Certificate Revocation: Mechanics and Meaning

Barbara Fox[⋆] and Brian LaMacchia

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
{bfox,bal}@microsoft.com

Abstract. Revocation of public key certificates is controversial in every aspect: methodology, mechanics, and even meaning. This isn't so surprising, though, when considered in the context of current public key infrastructure (PKI) implementations. PKIs are still immature; consumers, including application developers and end-users, are just beginning to understand the implications of large-scale, heterogeneous PKIs, let alone PKI subtleties such as revocation. In this paper, which is the product of a panel discussion at Financial Cryptography '98, we illustrate some of the semantic meanings possible with current certificate revocation technology and their impact on the process of determining trust relationships among public keys in the PKI. Further, we postulate that real-world financial applications provide analogous and appropriate models for certificate revocation.

1 Introduction

At this stage of PKI deployment we've figured out how to issue certificates and evaluate them in a reasonably interoperable manner but not how to revoke them. In fact, when it comes to revocation, we can't even agree on who should do it, how it should work or even what it means.

In the most literal sense, to *revoke* a digital certificate is to *UNDO* a persistent signed statement. Revocation doesn't sound controversial or, at first glance, even very hard. At the root of the problem though is the maturity of the technology itself. While the concept of digital certificates has been around since the late seventies, only now do we see any plans by commercial certificate authorities and enterprise PKIs to issue them on a large scale. And as technologists, we've done a terrible job of explaining certificates and digital signatures.

Viable trust models already exist that do not include explicit "identity" certificates, most notably PGP ("Pretty Good Privacy" [2]). In key-centric PGP, key holders are responsible for notifying their correspondents of a key compromise — a form of self-revocation. After all, key compromise is what they *should* care about; for it allows the crook to impersonate them in a transaction. We do not tend to think of the *subject* of a certificate being able to unilaterally revoke that certificate, but that's exactly what is necessary if the subject knows its private key has been compromised. Key holders could "bulk" revoke all their certificates acquired with the same key through some kind of registry. This is a natural analog of credit card registries with the convenient side effect that multiple certificates issued against the same key establish an alias: Bfox@microsoft.com can easily become barb@skijump.com when two certificates are bound to the same key pair.

To make this phenomenon clear to consumers of public key systems including application developers and end-users, we first need to position certificates as merely *evidence for a decision* that the certificate acceptor needs to make. The certificate (or a chain of them) is intended to provide enough information for an acceptor to determine whether he will accept the bound signature for a transaction, period. Confusion arises however because actual certificate evaluation is usually buried deep

[⋆] The opinions expressed in this paper are those of the authors and are not necessarily those of Microsoft Corporation.

under the covers in some application or system. Most Internet-savvy users understand the concept of trusted *root certificates*, which are usually either "baked into" or installable in their browsers, and that a digital signature/certificate chain ending in one of these root certificates constitutes validation. Revocation, however, is a different matter — because it doesn't work yet. In other words, there's still time to make revocation make sense.

So far, though, we've manage to confuse certificates, which are solely intended to establish transitive trust through public keys, with lots of real-world analogies like driver's licenses and credit cards. In retrospect, this was probably a bad idea because many of these examples lose sight of the function and value of the signature key. The long-term cost of these false similes is that before revocation can make any sense we have to start over in explaining certificates themselves.

2 Meaning

What really matters in a public key certificate? For practitioners looking at this problem for the first time, the answer is obvious: *the issuer and subject public keys.* But those familiar with X.509-based public key infrastructures could counter that *the distinguished names of the issuer and subject* alone provide enough information to verify a certificate chain.

The process of certificate path validation from an end-entity certificate of interest to a trusted root is the algorithmic foundation of establishing trust in public key systems. In turn, checking to determine whether a certificate has been revoked is just a function of the underlying algorithm.

The proposed IETF public key infrastructure standard, PKIX Part 1 [4], specifies the algorithm for determining whether a particular certificate chain is valid but does not describe how to discover certificate chains in the first place. Moreover, the semantics of PKIX certificate revocation are also underspecified, which puts crucial issues of meaning in the hands of individual implementations. In every case, the actual process of certificate revocation is deemed the responsibility of the signer even if it is at the request of the subscriber. In theory, the certificate authority has the most to lose with continued circulation of a bad certificate.

Let $C = c_0, c_1, ..., c_n$ be a chain of certificates where c_n is the end-entity certificate of interest, c_0 is a self-signed trusted root certificate, and c_k is signed by the subject of c_{k-1} for all $k = 1, ..., n$. By definition, if any certificate c_i in C is revoked then C is not a valid chain. Assume that no c_i in C is revoked, and let $C' = c'_0, c'_1, ..., c'_{k-1}, c_k, ..., c_n$ be a second chain of certificates from c_n to a trusted root (c'_0), with c'_{k-1} distinct from c_{k-1}. Certificates c_{k-1} and c'_{k-1} have the same subject public key (the issuer of the statement in c_k).

