Windowed Certificate Revocation

Patrick McDaniel Sugih Jamin
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122
{*pdmcdan,jamin*}*@eecs.umich.edu*

Abstract—**The advent of electronic commerce and personal communications on the Internet heightens concerns over the lack of privacy and security. Network services providing a wide range of security related guarantees are increasingly based on public key certificates. A fundamental problem inhibiting the wide acceptance of existing certificate distribution services is the lack of a scalable certificate revocation mechanism. We argue in this paper that the resource requirements of extant revocation mechanisms place significant burden on certificate servers and network resources. We propose a novel mechanism called** *windowed revocation* **that satisfies the security policies and requirements of existing mechanisms and, at the same time, reduces the burden on certificate servers and network resources. We include a proof of correctness of windowed revocation and analyze worst case performance scenarios.**

I. Introduction

The use of certificates as an enabling technology for security in the Internet is commonplace. Every day, end-users trust certificate services to guarantee, among others, the authenticity of commercial web-sites, to ensure privacy of personal communication, or to state commitment to legally binding contracts. However, the underlying certificate services are subject to a number of known vulnerabilities. One primary vulnerability is the lack of support for *certificate revocation*.

Researchers and standards bodies have argued at great length over possible architectures for providing an authenticated service under which public key certificates can be securely distributed. A central point of contention in these discussions is the design of mechanisms providing efficient and scalable revocation. Revocation is the process whereby a previously issued certificate is invalidated.

In this paper, we investigate *windowed revocation*, a novel approach to certificate revocation.[1] Windowed revocation addresses the inherent scalability problems of traditional revocation mechanisms by allowing the tradeoffs between resource usage and security requirements to be tuned to the target environment. Security requirements, as defined by bounded timeliness, can be met on a *per certificate* basis. Windowed revocation can be integrated with a large number of architectures and optimizations present in existing certificate services.

Patrick McDaniel's research is supported in part by the NASA Graduate Student Researchers Program, Kennedy Space Center, Grant Number 10-52613.

Sugih Jamin's research is supported in part by the NSF CAREER Award ANI-9734145 and the Presidential Early Career Award for Scientists and Engineers (PECASE) 1998. Additional funding is provided by MCI WorldCom, Lucent Bell-Labs, and Fujitsu Laboratories America, and by equipment grants from Sun Microsystems Inc. and Compaq Corp.

[1] Patent pending.

A *certificate* is a data structure that defines an association between an entity (the *principal*) and a public key. A trusted authority, called a *Certification Authority* (CA), states its belief in the validity of the association by digitally signing the certificate. Several certificate distribution architectures, called Public Key Infrastructures (PKI), employ models not based on trusted third party (CA) certificate distribution (see Section V). As windowed revocation may be applied to both CA and non-CA environments, we refer to the generic certificate authenticating body as an *issuer*.

Certificate revocation is the mechanism under which an issuer can revoke the association before the end of its documented *lifetime*. The issuer may wish to revoke a certificate because of the loss or compromise of the associated private key, in response to a change in the owner's access rights, a change in the relationship with the trusted third party, or strictly as a precaution against cryptanalysis [1]. As stated by the issuer or some other authoritative entity, the *revocation state* of a certificate indicates the validity or cancellation of its association. A *verifier* determines the revocation state through the *verification* of the certificate. A *window of vulnerability* states a bound on the use of a revoked certificate. Intuitively, the window of vulnerability provides the granularity of revocation notification, and indirectly defines the quality of service afforded by a revocation mechanism.

Windowed revocation addresses the requirements of certificate services through the definition of secure and scalable revocation facilities. Central to the construction of the windowed revocation was an analysis of the needs of certificate distribution architectures. This analysis lead to an understanding of the following design objectives as requirements not only for windowed revocation, but for any future revocation service:

1. **Correctness** - All verifiers must be able to correctly determine the revocation state of a certificate within well-known (time) bounds.
2. **Scalability** - The costs associated with the management, retrieval, and verification of certificates should increase at a rate slower than increases in the size of the serviced community.
3. **General Guarantee Statement** - Any revocation service must be able to support guarantees consistent with existing security policies and requirements.

As with many security solutions, certificate revocation mechanisms are subject to the fundamental tradeoff between security and scalability. Solutions with strict security requirements consume more resources than systems with more relaxed security requirements. Our proposed approach provides parameters used to manage these tradeoffs through the incorporation of the following design principles into the key revocation mechanism:

1. *Revocation window:* By bounding the time over which the revocation of a certificate is announced, we limit the size of such announcements.
2. *Certificate caching:* A cached certificate may be used until it expires, is revoked, or a pre-specified time-to-live (TTL) is reached. The expiration of a TTL indicates that the associated entity's policy requires the certificate to be revalidated.
3. *Scheduled Announcement:* By stipulating that issuers generate revocation announcements at a documented schedule, we allow verifiers to detect lost announcements.

In this paper we describe windowed revocation and assess its viability as scalable revocation service. In the next section we identify two classes of existing revocation technologies, and introduce windowed revocation as a mechanism for achieving scalable certificate revocation. Section III analyzes the scalability, correctness, and semantics of windowed revocation. Section IV assesses the performance of windowed revocation. Section V gives a brief overview of work related to certificate revocation. We conclude in Section VI.

II. Certificate Revocation

As previously noted, the purpose of revocation is to nullify the association stated by the existence of a digitally signed certificate. In this section we explore the resource requirements of extant revocation mechanisms and describe windowed revocation.

A. Implicit and Explicit Revocation

We recognize two fundamental approaches used to distribute revocation state: explicit and implicit. In certificate distribution architectures that employ explicit revocation, each issuer explicitly states which certificates are revoked, and indirectly which are not revoked. In X.500 [2] based systems, each issuer periodically generates a list of certificates that have been revoked, but have not yet expired. The presence of the certificate in the list,[2] called a *Certificate Revocation List* (CRL), explicitly states revocation. A discussion on the semantic limitations of CRLs is given in Section III-A. The canonical CRL based architecture is the Privacy Enhanced Mail (PEM) [3], [4] system, an architecture originally designed for the distribution of certificates used to secure electronic mail.

