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*

November 1, 1999

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 a trace-based performance study illustrating the scalability and general applicability of windowed revocation.

1 Introduction

Over the past several years, the use of the Internet has grown immensely. Applications on the Internet allow geographically distant users to communicate, leading to social, educational, and commercial interactions that were previously impossible. Unfortunately, because of the openness of the Internet, the form and content of these interactions are vulnerable to attack. Limiting these vulnerabilities is essential to the future success of these applications and the continued growth of the Internet.

A popular approach to securing communication over large networks is to use public keys. Researchers and standards bodies have argued at great length over possible architectures for providing an authentication service under which public key certificates can be securely distributed. A central point of contention in these discussions is the mechanisms over which public keys are revoked.

A *certificate* is a data structure that defines an association between an entity (the *principal*) and a public key. A trusted authority, called a *Certificate Authority* (CA), states its belief in the validity of the association by digitally signing the certificate.[1] Certificate revocation is the mechanism under which a CA can revoke the association before its documented expiration. The CA 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 [FL98]. As stated by the CA, 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.

In this paper we investigate *windowed revocation*, a novel approach to certificate revocation within a global certificate distribution service, called a *Public Key Infrastructure* (PKI). The central design objectives of windowed revocation are:

[1] Several Public Key Infrastructures employ models not based on trusted third party (CA) certificate distribution (see Section 4). Although windowed revocation may be applied to both CA and non-CA environments, for simplicity we only describe windowed revocation within CA based architectures.

1. **Correctness** - All entities within the PKI 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** - Windowed revocation must be able to support guarantees consistent with existing security policies and requirements.

As with many security solutions, certification revocation mechanisms are subject to the fundamental tradeoff between security and scalability. Solutions with strict security objectives require more resources than systems with more relaxed security objectives. Thus, security requirements have a direct influence on scalability. Our proposed architecture provides a flexible framework for managing this tradeoff by incorporating 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. *Push delivery:* With limited revocation announcement size, we can contemplate the active delivery of this information to verifiers. This reduces the load on the CAs by curtailing the number of verifier initiated retrievals.

3. *Certificate caching:* A cached certificate may be used until it expires, is revoked, or the issuer 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.

4. *Scheduled Announcement:* By stipulating that CAs generate revocation announcements at a documented schedule, we allow verifiers to detect lost announcements.

5. *Multicast delivery:* Given verifiers' ability to detect missing revocation announcements, we can use unreliable transport protocol without sacrificing the security of certificate revocation. This allows the use of IP multicasting, where available, to further reduce the bandwidth requirements of the revocation mechanism.

6. *Lazy verification:* Verification of a cached certificate's revocation state is postponed until the certificate is used. By deferring retrieval of lost CRLs, we reduce the load on CAs; by deferring certificate verification and CRL processing, we to reduce the load on verifier hosts.

In this paper we describe windowed revocation and present results of a series of simulation experiments designed to assess windowed revocation's viability as revocation mechanism within PKI architectures. In the next section we define and illustrate windowed revocation. Section 3 presents performance characteristics of windowed revocation observed from simulation experiments. Section 4 gives a brief overview of work related to PKI systems and certificate revocation. We conclude in Section 5. We prove the correctness of windowed revocation in Appendix A.

2 Architecture

In this section, we develop a working definition of a Public Key Infrastructure and present *windowed revocation* as a mechanism for providing a provable bound on the use of revoked certificates.

2.1 Public Key Infrastructure

A common approach in designing Public Key Infrastructures (PKI) is the definition of a *Certification Hierarchy*. The certification hierarchy is a collection of certificate authenticating bodies called *Certification Authorities* (CA) organized into one or more trees. Fig. 1 presents an example Privacy Enhanced Mail [Ken93] hierarchy. The leaves of each tree represent certificates for hosts, users, or services. The topology of the tree infers a hierarchy of authentication, where parent CAs assert the validity for the certificates of all its immediate children.

Each CA is responsible for the registration, distribution, and potential revocation of the certificates

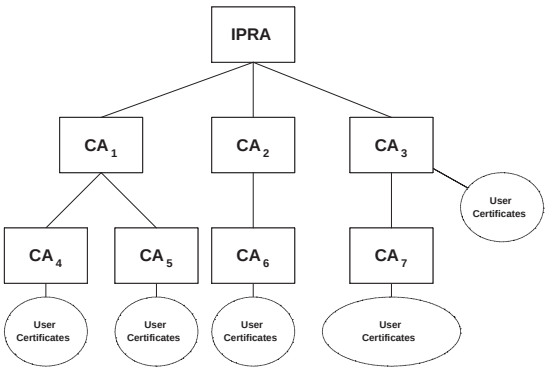

Figure 1: PEM Hierarchy

for its immediate children in the hierarchy. Registration is the process whereby, after the child provides credentials, the CA digitally signs the child's certificate. Any entity wishing to validate the authenticity of a registered certificate need only verify the CA's digital signature.

There exists a bootstrapping problem with the certificate hierarchy approach. When authenticating a certificate, the verifier must also authenticate the certificate signer's certificate. The channel over which the signer's certificate is received may not be known or trusted by the verifier, so the signer's certificate must also be authenticated. Logically, it would appear that the certificate authentication process does not terminate.

A widely accepted compromise used to address this problem is the integration of *trusted points* into verifier software. During certificate verification, the verifier recursively traverses a logical path (called the *certification path*) from the certificate to some *trusted point*. All certificates along the path are verified as described above, save the trusted point's certificate. A certificate for a trusted point authority is typically installed with the verifier software and manually updated as needed. PEM requires all verifiers trust the root entity, called an Internet Policy Registration Authority (IPRA).

As the IPRA CA is the trusted point for all verifiers in PEM, the certification path begins with the IPRA and traverses the tree toward the leaf (certificate). Thus, certification paths are known *a priori* by the location of the certificate within the tree. The authentication of a certificate requires the acquisition of all certificates and associated revocation state information within the certification path.

The Secure DNS (DNSSec) [Gal96, EK99] system leverages the vast installed base of the Domain Name System [Moc87] to support certificate distribution. DNS name servers perform CA functions, and the root server (.) is the trusted point for all verifiers. Certification paths mirror DNS name resolution, where the path is constructed from the root to the certificate holder.

