Nonmonotonicity, User Interfaces, and Risk Assessment in Certificate Revocation (Position Paper)

Ninghui Li[1] and Joan Feigenbaum[2]

[1] Department of Computer Science, Stanford University, Gates 4B, Stanford, CA 94305-9045, USA
ninghui.li@cs.stanford.edu

[2] Department of Computer Science, Yale University, PO Box 208285, New Haven, CT 06520-8285, USA
jf@cs.yale.edu

Abstract. We consider certificate revocation from three high-level perspectives: temporal nonmonotonicity, user interfaces, and risk management. We argue that flawed understanding of these three aspects of revocation schemes has caused these schemes to be unnecessarily costly, complex, and confusing. We also comment briefly on some previous works, including those of Rivest [16], Fox and LaMacchia [5], and McDaniel and Rubin [11].

Keywords: Certificates, Revocation, PKI, CRL

1 Introduction

Public-Key Infrastructure (PKI) is an important enabling technology for e-commerce. However, the use of PKI can be limited by the cost, complexity, and sometimes confusion attributable to revocation. There has been a lot of debate over the meaning of certification and revocation [5, 11, 13, 16], and different revocation mechanisms have been proposed [1, 4, 9, 10, 12, 15, 19, 8, 14]. In this paper, we argue that revocation is complex and confusing for the following reasons.

- Revocation makes certification nonmonotonic. More precisely, in a PKI that has revocation, the validity of a certificate is nonmonotonic with respect to time, *i.e.*, a certificate may go from valid to invalid as time passes.
- A PKI has a user interface and internal entities and mechanisms that implement this interface. In the literature, this distinction is not always drawn clearly, and thus discussions of user-interface issues and internal-mechanism issues are often intermingled.
- Traditionally, revocation schemes have been viewed as methods to provide "security" instead of methods to control risk. This view limits the ways in which revocation mechanisms are used and analyzed.

In this paper, we consider certification and revocation from the perspective of these three issues. We separate the user interface (UI) of a PKI from the internal mechanisms of a PKI and argue that the UI should be as simple as possible: It should provide only the information needed by the users and hide the rest. In particular, it is desirable for a PKI to have a monotonic user interface: Every piece of information shown through the interface should have a meaning that is monotonic with respect to time. In fact, the UI's of most existing PKI's can be made monotonic by making time an explicit element.

We also argue that revocation is a risk-management tool. Risk associated with a PKI cannot be completely removed, but it can be analyzed and controlled. With revocation, users control risk by, for example, setting recency requirements for certificate acceptance. Smaller recency requirements lead to lower risk but require higher communication and/or computation cost. Setting the right recency requirement requires risk analysis and balancing the risk and the cost. It is clear that different applications have different risk requirements and that different users have different preferences in the risk-cost balance. Therefore, a PKI aiming to support multiple applications should provide a revocation interface that is tunable. Users should be able to set different recency requirements based on their needs and resources.

The UI of a PKI should also be helpful in auditing, *e.g.*, it should be easy to obtain a proof that a certificate was valid at a particular time in the past. This is useful for detecting fraudulent transactions after they occur. It is also useful when a user's risk is assumed by a third-party insurer, and the insurer requires the user to provide a proof that she has followed the insurer's policy in a transaction.

2 Background

A *public-key certificate* (*certificate* for short) is a data record digitally signed by a private key; the entity that possesses the private key and signs the certificate is called the *issuer*, or the *certification authority (CA)*, of this certificate. Data in a certificate include a public key, which we call the *subject key* (of this certificate), and some information about the *subject-key holder* (*holder* for short), *i.e.*, the entity that holds the private key corresponding to the subject key. A certificate binds the subject key and the information together. For example, a certificate may bind the distinguished name (DN) of an entity and its public key. A certificate may also express implicitly some trust the issuer has in the holder. For example, a CA-to-CA certificate often implicitly suggests the trustworthiness of the holder, in addition to establishing a DN-to-public-key binding. In the following, we use *binding* to mean both the binding of the subject key to the other data in the certificate and the implicit trust semantics.

