PKI: It's Not Dead, Just Resting

Peter Gutmann
University of Auckland

Abstract

Despite enthusiastic predictions in the trade press, an X.509-style PKI has so far failed to eventuate to any significant degree. This paper looks at some of the reasons behind this, examining why a pure X.509-style PKI may never appear outside a few closed, highly-controlled environments such as government agencies. On the other hand there are many instances in which situation- and application-specific uses of certificates can be employed in a manner that avoids the shortcomings of X.509's one-size-(mis)fits-all approach. The paper examines a number of these situation-specific approaches to working with certificates, and concludes with a collection of useful design rules to consider before embarking on a PKI project.

1. Introduction

After some false starts, X.509 has slowly evolved into its current incarnation as an extremely flexible PKI model, especially when its capabilities are being described by X.509 proponents. However, like other flexible objects such as rubber screwdrivers and styrofoam broadswords, it sacrifices a little bit of utility in trying to be all things to all people. Its main problem is that generic, all-purpose identity certificates issued by third-party CAs are not, in general, what the marketplace is demanding. Instead, the marketplace has developed, and continues to develop, more economically efficient, useful, and imaginative business models.

Unfortunately the original problem that X.509 certificates were designed to solve, access control to an X.500 directory, is nothing like the problem or problems that need to be solved today. This creates a severe impedance mismatch between real-world business demands and the traditional X.509 model, as real-world usage must be shoe-horned into the X.509 model in order to work with certificates. An additional complication is the fact that the X.509 model, tied to X.500/LDAP directories, hierarchical structures (with cross- and bridge-certification as proposed band-aids), offline revocation, and assorted other design decisions stemming from its X.500 origins, is unsuited for real-world use [1][2][3][4], which would instead typically use or require the use of standard business tools and methods such as relational databases, a non-hierarchical organisation, and online validity/authorisation checking.

The solution to this problem (or solutions, since there are many possibilities) is to adapt the PKI design to the real world rather than trying to constrain the real world to match the PKI. The problem-solving literature distinguishes these two approaches as "strong" and "weak" methods, where strong methods are ones designed to solve a specific type of problem while weak methods are general-purpose, one-size-misfits-all ones: "A strong method, like a specific size of wrench, is designed to fit and do an optimal job on one kind of problem; a weak method, like a monkey wrench, is designed to adjust to a multiplicity of problems, but solve none of them optimally" [5]. A related concept is that of cognitive fit, matching the tools and techniques that are used to the task to be accomplished [6][7].

The remainder of this paper discusses some of the problems inherent in the weak approach used by the standard X.509 PKI model, and proposes a variety of alternative approaches that range from simple workarounds through to designing the application to sidestep the problem entirely. Although it is sometimes claimed that many of these issues are known, this knowledge exists mostly as folklore among PKI theoreticians. No PKI standard, no RFC, no vendor whitepaper ever mentions these issues. To those not steeped in PKI lore (and folklore), they are virtually unknown. This paper exists to correct this situation.

2. Background

The pre-history of PKI goes back to Diffie and Hellman's seminal paper on public-key cryptography [8], which proposed a key directory called a Public File that users could consult to find other users' public keys. The Public File protected all communications with users by signing them, and would today be called a trusted directory. Realising some of the shortcomings of this approach, which include the potential performance bottleneck, the fact that it presents a very tempting target for attackers, and the fact that disabling access to the directory also disables users' ability to communicate securely, Kohnfelder in 1978 proposed the concept of certificates [9]. These separate the signing and lookup functions by allowing a CA to bind a name to a key through a digital signature and then store the resulting certificate in a repository. Since the repository no longer needs to be trusted and can be replicated, made fault-tolerant, and given various other desirable properties, this removes many of the problems associated with a trusted directory.

2.1 X.509

Some years after Kohnfelder's thesis was published, the use of certificates was incorporated into X.500, a global directory of named entities administered by monopoly telcos. The X.500 directory proposed a hierarchical database model (at a time when the rest of the world had abandoned this approach for the more powerful relational model), with the path through the directory being defined by a series of relative distinguished name (RDN) components that together form a distinguished name (DN). At the end of the path is an entry that contains one or more attributes containing the actual data. In order to protect access to the directory, various access-control mechanisms were proposed, ranging from simple password-based measures (with or without hashing to protect the password) to the then relatively novel approach of using digital signatures for access control. In the case of signature-based access control, it was envisaged that each portion of the directory would have CAs attached to it that would create certificates for access control purposes. An example of this configuration is shown in Figure 1.

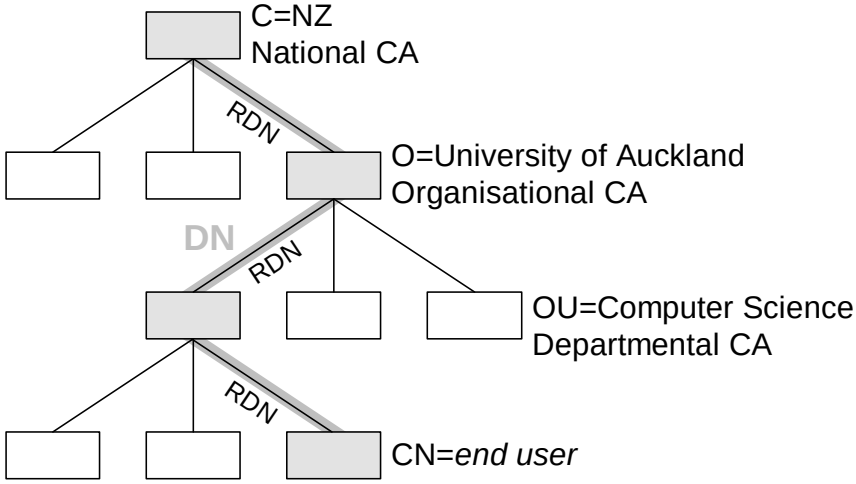

Figure 1: X.500 directory and certificate model

The original X.509v1 certificate structure very clearly shows these origins. There is an issuer DN and a subject DN (to place the certificate in the directory), a validity period, and a public key. There is no indication of whether the certificate belongs to a CA or an end entity (since this information is implicit from the directory), what the key in the certificate can be used for (there was only one use, directory authentication), what policy the certificate was issued under (again, there was only one policy, for authentication), or any of the other information without which no current certificate can be complete. This information was explicitly excluded from certificates because the only intended use was for directory access control [10]. Although no real directories of this type were ever seriously deployed, PKI designers and users have had to live with the legacy of this approach ever since.