Many of the proposed PKI architectures [MJ98a, CY97] define a hierarchy similar to the PEM architecture described above, but differ in the trusted points and mechanism for certificate revocation. For simplicity, throughout our description of windowed revocation we assume a singly rooted hierarchy. However, windowed revocation is in no way dependent on a single trusted root hierarchy.

2.2 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. The trade-off between security and scalability we alluded to in Section 1 is formally expressed in what we call the *window of vulnerability*. The window of vulnerability describes the maximum time that any verifier may unknowingly use a revoked certificate. Intuitively, the window of vulnerability provides the granularity of revocation notification, and thus the security afforded by a revocation mechanism.

We recognize two fundamental approaches used to distribute revocation state: explicit and implicit. In PKI architectures that employ explicit revocation, each CA explicitly states which certificates are revoked, and indirectly which are not revoked. In X.500 [Cha94] based systems, each CA 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 2.4. The canonical CRL based PKI is the

[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.

Privacy Enhanced Mail (PEM) [Ken93] 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 4.

In PKI architectures that employ implicit revocation, the revocation state is implicitly stated in a verifier's ability to retrieve the certificate. Any certificate retrieved from the issuing CA 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) [Gal96, EK99] 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 PKI may be subject to a number of masquerading and replay attacks. Providing these guarantees for each certificate retrieval may limit the scalability of the PKI.

A central parameter to PKIs employing implicit revocation is the length of the certificate TTL. PKI administrators 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 CA, or both.

2.3 Windowed Revocation

In windowed revocation, we use explicit notification as the primary revocation mechanism. CRLs are generated per a CA-specified schedule documented in the associated certificates. Revoked certificates are included in scheduled CRLs for a period equal to their *revocation window*. The size of a certificate's revocation window is specified by the CA and documented in the certificate. The revocation window limits the length of time a certificate may be cached without further validation via a more recent CRL. 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 an associated CRL during the revocation window, it must drop the certificate from its cache. A verifier acquires a CRL either through active retrieval from the CA or by passively receiving one pushed by the CA.

The scalability of traditional explicit PKI architectures is limited by the requirement that verifiers actively retrieve CRLs. Windowed revocation mitigates the costs of CRL delivery by using a push mechanism, where available. Each entity holding a cached certificate may passively listen for revocation announcements from the corresponding CA. Therefore, verifiers subscribing to the CRL push delivery

[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 [Gal96] 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 [EK99] 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.

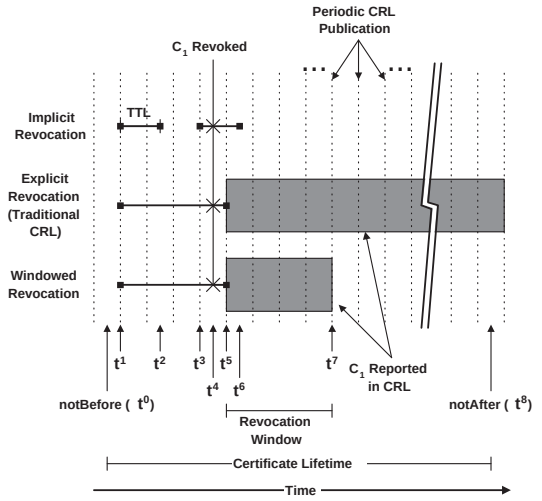

Figure 2: Implicit, explicit, and windowed revocation in PKI architectures.

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 CA (or refresh the certificate by re-acquiring it). Hence CRL push delivery may use unreliable transport protocol, such as IP multicasting. Note that the use of unreliable transport protocol does not affect the security of CRL delivery (see Section 2.4).

We illustrate implicit, explicit, and windowed revocation in Figure 2. 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 CA 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 through the *revocation window*. When the certificate is retrieved (t^1) it is guaranteed to be fresh and unrevoked. After revocation (t^4), the CA need only include the certificate in the CRL for one revocation window (t^5 to t^7). At t^7, the CA knows that one of the following two cases has occurred at each host 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 host at the end of the revocation window, hence the CA can discontinue announcing the revocation. After the revocation window has been reached, the CA 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. Similar to explicit revocation, the window of vulnerability in windowed revocation is equal to the periodicity of CRL publication (see Appendix A for a correctness proof).

When the CRL associated with a certificate cannot be obtained, the certificate must be re-acquired. As CAs are prohibited from returning revoked certificates, and the retrieval process is freshness and authenticity protected [MJ98b], all retrieved certificates are guaranteed to be both fresh and unrevoked. Thus, a byproduct of the certificate acquisition protocol is an instantaneous proof of the revocation state of the certificate. Therefore, if a recent CRL cannot be obtained, the revocation state can be determined by the direct re-acquisition of the certificate.

By providing low cost delivery of CRLs in the average case (with multicast CRL delivery) we avoid the vast amount of active CRL retrievals normally associated with traditional PKI architectures (see Section 3). In the aberrant case, where the most recent CRL has not been received, we provide a means of

recovery through direct retrieval.

In reference [MJ98b], we present extensions to X.509 v3 certificate format to support windowed revocation. In the same reference we also discuss potential extensions to windowed revocation, such as the use of *freshness CRLs*, and our handling of implementation issues such as relationship between a CA and its directory, revocation of the trusted point certificate, secure retrieval of certificates, providing push delivery where IP multicasting is not available, etc.

2.3.1 Certificate Cache Management

We present the following algorithm used by the verifier to determine the revocation state of a cached certificate. In the following text, a distinction is made between the *last published CRL* and the *last acquired CRL*. The last published CRL is the last CRL generated by the CA previous to the verification of the certificate. The last acquired CRL is the last CRL acquired by the verifier.

1. If the difference between the current time and the time the certificate was acquired is less than the CRL publication period, the certificate may be used.
2. If the last published CRL has been acquired from the CA and the certificate has not been revoked, it can continue to be used.
3. If the last published CRL has not been acquired:
   (a) If the difference between the current time and the last acquired CRL is less than the revocation window, the last published CRL is retrieved. Once retrieved, the CRL is used to determine the revocation state of the certificate.
   (b) If the difference between the current time and the last acquired CRL is greater than the revocation window, the certificate is dropped and must be re-acquired. The expiration of a revocation window indicates that revocation announcements for the associated certificate may have been missed.
   (c) If the last published CRL cannot be retrieved, the certificate is dropped from the cache, and must be re-acquired from the CA.