Normally, a certificate has a validity period that includes a beginning time and an ending time; the issuer only vouches for the binding during this period. However, even before the validity period of a certificate ends, things may happen to make the information in the certificate invalid, *e.g.*, the subject-key holder may report that the private key has been stolen or lost, the issuer may suspect that the private key has been stolen from the holder or has been given away by the holder, or the binding may be shown to be no longer accurate. The traditional approach to certificate revocation is certificate revocation lists (CRL's) [8]. A CRL, signed by a CA, contains an issuing time t and a list of entries, each of which contains the serial number of a certificate that was issued by this CA, has not expired,[1] and has been revoked at t.

An architectural model of PKI is given in [8, 2]. In this model, a PKI has *end entities*, *PKI management entities*, and *repositories*. PKI management entities include CA's and, optionally, *registration authorities (RA's)*, to which CA's delegate certain management functions. Repositories are systems that store and distribute certificates and revocation data such as CRL's.

Here, we recommend a slightly different architecture. End entities are "users" of a PKI; thus the interface between end entities and the rest of a PKI is the user interface (UI) of the PKI. We further distinguish between two kinds of end entities: subject-key holders and entities that use certificates in making decisions, which we call *acceptors* or *verifiers*. In this paper, we focus on the acceptors's view of a UI, *i.e.*, the interface for providing information to help acceptors decide whether to accept a certificate, as opposed to the interface for requesting certificates.

As a general design principle for UI, we have the following.

Recommendation 1 *The UI of a PKI should be clear and simple. It should provide only the information needed by end users, and it should hide everything else.*

We also stress that, to be clear, a UI should precisely specify, for each piece of information it exposes to users, the meaning and the expected action.

We now review four kinds of user interfaces for PKI's, focusing on the data provided through the UI's.

- The first kind of UI's have certificates that cannot be revoked. This is the simplest kind. Certificates are valid for their life times, which are typically short.
- The second kind of UI's have certificates and CRLs. The standard X.509 PKI belongs to this kind. The characteristic of a CRL is that one piece of data (*i.e.*, the CRL) provides the current status of all the certificates issued by a CA. This is good for acceptors who process lots of certificates. However, the size of one typical CRL is quite large, and so the communication cost might be too high for acceptors who process only a small number of certificates.
- The third kind of UI's have certificates and validity proofs for individual certificates. Such proofs are much shorter than typical CRL's, but they can only prove the validity of one or several certificates. Examples of these kind include OCSP (Online Certificate Status Protocol) [14], CRS [12], *etc.* CRT

[1] According to [8], a revoked certificate should appear in at least one CRL after it has expired.

(Certificate Revocation Tree) [9] and 23CRT [15] provide both short proofs for one certificate and CRL-like data at the same time.

- The fourth kind of UI's have certificates and revocation notices. See, *e.g.*, the work of Wright *et al.* [19]. In these UI's, acceptors who are interested in the status of a certificate register themselves with someone who distributes revocation notices for the certificate, *e.g.*, the CA. When a certificate is revoked, the CA broadcasts this information to all interested parties.

3 A Monotonic Interface for PKI

Revocation leads to nonmonotonicity. When more certificates are revoked, fewer are valid; the amount of validity information decreases when the amount of revocation information increases. Normally, revocation information increases over time, *i.e.*, as time passes, more certificates are revoked. Therefore, when a PKI allows revocation, the validity information is temporally nonmonotonic. More specifically, a certificate valid at time t_0 may become invalid at a later time t_1.

The nonmonotonicity introduced by revocation is similar to the notion of "negation-as-failure" in the logic-programming and nonmonotonic-reasoning literature. Negation-as-failure means that, to conclude "not r," one needs to try every way to prove r; if they all fail, then "not r" is concluded. In a PKI with revocation, one needs to prove "not revoked(cert)" at the time at which one decides whether to accept a certificate. To prove "not revoked(cert)," conceptually, one needs *complete* information about revoked(). Because the information about revoked() increases with time, one needs *current* information in order to conclude safely "not revoked(cert)." In a distributed system, distributing absolutely current information to all concerned parties is impossible. The best one can do is to deliver recent information. Even this is quite expensive in large-scale distributed systems. This is a major source of difficulty in revocation.

Recommendation 2 *The difficulty of revocation is caused by temporal nonmonotonicity, and thus a PKI should provide an interface that is monotonic.*

In fact, when viewed appropriately, existing PKI's have such an interface. In the following, we give a monotonic semantics of certificates and information provided by revocation mechanisms.

Without revocation, the meaning of a certificate is monotonic. A certificate means that the issuer vouches for the binding in the certificate for the validity period. Anyone who sees the certificate can check whether it has expired and decide whether to use it.