One of the main conceptual problems of this approach was that it turned simple public keys into capabilities, tickets that control access rights and that an end entity can use to demonstrate their access to an object. Capabilities have the problem that they make access review (deciding who has access to what, since a capability can be easily passed on to others) and revocation extremely tricky [11]. A standard (if clunky) way to address this issue is to change the name of the object referred to in the capability in order to render it invalid. If selective revocation is requires, this can be performed by using multiple alias names for an object, one per capability.

X.500 tried to address the revocation problem through certificate revocation lists (CRLs), a digital analogue of 1970's-era credit-card blacklists, which in turn were modelled after even earlier cheque blacklists. The standard was rather vague as to how this was supposed to work, and left all the details up to the CA and/or directory. Other options that were provided included simply replacing the revoked certificate with the new one, notifying the certificate owner "by some off-line procedure", and various other approaches [12]. The issue of revocation of capabilities/certificates is in fact so thorny that an entire section of this paper has been devoted to it.

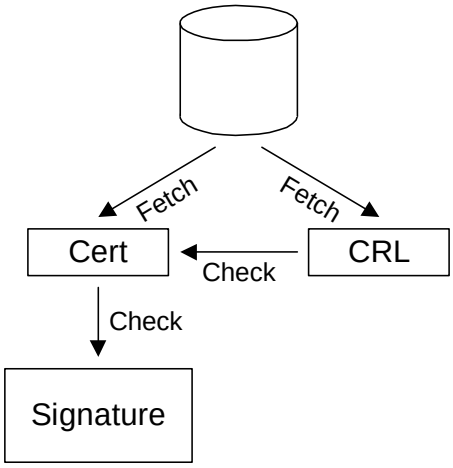

Figure 2: X.509 certificate usage model

The final form taken by the X.509 certificate usage model, in this case for general digital signature verification rather than direct directory authentication, is shown in Figure 2. The relying party, wishing to verify a signature, fetches a certificate from a repository, fetches the CRL from the same or another repository, checks the certificate against the CRL, and finally checks the signature against the certificate. Originally the repository was intended to be the X.500 directory, in practice as applied today it's anything from flat files, a relational database, Berkeley DB, and the Windows registry, through to a hardcoded local certificate or a certificate included with the data it authenticates. Occasionally, it really is fetched from a directory.

2.2 Identity vs. Authorisation Certificates

In abstract terms, an X.509 certificate can be thought of as a signed *n*-tuple that asserts a predicate $p(x_1, x_2, x_3, \ldots x_n)$ over the fields it contains. Unfortunately, there is no way to indicate exactly what that predicate is. Some examples of required predicates might include has_read_access_to or can_withdraw_money_from, while the only real predicate that an X.509 certificate can offer is the tautological is_an_X509_certificate [13].

SPKI certificates, in contrast, assert a user-defined predicate specified by the issuer of the certificate, so that the relying party can make meaningful authorisation decisions based on the contents of the certificate [16][17]. These predicates may be arbitrarily complex, going beyond the basic can_withdraw_money_from example given above to more specialised forms such as can_withdraw_money_from_account_X_up_to_$Y_per_day, a predicate useful for handling ATM withdrawals. In addition, SPKI contains the necessary mechanisms for automatically evaluating and processing these authorisation decisions. The X.509 world, consisting purely of identity certificates, provides no equivalent for this capability.

2.3 Problems with Identity Certificates

The abstract model given in section 2.1, while simple, hides a great many problems. The biggest of these is reflected in the simple phrase "fetches a certificate from a repository". Since the concept of a global distributed directory (or even a less ambitious local directory) was never realised, there's no clear idea where to fetch a certificate from, and if you have a certificate there's no clear idea where to fetch its CRL from. This is a well-known PKI issue, and is termed the "Which directory?" problem. The solution that was adopted, and that works reasonably well in practice, was to include any certificates that might be needed wherever they might be needed. For example, an S/MIME signature usually includes with it all the certificates needed to verify it, and an SSL server's communication to the client usually includes with it the certificates needed to protect those communications. Obtaining a new certificate is handled either by out-of-band means (for example mailing a user and asking for the certificate) or by a lazy update mechanism in which applications keep copies of any certificates they may come across in case they're useful in the future. This approach nicely solves the certificate distribution problem, at the expense of moving the load across to an even harder problem, the certificate revocation problem, which is covered in section 3.

Even if the user knows which directory to look in, there's no way to determine which DN should be used to find a certificate, or which of a number of identical names you're searching on belongs to the person whose key you're

interested in. This is another standard PKI issue, the "Which John Smith?" problem. As a result, the post-X.500 form of X.509 which is in use today turns a key distribution problem into an equally intractable name distribution problem. Although it's possible to disambiguate names through ad hoc measures such as adding the last four digits of a user's social security number to the DN (as was used in one project which found that there were people with the same first, middle, and last name in the same OU), the result is a DN which is unique but useless for name lookups since no third party will know how to construct it. X.501's name design criteria would have made things even more difficult by more or less disallowing name forms that were amenable to automated processing [14], but the issue was resolved in an amicable manner by everyone ignoring the design criteria.

The naming problem has been solved in a similar manner both within the X.509 framework and outside it with other certificate designs such as PGP and SPKI/SDSI. SDSI realised that globally unique names would never work except in a few special cases, and that all that was needed was a name that was meaningful within a limited community [15]. For example while the name "John Smith" is meaningless when the community is "USA", it's meaningful when the community is restricted to "People who are allowed to log on to this file server". SPKI then paired SDSI names with the concept of using the public key as an identifier to provide global uniqueness [16][17].