At the time of retrieval, two timers are associated with each cached certificate: the *clean timer* (π) and the revocation window timer. The *revocation window timer* is set to the revocation window (w) times the CRL publication period (p). If we denote the time of CRL publication as t^{CRL}, the clean timer associated with each un-revoked certificate, after acquisition of the new CRL, is reset to $t^{CRL} + \pi$, and the revocation window timer is reset to $t^{CRL} + wp$. Revoked certificates are removed from the cache.

The clean timer is set by a verifier to a value commensurate with its security policies and requirements. Without loss of generality, in this section we assume clean timer equal to the CRL publication period. See Section 2.5 for further discussion on verifier selection of clean timer sizes.

As clean timers expire, the associated entries are marked "dirty." Certificates with unexpired clean timers may be used without further verification. Because the certificate acquisition process provides an instantaneous proof of its non-revoked status, certificates with unexpired clean timers are provably within their windows of vulnerability.

After the initial clean timer expires, we can do one of two things: either (1) use CRLs to re-assert certificates' non-revoked status, or (2) perform lazy verification and revalidate a dirty certificate only when it is needed again. In the former case, since CRLs are acquired 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. In the latter case, if the certificate is to be used within its revocation window, the last published CRL will be consulted for its revocation status; otherwise, a certificate with expired revocation window timer will be automatically dropped from the cache and must be re-acquired if it is to be used again. If a certificate is to be validated within its revocation window but the last published CRL cannot be acquired, the certificate must also be dropped and re-acquired. Given the high cost of signature verification, we opted for the latter case in our design (see Section 3.7 for performance data). For the same reason, when a CRL

is consulted to validate a certificate, all cached certificates associated with the CRL are revalidated or revoked at the same time.

We now illustrate the certificate cache management process with an example. In this example, the CRL publication period for the CA associated with 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.

By definition, the CRLs published by the CA at time $t+2$ and $t+3$ will contain the revocation of certificate C_1. The revocation of certificate C_2 will be included 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 an end-user host (H_1) whose cache contains both certificates C_1 and C_2. Assume that the host received the CRL published at time $t+1$. Thus at time $t+1$, the host set the revocation window timer for both C_1 and C_2 to $t+3$. We now describe several possible scenarios relating to this example.

If certificate C_1 is accessed by an end-user between $t+2$ and $t+3$, the host must acquire the CRL published at time $t+2$. If this process fails, the host will drop and attempt to re-acquire the certificate.

In the case when both CRLs at time $t+2$ and $t+3$ cannot be acquired, the host is unable to determine the revocation state of either C_1 or C_2. The revocation window timers of both certificates expire at time $t+3$, and the host will remove both certificates from its cache.

Now consider a second host (H_2) who retrieves certificate C_2 at time $t+2$. It knows at the time of retrieval that C_2 is fresh and unrevoked, so it sets the clean timer associated C_2 to expire at $t+3$ and the revocation window timer to expire at time $t+4$. The certificate is handled as in the previous case, with the exception of the different timer expirations.

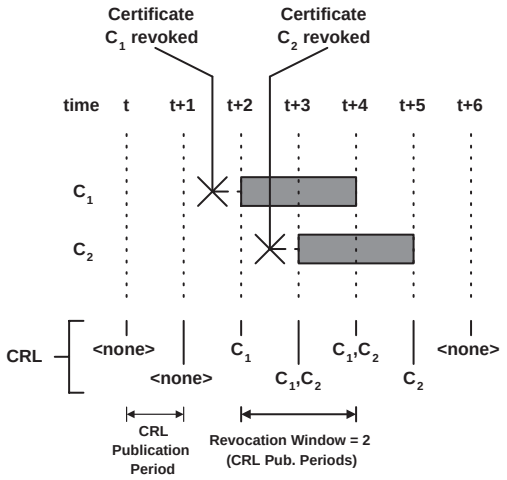

Figure 3: Example CRL generation - In this example, we show the revocation of certificates C_1 and C_2 and their inclusion in subsequent CRLs.

Note that while the *size* of the revocation window is the same in all hosts for a given certificate, the *start time* of the revocation window timer itself is not. In each host, 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 by the local network administrator.

2.4 Scalability of Design

Windowed revocation is scalable both in its bandwidth requirements and the size of the supported community. As indicated throughout this paper, the scalability of windowed revocation is based on its use of the revocation window and 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 CAs.

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. While a push mechanism for CRL delivery is mentioned in [Pro94, FB97], we are not aware of any existing design that uses the push mechanism with provable correctness.

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 [DC90]. 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 signature, and as such are as secure as the signers' CRL generation process [MJ98b].

2.5 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 from the CA. The revocation window allows the CA to control the resources required to process CRLs. Smaller revocation windows reduce the size of CRLs, but require hosts to re-acquire certificates more frequently.

An advantage of this approach is that a CA 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 CAs 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 [Riv98], 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 PKIs with strictly 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 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 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 [Riv98]:

> *"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.

2.6 Correctness

We prove the correctness of windowed revocation in Appendix A. We prove that *the length of time any revoked certificate may be used is bounded by the length of the verifier's clean timer*. Hence a verifier can control its window of vulnerability without loss of correctness. Each verifier may have differing demands, all of which may be met safely.

3 Performance Evaluation

In this section we present results from several trace-based simulations. We evaluate the performance of windowed revocation and compare it against those of both explicit (PEM) and implicit (DNSSec) revocation mechanisms. To understand the behavior and benefits of revocation window independently from those of multicast push delivery, our simulations of windowed revocation, except those in Sections 3.6 and 3.7, do not implement push delivery; instead, verifier hosts actively, and reliably, retrieve CRLs from CAs if a cached certificate is to be revalidated within its revocation window.