When revocation is possible, the meaning of a certificate becomes more complicated. In [16], Rivest discussed the following guarantee for standard certificates: "*This certificate is good until the expiration date. Unless, of course, you hear that it has been revoked.*" Rivest argued that this guarantee is not very useful, because the acceptor is always required to check whether a certificate has been revoked; he proposed a different general certificate guarantee: "*This certificate is definitely good from T_1 until T_2. The issuer also expects this certificate* *to be good until T_3, but a careful acceptor might wish to demand a more recent certificate. This certificate should never be considered valid after T_3.*"

The above guarantee is a combination of nonrevokable certificates and standard revokable certificates. It means that a certificate is nonrevokable from T_1 to T_2 and then is a standard certificate. We argue that this interpretation of certificate is still problematic. The meaning of this certificate is still nonmonotonic from T_2 and T_3.

A certificate states what its CA believed when the certificate was issued. This belief may change over time, and this change may be reflected by revocation. This is the cause of nonmonotonicity. However, the fact that the CA believed the content of the certificate at the time when it was issued doesn't change over time. Therefore, we can give a certificate a temporally monotonic meaning if we take the issuing time as part of the meaning of a certificate.

We now introduce a simple logic for representing meaning of certificates. A statement in this logic takes the following form:

– At time t_0, X believes b to be true in $[t_1, t_2]$, where $t_1 \le t_2$.

We call t_0 the *fresh time* of this statement. This logic has the following two inferencing rules:

1. If, at t_0, X believes b to be true in $[t_1, t_2]$, then, at t_0, X believes b to be true in any $[t', t'']$ such that $t_1 \le t' \le t'' \le t_2$.
2. If, at t_1, X believes b to be true in $[t_1, t_2]$, then, at any time t_0 such that $t_0 < t_1$, X believes b to be true in $[t_1, t_2]$.

Note that this logic doesn't interpret the belief b. In particular, it doesn't relate beliefs b and $\neg b$. Also note that one cannot express "disbeliefs" in this logic.

The first inferencing rule is straightforward and quite standard [18]. The second rule says that, if X believes something at time t_1, then X has been believing it at all times up to t_1. This is certainly false for general beliefs; however, it seems appropriate for our purpose, *i.e.*, monotonic reasoning about certificates and revocation. Next we show that certificates and revocation data such as CRL's can be represented by statements in this logic.

Recommendation 3 *We propose the following interpretation of certificates: At issuing time t_0, the issuer believes the information in this certificate to be true from t_1 to t_2.*

This reading is temporally monotonic; it is always true at any time after t_0. Note that our interpretation makes issuing time an explicit part (the fresh time) of the meaning of a certificate.

A certificate states the issuer's belief at the issuing time t_0, and one can view revocation schemes as mechanisms to reconfirm the issuer's belief at a later time. If one only has a certificate issued at t_0, the fresh time of the binding in the certificate is t_0. If one also obtains a proof that a certificate has not been revoked at a later time t_1, then one can update the fresh time to t_1.

Consider the case that, at time t_u, an acceptor wants to use a certificate that has validity period $[t_1, t_2]$, issuing time t_0, and fresh time t_f. The acceptor should check that, at a time in recent past, say, within a fresh requirement dt, the issuer of this certificate still believed the binding to be true at current time t_u. In other words, the verifier needs to check that at time $t_u - dt$, the issuer still believed the binding to be true in $[t_u, t_u]$. Following the inferencing rules, the verifier needs to check that $t_u \in [t_1, t_2]$ and that $t_f \geq t_u - dt$. The choice of dt is a policy that the acceptor needs to decide. If one doesn't want to check revocation, one can set dt to ∞, then $t_f \geq t_u - dt$ is always true.

Most existing certificate formats only have two time fields: *not-before* and *not-after*, and it is often assumed that the *not-before* time is the same as the issue time. If one is willing to make this assumption, one can interpret existing standard certificates as in Recommendation 3. However, we think that a certificate should have a separate issue time in order to allow post-dated certificates to be issued. A post-dated certificate can be revoked even before its validity period begins.