PGP solved the problem in a less rigorous, but equally effective, manner. Users were allowed to choose any kind of identifier they wanted for certificates, which generally consisted of an email address and a user name to go with the address. Since the email address was unique, and PGP was used mostly for email communications, this worked out reasonably well. For identifying keys internally, PGP also used the public key.

Both PGP and SPKI in effect employ the same conceptual model, using a locally meaningful identifier within a specific domain. If PGP's case the domain is implicitly set to "email addresses", with SPKI it can be implied by the restricted community in which the certificate is used, for example to authenticate to a particular server. Other examples of such schemes are credit card numbers, bank account numbers, and social security/tax identifiers that, although they may be easily confused when presented without any disambiguating context, are meaningful in the particular domain of application.

X.509, on the other hand, couldn't adopt such a simple solution. Since X.509 was intended to be all things to all people, taking a restricted application domain and creating an application-specific solution for it was out of the question. On the other hand X.500 DNs fitted no real-world domain, and weren't understood by the vast majority of people using them. The result was that, except for a few carefully managed, centrally-controlled schemes, users employed a de facto local naming scheme in which they crammed anything they felt like into the DN, so that it became a (mostly) meaningless blob whose sole purpose was to uniquely identify a public key (the only generally meaningful element might be, as with PGP, an email address or URL and a user name). As a somewhat extreme example of this, the author recently encountered a collection of certificates that appeared to originate from Sweden because the administrator who set up the system had copied a magic DN formula from something else that was (presumably) located in Sweden. In the resulting certificates, only the common names and email addresses made any sense.

X.509v3 also added an alternative identifier, again based on the public key.

All three approaches, two by design and one by coincidence, therefore eventually found the same solution of using a locally unique identifier such as an email address and user name within an (implicit or explicit) specific domain, and a (probabilistically) globally unique blob derived from the public key. The identification issue was therefore mostly resolved, leading to the next problem: Revocation.

3. Revocation

As has already been mentioned earlier, certificate revocation is something that doesn't really work, particularly when attempts are made to implement it in the manner envisaged in X.509 [18]. The main reason for this is that the use of CRLs violates the cardinal rule of data-driven programming which is that once you have emitted a datum you can't take it back [19]. Viewing the certificate issue/revocation cycle as a proper transaction-processing (TP)-style transaction, the certificate issue becomes a `PREPARE` and the revocation a `COMMIT`, however this means that nothing can be done with the certificate in between the two because this would destroy the atomicity and consistency properties of the transaction (TP mechanisms provide a useful analytical model for examining certificate management operations, and will be the subject of a future paper [20]). Allowing for other operations with the certificate before the transaction has been committed results in non-deterministic behaviour, with the semantics of the certificate being reduced to a situation described as "This certificate is good until the expiration date, unless you hear otherwise" [21] which is of little value to relying parties. It is for this reason that digital signature laws such as the UNCITRAL Model Law on

Electronic Signatures [22] don't even touch these revocation-related issues. This problem is complicated by the existence of two schools of thought on how revocation information should be reported, the accuracy school, which holds that the indicated revocation time should be the time reported by the user (even if it leads to backdated revocations), and the consistency school, which holds that the indicated revocation time should be the time of CRL issue (even if it leads to losses due to delays in marking a certificate as revoked). Neither of these options are particularly palatable.

3.1 Problems with CRLs

CRLs have a number of problems that make them difficult to work with and unreliable as a certificate status propagation mechanism (at a recent conference that covered PKI implementation experiences, speakers for large organisations such as Boeing and J.P.Morgan specifically singled out revocation as a major problem area [23][24]). The general problem that needs to be solved, and the one that CRLs don't really solve, is that critical applications (which usually mean ones where a lot of money is involved, but can be extended arbitrarily to cover whatever the relying parties consider important enough) require prompt revocation from a CRL-centric world view, or require real-time certificate status information from a more general world view. Attempting to do this with CRLs runs into a number of problems.

The CRL concept is based on the credit-card blacklists that were used in the 1970's in which credit card companies would periodically print booklets of cancelled card numbers and distribute them to merchants. The merchants were expected to check any cards they handled against the blacklist before accepting them. The same problems that affected credit-card blacklists then are affecting CRLs today: the blacklists aren't issued frequently enough to be effective against an attacker who has compromised a card or private key, they cost a lot to distribute, checking is time-consuming, and they are easily rendered ineffective through a denial-of-service attack (to ensure that your card is never blocked through a blacklist, make sure the blacklist is never delivered).

When a CA issues a CRL, it bundles up a blacklist of revoked certificates along with an issue date and a second date indicating when the next blacklist will become available. A relying party that doesn't have a current CRL is expected to fetch the current one and use that to check the validity of the certificate. In practice this rarely occurs because users and/or applications don't know where to go for a CRL (the "Which directory" problem covered in section 2.3), or it takes so long to fetch and check that users disable it (in one widely-used application enabling CRL checking resulted in every operation that used a certificate being stalled for a minute while the application groped around for a CRL, after which it timed out and processing continued as before), or they simply can't be bothered and put things off until they can't do anything any more (assuming they even know the significance of a revoked certificate and don't just interpret the failure to perform as an error in the application) [25]. Other applications treat all certificates from a CA as invalid once the current CRL has expired, possibly a move to force users to fetch new CRLs rather than just ignoring the whole issue, although in this case the problem was solved when users discovered that if they simply deleted the CRL their application would "work" again. In addition, most CRL-using applications will continue to use the current CRL (no matter how out of date it is) until its expiry date, turning the revocation check into an empty ritual in which the same obsolete information (the default Microsoft CRL interval, for example, is a full week) is consulted again and again to determine a certificate's "current" validity [26].