Verifiers retrieve and cache the latest CRL during the certificate verification process. Because CRLs are the only medium from which revocation state can be obtained, the window of vulnerability in explicit revocation is equal to the periodicity of CRL publication. A revoked certificate is included in a CRL from the time it is revoked until its validity period expires.[3] Given that revocation is announced until certificate expiration, and the certificate lifetime is commonly measured in years, even modest revocation rates may induce large CRLs.

Another potential scalability limitation of explicit schemes is that verifiers may become synchronized around CRL publication. When a verifier determines that a new CRL has been published, she may immediately attempt to retrieve it. Thus, many verifiers may request the CRL at or near the moment of publication. The burst of requests immediately following CRL publication, which we call *CRL request implosion*, may cause network congestion and introduce latency in the certificate verification process. A number of approaches designed to reduce the costs associated with CRL acquisition and construction have been proposed in the literature. We describe several of these approaches in Section V.

In certificate distribution architectures that employ implicit revocation, revocation state is implicitly asserted through the verifier's ability to retrieve the certificate. Any certificate retrieved from the issuer is guaranteed to be valid at or near the time of retrieval. Associated with each certificate is a *time-to-live* (TTL) which represents the maximum time the certificate may be cached. Thus, in implicit revocation, the window of vulnerability is exactly the TTL. The Secure DNS (DNSSec) [5], [6] architecture uses a form of implicit revocation.[4]

In implicit revocation, the certificate retrieval protocol must have freshness and authenticity guarantees. Without such guarantees, the verifier may be subject to a number of masquerading and replay attacks. Providing these guarantees for each certificate retrieval may limit the scalability of the system.

A central parameter of implicit revocation is the length of the certificate TTL. Issuers must trade-off security (as stated by the bound on revoked certificate use) with the frequency of retrieval. A long TTL may expose the verifier to a revoked certificate. A short TTL requires the verifier to re-acquire the certificate frequently. In extant systems, each retrieval requires heavyweight operations by the verifier, the issuer, or both.

B. Windowed Revocation

A key observation driving the design of windowed revocation is that, like many other communication services, certificate usage exhibits *reference locality*. Windowed revocation takes advantage of certificate locality by optimizing costs in the average case (i.e. reducing costs for frequently used certificates) at the expense of the exceptional case (i.e. increased costs when certificates are infrequently used). Moreover, the degree to which the average case performance improves at the expense of the exceptional case may be tuned through algorithm parameters.

The security requirements of certificate services is driven not only by the certificates themselves, but also by the operations for which they are used. Some application operations necessarily require strong guarantees. For example, it may be necessary to support real-time revocation state during validation of signatures on legally binding contracts. Thus, it is incumbent on any general purpose revocation service to support such "security critical operations". However, supporting these guarantees in the average case may lead to significant performance problems. Similarly, some certificates have naturally stronger security requirements than others. For example, certificates used

[2] The entire certificate is generally not present in the list, but is referenced by some unique identifier. This identifier is commonly known as a serial number.

[3] In most existing approaches, a certificate's lifetime is defined by an explicitly stated validity interval. If unrevoked, a certificate is valid from the `notBefore` to `notAfter` timestamp fields included in the certificate. The certificate is assumed invalid at any time outside this interval. A certificate *expires* when the `notAfter` time is reached.

[4] The original DNSSec [6] operates in an *off-line* mode that provides a high degree of scalability at the cost of a loose bound on the window of vulnerability. Later modifications to DNSSec [5] provided a *transactional authenticity* mode which is roughly equivalent to our definition of implicit revocation. To the first order of approximation, the off-line mode can be considered a looser form of implicit revocation.

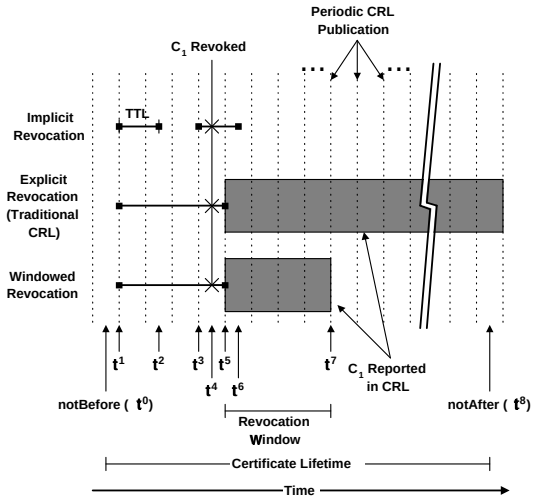

Fig. 1. Implicit, explicit, and windowed revocation.

for signing purchase orders may require stronger security guarantees than certificates used for authenticating enterprise internal web-pages. Windowed revocation provides support for both critical operations and certificates with varying security requirements.

We now describe the windowed revocation algorithm. In windowed revocation, revocation is stated by an issuer in two ways; implicitly during initial certificate acquisition, and explicitly through periodically published CRLs. Retrieved certificates are guaranteed to be non-revoked, fresh, and authentic. Subsequent validation of certificates' revocation state is achieved primarily through CRLs. CRLs are generated at a uniform rate documented in each certificate, called a *CRL publication period*. Revoked certificates are included in scheduled CRLs for a period equal to their *revocation window*. The revocation window states the length of time a certificate may be cached without further validation. The revocation window is specified by the issuer and documented in each certificate. Verifiers acquire CRLs from issuers directly. We consider an alternative mechanism for CRL retrieval in section II-C.

Because revocation is explicitly stated in the CRL only for the revocation window, the verifier will have no means of determining the correct revocation state afterwards. Therefore, if a verifier does not acquire the CRL during the revocation window, the certificate must be dropped from the verifier's cache.

We illustrate implicit, explicit, and windowed revocation in Figure 1. In the figure we show the lifetime of a certificate C_1, which has a documented validity period from `notBefore` (t^0) to `notAfter` (t^8). At time t^4, C_1 is revoked. Assume C_1 is verified at times t^1 and t^3 in each example.