A CRL issued at t_1 is a claim that all certificates that are not listed should have a fresh time t_1 or later. Some argue that one can criticize CRL's because they make negative statements. We disagree. Although a notice that some certificates have been revoked is negative, a list of all revoked certificates provides positive information, because all those certificates that are not listed are still valid. In some cases, this is more efficient than listing all nonrevoked certificates. There is also the argument that a CRL doesn't provide positive information, because it doesn't prove the existence of a certificate. We disagree with this, too. The purpose of revocation is to complement certification, not to replace it. The purpose of a CRL is not to prove that a binding is valid but rather to update the fresh time of an existing proof (a certificate). One has to have a certificate first before caring about revocation.

Similarly, responses of the Online Certificate Status Protocol (OCSP) [14] and information from other revocation schemes can all be viewed as proofs that something is still believed at a later time.

In section 2, we reviewed four kinds of UI's for PKI. Among them, only the last kind, *i.e.*, the one that uses revocation notices, cannot be interpreted as in Recommendation 3. A revocation notice is a piece of negative information. If it fails to reach an acceptor, then the acceptor may accept a revoked certificate as valid.

We want to stress the point that the difficulty of revocation is caused by *temporal* nonmonotonicity. Because revocation information changes with time, one needs sufficiently recent information about revocation. Some previous work tries to make certification with revocation monotonic; however, this work does not address the time issue. In [6], Gunter and Jim argued that revocation information can and should be handled in the same way as certificates and that their system Query Certificate Manager (QCM) with revocation is monotonic. QCM has dual notions of positive sets and negative sets, *e.g.*, a CRL is a negative set. For a positive set, a QCM certificate states that an element is a member of the set. For a negative set, a QCM certificate states that an element is not a member of the set. An environment is a set of QCM certificates. In [6], the claim that QCM with revocation is monotonic means that a larger environment always leads to more conclusions. However, an environment itself is nonmonotonic with respect to time; more specifically, a QCM certificate for a negative set itself may go from true to false as time passes. For a user to decide whether to accept a certificate, she needs to forget an old environment and get a sufficiently recent one. This doesn't decrease the amount of information that needs to be transmitted.

4 The Semantics of Revoking a Certificate is to Cancel It

A certificate may be revoked for several reasons. In [5], Fox and LaMacchia argued that revocation for different reasons should have different semantics. When a verifier knows that a certificate has been revoked, the verifier should remove the revoked certificate from any certificate chain (or graph) that she is using. In other words, revoking a certificate cancels it. A question that follows is whether revoking a certificate should do more than that. Consider an example given in [5].

Example 1. Let $C = c_0, c_1, \ldots, c_n$ be a chain of certificates, where c_n is the end-entity certificate of interest, c_0 is a self-signed, trusted-root certificate issued by K_0, and each c_i, for all $i = 1, \ldots, n$, is signed by the private key corresponding to K_{i-1}, the subject key of c_{i-1}. Let j be an integer in $[1..n-1]$, and let $C' = c'_0, c'_1, \ldots, c'_j$ be a second chain of certificates from K_0 to K_j. Suppose that the certificate c'_j is revoked and that all other certificates in the two chains are valid. If these are the only certificate chains that the user has that end in c_n, should the user accept the binding in c_n, or (equivalently in this acceptance decision) should c_j be treated as valid?

In [5], Fox and LaMacchia argued that whether c_j should be treated as valid depends on the reason for revoking c'_j. The certificate c'_j may be revoked in each of the following three cases:

(a) the key K_j has been compromised, in which case, c_j should be treated as revoked as well.

(b) the binding in c'_j is no longer valid, in which case, c_j should be treated as invalid if it contains the same binding as c'_j.

(c) the binding may still be valid, but the issuer doesn't want to vouch for it anymore, in which case, c_j should still be valid.

Although it is desirable to revoke all certificates concerning a compromised private key, we argue that this should be done internally, *i.e.*, on the other side of the PKI's UI from the external one that is exposed to users.

Interpreting revocation of c'_j as revoking c_j as well enlarges the domain over which an acceptor needs complete information. To use c_j, one not only needs to know that "not revoked(c_j)" but also needs to know "not revoked(c'_j)," for all c'_j's that are somehow related to c_j. This has the effect of changing the trust relationship. Under this interpretation, one certificate path is not enough. To use a certificate, one needs to have all the CA's agree that a private key has not been compromised or that a binding is valid; any CA can veto a binding by issuing a certificate and then revoking it for key-compromise reasons. This is not just expensive — it may also be undesirable.