Moving beyond the user interface problems (which will inevitably affect any certificate status mechanism, no matter how sophisticated), there are a number of practical issues with CRLs. In order to guarantee timely status updates, it's necessary to issue CRLs as frequently as possible, however the more often a CRL is issued the higher the load on the server that holds the CRL, on the network over which it is transmitted, and (to a lesser extent) on the client that fetches it. Issuing a CRL once a minute provides moderately timely revocation (although still not timely enough for cases such as Federal Reserve loans where interest is calculated by the minute) at the expense of a massive amount of overhead as each relying party downloads a new CRL once a minute. This also provides a wonderful opportunity for a denial of service attack on the CA in which an attacker can prevent the CRL from being delivered to others by repeatedly asking for it themselves. On the other hand delaying the issuance to once an hour or once a day doesn't provide for any kind of timely revocation.

Another problem encountered with CRLs is that there are no real mechanisms available for charging relying parties for revocation checking. When a CA issues a certificate the user is charged a fee by the CA, with the amount being charged depending on how much checking the CA does before it issues the certificate. On the other hand CAs are expected to create and issue CRLs for free. Neither the user nor the CA are in a good position to know how often the certificate will have to be validated, or by whom, or what degree of latency will be acceptable. This is not merely a

non-incentive for CAs to pay much attention to CRLs but an active disincentive, since creating and distributing them requires processing time, one or more servers, and significant amounts of network bandwidth (CRLs can become quite large, and many clients can fetch these large CRLs). For PKIs sized at tens of thousands of users, multi-megabyte CRLs are not uncommon, and the potential size of CRLs for hypothesised national CAs/PKIs is unpleasant to contemplate. For example the Johnson and Johnson PKI, with 60,000 users, had a CRL over a megabyte in size after its first year of operation [27], requiring that users fetch several million times more data than necessary for any certificate check (the entire CRL had to be fetched to obtain the effect of a boolean valid/not valid flag). The problem was "solved" in this instance by only issuing a CRL once a week and caching old copies of the CRL to turn it into the ritual check described earlier, both being relatively common approaches whenever this problem raises its head. The problem can also be addressed by protocols such as OCSP (see section 3.3), which allow for status queries to be signed by relying parties, thus allowing the CA to bill them for the query.

A more general form of the problem of who pays for revocation is the problem of who pays for general PKI operations. For example in the SET PKI the issuing bank carries the cost (and associated risk) of handling certificate enrolment and issue while the acquiring bank obtains all the benefits. While this could be mitigated to some extent by sharing the cost across both banks, it was one of the many nails in SET's coffin. Another approach to the problem is to charge a per-certificate-use transaction fee to cover the cost of running the PKI. In one project run by the US General Services Administration, the cost ranged from 40¢ to $1.20 each time the certificate was used [1], a considerable disincentive for the use of certificates. In Sweden, certificates issued to citizens are free, but a validity check costs € 0.25, the price apparently inspired by the cost of a postage stamp [28]. This problem was foreseen as long ago as 1994 in a MITRE study which estimated that most of the costs of running a PKI arose from the cost of managing revocation [29]. The approach that is currently more generally used is to bury the per-transaction costs elsewhere, although how well this will work when the PKI project leaves the pilot stage remains to be seen.

One approach that has been proposed to address this is to view the issue in terms of a quality-of-service (QoS) problem in which users can pick and choose the PKI services and capabilities that they require based on the value and constraints of their transactions [30]. With this approach, certificate users are allowed to tune their PKI requirements based on parameters such as certificate freshness, time for validation, cost to validate, time to issue, and time to revoke/invalidate, rather than having to use the normal take-it-or-leave-it approach in which a single solution (with the QoS target being "whatever the X.509 standard says") is expected to meet all needs. This is one of the few proposals utilising the X.509 framework that approaches the problem as an exercise in meeting user requirements rather than of forcing a fixed standard onto whatever the user wants to do.

3.2 Proposed Workarounds for CRL Shortcoming

A problem with CRLs with built-in lifetimes is that they all expire at the same time, so that every relying party will fetch a new CRL at the same time, leading to huge peak loads whenever the current CRL expires. In effect CRLs contain their own built-in distributed denial-of-service (DDOS) attack. One proposed solution to this problem is to stagger CRL expiry times for different classes of certificates so that they don't all expire at the same time, although scheduling the CRL times while still providing some form of timeliness guarantee is sure to prove a tricky exercise. Another approach is to over-issue CRLs so that multiple overlapping CRLs exist at one time, with a relying party who fetches a CRL at time n being fed a CRL that expires at time $n + 5$ and a relying party who fetches it at time $n + 2$ being fed one that expires at time $n + 7$. Assuming that CRL fetching patterns follow a certain distribution, this has the potential to lighten the peak loads on the server somewhat [31] at the expense of playing havoc with CRL semantics.

Yet another approach is to segment CRLs based on the perceived urgency of the revocation information, so that a CRL with a reason code of "key compromise" would be issued more frequently than one with a reason code of "affiliation changed". This segmenting doesn't reduce the request rate but can reduce the amount of data transferred so that it takes less time to service an individual request. Segmenting CRLs has the side-effect that it introduces security problems since an attacker who has performed a key compromise can use the compromised key to request that a revocation for it be placed in a low-priority CRL, ensuring that the key is both regarded as safely revoked by the CA but at the same time will still be valid until the next low-priority CRL is issued at the end of the day or week [32]. Solving this problem requires very careful protocol design and extensive amounts of checking and safeguards at every step of the revocation process.

Yet another approach involves delta CRLs, which have one large long-term CRL augmented by a number of smaller, short-term CRLs that modify the main CRL. There appears to be little real-world experience in the use of any of these mechanisms, although discussions on PKI mailing lists indicate that attempts to implement them will prove to be an

interesting experience. Beyond this there exist even more approaches to the problem, tinkering with CRL mechanisms is a popular pastime among PKI researchers.

One possible solution to this problem is used in SET, which takes advantage of the fact that certificates are tied to credit cards to avoid the use of CRLs altogether. SET cardholder certificates (which are expected to be invalidated relatively frequently) are revoked by revoking the card that they are tied to, merchant certificates (which would be invalidated far less frequently) are revoked by removing them from the acquiring bank's database, and acquirer payment gateway certificates (which would almost never be invalidated) are short-term certificates that can be quickly replaced. This process takes advantages of existing mechanisms for invalidating certificates, or designs around the problem so that revocation in the manner expected of other PKIs is never needed. A similar type of scheme that designs around the problem is used in Account Authority Digital Signatures (AADS, ANSI X9.59), a simple extension to existing account-based business infrastructures in which public keys are stored on a server that handles revocation by removing the key. To determine whether a public key is valid, the relying party fetches it from the server [33][34][35]. The original (and most widely-used) ssh key management system works along similar lines, tying keys to Unix user accounts [36][37].