As indicated in Section 2.2, the limiting factor in the scalability of any revocation mechanism is the level of its resource consumption. In the following analysis, we estimate the resource consumption of a revocation mechanism by two metrics: the number of signatures generated and verified and the amount of network bandwidth generated, for a given workload. The high cost of public key cryptographic operations makes signature generation the dominating factor in CPU consumption at CAs (the generation of a RSA digital signature using a 1024 bit key requires .97 seconds on a Sparc II [Sch96]); similarly, signature verification dominates verifier host CPU consumption.[5] The number of certificates and CRLs delivered is used to compute bandwidth consumption.

Before presenting quantitative performance data, we first present the worst-case performance analysis of windowed revocation in the next section. We found that CPU consumption of windowed revocation is upper bounded by implicit revocation and bandwidth consumption upper bounded by explicit revocation. In Section 3.3, we present results from trace-driven simulations confirming these findings in the average case. The effect windowed revocation has on certificate caching is presented in Section 3.4. Our claim that explicit and implicit revocations are special cases of windowed revocation (see Section 2.3) is graphically illustrated in Section 3.5, where we experimented with various settings of the protocol parameters.

Finally, we turn our attention to the benefits of multicast push delivery and lazy verification in Sections 3.6 and 3.7.

3.1 Worst Case Performance

The worst-case CPU usage scenario for windowed revocation is 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 CA can only return fresh and unrevoked certificates, each certificate acquisition requires an expensive cryptographic operation at the CA. Hence if all certificate revalidation occurs outside their revocation window, windowed revocation degenerates into implicit revocation. For a given workload, 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.[6] 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, without push delivery, windowed revocation behaves similar to explicit revocation. However, because the CRLs in explicit revocation are larger, the total bandwidth consumed in traditional explicit revoca-

[5] The costs at the CA for providing freshness and authenticity guarantees may be amortized over a number of certificate requests [EK99]. These approaches are similarly applicable to windowed revocation [MJ98b] and can be factored out in our study.

[6] 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.

tion will be an upper bound on that of windowed revocation.

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

3.2 Simulation Setup

Following the motivation behind the design of DNSSec, we model our simulated PKI on a DNS (Domain Name Service) hierarchy. To drive the simulations, we collected a trace of DNS name resolution requests and used them to model certificate requests. The lack of deployed PKI systems available for study precludes us from collecting real certificate request traces. We argue that DNS likely has the usage characteristics that PKI architectures will encounter. Similar to certificate requests, DNS requests are most often used as precursors to session initiations [DOK92].

We collected 2,857,654 DNS requests representing 14,999 name servers and 33,989 hosts. The data collection was done on our departmental primary name server over a one week period, between 1:25pm, Monday, October 26, 1998 and 3:02pm, Monday, November 2, 1998.

In our simulation, each timestamped DNS request is interpreted as a certificate request. The model PKI contains 14,999 CAs and 33,989 hosts. The modeled entities retrieve certificates and CRLs, subject to the timestamps, protocol, and architecture parameters.

The trace data used to drive the simulations described in this section contain all the DNS traffic within our local network environment. The trace includes a complete recording of the departmental nameserver and local host traffic, but contains only partial data for external nameservers and DNS clients. Because the trace data does not contain all the DNS name lookups for external nameservers and clients, we use it to model only the PKI traffic of a local environment. Thus, we present performance data for a departmental CA modeled from the local nameserver, and 2000 verifier hosts modeled from the local DNS clients. While the environment modeled is limited in scope, it is sufficient for the purposes of understanding the salient features of windowed revocation, its performance relative to existing approaches, and to demonstrate its design flexibility.

The performance data presented throughout this section was generated as follows. The cited bandwidth statistics represent the total number of bytes transmitted over the network interface of a CA modeled from the local departmental nameserver. The number of bytes is calculated from the number of certificates and CRLs sent by the modeled CA. In those instances where certificate retrieval must be secure (e.g. in windowed revocation and implicit mechanisms), each acquisition requires one signature generation by the CA. Recall that in PEM, because the revocation state of certificates is only asserted by CRL, they need not be securely retrieved. Additionally, each CRL publication requires the generation of a digital signature. Therefore, the reported number of signatures generated is calculated from the number of certificates securely acquired and the number of CRLs published. Verifier related performance data is calculated by averaging hourly samples taken at each of the 2000 verifier hosts modeled from the local DNS clients.

The implementation of PEM in our simulator models the DNS root as the IPRA and each name server, a CA. The CRL publication period is set to 12 hours. Our simulation of DNSSec implements only *transaction authenticity* mode, and we assume that zone signatures do not expire during the simulation [EK99]. DNSSec servers do not allow recursive requests. To ease comparison of simulation results, we set the TTL of all simulated certificates in the implicit scheme to 12 hours. Unless otherwise noted, the CRL publication period under windowed revocation is also set to 12 hours and the revocation windows of all simulated certificates are set to 4. The trusted point for all simulations is the root CA, and all hosts are assumed to have infinite certificate and CRL cache sizes (the cache size never goes beyond 20 MB in all cases).

UNIX password change data collected from the modeled domain is used to estimate revocation rates. In analyzing all password changes logged over the past five years (1993-1998), we found that in the studied domain a password was changed, on average, once every 9 hours. If certificate revocation maintained this rate, and each revocation occurred 6 months into a one year certificate lifetime, the resulting PEM CRL would contain about 500 entries and be greater than 5 kilobytes. While 5 kilobytes may

appear small, CRLs are delivered many times over short periods. Thus, CRL size has a multiplicative effect on bandwidth usage. In the simulated PEM environment, the average bandwidth consumed per hour for the delivery of CRLs is over 1 megabyte. Under the same assumptions, a windowed revocation CRL would contain on the average 5 certificates, be about 90 bytes, and consume a little over 4 kilobytes per hour for CRL delivery.

3.3 Resource Consumption

In this section, we compare windowed revocation to explicit and implicit revocation mechanisms found in the PEM and DNSSec PKIs, respectively. Recall that to understand the behavior and benefits of revocation window independently from those of multicast push delivery, we do not implement CRL push delivery in all experiments described here.