We believe that revocation schemes shouldn't change the trust relationships of a PKI. If a CA wants to revoke a certificate whenever another CA revokes a related certificate, it should make this arrangement behind the user interface. If a user needs more than one source to confirm a binding, *e.g.*, a separate proof that the private key has not been compromised, then this should be clearly specified by the user's policy; it shouldn't be accomplished indirectly with revocation.

Recommendation 4 *Revocation of a certificate should cancel the certificate and do nothing else.*

5 Revocation Provides Risk Management for PKI

Traditionally, computer-security mechanisms try to ensure that insecure things do not happen. In [20], an alternative view is given. We summarize it as follows: Complex systems can be secured only up to a point. Insecurity always exists and cannot be destroyed. The question one should ask is not whether a system is secure, but how secure that system is relative to some perceived threat (page 119 of [20]).

That insecurity always exists is precisely the situation in a global-scale public-key infrastructure. Total security is unattainable, even under the unrealistic assumption that revocation information can be delivered to everyone instantaneously. A private key may be compromised long before the compromise is discovered and the certificates for the key revoked. This cannot be handled by revocation schemes, but it should be taken into consideration when analyzing the risk inherent in a PKI.

When we acknowledge that risk always exists, we can view revocation schemes as a way to control risk. Traditionally, it is often implicitly assumed that everyone should get the most recent CRL. One piece of evidence for this assumption is that each CRL has a next-update field; a CRL is assumed to be expired after that date, and a newer CRL is needed.

However, when taking the risk-management view of CRL's, it is clear that one doesn't always need the most current CRL. Instead, one should set recency requirements as a matter of policy. As long as a user has a CRL that is recent enough, it should be okay. More strict recency requirements have lower risk, but they have higher communication costs. Because risk is application-dependent, different applications and users have different recency requirements. Therefore, we have the following recommendation.

Recommendation 5 *A PKI that serves diverse applications should provide flexible revocation schemes that can be tuned to support different recency requirements.*

Whoever is exposed to the risk of wrongfully accepting a certificate should set recency requirements. However, which party has higher risk has been debated. In [16], Rivest argued that recency requirements must be set by the acceptor of a certificate, not by the certificate issuer, because the acceptor is the one who is at risk if her decision is wrong. In [11], McDaniel and Rubin disagreed. They argued that, in business-to-consumer e-commerce scenarios, in which consumers (or their browsers) need to decide to whether to accept a merchant website's certificate as valid to establish a secure connection, consumers are usually transferring credit-card numbers through the connection and thus only have limited liability. Conversely, the merchant risks its reputation for unsafe operation; therefore, the risk is actually higher for merchants than for acceptors. In [5], Fox and LaMacchia said that "In theory, the certificate authority has the most to lose with continued circulation of a bad certificate."

In the above B2C credit-card transmission scenario, several parties are at risk in one fraudulent transaction. Both the merchant and the certification authorities risk some reputation damage, but their risk is limited and less tangible. Note that we are talking about the damage caused by *one* fraudulent transaction, *i.e.*, one in which a consumer accepts a certificate that she shouldn't and as a result sends her credit card number to an intruder, *not* the damage caused by revocation of a certificate. When a merchant's certificate is revoked because of key compromise, the merchant has *already* suffered great loss of reputation, even before a single fraudulent transaction occurs. This means that a merchant has high incentive to protect its private key, but it doesn't necessarily have higher risk than a customer in *one* fraudulent transaction. One can argue that more fraudulent transactions will do more damage to the merchant's reputation, but it is hard to quantify how much damage each additional fraudulent transaction does. More importantly, there is no way for merchants or CA's to enforce the requirements. They can make suggestions about the recency requirements suitable for particular kinds of transactions; CA's should also make revocation information available. However, if customers don't follow these suggestions, there isn't much that CA's or merchants can do. Thus, they are mostly free of reputation damage as long as they have made good suggestions.

The acceptor has the primary risk. In some cases, the acceptor is protected by insurance, and her risk is limited to a small deductible. In that case, the insurer has the highest risk. This is the case in the credit-card scenario. Note that credit-card numbers aren't the only kind of information transmitted through secure connections. In online banking or trading, a customer may be transmitting account numbers and PINs, which can be much more valuable than credit-card numbers.

