No} + L_{hash})\big)$. With $U = \frac{M}{\delta}$, the total daily update costs become: $D\_Bw_{CA-CMAE} = U \cdot Bw_{CA-CMAE}$, $D\_Ovh_{CA} = U \cdot C_{sign}$, and $D\_Ovh_{CMAE} = U \cdot C_{verify}$.

Under the steady-state condition, there are N valid certificates and $Nb/2$ (or $0.05N$ when b=0.1) revoked certificates in the system. The majority of the certificates are thus valid.[1] The expected average overhead of a verifier in validating *one* hash token is: $Ovh_{Ver} = \frac{\ell+1}{2} \cdot C_{hash}$. The corresponding daily query costs are: $D\_Bw_{CMAE-\forall Ver} = Q_{daily} \cdot L_{hash}$, $D\_Bw_{CMAE-\exists Ver} = Q_{V\_issued\_daily} \cdot L_{hash}$, $D\_Ovh_{CA} = 0$, $D\_Ovh_{CMAE} = 0$, and $D\_Ovh_{Ver} = Q_{V\_issued\_daily} \cdot Ovh_{Ver}$.

For the storage costs, note that the CA can remove the subchains it has released. Thus, the storage requirements in CRS are: $Stor_{CA} = \frac{N\delta}{\beta M} \cdot \frac{\ell^2+3\ell}{2} \cdot L_{hash} + \lfloor Nb/2 \rfloor \cdot L_{hash}$, and $Stor_{CMAE} = Bw_{CA-CMAE}$.

CREV-I. The CA sends an OCSP Response message every $d_{CA} = \delta$ minutes. In a day, each EVCP thus performs $S = \frac{M}{t_{CREV\_I}}$ session establishments, and receives $U = \frac{M}{\delta}$ OCSP Response messages. We use $L_{CREV\_I\_Msgs} = 1{,}577$ bytes to denote the length of all messages in a session establishment, L_{OCSP_Resp} to denote the length of an OCSP Response message, and L_T to denote the length of the timestamp or a CA's nonce.

The total daily update costs are: $D\_Bw_{CA-\forall EVCP} = N \cdot (S \cdot L_{CREV\_I\_Msgs} + U \cdot L_{OCSP\_Resp})$, $D\_Bw_{CA-\exists EVCP} = S \cdot L_{CREV\_I\_Msgs} + U \cdot L_{OCSP\_Resp}$, $D\_Ovh_{CA} = N \cdot S \cdot (2 \cdot C_{verify} + C_{sign}) + N \cdot U \cdot C_{sign}$, and $D\_Ovh_{EVCP} = S \cdot (2 \cdot C_{sign} + C_{verify}) + U \cdot C_{verify}$.

The total daily query costs are as follows: $D\_Bw_{EVCP-\forall Ver} = Q_{per\_cert\_daily} \cdot L_{OCSP\_Resp}$, $D\_Bw_{EVCP-\exists Ver} = Q_{V\_issued\_daily} \cdot L_{OCSP\_Resp}$, with $D\_Ovh_{CA} = 0$, $D\_Ovh_{EVCP} = 0$, and $D\_Ovh_{Ver} = Q_{V\_issued\_daily} \cdot C_{verify}$.

The storage requirements are: $Stor_{CA} = 0$ and $Stor_{EVCP} = L_T + L_{OCSP\_Resp}$.

CREV-II. We set the hash-chain update interval in CREV-II (d) to δ minutes. Each EVCP thus performs $S = \frac{M}{t_{CREV\_II}}$ session establishments daily, and receives $U = S \cdot \ell_{CREV\_II} = \frac{M}{\delta} - S$ hash-token updates daily. We use $L_{CREV\_II\_Reply} = 605$ bytes to denote the length of *Session Reply* message, and $L_{CREV\_II\_Msgs} = 1{,}615$ bytes to denote the length of all messages in a session establishment of CREV-II.