Fig. 4 shows the bandwidth usage of PEM, windowed revocation, DNSSec, and an idealized X.509 based implicit mechanism.[7] The x-axis shows the time (in hours) since the start of the trace. The y-axis shows the total bandwidth usage. The total bandwidth usage includes that from certificate acquisitions and CRL retrievals. Note the presence of the diurnal pattern normally seen on network traffic traces. The size of certificates in DNSSec[8] ($\approx$ 242 bytes) is significantly smaller than the X.509v3 certificates ($\approx$ 1024 bytes) used by both PEM and windowed revocation. In light of the difference in certificate sizes, it is not surprising that DNSSec generates less bandwidth than both of the other schemes. Fig. 5, however, shows that DNSSec induces the CAs to generate more digital signatures per hour than windowed revocation. The x-axis again shows the time (in hours) since the start of the trace; the y-axis shows CPU consumption expressed as the number of signatures generated. Re-verification in windowed revocation may be achieved by retrieving a CRL, thus avoiding some of the signature creations required by DNSSec in all cases. Note that PEM generates a constant number of signatures (one signature per publication period in signing the CRL), and thus its CPU requirements are not shown in Fig. 5.

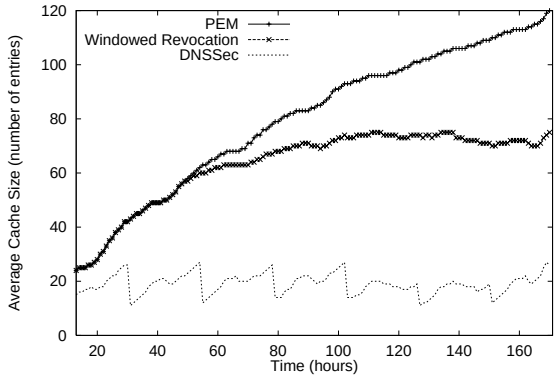

Figure 6: Average Cache Size: average cache size of hosts in the studied domain. In these experiments, no CRL push mechanism is simulated.

Fig. 4 further shows that windowed revocation consumes 83% less bandwidth than PEM. This can be attributed to two factors, the number of CRLs delivered and the size of the CRLs. For this workload, windowed revocation transmitted about 65% less CRLs than PEM. Unlike windowed revocation, each time a PEM verifier retrieves a certificate, it must also retrieve the latest CRL. Under the protocol parameters above, the average reporting period for a revocation is 2 days in window revocation and 6 months in PEM. The size of a CRL in both PEM and windowed revocation is a direct result of the revocation rate and the length of time a revocation is announced. In the simulated environment, the ratio of PEM to windowed revocation CRL sizes is exactly the ratio of the reporting period, or about $1:90$. Therefore, the cost associated with the delivery of a singular CRL in PEM is about 90 times greater than in windowed revocation.

In Fig. 4, the line labeled "Implicit" shows the bandwidth costs of an idealized implicit mechanism distributing X.509v3 certificates. In our modeled environment, certificate revalidation via CRL consumes less bandwidth than direct certificate acquisition. As a result, windowed revocation uses 40% less bandwidth than a strictly implicit mechanism distributing the same certificates.

[7] The idealized implicit mechanism is an approximation of the Online Certificate Status Protocol (OCSP) [MAM+98]. OCSP is described in the Related Work section of this paper.

[8] Certificates in DNSSec are called key resource records (RR). Key resource records are semantically identical to the certificates found in other PKIs.

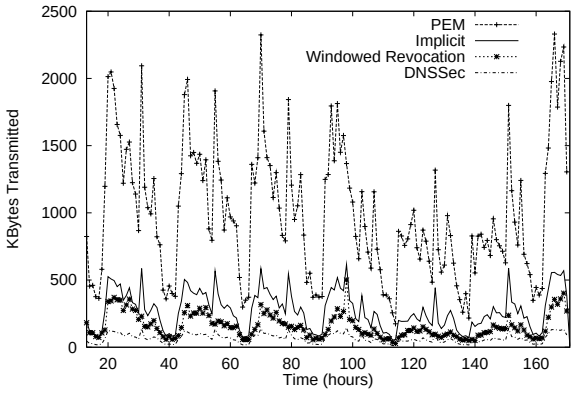

Figure 4: Total bandwidth usage calculated from estimation of certificate and CRL acquisitions. In these experiments, no CRL push mechanism is simulated.

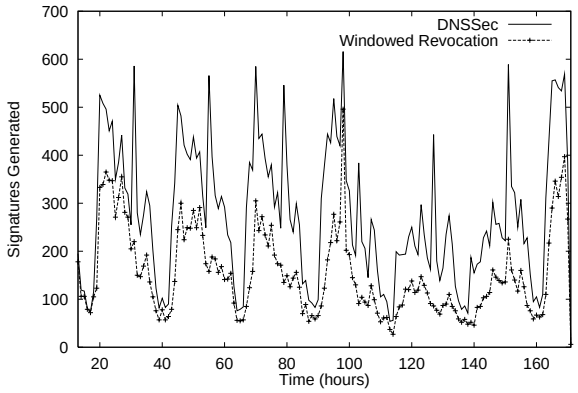

Figure 5: Signatures generated per hour in the simulated environment. In these experiments, no CRL push mechanism is simulated.

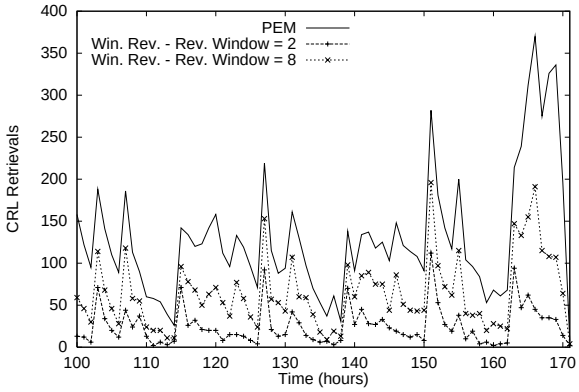

Figure 7: Total bandwidth costs in a series of simulations with varying revocation window sizes (CRL publication period = 12 hours).

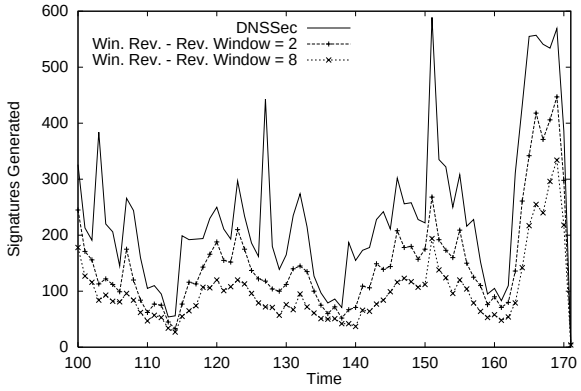

Figure 8: CPU costs in a series of simulations with varying revocation window sizes (CRL publication period = 12 hours).