Now, because revocation in the PKIX model applies to a particular certificate, it is possible to revoke c'_{k-1} without revoking c_{k-1}. Then chain C' is an invalid chain while C is still valid. Is this consistent? It depends on the semantics of the revocation. The certificate c'_{k-1} is a statement by an *issuer* $I_{c'_{k-1}}$ that binds together *subject name* $SN_{c'_{k-1}}$ and a *subject public key* $SPK_{c'_{k-1}}$. That is:

$$I_{c'_{k-1}} : SN_{c'_{k-1}} \longleftrightarrow SPK_{c'_{k-1}}$$

where this statement is read, "Issuer $I_{c'_{k-1}}$ states that there is a binding (or relationship) between the public key $SPK_{c'_{k-1}}$ and the identity information $SN_{c'_{k-1}}$."

So, revoking c'_{k-1} (e.g. adding c'_{k-1} to a "certificate revocation list" (CRL) signed by $I_{c'_{k-1}}$) could mean "UNDO" any of the following:

(a) UNDO $SPK_{c'_{k-1}}$, the subject public key (that is, no longer trust the subject public key for anything because it has been compromised),
(b) UNDO $SN_{c'_{k-1}} \longleftrightarrow SPK_{c'_{k-1}}$, the relationship between the subject public key and the subject name/identity information, because that binding is no longer valid, or

(c) UNDO $I_{c'_{k-1}}$:<binding>, the relationship between the certificate issuer and the binding between subject public key and identity information, because the issuer is no longer willing to vouch for the binding, although it may still be true.

Each of these cases means something different, and chain processing in the presence of revocation information acts differently in each case.

Consider the first case (a) above, where revocation of c'_{k-1} denotes compromise of the subject public key. In this case, the fact that c'_{k-1} is revoked should cause *all* certificate paths that involve the subject public key $SPK_{c'_{k-1}}$ to no longer be valid. So, not only is C' now invalid, but C itself is now invalid as:

$$SPK_{c_{k-1}} == SPK_{c'_{k-1}} == I_{c_k}$$

Ideally, if any certificate for a given subject public key is revoked for reasons of key compromise, all such certificates would immediately be revoked, but obviously we cannot guarantee this behavior. Thus, it may be argued that relying parties have a duty to check revocation status on all certificates naming a particular subject public key even if they themselves are not relying on those certificates for chain-building.

Case (b), direct revocation of the subject name-subject public key binding by the issuer, is a fuzzier situation. If the subject names in the two certificates c_{k-1} and c'_{k-1} are distinct (i.e. $SN_{c_{k-1}} \neq SN_{c'_{k-1}}$) then revocation of c'_{k-1} should have no impact on acceptance of c_{k-1} itself. That is, chain C' is invalid because of the revocation but C is still a perfectly valid chain. Notice, however, that if the subject names in the two certificates *are* equal then relying parties could reasonably choose to reject C (or at least suspect it) if the name information $SN_{c_{k-1}} == SN_{c'_{k-1}}$ is important to their particular application.

Finally, in case (c) we have revocation of certificate c'_{k-1} because the issuer of that certificate no longer has a relationship with the subject public key. Revocation here speaks not to the validity of the name-key binding but rather to a lack of contractual obligation. Revocation of c'_{k-1} should not in any way impact chain C as there is no authorization statement from the issuer of c'_{k-1} concerning the validity of the subject public key $SPK_{c'_{k-1}}$ itself (which would make it case (a)) or the name-key binding (case (b)).

The fact that the act of revoking a particular certificate needed to be qualified with intended semantics was recognized by the authors of the X.509 standard. In X.509 Version 2 Certificate Revocation Lists (CRLs), it is possible to include a *reason code* extension on each and every entry in the CRL. Reason codes are semantics modifiers and can specify situations such as:

- Key compromise
- CA compromise
- Affiliation change (including subject name changes)
- Superseded
- Cessation of operation (the certificate is no longer needed for its original purpose)

Thus, one could conceivably decide whether a particular revocation fell into cases (a)–(c) above based on reason code, assuming one was present in the CRL and that each possible reason code could be assigned to a particular case. (This is not currently true in the PKIX specification as some reason codes have semantics orthogonal to the separation described above. For example, reason code 6, "certificateHold," means that the certificate in question has been "suspended" pending the outcome of some process. Clearly the suspended certificate could fall into any of cases (a)–(c).)

Even if reason codes were always present (which is not required by PKIX) and defined to fall into exactly one semantic category for meaning, they only solve the problem if this information is always available to the chain-building algorithm. This is clearly not the case, as any revocation that falls into case (a) (including reason code 1, "keyCompromise") may impact certificate chains that do not include the particular revoked certificate. Thus, until we come up with a better mechanism for moving revocation information around, and binding that revocation information to the proper statement being undone by the revocation, use of revocation information can add significant ambiguity to the chain-building process.