When an acceptor is insured by someone, all or part of the acceptor's risk is transferred to the insurer. Then the insurer has a strong incentive to set and enforce recency requirements. However, at the point of decision, only the acceptor can enforce the recency standard, because it is she who actually decides whether to accept a certificate. Note that the granularity of the recency standard is limited by the revocation mechanisms available to the acceptor. If a PKI has

CRL's as the only revocation interface, and CRL's are issued at time interval δt, then no one can operate with a recency requirement that is smaller than δt. Although an insurer cannot enforce a recency requirement when the transaction occurs, the insurer can do so when something goes wrong and the acceptor makes a claim. The insurer can set a recency requirement and require the acceptor to provide proof that she has followed the requirement in the transaction.

Recommendation 6 *The UI of a PKI should support auditing.*

For example, a PKI that supports CRL's could maintain a CRL that keeps all the revoked certificates and the time at which they are revoked, whether they are expired or not. The insurer or some other parties can use this CRL to check whether a certificate is valid at a time in the past. Besides allowing a certificate to be revoked, PKIX also allows a certificate to be put on hold (temporarily disabled) and then activated again [8]. It is difficult for the above scheme to deal with such certificates, because the notion of "certificates-on-hold" significantly complicates revocation. When certificates can only be revoked, the revocation status of a certificate is temporally monotonic (although the validity status is not). Thus a certificate can only go from valid to revoked. Recording the time of this change or the fact that such a change has not occurred determines the status of the whole life of a certificate. When a certificate can be put on hold, the revocation status of a certificate is not temporally monotonic. To know the status history of a certificate, one needs to know all the changes that have occurred in the past. It is even harder to figure out whether a particular acceptor believes that a certificate is valid or not at a time in the past, especially when acceptors have different recency requirements. Therefore, we argue that it is better to disallow this notion of certificate-on-hold. As an alternative, the CA can revoke the certificate and later issue a new certificate with the same binding when needed.

Recommendation 7 *We recommend not allowing certificates to be put on hold, in order to simplify auditing and the semantics of revocation.*

In addition to being useful in scenarios that involve insurers, auditing can also be used for earlier detection of potentially fraudulent transactions. Consider an acceptor that has a CRL issued at time t_0 and is scheduled to obtain a new CRL at a later time t_1. Suppose that the CRL at time t_0 doesn't contain the certificate c. Then, at any time between t_0 and t_1, the acceptor would accept c as valid. However, if the certificate c is revoked during this time, then this is potentially problematic. It would be useful to detect this when the acceptor obtains a new CRL at t_1. To use CRL to support this kind of auditing and to support variable recency requirements at the same time, a CRL should keep an revoked certificate longer than required by [8]. In [8], a revoked certificate is required to appear in at least one CRL after it has expired. If the certificate c is revoked then expired after t_0, and several CRL's are issued after c expired and before t_1, then the acceptor won't know that the transaction involving c is potentially problematic. One solution to this is to have a CA set two parameters for issuing CRL, δt and Δt, where a new CRL is issued every δt and a revoked certificate will be kept on CRL for Δt after it has expired. Anyone who uses CRL's should set a recency requirement that is between δt and Δt.

Different revocation mechanisms have been proposed, and there has been extensive debate over which revocation mechanisms are the best and who should provide recency proofs. We think that the answers depend on the specific application and scenarios. No one scheme fits all scenarios. For example, CRL's work well when there are a small number of acceptors who have high communication capacity and who process lots of requests from a large number of certificate holders. This is often the case in an intranet setting, *e.g.*, an internal web server authenticating employees using certificates. In this case, it is more efficient for the web server to obtain and check a CRL than for certificate holders to be required to obtain and present proofs.

On the other hand, CRL's are not suitable in B2C e-commerce scenarios, in which customers' browsers are acceptors. There are a large number of acceptors, each of which processes only a small number of requests. Furthermore, acceptors often have limited network bandwidth. It is not efficient to have every browser deal with CRL's. It is better to have the server obtain a recency proof and reuse it with different browsers. In this case, revocation mechanisms that can generate short validity proofs for certificates are needed. The fact that existing PKI's lack the ability to provide short validity proofs is one reason that revocation is not used in B2C e-commerce scenarios.

6 Conclusions

In summary, a PKI should have a clear and simple user interface that is temporally monotonic and supports functionality needed for applications. Depending on the application, it may be necessary for a PKI to support tunable revocation services and auditing.

7 Acknowledgements

The first author is supported by DARPA contract N66001-00-C-8015. The second author is supported in part by DARPA grant AF F39502-99-1-0512.