SPKI takes a slightly different approach by implicitly making validation part of the certificate-processing operation. SPKI prefers revalidation, representing a positive statement about a certificate's validity, to CRLs, which represent a negative statement, since positive assertions are much more tractable than negative ones (consider, for example, the relative difficulties of proving "Aliens exist" vs. "Aliens don't exist"). This transforms the X.509 assertion "This certificate is good until the expiration date, unless you hear otherwise" into "This certificate is good until this time". The time interval in this case is far less than the traditional year granted to X.509 certificates (or 20 years for major CA certificates [38]), but is instead based on a risk analysis of potential losses due to excessively long certificate validity periods. In order to avoid clock skew problems (which is virtually guaranteed in Windows machines, which can be out by minutes, hours, days, or even years in extreme cases [39]), SPKI also allows for one-time revalidations which guarantee that the certificate is valid for a single transaction (other applications of this concept are covered in section 5.1).

A concept that has been proposed for avoiding revocation is to use extremely short-lived certificates [21][40]. This is explicitly addressed in SPKI, and can be implemented in X.509 by fiddling with certificate validity periods. Apart from requiring precisely synchronised clocks for all principals, it doesn't quite avoid the revocation problem since there's still a need for some form of revocation/prevention of revalidation if a private key is compromised [41].

Another approach that can be taken to avoid revocation is to try to address the problem in an application-specific manner. Consider the case in which authority-to-individual communications such as for tax filing purposes need to be secured. The obvious solution is to use S/MIME or PGP-secured email. A much simpler solution is to use an SSL web server with appropriate access control measures. "Revocation" is handled by disabling access for the user and is therefore instantaneous (there's no CRL propagation delay), consistently applied (you don't have to worry whether the client software will check for revocation or not), and effectively administered (from the server containing the data, not an external CA). Other issues such as the "Which directory" and "Which John Smith" problem also disappear, since everyone knows who the tax department is and the tax department knows who its "users" are via the domain-specific ID described in section 2.3.

SSL itself already provides an example of revocation being handled in an application-specific manner. Using the X.509 CRL reason codes as usage cases, "key compromise" is unlikely to be useful unless the attacker helpfully informs the server administrator that they've stolen their key, "affiliation changed" is handled by obtaining a new certificate for the changed server URL (this is in effect the capability-revocation approach described in section 2.1, although it becomes somewhat impractical when the subject is a person), "superseded" is handled in the same way, and "cessation of operation" is handled by shutting down the server. In none of these cases is revocation of much use. It is for this reason that the whole SSL certificate management process has been referred to as "certificate manufacturing" rather than PKI [42], since the only real infrastructure component present is the one that, once a year, exchanges the client's credit card number for a collection of bits.

3.3 Online Revocation Authorities

A recent solution that has been proposed for the revocation checking problem is the Online Certificate Status Protocol (OCSP) [43]. The OCSP model provides a responder that can be queried online and that relies on CRLs or other, unspecified certificate management mechanisms to provide revocation information about a certificate. The model for OCSP usage is shown in Figure 3. Compare this with the CRL-based model of Figure 2, in which obtaining up-to-date

certificate status information could entail downloading a CRL with thousands of entries every few minutes, delta CRLs and other kludges notwithstanding. This represents an extraordinarily inefficient means of disseminating revocation information, since it's necessary for a relying party to repeatedly fetch a huge number of totally irrelevant entries in order to obtain status information for the one certificate that they care about, at the same time placing a heavy load on the CA that sources the CRLs [44]. In DNS terms, this operation is like doing a full zone transfer for each address lookup operation.

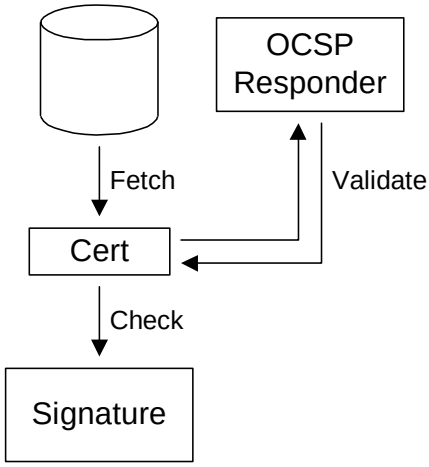

Figure 3: Certificate usage model with OCSP responder

In effect the OCSP responder functions as a special-purpose CRL creation mechanism, which solves many of the problems inherent in monolithic CRLs by creating a one-off, fresh, single-entry CRL in response to a query. This property also comes with a cost, however. Instead of being able to prepare a CRL as a background, offline operation, the CA must now perform a certificate lookup/check and OCSP pseudo-CRL creation operation for each query (although alternative, somewhat more lightweight approaches have also been proposed [45]). In order to make OCSP economically feasible, it's necessary for the CA to charge for each revocation check in order to cover their costs. This is provided for in OCSP by signing requests to identify the sender for billing purposes, an approach used in the Identrus PKI.

This leads to an immediate split in revocation checking mechanisms based not on abstract technical concepts but on real-world, economic terms. It also illustrates a special case in which CRLs are, despite all of their shortcomings, valuable. This occurs when a revocation check is, quite literally, worthless. For example consider signed executables such as ActiveX controls. A vendor buys a (relatively) cheap code-signing certificate from a commercial CA and signs all the software they like, which is distributed over as many machines as they can get it to. With an installed base of some hundreds of millions of Windows machines, all containing hundreds of ActiveX controls, the costs involved in providing any sort of useful online revocation checking service for code-signing certificates would be astronomical. Low-assurance email certificates are another example where serious revocation handling isn't financially feasible.