3.4 Effect on Certificate Caching

Fig. 6 shows the average verifier cache size (on the y-axis) under PEM, windowed revocation, and DNSSec. DNSSec drops a certificate from its cache at the expiration of the certificate's TTL. Should the certificate need later re-verification, it must be re-acquired from the CA. Windowed revocation, on the other hand, simply marks a certificate that has been cached longer than its clean timer as dirty. Should the certificate need re-verification before expiration of the certificate's revocation window, only the last published CRL needs to be acquired and processed. Hence we see that windowed revocation caches almost three times as many certificates as DNSSec at the end of the simulations shown in Fig. 6. We consider the difference in the number of cached certificates as DNSSec's "lost opportunity" to provide better performance. DNSSec forces the re-acquisition of a large number of certificates that have not been revoked and would still have been validly cached under windowed revocation.

Fig. 6 also shows that at the end of the simulations, PEM caches twice as many certificates as windowed revocation. A certificate stays in the verifier's cache only if it is requested before the expiration of its revocation window. A dirty certificate with an expired revocation window is dropped from the cache. Since PEM does not have the concept of revocation window, a cached certificate stays cached until its revocation or the expiration of its lifetime. Fig. 6 shows

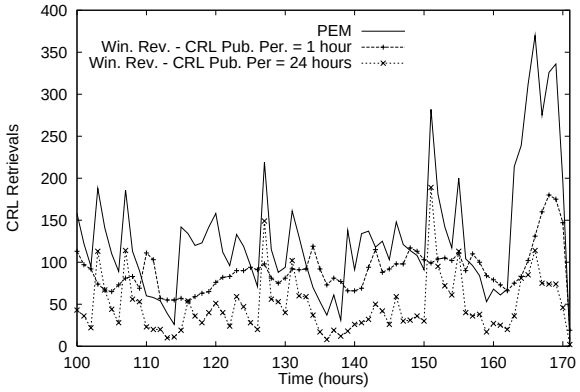

Figure 9: Total bandwidth costs in a series of simulations with varying CRL publication periods (revocation window size = 4).

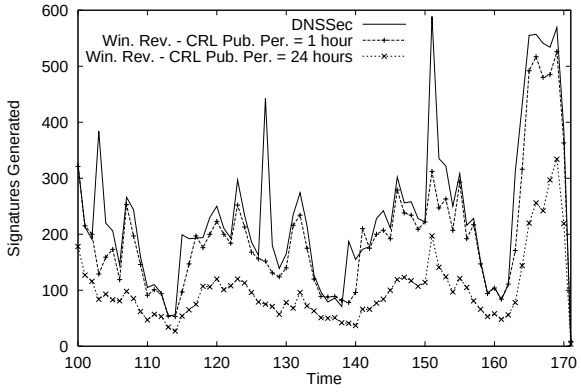

Figure 10: CPU costs in a series of simulations with varying CRL publication periods (revocation window size = 4).

that half of the cached certificates under PEM are not accessed within four times the CRL publication period and are thus good candidates for replacement should the cache overflow.

3.5 Protocol Parameters

The next set of experiments were designed to identify the effects of protocol parameter values on performance. We investigated the two primary windowed revocation parameters: revocation window size and CRL publication period. Again, we assume no multicast CRL delivery in these experiments.

The size of the revocation window determines the length of time a certificate will be included in the CA's CRLs, and indirectly, the length of time any verifier may cache a certificate without further verification. Intuitively, as the revocation window size increases, it is more likely that a certificate will require refreshing via CRL than by being dropped and reacquired. Thus, longer revocation windows should require more CRLs to be delivered, but less direct certificate acquisitions. This intuition is supported by the results of several simulations presented in Figs. 7 and 8. From Fig. 6, we determine that it takes about 100 hours for the certificate cache to warm up. Hence we only report resource consumption from the 100 hour on in the next five figures.

These results serve to illustrate the fundamental tradeoff in windowed revocation: that between network bandwidth (CRL size and request rate, as determined by the revocation window) and CA load (number of signatures generated).

Results of simulations with varying CRL publication periods are given in Fig. 9 and 10. These figures illustrate that smaller CRL publication periods increase the load on the CA. As the CRL publication period decreases, the length of the revocation window, which is expressed as multiples of CRL publication periods, also decreases, increasing the probability of certificates being revalidated beyond their revocation windows. The figure also indicates that the total number of CRLs delivered increases as the CRL publication period decreases. Thus, decreases in CRL publication period lead to increases in all P-KI costs.

3.6 Benefits of Multicasting

We have so far concentrated our study on the behavior and benefits of windowed revocation without multicast push delivery. We now turn our attention to the benefits of push delivering CRLs. In the simulated environment with a CRL publication period of 12 hours and a revocation window of 4 (48 hours), we observed that only 3% of the total bandwidth is consumed by the delivery of windowed revocation CRLs. Hence the use of a push mechanism for CRL delivery under this scenario will reduce bandwidth consumption by less than 3%. In contrast, when applied to traditional PEM, a CRL push mechanism reduces bandwidth consumption by 53%. This demonstrates that when the cost of certificate acquisition is constant, the advantage of pushed CRLs increas-

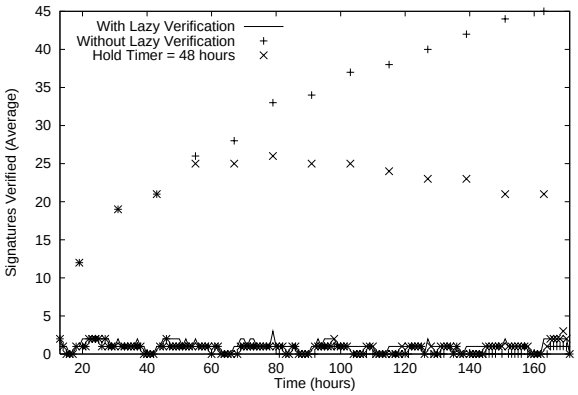

Figure 11: Average number of signatures validated by the verifier with and without lazy verification.