In traditional explicit revocation, the certificate and last generated CRL is retrieved at time t^1. Each subsequent use (e.g. at time t^3) of the certificate requires that the most recent CRL be checked for a revocation announcement. Because a cached certificate is only authenticated as required by use, there is no bound on the time in which a CRL may be retrieved by the user. Therefore, the issuer must announce the revocation of each certificate starting from the CRL immediately following the revocation of the certificate (t^5) until the expiration time of the certificate (t^8).

In implicit revocation, the user securely retrieves and caches C_1 at time t^1. No further verification is performed between t^1 and the expiration of the certificate's TTL at t^2 ($t^2 = t^1 + TTL\ length$). At t^2, the certificate is dropped. The certificate need not be re-acquired until it is needed again at time t^3. Because verification is performed only during retrieval, the revocation of C_1 will not be discovered until it is dropped due to the expiration of the TTL at time t^6 ($t^6 = t^3 + TTL\ length$) and re-acquired afterward.

Windowed revocation bounds the time at which a certificate may be cached without further validation through the issuer-specified *revocation window*. When the certificate is retrieved (t^1) it is guaranteed to be fresh and unrevoked. After revocation (t^4), the issuer need only include the certificate in the CRL for one revocation window (t^5 to t^7). At t^7, the issuer knows that one of the following two cases has occurred at each verifier caching C_1: either 1) a CRL was acquired within the revocation window, and C_1 was dropped, or 2) the revocation window has expired, and C_1 was dropped. In either case, windowed revocation stipulates that the certificate will no longer be cached by any verifier at the end of the issuer-specified revocation window. Hence, the issuer can discontinue announcing C_1's revocation. After the revocation window has been reached, the issuer may purge the revoked certificate from its internal lists. Unless needed for some other purpose, such as support for non-repudiation, no master list of revoked certificates is required.

When a CRL cannot be obtained, or more recent revocation state than provided by the CRL is required (as in the "critical operations" described above), the certificate must be dropped and re-acquired. As issuers are prohibited from returning revoked certificates, and the retrieval process is freshness and authenticity protected [7], all retrieved certificates are guaranteed to be both fresh and unrevoked. Therefore, real time revocation state can always be determined by the direct acquisition of the certificate.

In [7], we present extensions to X.509 v3 certificate format to support windowed revocation. Also discussed are a number of implementation details including, among others, revocation of the issuer certificate and the certificate acquisition protocol.

B.1 Certificate Cache Management

Specified by the verifier and assigned to each received certificate is a *clean timer* (π) value. The size of the clean timer represents a verifier policy stating maximum latency of revocation state they are willing to accept, and is the direct statement of the desired window of vulnerability. As the needs of the verifier are context dependent, clean timers are set to a value commensurate with the security policies and requirements of each acquired certificate. To simplify exposition and without loss of generality, we assume the clean timer is equal to the CRL publication period in this section. We defer further discussion of the verifier selected clean timer values to Section III-B.

We present the following algorithm used by the verifier to determine the revocation state of a cached certificate. As previously described, each certificate includes fields defining the revocation window (w) and the CRL publication period (p).

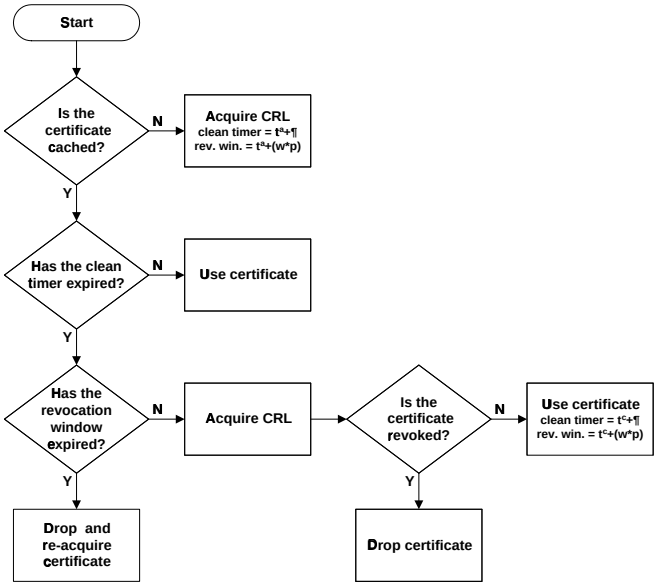

Fig. 2. Verifier cache management algorithm. This algorithm is executed before for each use of a certificate. The time a certificate is acquired is denoted t^a, and the time a CRL is published as t^c. The CRL publication period is denoted p, the revocation window w, and the verifier selected clean timer π.

At the time of retrieval, two timers are associated with each cached certificate: the *clean timer* (π) and the revocation window timer. The clean timer is set as described above. The *revocation window timer* is set to the revocation window (w) times the CRL publication period (p). Both timers are reset after each subsequent acquisition of a CRL. In response to the acquisition of a CRL created at time t^c, the clean timer is reset to $t^c + \pi$, and the revocation window timer is reset to $t^c + wp$. Revoked certificates are dropped from the cache.

As clean timers expire, the associated entries are marked "dirty." Certificates with unexpired clean timers may be used without further verification. In Section III-C, we prove that no revoked certificate can be used beyond the clean timer (and thus the desired window of vulnerability is always preserved).

Under windowed revocation, we use CRLs to re-assert cached certificates' non-revoked status. If the verifier does not hold the most recent CRL from the issuer, it is directly acquired. Once the CRL has been obtained, the certificate clean timer and revocation window timers are reset (if it has not been revoked), or dropped (if it has been revoked). In the normal case, the revocation states of unrevoked certificates will be updated as needed after CRL acquisition.

A certificate is dropped from the cache when its revocation window timer expires. Subsequent use of a dropped certificate requires its re-acquisition from the issuer.

Figure 2 describes the verifier certificate cache management algorithm used when clean timers are longer than or equal to CRL publication times. When clean timers are shorter than CRL publication times, certificates must be dropped at the expiration of clean timers and windowed revocation reduces to implicit revocation.