As a result, there is a strong financial incentive for CAs to do as little as possible in terms of handling revocations for these certificates beyond paying lip service in the form of an infrequently-issued CRL located at some semi-documented location. In scenarios such as this, CRLs are perfect. Another special-case potential use for CRLs is as a form of zone transfer from one repository to another when a large number of certificates need to be revoked at once, for example due to a company division closing down. A final special-case situation in which CRLs are useful is when there exists some statutory or contractual obligation to use them, so that a relying party needs to be able to claim CRL use for due diligence purposes or to avoid liability in case of a dispute.

3.4 Problems with OCSP

Although OCSP has many advantages over CRLs, its major shortcoming is that instead of providing a simple yes/no response to a validity query, it instead uses multiple, non-orthogonal certificate status values because it can't provide a truly definitive answer. The possible responses to a query are "not-revoked" (confusingly labelled "good" in the OCSP specification), "revoked", and "unknown", where "not revoked" doesn't necessarily mean "OK" and "unknown" could mean anything from "this certificate was never issued" to "it was issued but I couldn't find a CRL for it". This leads to

the peculiar situation of a mechanism labelled an online certificate status protocol that, if asked "Is this a valid certificate" and fed a freshly-issued certificated can't say yes, and if fed an Excel spreadsheet or an MPEG of a cat, can't say no. Contrast this with another online status protocol in use for the last decade or so, credit card authorisation, for which the returned status information is a straightforward "Authorised" or "Declined" (with an optional side order of reasons as to why it was declined).

This vagueness is at least partially the fault of the original CRL-based certificate status mechanism since a CRL can only provide a negative result, so that the fact that a certificate is not present in a CRL doesn't mean that it was ever issued or is still valid. For some OCSP implementations the "I couldn't find a CRL" response may be reported as "not revoked/good", or will be interpreted as "good" by relying parties since it's not the same as "revoked" which is assumed to be "not good" (opinions on the exact semantics of the various responses vary somewhat among implementers).

The fundamental problem with blacklist-based approaches such as CRLs and OCSP is that they ask entirely the wrong question — "Has this been revoked?" — when in fact what's required is an answer to the question "Is this currently valid?". This problem is not something that can be easily fixed, because that's the only question that a blacklist is capable of answering. SPKI, in contrast, gets it right with its use of a certificate revalidation process.

A related problem (which affects CRLs more than OCSP itself) is that, as with their 1970s credit-card-blacklist cousins, they represent an inherently offline operation in an almost completely online world. Credit card vendors realised this when they went to full online verification of each transaction in the 1980s. As with online credit-card checks, a response to a query about a certificate only needs to return a simple boolean value, "The certificate is valid right now" or "The certificate is not valid right now" (along with reasons for why it's not valid, there may be other, more complex additional requirements that are covered elsewhere [46]). Unfortunately, OCSP is incapable of doing this.

In an attempt to break free of this restriction, proposed protocols such as the Simple Certificate Validation Protocol (SCVP) [47] have appeared that make use of a server that either acts as a dumb repository or as a full certificate chain validation system in which the relying party submits a collection of certificates and optional ancillary information such as policy information, and the server indicates whether the chain can be verified up to a trusted root. This service seems to be aimed mostly at thin clients who can't perform any path validation themselves, although there is some debate among proponents as to its merits relative to approaches like OCSP. A more stealthy approach is taken in the Data Validation and Certification Server Protocols (DVCS) [48], which provide as part of the protocol a facility more or less identical to SCVP but that has escaped controversy by presenting itself as a third-party data validation mechanism rather than a certificate status protocol.

Alongside the warring OCSP and SCVP camps and DVCS stealth approach, a number of other certificate status protocols have appeared and disappeared over time, including the Integrated CA Services Protocol (ICAP) [49], the Real-Time Certificate Status Protocol (RCSP) [50], the Web-based Certificate Access Protocol (WebCAP) [51], Peter's Active Revocation Protocol (PARP) [52], the Open CRL Distribution Process (OpenCDP) [53], the Directory Supported Certificate Status Options (DCS) [54], and many, many others (the protocol debate has been likened to religious sects arguing over differences in dogma [55]). Since the author believes in freedom of religion, this paper doesn't contain any comments on particular protocols, except to note that there are enough to choose from to suit anyone's tastes.

4. Certificate Chains

Once we go beyond single certificates, things become even trickier than what was presented above. The initial problem we encounter with multiple certificates is that of constructing a path from a leaf certificate to a trusted top-level CA and validation of certificate chains once the path has been built. This can become almost intractable in the presence of cross-certification in which CAs in disjoint hierarchies cross-certify each other, since there can be multiple certificate paths leading from a given leaf certificate, all with different semantics. Certificate paths can now contain loops, and in extreme cases the semantics of a certificate can change across different iterations of the loop [56] (whether certificate paths are Turing-complete is an open problem). Cross-certification turns the hierarchy of trust into the spaghetti of doubt, with multiple certificate paths possible from leaf to roots, as shown in Figure 4. Unfortunately the use of cross-certificates is currently being advocated as a means of tying together disparate PKI projects. An alternative, bridge CAs (also known as overseer CAs when they were initially developed for the Automotive Network Exchange (ANX) program), avoids this problem to some degree by adding a single super-root that bridges two or more root CAs.

In addition to this explicit cross-certification, most current PKI software employs a form of implicit cross-certification in which all root CAs are equally trusted, which is equivalent to unbounded cross-certification among all CAs. This means that, for example, any certificate can be trivially replaced by a masquerader's certificate from another CA, since both CAs are equally trusted [57]. Correcting this problem is extremely difficult: Netscape 6 contains just under a hundred built-in root CAs requiring around 600 mouse clicks to disable, and MSIE 6 contains over a hundred built-in certificates that require around 700 mouse clicks to disable. Many of these CAs are completely unknown, follow dubious practices such as using 512-bit root keys or keys with 40-year lifetimes, appear moribund, or have had their CA keys on-sold to various third parties when the original owners went out of business [58]. With the implicit universal cross-certification that exists in this environment, the security of any certificate is reduced to that of the least trustworthy CA, who can issue a bogus certificate to usurp the legitimate one, at the same level of trust.