3 Mechanics

So, how does the revocation notification get propagated? Certificates come attached to all kinds of mobile objects ranging from signed forms to ActiveX controls, so considerable thinking has been put into how to avoid having revocation notices chase certificates. These include simple techniques like issuing certificates with short validity periods (certificates just expire on their own), directory-based revocation certificates (in effect an "anti-certificate") and online status checking. While each of these approaches may reduce the problem to some extent, they all rely on two independent but closely related factors:

- Good network connectivity, to either perform online status checks or frequently refresh certificates; and
- The certificate acceptor application (or person) has to care enough to check.

Technically, the "revocation mechanics problem" boils down to *on-line vs. off-line* operation and *open vs. closed* systems. In the traditional closed, on-line system, for example within an enterprise, a trusted directory is the obvious repository for key status information. But how realistic is this model in the long term? Gartner [3] estimates that by 2001, 60 million users will be relying on intermittent, low bandwidth, unreliable network connections and notebooks continue to be the fastest growing segment of the PC market? Furthermore, in the age of the Internet where are the examples of truly closed systems?

Possibly the only large-scale specimen of a closed system is the bank credit card authorization network within the United States, and in its Internet implementation (SET), there is currently no support for revocation. It has been argued that SET does not need a revocation mechanism because a SET certificate is just a proxy for a credit card that exists in the physical world. Only the account number requires verification — and legacy systems work fine for this task. Thus, if the rest of public key systems are trending towards *all open* and *often off-line* in their degenerate state, we have to make some form of revocation work within those constraints.

The most common proposed method for distributing revocation information requires an issuing certificate authority to publish a signed list of revoked certificates (a CRL). These lists may be actively distributed to users or simply made available to interested applications via cached file references, URLs, or directory queries. In any case, it is up to the individual application to determine whether it will enable CRL checking by default and how it will handle revoked certificates. In practice, applications that must rely on HTTP do not generally check CRLs by default for performance reasons, and even when such applications do perform revocation checks they only warn users of revoked certificates. These are two separable problems.

Applications' not checking for revocation should not be wholly credited to some fear of degraded performance. Robust revocation information just isn't available yet and, more importantly, the legal landscape isn't littered with case law allocating liabilities for misinformation. After all, revocation of a certificate only means that *either* one of the two parties chose to terminate the binding of the public key to some identity or authorization.

A possibly larger issue is timeliness. There is no magic in how revocation data gets to a client in an enterprise setting: it's either a signed wad of certificates or their hashes pushed at some designated time, or, if he happens to be online, the client can fetch it. In either case, there is some amount of latency. Revocation is after all a *process*, not an event, and as such implies some responsibility (if not outright liability) for correctness. Due diligence requires the expenditure of time and effort somewhere in the system.

Perhaps the whole certificate revocation concept itself is too blunt an instrument to be really useful without some re-engineering. While PKI architects are worrying about hundreds of millions of certificates floating around, systems designers are trying to figure out how to "recall" individual Java applets without impacting code size or runtime overhead. Revoking a software publisher's certificate clearly doesn't solve the problem.

Dealing with revoked certificates presents another set of problems for applications. When a revoked certificate is encountered, can an application simply prompt the user for a decision, or must it invoke some policy to determine what to do? There isn't a definitive answer yet; the only widespread applications that actually use certificates are secure channels and mail. In fact, the only people talking about public key infrastructures these days are technologists and lawyers; the business types haven't shown up yet to the party.

When those business types *do* arrive, though, the applications in which they will most likely be interested deal with something everybody cares about: money. Signed transactions with assigned monetary value and properties of non-repudiation are the obvious next step to legitimacy for public key systems and the certificates that support them. Revocation will begin to *mean* something.

The credit card system provides both bad and good examples of how things could work. *Bad* because the issuer of the "certificate" (aka the credit card) is always a party to any transaction using it, and because the certificate remains the property of its issuer (which means he can govern its use through subscriber agreements). For example, the card cannot legally be used for identification (cashing a check) and no other identification (a driver's license) may be required by a merchant to accept it.

Credit cards provide a *good* example because the card itself is only a small component of a complete (and highly mature) risk management system. When a merchant goes online to authorize a credit card he not only associates a value and a signature with a transaction, but he simultaneously *sells his own risk* to his bank for a fixed discount rate. His bank, in turn, sells that risk to the card's issuer or assumes it himself (which, by the way, is the default outside the US where communication costs are high). As soon as the merchant makes the decision to authorize the transaction, he's *committed* to sell it. The same holds true for his bank. In each case, though, the decision to sell the risk (i.e. go online) is the decision of the risk holder.