We now illustrate the certificate cache management process through several examples. In these examples, we state the CRL publication period for the issuer of certificates C_1 and C_2 is equal to 1 (where a CRL is generated at $t, t+1, t+2, \ldots$). The revocation window sizes documented in both C_1 and C_2 are 2 (times the CRL publication period). Between $t+1$ and $t+2$, certificate C_1 is revoked. Between $t+2$ and $t+3$, certificate C_2 is revoked. Figure 3 describes the revocation and subsequent inclusion in CRLs of C_1 and C_2. We assume that all verifiers set clean timers to the CRL publication period (1).

By definition, the CRLs published by the issuer at time $t+2$ and $t+3$ will contain the revocation of certificate C_1. The revocation of certificate C_2 will be returned in the CRLs published at time $t+3$ and $t+4$. The CRL published at time $t+4$ will no longer contain the revocation state of certificate C_1. In Figure 3, the inclusion of a certificate in published CRLs is indicated as shaded boxes. Note that any CRL request will return the most recently published CRL. Thus, the response to a CRL request received between time $t+3$ and $t+4$ will include the CRL published at $t+3$.

Consider a verifier (V_1) whose cache contains both certificates C_1 and C_2. Assume that the verifier received the CRL published at time $t+1$. Thus at time $t+1$, the revocation window timer for certificates C_1 and C_2 are set to $t+3$. We now describe several possible scenarios relating to this example.

If V_1 wishes to use certificate C_1 is between $t+2$ and $t+3$, because the clean timer for C_1 expired at $t+2$, she must acquire the CRL published at time $t+2$. If the CRL is successfully obtained, she will note the revocation of C_1 and and drop it. If the CRL cannot be obtained, V_1 will drop and attempt to re-acquire the certificate.

When the CRLs published at time $t+2$ and $t+3$ cannot be acquired, the verifier is unable to determine the revocation state of both C_1 and C_2. The revocation window timers of C_1 and C_2 would expire at time $t+3$, and the certificates would be dropped from the cache.

Now consider a second verifier (V_2) who retrieves certificate C_2 at time $t+2$. Because he knows at the time of retrieval that C_2 is fresh and unrevoked, he sets the clean timer associated with C_2 to expire at $t+3$ and the revocation window timer to expire at time $t+4$. V_2 may freely use the certificate until $t+3$, after which it must be revalidated via CRL or re-acquisition. If not revalidated at any time before $t+4$, C_2 is dropped.

Note that while the *size* of the revocation window is the same at all verifiers for a given certificate, the *start time* of the revocation window timer itself is not. In each verifier, the revocation window is reset each time the validity of a certificate is asserted.

We address the latencies incurred by the delivery of CRLs by stipulating that clean timers must factor in the propagation delay. The propagation delay is a short period that estimates the maximum time needed for the generation and delivery of the CRL. This value is site dependent, and must be set locally.

C. Push Delivery

The scalability of traditional explicit mechanisms is limited by the requirement that verifiers actively retrieve CRLs. Where available, the use of push delivery may mitigate the costs of CRL acquisition. Using push delivery, each entity holding a cached certificate may passively listen for revocation announcements from the corresponding issuer (or appropriately authenticated

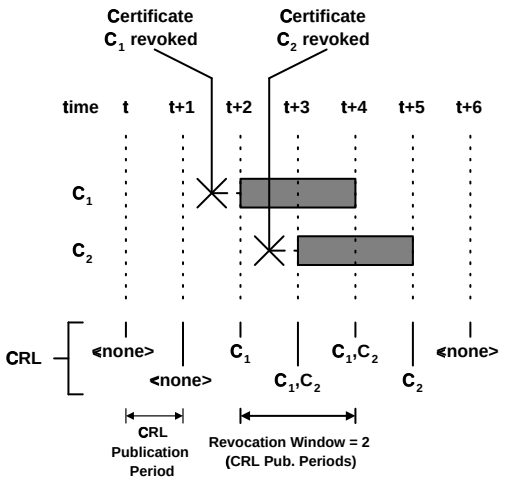

Fig. 3. Example CRL generation - In this example, we show the revocation of certificates C_1 and C_2 and their inclusion in subsequent CRLs.

CRL service). Therefore, verifiers subscribing to the CRL push delivery service can verify cached certificates without incurring the costs of direct CRL acquisition. If a pushed CRL is lost in transit and it is required by a verifier, the verifier may retrieve it from the issuer. If the verifier is unable to retrieve the CRL, the certificate may be revalidated by re-acquiring it. Hence, CRL push delivery may use unreliable delivery protocol, such as IP multicasting [8]. Note that the use of unreliable delivery does not affect the security of CRL delivery (see Section III-A).

Since pushed CRLs are received and processed at the time of their publication, cached certificates not revoked by the last published CRL will never be marked dirty and may continue to be used. In this case, we use CRL publication as a form of cache invalidation message.

While a push mechanism for CRL delivery is mentioned in [9], [10], we are not aware of any existing design that uses the push mechanism with provable correctness. Where IP multicast is not available, several known techniques for providing scalable data distribution may be used to provide push delivery. While push delivery may improve the performance of windowed revocation, windowed revocation does not depend on it to operate correctly.

III. Analysis

In this section we analyze the effectiveness of windowed revocation as a scalable mechanism for the distribution of revocation state.

A. Scalability of Design

Windowed revocation is scalable both in its bandwidth requirements and the size of the supported community. The scalability of windowed revocation is based on its use of the revocation window and may be enhanced by the use of CRL push delivery. By limiting the size of CRLs through the use of the revocation window, we reduce the costs associated with their distribution.

Through certificate caching, we attempt to scale the total number of supportable verifiers. Given our reduced CRL size, we can push deliver CRLs to verifiers. This allows verifiers to passively maintain the validity of their cached certificates without having to independently request information from the issuers. We avoid unnecessary validation by allowing verifiers to postpone the verification of a cached certificate's revocation state until the certificate is to be used. Also, lost CRLs are reliably retrieved only when a certificate verification is needed.