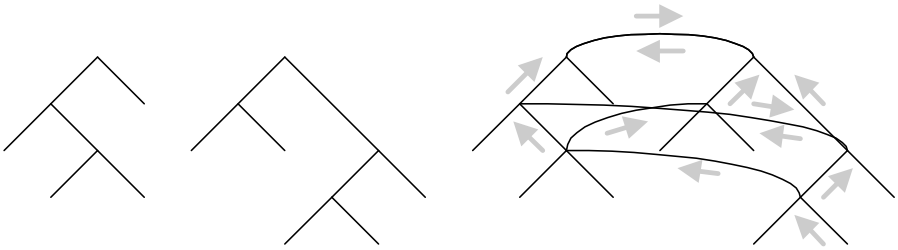

Figure 4: Certificate hierarchy (left) with cross-certification (right)

Going beyond the basic issue of path construction, the revocation checking problem also experiences a huge increase in complexity, jumping from a single lookup and check to being proportional to the square of the depth of the issuance hierarchy [59]. This is illustrated in Figure 5, which depicts a hierarchy of depth 3. In order to verify a certificate, the relying party needs to fetch the certificate for its issuing CA and its CRL. In order to verify the CRL it needs the certificate of the revocation authority (RA), which in turn has its own certificates and CRLs. Checking a simple certificate via the mechanisms envisaged in X.509 thus results in an exponential growth in complexity, with many certificate/CRL fetches and signature checks being necessary to verify a single certificate (in practice the growth isn't fully exponential, since the number of authorities drops as we get closer to the root). In the original X.509 design with a single hierarchy and everything tied to a directory this wasn't a problem, but with current approaches it requires alternative solutions such as revocation and validation authorities of the kind discussed further on.

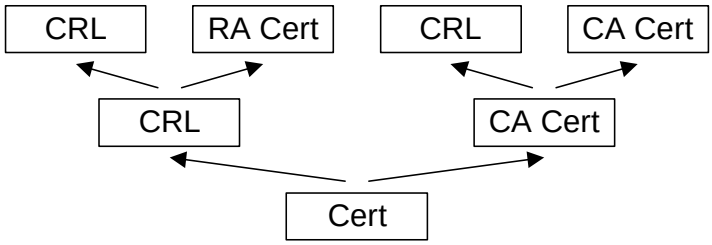

Figure 5: CRL-based revocation checking for a chain of length 3

The solution to this problem, again provided by OCSP, is to use an access concentrator or gateway that takes an entire chain and farms the revocation checking out to one or more OCSP responders and/or CRL-based implementations. The relying party communicates with a single gateway that does all the work of revocation checking, as shown in Figure 6. This is the approach used by Identrus, who use gateways called transaction coordinators (TCs) to provide real-time certificate status information (along with many other things) to its members. Although the diagram depicts an OCSP gateway talking to a number of OCSP responders, the actual implementation could take any of a number of forms, for example in the Identrus case the TC could in turn talk to further TCs that front for responders. Since the relying party sees only the OCSP gateway, they're not concerned about how the actual checking is done. Another advantage of the gateway approach, at least in the Identrus case, is that it makes billing easier. Since Identrus charge for each transaction, which comes with certain guarantees that are absent with general commercial CAs, users want to submit an entire collection of certificates at once, and be charged once, rather than being charged individually for each certificate.

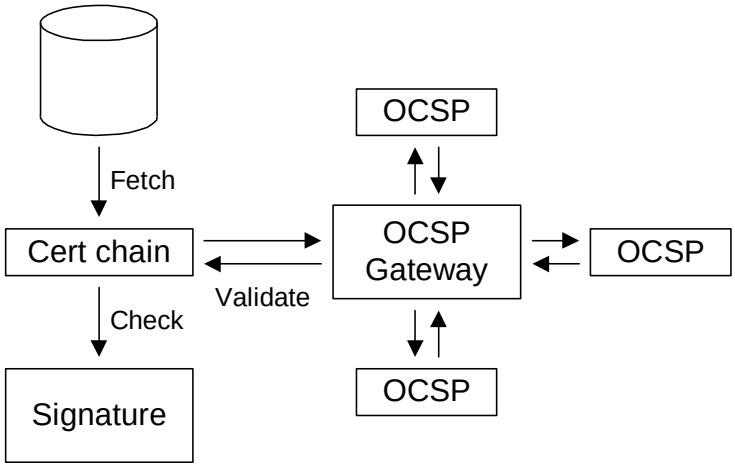

Figure 6: Offloading the revocation checking process

Working with certificate chains also leads to an extreme case of the "Which directory" problem, in which it's not just necessary to locate a single directory but requires locating multiple directories (and locations in directories) for the different CA/RA certificates and CRLs. Some attempts have been made to work around this by encoding location URLs in certificates, but this information is usually not present and when it is present is frequently wrong [60]. The problem is made even worse in the presence of cross-certificates, since it's now necessary to locate all certificates on all possible paths. This problem is, in general, intractable since it's not possible to determine whether further paths exist based on certificates in as-yet undiscovered repositories.

A proposed solution to this problem has been the use of path construction servers (PCS), a type of smart repository that offloads the chain building process from the end user [61][62]. While effective in theory, it simply offloads all of the problems to the PCS server, while adding the problem of communicating certificate selection criteria (acceptable policies, CAs, certificate types, path lengths, and so on) from the client to the server. This situation doesn't occur when the chain building is performed at the client, and requires a means of specifying complex constraints for use when building the chain (this is currently an unsolved problem). The main benefit of a PCS server is fast response time, since it can function like a web crawler and cache as much information as possible for use with future queries.

Extending this even further is the concept of a path validation server (PVS), which offloads the validation process as well as the path construction process [61]. This extends the problems of PCS a step further, since even more constraint information must be communicated to the server. Both of these approaches, which effectively implement PKI-crawler analogues to standard web-crawlers, also suffer from the usual "Which directory" problem. Finally, even more so than OCSP, the charges incurred by this type of operation are likely to be substantial.