The total daily update costs are: $D\_Bw_{CA-\forall EVCP} = N \cdot (S \cdot L_{CREV\_II\_Msgs} + U \cdot L_{hash})$, $D\_Bw_{CA-\exists EVCP} = S \cdot L_{CREV\_II\_Msgs} + U \cdot L_{hash}$, $D\_Ovh_{CA} = N \cdot S \cdot (2 \cdot C_{verify} + C_{sign})$, and $D\_Ovh_{EVCP} = S \cdot (2 \cdot C_{sign} + C_{verify}) + U \cdot C_{hash}$.

The expected verifier's average query cost is: $Ovh_{Ver} = \frac{\ell_{CREV\_II}+1}{2} \cdot C_{hash} + C_{verify}$. The total daily costs are: $D\_Bw_{EVCP-\forall Ver} = Q_{per\_cert\_daily} \cdot (L_{hash} + L_{CREV\_II\_Reply})$, $D\_Bw_{EVCP-\exists Ver} = Q_{V\_issued\_daily} \cdot (L_{hash} + L_{CREV\_II\_Reply})$, $D\_Ovh_{CA} = 0$, $D\_Ovh_{EVCP} = 0$, and $D\_Ovh_{Ver} = Q_{V\_issued\_daily} \cdot Ovh_{Ver}$.

The storage requirements are: $Stor_{CA} = \frac{N}{\ell_{CREV\_II}} \cdot \frac{\ell^2_{CREV\_II}+3\ell_{CREV\_II}}{2} \cdot L_{hash}$, and $Stor_{EVCP} = L_T + L_{CREV\_II\_Reply} + L_{hash}$.

[1] Furthermore, in practice, queries on revoked certificates are expected to decrease over time due to the increasing usages of the valid replacement certificates.

Table 3. Cost comparison of various schemes with $\delta = 1$ hour. U and Q denote the costs due to update and query respectively.

Entity	Daily Costs	Unit	CRL	OCSP	CRS	CREV-I	CREV-II
CA	D_Ovh_{CA} (U+Q)	sec	0.036	$\mathbf{1.07\times10^{6}}$	3.39×10^{-5}	3876	324
	$Stor_{CA}$	MB	0.19	0	**8355.24**	0	13.35
	$D\_Bw_{CA-CMAE}$ (U)	MB	4.47	-	64.89	-	-
	$D\_Bw_{CA-\forall EVCP}$ (U)	MB	-	-	-	1351.36	350.00
	$D\_Bw_{CA-\forall Ver}$ (Q)	MB	-	$\mathbf{3.15\times10^{5}}$	-	-	-
CMAE	D_Ovh_{CMAE} (U+Q)	sec	0.0017	-	0.0017	-	-
	$Stor_{CMAE}$	MB	0.19	-	2.70	-	-
	$D\_Bw_{CA-CMAE}$ (U)	MB	4.47	-	64.89	-	-
	$D\_Bw_{CMAE-\forall Ver}$ (Q)	MB	$\mathbf{1.34\times10^{8}}$	-	1.37×10^{4}	-	-
EVCP	D_Ovh_{EVCP} (U+Q)	sec	-	-	-	0.0078	0.0061
	$Stor_{EVCP}$	MB	-	-	-	4.53×10^{-4}	6.11×10^{-4}
	$D\_Bw_{CA-\exists EVCP}$ (U)	MB	-	-	-	0.014	0.0035
	$D\_Bw_{EVCP-\forall Ver}$ (Q)	MB	-	-	-	3.15	4.29
Verifier	D_Ovh_{Ver} (Q)	sec	0.0017	0.0017	**0.042**	0.0017	0.0017
	$D\_Bw_{CA-\exists Ver}$ (Q)	MB	-	0.011	-	-	-
	$D\_Bw_{CMAE-\exists Ver}$ (Q)	MB	**4.47**	-	4.58×10^{-4}	-	-
	$D\_Bw_{EVCP-\exists Ver}$ (Q)	MB	-	-	-	0.011	0.014
Revocation Latency		mins	60	≈ 0	60	60	60

4.2 Performance Evaluation of CREV and Other Schemes

We evaluated the models in Sect. 4.1 with parameter values for the two scenarios given in Table 2. The objective here is to have a quantitative comparison of the costs incurred by the various schemes under common conditions and basic costs.