As verifiers passively receive CRLs immediately following publication, the effects of CRL request implosion may be decreased or eliminated. In the normal case, the CRLs will arrive shortly after publication, alleviating the need for their direct acquisition.

Our use of IP multicasting in CRL push delivery minimizes network bandwidth usage by not duplicating data transmission to multiple destinations where their paths overlap. For scalability reasons, IP multicasting uses the unreliable transport protocol, UDP, for data delivery. Our ability to use unreliable transport protocol for push delivery of CRLs rests fundamentally on the use of documented scheduled intervals. A verifier with a cached certificate knows the periodicity at which CRLs are expected. If a CRL is not received at the expected time and a certificate validation is needed, the verifier uses a reliable transport protocol to revalidate the certificate.

An important distinction to note is that our use of unreliable transport protocol in no way affect the security of received CRLs. The security of received CRLs is based on digital signatures, and as such are as secure as the signers' CRL generation process [7].

B. General Guarantee Statement

We bound the time in which a revoked certificate can be used by its associated clean timer. Any certificate which is cached longer than its clean timer is subject to verification explicitly through a fresh CRL, or implicitly by re-acquisition of the certificate from the issuer. The revocation window allows the issuer to control the resources required to process CRLs. Smaller revocation windows reduce the size of CRLs, but require verifiers to re-acquire certificates more frequently.

An advantage of this approach is that an issuer using windowed revocation can mimic traditional key revocation mechanisms. By setting the revocation window equal to the maximum lifetime of any certificate, the CRLs generated will be functionally equivalent to those found in explicit revocation systems. In this way, no cached certificate will ever have its revocation window timer expire before the certificate expiration date. To mimic implicit revocation, windowed revocation issuers simply set the CRL publication period to 0 and never publish CRLs. This forces all certificates to be re-acquired after their clean timers expire.

In [11], Rivest exposes a fundamental limitation of CRLs: verifiers' inability to control the window of vulnerability. With traditional CRLs, a verifier receiving signed content must accept the validity of that content based on revocation information which is only as recent as the latest CRL publication. While this problem exists in explicit revocation, windowed revocation allows verifier control over the window of vulnerability through the direct acquisition of certificates. In acquiring the certificate, the verifier obtains an instantaneous proof of the revocation state

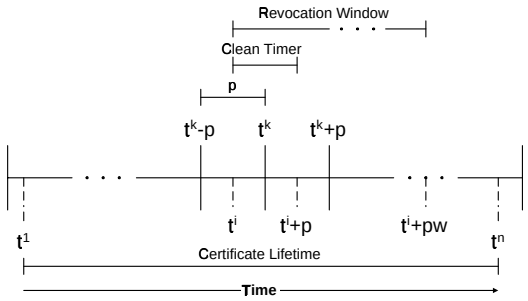

Fig. 4. We show the lifetime of certificate C, which is valid from t^1 to t^n. At time t^i, a verifier retrieves the certificate. In response, the clean and revocation window timers are set to $t^i + \pi$ and $t^i + wp$, respectively, where π is the verifier selected clean timer value, p is the CRL publication period of the issuer, and w is the revocation window. The issuer publishes CRLs at times $\ldots, t^k - p, t^k, t^k + p, \ldots$

of the certificate. Verifiers who wish to retrieve revocation state at rates faster than the CRL publication period can do so by setting a certificate's clean timer to a period smaller than the CRL publication period, and setting the revocation window timer to 0. In this case, the certificate is dropped after the clean timer expires.

To summarize, the guarantee provided by windowed revocation is *exactly* the *general certificate guarantee* proposed by Rivest in [11]:

> This certificate is definitely good from (date-time-1) until (date-time-2). The issuer also expects this certificate to be good until (date-time-3), but a careful acceptor [i.e. verifier] might wish to demand a more recent certificate. This certificate should never be considered as valid after (date-time-3),

where *(date-time-1)* is the time a certificate is retrieved or a CRL is acquired and processed, *(date-time-2)* is *(date-time-1)* plus the CRL publication period, and *(date-time-3)* is the end of the certificate lifetime.

C. Correctness

In this section, we formally prove the bound on the use of revoked certificates under windowed revocation, assuming that the push delivery mechanism described in Section II-C is not implemented.[5] In Figure 4, we describe the lifetime of certificate C. C is valid from time t^1 until its expiration at time t^n. CRLs are generated by the issuer at the publication period p. π is the clean timer value selected by the verifier. The revocation window of C is w. We denote an arbitrary CRL publication time as t^c. At time t^i, C is retrieved and cached by a verifier. At some time t^r, C is revoked. Before presenting the proof, we formally define the two central properties of windowed revocation.

Property 1: *Fresh Certificate Retrieval.* This property ensures that all certificates are fresh and unrevoked at the time of retrieval. More formally, $t^r > t^i$ holds for the retrieval and revocation of any certificate C.

[5]The use of push delivery does not affect the bound on the use of revoked certificate, or on the provable correctness of windowed revocation. We omit its inclusion in this section for simplicity.

Property 2: *Windowed Revocation.* This property ensures that all revoked certificates are included in the CRLs published within the documented revocation window. Formally,

$C \in CRL^j$ for all CRLs published at $t^c + mp$, where $\min(t^c)|t^c > t^r, 0 \leq m \leq w$.

Intuitively, t^c is the CRL publication time immediately following the revocation, i.e. the publication time of the first CRL that contains the revocation. CRL^j denotes the CRL published at time t^j.

Theorem 1: *The length of time any revoked certificate may be used is bounded by the length of the clean timer (π).*[6]

Proof: After retrieval, the initial clean timer for C is set to $t^i + \pi$, and the revocation window timer is set to $t^i + wp$. It is sufficient to show the theorem holds for verifications (and use) of C at time t^δ, for all $t^\delta \geq t^i$.

- Case 1: $t^\delta < t^i + \pi$: The certificate is verified before the initial clean timer expires.