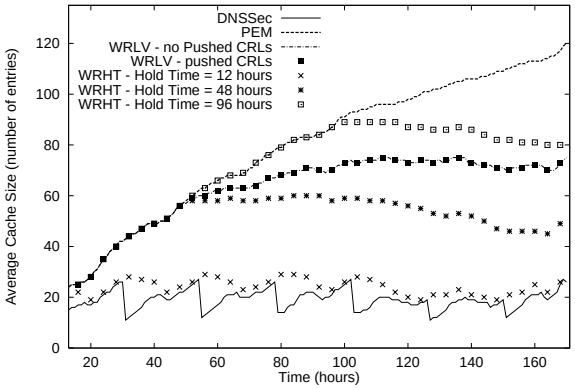

Figure 12: Average verifier cache size for PEM, DNS, and windowed revocation with a range of hold times. (*note: Several data points have been omitted to increase legibility.*)

es as the size of CRLs increases. Windowed revocation alone, even without pushed CRLs, uses 83% less bandwidth than traditional CRL based mechanisms. We expect that in communities larger than the one we have simulated, CRL size will be larger and thus the advantage of pushed CRLs under window revocation should increase with the size of the serviced community.

3.7 Benefits of Lazy Verification

In Section 2.3.1, we made a design decision not to use periodically published CRLs as cache invalidation messages, instead we opted to perform lazy verification and revalidate a dirty certificate only when it is needed again. When used in conjunction with pushed CRLs, lazy verification means that a CRL received from a CA is not processed until a certificate associated with the CRL needs to be revalidated. All our performance results so far are from windowed revocation with lazy verification. We now present performance data informing this design decision.

In our simulated environment, all CAs maintain the same publication period and generate CRLs on the same schedule. Using CRLs as cache invalidation messages means that at the time of CRL publication, each verifier will receive a CRL from each CA from which it acquired a currently cached certificate. The influx and subsequent processing of these CRLs cause periodic bursts of signature validations at the verifier hosts, illustrated by the line labeled "Without Lazy Verification" in Fig. 11. This behavior is clearly undesirable. This problem is similar to those experienced by a number of push based Internet services. Without careful engineering, the costs associated with push mechanism may increase network congestion and client host load.

Another salient feature apparent of the line labeled "Without Lazy Verification" in Fig. 11 is the continuing rise of CRL processing cost over time. Under windowed revocation, certificates refreshed via CRLs will never be dropped from cache. Assuming infinite cache size, the cache will thus hold every certificate ever acquired, and continue to receive the associated CRLs. As can be seen from the line labeled "With Lazy Verification" in the same figure, the undesirable effects of synchronized CRL delivery and ever increasing cache occupancy are not present. Lazy verification avoids these effects by not refreshing certificates at every CRL publication. Thus, unused certificates are dropped due to the expiration of the revocation window timer. (We call windowed revocation with lazy verification WRLV and without lazy verification WR henceforth.)

While lazy verification successfully prevents synchronized bursts of signature validations and alleviates the bursts of bandwidth consumption associated with CRL deliveries, it does not completely remove such bursts in bandwidth demand. Hence to further study the effect of certificate holding time on bandwidth demand, we experimented with windowed revocation without lazy verification but with

a *hold timer* associated with each cached certificate (WRHT henceforth). The hold timer specifies the maximum time a certificate may be cached without verifier access. Certificates not accessed before the expiration of its hold timer are dropped from cache; when a certificate is accessed its hold timer is reset to its initial value.

The line labeled "Hold Timer = 48 hours" in Fig. 11 shows the result of a simulation of WRHT with a hold timer value equal to the revocation window (48 hours). In this case, the number of signatures does not increase linearly over time as in the WR case, but reaches a steady state, with smaller load bursts at CRL publication times.

In Fig. 12, the average verifier certificate cache sizes for a range of simulations with varying hold timer values are shown. For comparison, we also include the average verifier cache sizes from simulations of the PEM and DNSSec systems. Note that the cache sizes increase linearly with hold timer length. A hold timer equal to the DNSSec TTL shows a smoothed approximation of DNSSec caching. The cache operates as in DNSSec, but avoids the periodic drops (and subsequent re-acquisitions) of certificates associated with TTL expirations. Notice also that WRLV with pushed CRLs has the exact same cache characteristics as that of WRLV without pushed CRLs.

Given revocation window size of 4 (48 hours), the performance of WRHT with hold timer of 48 hours tracks those of WRLV up to 48 hours. After which, their performance diverges with the WRHT caching less certificates. WRLV refreshes all unrevoked cached certificates associated with a CRL whenever the CRL is processed and drops only those certificates that have not been refreshed within their revocation windows. WRHT drops certificates that have not been accessed for a period of time, regardless of their clean and revocation window timers. Therefore, we see smaller cache occupancy under WRHT than under WRLV when the hold timer value is equal to the revocation window. Note that the trend between hours 110 and 160 towards smaller cache sizes is due to the reduced number of requests on the system. These hours span a Saturday and Sunday, when less data is requested by verifiers.

In conclusion, in environments where pushed CRLs are synchronized and the bursts of bandwidth demand is intolerable, system administrators can automatically drop cached certificates that have not been used for a specified amount of time. This mechanism, which we call the *hold timer*, is independent of and does not effect the correct operation of windowed revocation.

4 Related Work

The Privacy Enhanced Mail [Ken93] 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 the CRLs have been proposed [AZ98, HFPS98], many of which have been included in the IETF Public Key Infrastructure Working Group (PKIX) draft standards.

CAs supporting delta CRLs [HFPS98] 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.

In systems that use freshness CRLs [AZ98], 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 [NN98, Koc98, Mic96]. Using authenticated dictionaries, verifiers need not retrieve the entire CRL, but request only enough information to validate the certificate. These approaches often involve heavyweight cryptographic operations, long interactive protocols, and/or significant CA resources.

The Online Certificate Status Protocol (OCSP) [MAM+98] 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.

There is a direct parallel between global certificate and name-space management. In recognition of this fact, the authors of DNSSec [Gal96, EK99] 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) [Zim94] 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) [RL96, BFL96] and the closely related Simple Public Key Infrastructure (SPKI) [Ell98] 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.