All hash values are generated using SHA-1, and signatures are created using RSA with a 1024-bit modulus. The overheads of the basic cryptographic operations, based on the Crypto++ 5.6.0 Benchmarks (`http://www.cryptopp.com/benchmarks.html`) on Intel Core-2 PC with 1.83 GHz CPU, are as follows: C_{sign} = 1.48 ms as the cost of a digital signature (RSA-1024) generation;

Table 4. Cost comparison of various schemes with $\delta = 10$ minutes

Entity	Daily Costs	Unit	CRL	OCSP	CRS	CREV-I	CREV-II
CA	D_Ovh_{CA} (U+Q)	sec	0.21	$\mathbf{1.33\times10^{6}}$	2.03×10^{-4}	2.26×10^{4}	1296
	$Stor_{CA}$	MB	0.19	0	$\mathbf{5.01\times10^{4}}$	0	19.07
	$D\_Bw_{CA-CMAE}$ (U)	MB	26.83	-	389.35	-	-
	$D\_Bw_{CA-\forall EVCP}$ (U)	MB	-	-	-	7506.56	1491.55
	$D\_Bw_{CA-\forall Ver}$ (Q)	MB	-	$\mathbf{3.94\times10^{5}}$	-	-	-
CMAE	D_Ovh_{CMAE} (U+Q)	sec	0.01	-	0.01	-	-
	$Stor_{CMAE}$	MB	0.19	-	2.70	-	-
	$D\_Bw_{CA-CMAE}$ (U)	MB	26.83	-	389.35	-	-
	$D\_Bw_{CMAE-\forall Ver}$ (Q)	MB	$\mathbf{1.68\times10^{8}}$	-	1.72×10^{4}	-	-
EVCP	D_Ovh_{EVCP} (U+Q)	sec	-	-	-	0.034	0.024
	$Stor_{EVCP}$	MB	-	-	-	4.53×10^{-4}	6.11×10^{-4}
	$D\_Bw_{CA-\exists EVCP}$ (U)	MB	-	-	-	0.075	0.015
	$D\_Bw_{EVCP-\forall Ver}$ (Q)	MB	-	-	-	3.94	5.36
Verifier	D_Ovh_{Ver} (Q)	sec	0.0021	0.0021	**0.32**	0.0021	0.0022
	$D\_Bw_{CA-\exists Ver}$ (Q)	MB	-	0.013	-	-	-
	$D\_Bw_{CMAE-\exists Ver}$ (Q)	MB	**5.59**	-	5.72×10^{-4}	-	-
	$D\_Bw_{EVCP-\exists Ver}$ (Q)	MB	-	-	-	0.013	0.018
Revocation Latency		mins	10	≈ 0	10	10	10

$C_{verify} = 0.07$ ms as the cost of a digital signature (RSA-1024) verification; and $C_{hash} = 0.40\,\mu s$ as the cost of computing a hash (SHA-1).

Table 3 and Table 4 show the overheads of the all revocation schemes under the evaluation scenarios. The daily computational overhead for certificate creation (of $\frac{N(1+b)}{\beta}$ certificates), D_Ovh_{CA}, is 0.45 second in all the schemes.

We can see that CREV schemes offer a good trade-off between the costs incurred on the CA, CMAE and EVCP, while incurring low costs on the verifier. Even for δ=10 minutes, the CA's daily computational cost (D_Ovh_{CA}) in CREV-I is 2.26×10^4 seconds (∼6 hrs), much lower than 1.33×10^6 seconds (∼370 hrs) in OCSP. Due to the use of hash chaining, CREV-II incurs an even much lower D_Ovh_{CA}. Compared with other schemes, CREV schemes are thus more viable for deployment when a near real-time timeliness guarantee is needed.