  $t^i \leq t^\delta < t^i + \pi$, [*from case definition*]
  $t^i < t^r$, [*by property 1*]
  $\Rightarrow t^\delta - t^r < \pi$,

  so the theorem holds.
- Case 2: $t^i + \pi \leq t^\delta < t^i + wp$: The certificate is verified after the initial clean timer expires, but before the revocation window expires.

  a) If $\pi < p$, the verifier requires a window of vulnerability tighter than the CRL publication time, so the certificate must be dropped after the clean timer expires. The theorem holds.

  b) If $\pi \geq p$ and C is not marked dirty, then there exists some CRL^j published at time $t^j < t^r$ that was received by the verifier. At t^j, we know C has not been revoked. The clean timer has not expired, so $t^\delta - t^j < \pi$.

  Therefore,

  $t^\delta - t^j < \pi$, [C *is not marked dirty*]
  $t^j < t^r$, [$C \notin CRL^j$]
  $\Rightarrow t^\delta - t^r < \pi$.

  Intuitively, a certificate having an unexpired clean timer means that it has not been longer than π since a statement of the certificate's non-revoked status has been received from the issuer, thus the theorem holds.

  c) If $\pi \geq p$, C is marked dirty, and the most recent CRL^j published at time t^j is retrieved.

  $t^\delta - t^j < p$ [*by definition*]
  $\pi \geq p$ [*from case definition*]
  $\Rightarrow t^\delta - t^j < \pi$,

  The information received in CRL^j is within π of the verification time (t^j). This indicates that the CRL is recent enough to be within the window of vulnerability defined by the clean timer value.

  If $t^r > t^j$, $C \notin CRL^j$, the clean timer is reset to $t^j + \pi$. This case reduces to case 2(b).

  If $t^r \leq t^j$, then it suffices to prove $C \in CRL^j$. By property 2, $C \in CRL^j$ if and only if

  $t^c \leq t^j \leq t^c + wp$,

[6]Note that the bound on the use of revoked keys is actually the clean timer length plus the propagation delay value. For simplicity and without loss of correctness, we omit mention of the propagation delay value.

where t^c is $\min(t^c)|t^c > t^r$, the CRL publication on or immediately following t^r. From this, we can conclude that:

$\Rightarrow t^c \leq t^j$,
$t^i < t^r$, [*by property 1*]
$t^r \leq t^c$, [*by property 2*]
$\Rightarrow t^i < t^c$,
$\Rightarrow t^i + wp < t^c + wp$,
$t^j < t^i + wp$, [*from case definition*]
$\Rightarrow t^j < t^c + wp$.

Hence:

$\Rightarrow t^c \leq t^j < t^c + wp$,

and

$\Rightarrow C \in CRL^j$.

So the theorem holds. A similar argument holds for certificates whose revocation window is reset in response to a received CRL.

• Case 3: $t^\delta \geq t^i + wp$: The revocation window timer expired, so the certificate is dropped. Thus, the theorem holds. (see Case 2(c) for a description of reset revocation window timers.) □

IV. Performance Analysis

In this section, we assess the worst case and simulated performance of windowed revocation.

A. Worst Case Performance

This section investigates the resource costs of windowed revocation in the worst case. In the following text, we estimate the resource consumption through two metrics: the number of signatures generated and verified and the amount of network bandwidth generated. The high cost of public key cryptographic operations makes signature generation the dominating factor in CPU consumption at issuers (the generation of a RSA digital signature using a 1024 bit key requires .97 seconds on a Sparc II [12]); similarly, signature verification dominates verifier host CPU consumption. The number of certificates and CRLs acquired by verifiers determines the bandwidth consumption.

The worst case CPU usage scenario for windowed revocation is when clean timers are smaller than CRL publication times or when all cached certificates must be revalidated outside their revocation windows. Outside its revocation window, a certificate revalidation requires re-acquisition of the certificate. Since windowed revocation stipulates that issuer can only return fresh and unrevoked certificates, each certificate acquisition requires an expensive cryptographic operation at the issuer. Hence, if all certificate revalidation occurs outside their revocation window, windowed revocation degenerates into implicit revocation. For a given workload and window of vulnerability, the worst case CPU requirement of window revocation is thus the same as that of implicit revocation.

The worst case bandwidth usage for windowed revocation, assuming no push delivery, is when all cached certificates must be revalidated after their clean timers expired, but before their revocation windows expire—for example if the revocation window size is set to infinity.[7] In this scenario, each certificate access could potentially induce a CRL retrieval, causing bandwidth consumption to grow linearly with the number of certificates revalidated. Hence, in this worst case bandwidth consumption scenario, windowed revocation behaves similar to explicit revocation. However, because the CRLs in explicit revocation are larger, the total bandwidth consumed in traditional explicit revocation will be an upper bound on that of windowed revocation.

This worst case analysis further illustrates that implicit and explicit revocation are special cases of windowed revocation.

[7] This analysis assumes the bandwidth cost of CRL retrieval is greater than the cost of certificate acquisition. The actual total cost of CRL retrieval is dependent on revocation window size and revocation rate.

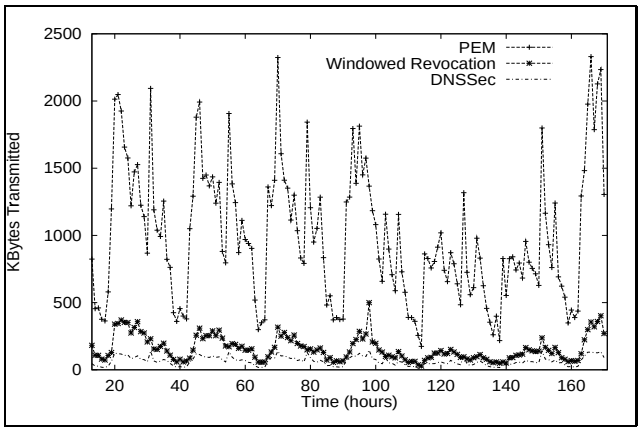

Fig. 5. Total bandwidth usage calculated from estimation of certificate and CRL acquisitions in DNS trace driven simulation. In these experiments, no CRL push mechanism is simulated.