The critical point here is that the *transaction* is being traded (i.e. bought and sold) between cardholder and merchant, between a merchant and his bank, and ultimately between banks. Each participant in the value chain discounts, or more precisely, *factors* it. The only difference between the certificate acceptance model and the old credit card acceptance model is that the risk metrics of the latter are built on years of experience with billions of transactions.

So, how does risk management relate to public key certificate revocation? It's possible that the controversy surrounding such basics as whether certificates state the right relationships for keys, who should be trusted to issue and revoke them, and how far that authority extends will go on as long as there are no risk models in place to test the assumptions. If the credit card system is any indicator of how public key infrastructures will actually roll out, then revocation will start with a simplistic blacklist like the "card recovery bulletin" and evolve based on something as fundamental as who's willing to pay for revocation information. The value of the transaction, or rather the effect of fraud, will determine whether it is verified off-line or on-line and what kinds of checks are performed.

In terms of the *revocation data*, credit card companies price this kind of information based on the risk characteristics of the individual transaction. For example, a bad merchant could be blacklisted for free because that information benefits the entire infrastructure, but individual cardholder status costs something. Again, it's all about risk. For certificate revocation lists, this would imply that revocation lists and online status checking of individual certificates would be available at different prices – but in no case free!

The core question then resolves down to, "To an acceptor, does a certificate have any independent value, or does it rely on *his* trust in its issuer?" Who is trusting whom? When the issuer *is a party to the transaction*, a certificate can certainly provide authentication along with some form of authorization. What's more interesting, though, is what happens when the issuer *is not a party* to the transaction. Will the acceptor of a certificate pay any attention to what an issuer intended as its use when issued, and does the acceptor even care that the stated binding is still valid?

It is still too early in the process of gluing public key infrastructures together to predict how relying third parties will view their relationships, real or implied, with certificate issuers. As we saw with credit cards in the seventies, it all resolves to a "certificate acceptance" problem. And this is the tough infrastructure play.

In a world where certificate issuers acquire some reputation (through a brand even?) then it might follow that the possession/revocation of a particular certificate carries more weight than the possession/revocation of another. Likewise, having ten certificates against a single signature key could pass for some form of digital credit rating. Sound familiar? If we're actually facing the digital variant of a known risk model, then elaborate infrastructures will almost certainly grow up to support certificate revocation and every intermediate state on the way to it.

The most obvious of these is aggregation of certificate-linked information into some "universal" revocation service paid for by the at-risk stakeholders. The Visa and Mastercard associations are existence-proofs of this concept at work for banks, but it's not clear that banks are going to be the only risk stakeholders in PKIs. Key-holders of all kinds, along with certificate acceptors, may well pay for some trusted, instant, and always available revocation registry — but only if the risk profile of the transactions justifies its cost.

This kind of service would undoubtedly have to offer more than notice of revocation. The real value in the credit card authorization network is that it provides fraud detection as well. If it's worth the trouble for a crook to compromise a signature key, then he's bound to use it in transactions. How can these be detected and what accompanying consumer protection must be in place?

4 Summary

Credit cards took almost twenty years to gain ubiquitous acceptance. While on the surface it appears to be a simple model, it isn't. But it just works. If the analogy holds and certificate revocation tracks with credit card authorization, the technology isn't really the gating factor. It's the infrastructure required to support it. Until *what the certificate stands for can be reliably verified and undone* — and the decision to status check it on-line or accept it off-line can be done seamlessly based on risk characteristics – then issuing it in the first place probably isn't worth doing.

5 Acknowledgements

The authors wish to acknowledge the Public Key Infrastructure Group at Microsoft, members of the Public Key Infrastructure (PKIX) Working Group within the IETF, and panel members (Joan Feigenbaum, Paul Kocher, Michael Myers, and Ron Rivest) at Financial Cryptography '98 for their valuable contributions to this paper.

References

1. Warwick Ford and Michael Baum, Secure Electronic Commerce, Prentice Hall, 1997.
2. OpenPGP Working Group, Internet Engineering Task Force. "OP Formats - OpenPGP Message Format," Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer, eds., work in progress. (Draft as of March, 1998, available from http://www.ietf.org/internet-drafts/draft-ietf-openpgp-formats-01.txt.)

3. J. O'Reilley. Information Security Strategies (ISS), Research Note, Key Issue Analysis, The Gartner Group, 21 July 1997.
4. PKIX Working Group, Internet Engineering Task Force. "Internet Public Key Infrastructure: X.509 Certificate and CRL Profile," R. Housley, W. Ford, W. Polk, D. Solo, eds., work in progress. (Draft as of March, 1998, available from http://www.ietf.org/internet-drafts/draft-ietf-pkix-ipki-part1-07.txt.)