References

1. Carlisle Adams and Robert Zuccherato, "A General, Flexible Approach to Certificate Revocation," June 1998.
http://www.entrust.com/resourcecenter/pdf/certrev.pdf
2. Carlisle Adams and Stephen Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols," IETF RFC 2510, March 1999. http://www.ietf.org/rfc/rfc2510.txt

3. David A. Cooper, "A Closer Look at Revocation and Key Compromise in Public Key Infrastructures," in *Proceedings of the 21st National Information Systems Security Conference*, pp. 555–565, October 1998. http://csrc.nist.gov/nissc/1998/proceedings/paperG2.pdf
4. David A. Cooper, "A More Efficient Use of Delta-CRLs," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 190–202, May 2000. http://csrc.nist.gov/pki/documents/sliding_window.pdf
5. Barbara Fox and Brian LaMacchia, "Certificate Revocation: Mechanics and Meaning," in FC'98 [7], pp. 158–164, 1998. http://www.farcaster.com/papers/fc98/fc98.ps
6. Carl A. Gunter and Trevor Jim, "Generalized Certificate Revocation," in *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, pp. 316–329, January 2000. http://www.cis.upenn.edu/~qcm/papers/popl00.pdf
7. Rafael Hirschfeld, editor, *Financial Cryptography: Second International Conference (FC'98)*, Lecture Notes in Computer Science, vol. 1465, Springer, February 1998.
8. Russell Housley, Warwick Ford, Tim Polk, and David Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," IETF RFC 2459, Janurary 1999. http://www.ietf.org/rfc/rfc2459.txt
9. Paul Kocher, "On Certificate Revocation and Validation," in FC'98 [7], pp. 172–177, 1998.
10. Patrick McDaniel and Sugih Jamin, "Windowed Certificate Revocation," in *Proceedings of IEEE Infocom 2000*, pp. 1406–1414, March 2000. http://www.eecs.umich.edu/~pdmcdan/docs/info2000.pdf
11. Patrick McDaniel and Aviel Rubin, "A Response to 'Can We Eliminate Certificate Revocation Lists?'," in *Proceedings of Financial Cryptography 2000*, February 2000. http://www.eecs.umich.edu/~pdmcdan/docs/finc00.pdf
12. Silvio Micali, "Efficient Certificate Revocation," Technical Report TM-542b, MIT Laboratory for Computer Science, March, 1996. ftp://ftp.lcs.mit.edu/pub/lcs-pubs/tm.outbox/MIT-LCS-TM-542b.ps.gz
13. Michael Myers, "Revocation: Options and Challenges," in FC'98 [7], pp. 165–171, 1998.
14. Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP," IETF RFC 2560, June 1999. http://www.ietf.org/rfc/rfc2560.txt
15. Moni Naor and Kobbi Nissim, "Certificate Revocation and Certificate Update," in *Proceedings of the 7th USENIX Security Symposium*, pp. 217–228, January 1998. http://www.wisdom.weizmann.ac.il/~kobbi/papers/revoke_usenix.ps
16. Ronald L. Rivest, "Can We Eliminate Certificate Revocation Lists?" in FC'98 [7], pp. 178–183, 1998. http://theory.lcs.mit.edu/~rivest/revocation.ps
17. Stuart G. Stubblebine, "Recent-Secure Authentication: Enforcing Revocation in Distributed Systems," in *Proceedings of the 1995 IEEE Symposium on Research in Security and Privacy*, pp. 224–234, May 1995. http://www.stubblebine.com/95oak.pdf
18. Stuart G. Stubblebine and Rebbeca N. Wright, "An Authentication Logic Supporting Synchronization, Revocation, and Recency," in *Proceedings of the Third ACM Conference on Computer and Communications Security*, pp. 95–105, March 1996. http://www.stubblebine.com/96ccs.pdf
19. Rebecca N. Wright, Patrick D. Lincoln, and Jonathan K. Millen, "Efficient Fault-Tolerant Certificate Revocation," in *Proceedings of the 7th ACM Confer-*

ence on Computer and Communications Security (CCS'2000), November 2000. http://www.research.att.com/~rwright/ccs00.ps
20. Committee on Information Systems Trustworthiness, National Research Council, *Trust in Cyberspace*, National Academy Press, 1999. http://www.nap.edu/html/trust/