B. Empirical Analysis

In this section we identify several results of a trace based performance study of windowed revocation. The lack of deployed certificate services available for study precludes us from collecting real certificate request traces, hence we use DNS traffic to model certificate usage. We argue that as DNS requests are most often used as precursors to session initiations [13], DNS access has similar characteristics to certificate usage. While this model of certificate use may not be perfect, it does demonstrate how, under a particular workload, windowed revocation may be tuned to meet the performance needs of the target community. A detailed account of this study can be found in [14].

Figure 5 describes the bandwidth consumption of a simulated issuer in windowed revocation, PEM (explicit revocation), and DNSsec (implicit revocation) over a one week period. In this study, we modeled our departmental DNS server as an issuer, each DNS record as a certificate, and estimated one revocation per 9 hours. The TTL, CRL publication period, and clean timer values were set to 12 hours in all experiments. The revocation window was set to 4 (48 hours). While not factoring significantly in our study, the lifetime of each certificate was set to 1 year.

An interesting observation that can be made from Figure 5 is that the performance of windowed revocation lies somewhere between the performance of implicit and explicit mechanisms. This feature can be explained by viewing explicit and implicit revocation as boundary cases of windowed revocation. As the revocation window approaches 0, the bandwidth consumption approaches that of implicit revocation. Conversely,

as the revocation window approaches the certificate lifetime, bandwidth use approaches that found in explicit mechanisms. Thus, through the selection of the revocation window, an issuer can select the desired performance on the continuum between implicit and explicit revocation. Because the revocation window used in our experiments is significantly closer to 0 than the certificate lifetime (1 year), its bandwidth usage more closely models that of the implicit revocation.

Although not shown, the CPU usage of windowed revocation has a similar property. As the revocation window approaches 0 or the certificate lifetime, the CPU usage approaches that seen by implicit or explicit revocation, respectively. This further demonstrates the flexibility of windowed revocation in managing resource tradeoffs. Bandwidth consumption grows and CPU usage decreases with revocation window size. Thus, tradeoffs between the utilization of CPU and network resources may be managed through the assignment of revocation windows.

V. Related Work

The Privacy Enhanced Mail [3], [4] architecture (PEM) stipulates that all revoked certificates in each domain be included in periodic CRLs. Due to the long lifetimes of certificates, the size of these lists made CRL distribution difficult. Several approaches to reducing the size of CRLs have been proposed [15], [16], many of which have been included in the IETF Public Key Infrastructure Working Group (PKIX) draft standards.

Issuers supporting delta CRLs [16] periodically publish a traditional CRL, called a base CRL, and, more frequently, delta CRLs that contain only revocation information generated since the last base CRL. Unlike CRLs in windowed revocation, delta CRLs continually increase in size between base CRLs. Furthermore, verifiers are required to acquire, validate, and cache the potentially large base CRLs. Note that windowed revocation may be used to augment delta CRL approaches.

In systems that use freshness CRLs [15], delta CRLs are generated at multiple rates. Verifiers retrieve CRLs generated at a rate commensurate with their security requirements. In windowed revocation, each verifier may acquire revocation state at any rate by dropping and re-acquiring certificates as needed. CRLs in windowed revocation may also benefit from multiple publication rates.

In an effort to reduce the costs of CRL processing, some systems present revocation information in authenticated dictionaries [17], [18], [19]. Using authenticated dictionaries, verifiers need not retrieve the entire CRL, but request only enough information to validate the certificate. A limitation of authenticated dictionaries is their requirement that each certificate be validated individually. Thus, the aggregation afforded by CRL based systems is not supported.

The Online Certificate Status Protocol (OCSP) [20] defines an implicit revocation mechanism to be used in conjunction with the explicit mechanism in PKIX PKIs. It does not attempt to reduce the resource consumption of the existing explicit mechanism.

The OCSP protocol embodies an implicit approach to retrieving revocation state. In this approach, revocation state is obtained through freshness and authenticity protected queries rather than by CRL. One extension to this online approach includes the establishment of symmetric session keys over which verifiers may securely obtain revocation state. The use of session keys allows the costs of session key establishment to be amortized over a number of certificate verifications. Because the responses by the issuer are generated in real time, the window of vulnerability is effectively reduced to zero. However, the costs of session key establishment and session key message processing places additional load on the issuers. The state held by the issuer will grown linearly with the number of verifiers. A better understanding of the workloads observed by issuers will indicate environments in which this approach will be useful.

In [21], Perlman and Kaufman introduce an approach that manages CRLs through the use of *blacklists*. In this approach, the issuer monitors CRL size. When some (time or CRL size) threshold is reached, **all** certificates are invalidated and reissued, whether they have been revoked or not. The time of re-issuance is indicated in each blacklist CRL through a *start date*. Any certificate issued before the start date is deemed invalid. Thus after each re-issuance, verifiers must drop all cache entries and acquire newly issued certificates. A limitation of this approach is that the cost of re-issuance grows linearly with the number of certificates. In issuers containing thousands of certificates or more, the cost of re-issuance may be prohibitive. Similarly, each verifier must purge all certificates from their cache after re-issuance. A simpler alternative to blacklisting is to create certificates with shorter lifetime. This alternative suffers similar shortcomings as the use of blacklists.

There is a direct parallel between global certificate and namespace management. In recognition of this fact, the authors of DNSSec [5], [6] designed an architecture for certificate distribution and revocation using the existing DNS service. As with DNS, certificates are retrieved from the source domain and held for a short time. Later validation is performed by re-acquisition of the certificate. As DNSSec requires each certificate to be digitally signed once per (short) configurable period, and that each response to a request with transaction authenticity enabled be digitally signed, it is unclear how well it will scale in large networks.

The Pretty Good Privacy (PGP) [22] system provides a suite of tools for generating, managing, and revoking certificates within a local environment. PGP does not specify certificate distribution or revocation protocols.