5 Conclusion

We have presented two lightweight and practical certificate revocation schemes, called CREV, based on a principal-hosted CSI revocation setting. We have also analyzed the trust and security requirements of the new setting, and how CREV can address them. Our cost analysis has shown the practicality of CREV when compared to several existing schemes even under the near real-time timeliness guarantee of 10 minutes. CREV offers a good balance of costs incurred on all the entities, while being very lightweight on the verifier. Furthermore, it provides good incentives for the involved entities to provide a secure and trusted revocation service. Thus, CREV gives a more scalable revocation service for real-time needs, and also addresses the constraints of mobile devices.

References

1. Lopez, J., Oppliger, R., Pernul, G.: Why Have Public Key Infrastructures Failed so Far? Internet Research 15(5), 544–556 (2005)
2. Gutmann, P.: PKI: It's Not Dead, Just Resting. Computer 35(8), 41–49 (2002)
3. ITU-T Recommendation X.509: Information Technology - Open Systems Interconnection - The Directory: Public-key and Attribute Certificate Frameworks (2000)
4. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (2008)
5. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (1999)
6. VeriSign, Inc., VeriSign Certification Practice Statement Version 3.8.1 (2009)
7. Iliadis, J., Gritzalis, S., Spinellis, D., de Cock, D., Preneel, B., Gritzalis, D.: Towards a Framework for Evaluating Certificate Status Information Mechanisms. Computer Communications 26(16), 1839–1850 (2003)
8. Micali, S.: Efficient Certificate Revocation. Technical report, MIT-LCS-TM-542b, Massachusetts Institute of Technology (1996)
9. Micali, S.: NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In: PKI Research Workshop (2002)

10. Muñoz, J.L., Forné, J., Esparza, O., Soriano, B.M.: Using OCSP to secure certificate-using transactions in M-commerce. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 280–292. Springer, Heidelberg (2003)
11. Berbecaru, D.: MBS-OCSP: An OCSP based Certificate Revocation System for Wireless Environments. In: Signal Processing and Information Technology (2004)
12. Aiello, W., Lodha, S., Ostrovsky, R.: Fast digital identity revocation. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 137. Springer, Heidelberg (1998)
13. Kocher, P. C.: On Certificate Revocation and Validation. In: Financial Cryptography (1998)
14. Naor, M., Nissim, K.: Certificate Revocation and Certificate Update. In: USENIX Security (1998)
15. Goyal, V.: Certificate Revocation Using Fine Grained Certificate Space Partitioning. In: Financial Cryptography and Data Security (2007)
16. Solworth, J. A.: Instant Revocation. In: European PKI workshop on Public Key Infrastructure: Theory and Practice (2008)
17. Solworth, J. A.: Beacon Certificate Push Revocation. In: Computer Security Architecture Workshop (2008)
18. Scheibelhofer, K.: PKI without Revocation Checking. In: PKI R&D Workshop (2005)
19. Lioy, A., Marian, M., Moltchanova, N., Pala, M.: PKI Past, Present and Future. International Journal of Information Security 5, 18–29 (2006)
20. Lim, T.-L., Lakshminarayanan, A.: On the Performance of Certificate Validation Schemes Based on Pre-Computed Responses. In: GLOBECOM (2007)
21. Zheng, P.: Tradeoffs in Certificate Revocation Schemes. ACM Computer Communication Review 33(2), 103–112 (2003)
22. Perlines Hormann, T., Wrona, K., Holtmanns, S.: Evaluation of Certificate Validation Mechanisms. Computer Communications 29(3), 291–305 (2006)
23. Jakobsson, M.: Fractal Hash Sequence Representation and Traversal. IEEE International Symposium on Information Theory, 437–444 (2002)