5 Conclusions and Future Work

In this paper, we presented a novel approach to key revocation in Public Key Infrastructures. *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 host is bounded by the announcement period, called the *revocation window*. Thus, all certificates will be verified: (1) explicitly by CRL or, (2) implicitly by retrieval. Through manipulation of the revocation window, CAs 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.

We are in the initial stages of constructing a reference implementation for windowed revocation. We plan to integrate the windowed revocation services with SSLeay [HY98], a widely-used session layer providing secure point to point communication. We intend to integrate windowed revocation into systems currently supporting the PKIX working group standards.

References

[AZ98] C. Adams and R. Zuccherato. A General, Flexible Approach to Certificate Revocation, June 1998. http://www.entrust.com/resources/whitepapers.htm.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 164–173, May 1996.

[Cha94] D. Chadwick. *Understanding X.500 : The Directory*. Chapman & Hall, London, 1994.

[CY97] D. Chadwick and A. Young. Merging and Extending the PGP and PEM Trust Models - The ICE-TEL Trust Model. *IEEE Network*, 11(3):16–24, May/June 1997.

[DC90] S.E. Deering and D.R. Cheriton. "Multicast Routing in Datagram Internetworks and Extended LANs". *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[DOK92] Peter B. Danzig, Katia Obraczka, and Anant Kumar. An Analysis of Wide-Area Name Server Traffic. In *Proceedings of ACM SIGCOMM 92'*, 1992.

[EK99] D. Eastlake and C. Kaufman. RFC 2065, Domain Name System Security Extensions. *RFC 2065, Internet Engineering Task Force*, January 1999.

[Ell98] C. Ellison. SPKI Requirements (draft-ietf-spki-cert-req-02.txt). *Internet Engineering Task Force*, October 1998.

[FB97] W. Ford and M. Baum. *Secure Electronic Commerce : Building the Infrastructure for Digital Signatures and Encryption*. Prentice Hall, Englewood Cliffs, New Jersey, 1997.

[FL98] B. Fox and B. LaMacchia. Certificate Revocation: Mechanics and Meaning, 1998. http://www.farcster.com/papers/fc98/.

[Gal96] J. Galvin. Public Key Distribution with Secure DNS. In *Proceedings of the 6th USENIX Security Symposium*, pages 161–170, July 1996.

[HFPS98] 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.

[HY98] T. Hudson and E. Young. SSLeay and SSLapps FAQ, September 1998. http://psych.psy.uq.oz.au/ ftp/Crypto/.

[Ken93] S. Kent. RFC 1422, Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. *RFC 1422, Internet Engineering Task Force*, February 1993.

[Koc98] P. Kocher. A Quick Introduction to Certificate Revocation Trees (CRTs), 1998. http://www.valicert.com/company/crt.html.

[MAM+98] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP (draft-ietf-pkix-ocsp-03.txt) (Draft). *Internet Engineering Task Force*, March 1998.

[Mic96] S. Micali. Efficient certificate revocation. Technical Report Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, 1996.

[MJ98a] P. McDaniel and S. Jamin. Key Distribution Hierarchy. Technical Report CSE-TR-366-98, EECS, University of Michigan, Ann Arbor, 1998.

[MJ98b] P. McDaniel and S. Jamin. Windowed Key Revocation in Public Key Infrastructure. Technical Report CSE-TR-376-98, EECS, University of Michigan, Ann Arbor, 1998.

[Moc87] P. Mockapetris. Domain Names - Concepts and Facilities. *RFC 1034, Internet Engineering Task Force*, November 1987.

[NN98] M. Noar and K. Nassim. Certificate Revocation and Certificate Update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, January 1998.

[Pro94] Produced by the MITRE Corperation for NIST. Public Key Infrastructure Study, Final Report. http://csrc.nist.gov/pki/documents/welcome.html, April 1994.

[Riv98] R. Rivest. Can We Eliminate Certificate Revocation Lists? In *Proceedings of Finiancial Cryptography '98*, pages 178–183, February 1998.

[RL96] R. Rivest and B. Lampson. SDSI A Simple Distributed Security Infrastructure. http://theory.lcs.mit.edu/ rivest/sdsi11.html, October 1996.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore, second edition, 1996.

[Zim94] P. Zimmermann. PGP user's guide. Distributed by the Massachusetts Institute of Technology, May 1994.

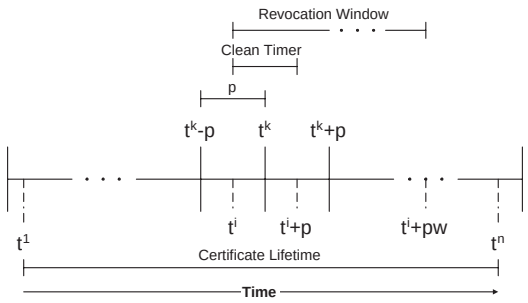

Figure 13: We show the lifetime of certificate host C, which is valid from t^1 to t^n. At time t^i, a verifier retrieves the certificate. In response, the clean and revocation window timers are set to $t^i + \pi$ and $t^i + wp$, respectively, where π is the verifier selected clean timer value, p is the CRL publication period of the CA, and w is the revocation window. The CA publishes CRLs at times $\ldots, t^k - p, t^k, t^k + p, \ldots$

A Proof of Correctness

In this appendix, we formally prove the bound on the use of revoked certificates. In Figure 13, 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 CA 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 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.

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.

Theorem 1: *The length of time any revoked certificate may be used is bounded by the length of the clean timer (π).*[9]

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.

  $$t^i \leq t^\delta < t^i + \pi, \quad \text{(\textit{from case definition})}$$
  $$t^i < t^r, \quad \text{(\textit{property 1})}$$
  $$\Rightarrow t^\delta - t^r < \pi,$$

  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.

[9] 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.

a) If $\pi < p$, then the certificate was dropped after the clean timer expires (see Section 2.5). Thus, 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 host. 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 certificates non-revoked status has been received from the CA, 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

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

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:

$\Rightarrow t^c \leq t^j$,
$t^i < t^r$, (*property 1*)
$t^r \leq t^c$, (*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.) □