The Simple Distributed Security Infrastructure (SDSI) [23], [24] and the closely related Simple Public Key Infrastructure (SPKI) [25] systems provide a language and toolkit under which user and group certificates can be created, distributed, and revoked. SDSI requires certificate owners to document a *reconfirmation* TTL. When this TTL expires, the validity of the certificate is required to be re-established. This is functionally equivalent to the implicit revocation mechanism found in DNSSec.

VI. Conclusions and Future Work

In this paper, we presented a novel approach to certificate revocation. *Windowed revocation* attempts to limit the size of CRLs by announcing revocation only for a documented period. The time a certificate can be held by a verifier is bounded by the announcement period, called the *revocation window*. Thus, all

certificates will be verified either (1) explicitly by CRL or, (2) implicitly by retrieval. Through manipulation of the revocation window, issuers may influence CRL sizes and the frequency with which certificates are retrieved. We allow an end-to-end push mechanism for CRL delivery using multicast. With push delivery, the costs and latencies associated with verifier initiated CRL retrieval can be alleviated.

Certificates are provably unrevoked within a verifier specified window of vulnerability. Because the window of vulnerability is under the control of the verifier, the security requirements of each certificate operation may be independently supported. Thus, verifiers may achieve the exact semantics required for each certificate validation at a minimal cost.

The performance of windowed revocation is bounded by existing implicit and explicit mechanisms. In the worst case, windowed revocation will consume no more network bandwidth than traditional CRL based solutions, and no more CPU resources than traditional implicit mechanisms. Further, the tradeoffs between resource consumption and security may be managed through the windowed revocation protocol parameters.

We are in the final stages of constructing a reference implementation for windowed revocation. We are integrating the windowed revocation services with SSLeay [26], a widely-used session layer providing secure point to point communication.

VII. Acknowledgements

We would like to Avi Rubin, Rebecca Wright, Atul Prakash, and the anonymous reviewers for their many helpful comments and insights. We would also like thank Eric Cronin for his work on the Windowed Revocation reference application.

References

[1] B. Fox and B. LaMacchia, "Certificate Revocation: Mechanics and Meaning," in *Financial Cryptography*, Rafael Hirschfeld, Ed., Anguilla, British West Indies, February 1998, vol. 1465, pp. 158–164, Springer.

[2] D. Chadwick, *Understanding X.500 : The Directory*, Chapman & Hall, London, 1994.

[3] S. Kent, "Internet Privacy Enhanced Mail," *Communications of the ACM*, vol. 36, no. 8, pp. 48–60, August 1993.

[4] S. Kent, "RFC 1422, Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management," *RFC 1422, Internet Engineering Task Force*, February 1993.

[5] D. Eastlake and C. Kaufman, "RFC 2065, Domain Name System Security Extensions," *RFC 2065, Internet Engineering Task Force*, January 1999.

[6] J. Galvin, "Public Key Distribution with Secure DNS," in *Proceedings of the 6th USENIX Security Symposium*, July 1996, pp. 161–170.

[7] P. McDaniel and S. Jamin, "Windowed Key Revocation in Public Key Infrastructure," Tech. Rep. CSE-TR-376-98, EECS, University of Michigan, Ann Arbor, 1998.

[8] S.E. Deering and D.R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 85–110, May 1990.

[9] Produced by the MITRE Corporation for NIST, "Public Key Infrastructure Study, Final Report," http://csrc.nist.gov/pki/documents/welcome.html, April 1994.

[10] W. Ford and M. Baum, *Secure Electronic Commerce : Building the Infrastructure for Digital Signatures and Encryption*, Prentice Hall, Englewood Cliffs, New Jersey, 1997.

[11] Ronald L. Rivest, "Can we eliminate certificate revocation lists?," in *Financial Cryptography*, Rafael Hirschfeld, Ed., Anguilla, British West Indies, February 1998, vol. 1465, pp. 178–183, Springer.

[12] Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore, second edition, 1996.

[13] Peter B. Danzig, Katia Obraczka, and Anant Kumar, "An Analysis of Wide-Area Name Server Traffic," in *Proceedings of ACM SIGCOMM 92'*, 1992.

[14] P. McDaniel and S. Jamin, "Windowed Certificate Revocation," Tech. Rep. CSE-TR-413-99, EECS, University of Michigan, Ann Arbor, November 1999.

[15] C. Adams and R. Zuccherato, "A General, Flexible Approach to Certificate Revocation," June 1998, http://www.entrust.com/resources/whitepapers.htm.

[16] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile (draft-ietf-pkix-ipki-part1-08.txt) (Draft)," *Internet Engineering Task Force*, June 1998.

[17] M. Naor and K. Nassim, "Certificate Revocation and Certificate Update," in *Proceedings of the 7th USENIX Security Symposium*, January 1998, pp. 217–228.

[18] P. Kocher, "On Certificate Revocation and Validation," in *Financial Cryptography*, Rafael Hirschfeld, Ed., Anguilla, British West Indies, February 1998, vol. 1465, pp. 172–177, Springer.

[19] S. Micali, "Efficient certificate revocation," Tech. Rep. Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, 1996.

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

[21] R. Perlman and C. Kaufman, "Method of Issuance and Revocation of Certificates of Authenticity Used in Public Key Networks and Other Systems," United States Patent 5,261,002, November 1993, Primary Examiner: B. Gregory.

[22] P. Zimmermann, "PGP user's guide," Distributed by the Massachusetts Institute of Technology, May 1994.

[23] R. Rivest and B. Lampson, "SDSI A Simple Distributed Security Infrastructure," http://theory.lcs.mit.edu/ rivest/sdsi11.html, October 1996.

[24] M. Blaze, J. Feigenbaum, and Jack Lacy, ""Decentralized Trust Management"," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, 1996, pp. 164–173, Los Alamitos.

[25] C. Ellison, "SPKI Requirements," *Internet Engineering Task Force*, May 1999, (draft) `draft-ietf-spki-cert-req-03.txt`.

[26] T. Hudson and E. Young, "SSLeay and SSLapps FAQ," September 1998, http://psych.psy.uq.oz.au/ ftp/Crypto/.