5. Closing the Circle

Gordon Bell once observed that the most reliable components of a system are those that aren't there. Based on this principle, if we could remove the need to perform revocation checking (at least in the X.509 sense), we could solve a significant portion of the PKI problem.

One way to address this is through the use of a PKI community (also known as a community of interest or COI), a restricted group of participants that agrees to play by certain rules. In this way it's possible to quantify risk reliably enough to make meaningful warranties to relying parties, either by requiring that all participants follow certain rules or by executive fiat/government decree. In contrast in an open environment in which a certificate represents general-purpose ID, the issuing CA is exposed to virtually unlimited liability unless they specifically disavow liability for their certificates, as many public CAs indeed do. Disavowing responsibility for identity in identity certificates seems somewhat ironic.

Since whoever accepts the risk can dictate the technology which is used, the PKI models used in these closed communities can differ radically from the traditional X.509 design. Communities are likely to be small (relative to the size of the entire Internet) and tied together by a common interest or policy requirements. The automated clearing house (ACH) network is an example of such a community which is tied together by both a very stringent set of

operating requirements and the operating rules of the ACH system. Other communities are systems such as Swift and Bolero, in which the members sign up to the rules of the community and are then obliged contractually to stand behind signatures made with their private keys. Further communities are built implicitly when a group of users work towards a common goal, for example the European 3-Domain SET initiative came about when several European banks and credit card vendors realised that the complexity and high cost of a distributed PKI was best addressed by creating a centrally-controlled and administered system [63]. Various other COI mechanisms are also being studied [64]. These communities manage risk by only admitting members who can afford to carry it, and by extension who have the means to manage it.

One final (unwritten) benefit of operating within a closed community is that, by signing up to a fixed, hardcoded set of rules that everyone agrees to play by, it's possible to avoid the ongoing feature creep inherent in PKI technology in which an endless flow of standards drafts and proposals results in a constant scramble by implementers and users to catch up. At some point the feature set can be frozen, everyone agrees to work within the given framework, and the PKI can be realised.

5.1 Bypassing Revocation Checking

Within a community, the problem of revocation can be addressed by collapsing the certificate-fetch-and-validation process even further than provided for through OCSP. Observe that what we're doing with both CRLs and OCSP is first fetching a certificate, and then immediately fetching revocation (or validity) information for the same certificate. Instead of first performing a query for a certificate and then immediately performing a second query to determine whether the certificate we just fetched was any good or not, we can combine the two into a single fetch of a known-good certificate from a server (or servers) known by the community. This process is shown in Figure 7. Although this now requires a trusted server, this is no different to OCSP, which also requires a trusted server to eventually perform the same function, but in a more roundabout manner. Note that this differs from the PCS concept in that it relies on a server operating within the rules of a community to provide "good" certificates to relying parties, while a PCS server attempts to construct a known-good path through an arbitrarily-complex PKI-crawling process.

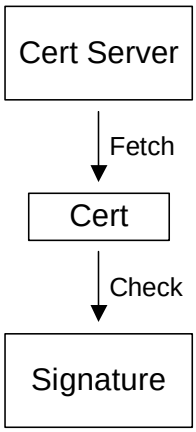

Figure 7: Avoiding revocation checking

This goes back to the original 1970s concept of public-key distribution in which keys were to be held in public directories or key distribution centres (KDCs) that handed out only known-good keys in response to queries [65]. A similar concept has been hypothesised for use in a general-purpose (rather than closed-community) PKI in the form of a trusted X.500 directory that stores known-good certificates for reference by relying parties (assuming they can figure out whether they're supposed to be looking for a certificate or userCertificate or cACertificate or crossCertificatePair or whatever other tag the directory might store certificates under). This approach, which relies on the existence of a reliable, scalable distributed X.500-style directory system appears to exist only in theory (at least one attempt at creating this type of trusted directory has been made in Germany, but the result is reportedly neither scalable nor reliable).

The process of turning two queries into one can be optimised even further. If bandwidth is a concern then instead of always fetching a certificate we can submit a hash of the certificate. If it's still valid the server returns a simple acknowledgement, if not it returns the replacement certificate or an indication that nothing is available. This concept is

similar to that of undeniable attestations [66], with the main difference being that the undeniable attestation scheme uses cryptographic mechanisms such as authenticated search trees whereas the approach given here relies on the use of established mechanisms such as security policies and auditing agreed upon by the members of the community and/or that have been determined sufficient to carry evidentiary weight in court. For example the US has a rule of evidence called the Business Records exception that allows records "kept in the course of a regularly conducted business activity" to be treated as evidence rather than mere hearsay like other computer-generated data [67],that comes with a large amount of legal precedent attached to it. It's probably preferable to rely on this type of mechanism than to become the test case for PKI.

5.2 Bypassing Certificates

In practice we can go even further than simply collapsing two queries into one. Since all we're really interested in is the public key, we can request only a copy of the appropriate key needed to perform an operation such as verifying a signature. This returns us to the original Public File approach of Diffie and Hellman. In fact this exact technique is already in use today by almost all certificate-using applications, which submit a request for a public key to some form of certificate store (for example a disk file, database, or the Windows registry) and obtain in response a key that, as far as the system knows (or cares) is associated with the given entity. Making the key lookup a remote (rather than local) query simply removes the administrative burden to a centralised location, allowing key management to be performed from a central location rather than being done in an ad hoc manner (or not at all) by end users. However, the advantages of centralised control and administration need to be balanced against drawbacks such as the fact that the authority that manages the operation can build fairly complete profiles of user activities based on certificate verification operations. Privacy issues, which are occasionally a concern but never appear to be a consideration in PKI design except for token measures by a few European CAs that allow the use of registered pseudonyms, are covered elsewhere [68].

There is one final step that can be taken which collapses the query-then-validate process into a single step. If we're going to trust a server to provide us with a known-good key, we may as well ask it to perform the validation for us as well, as shown in Figure 8. In this case we submit a signature to the server, and it indicates whether the signature is valid or not. This is analogous to the online credit-card-processing model mentioned previously, where if it's necessary to perform an online revocation/validity check anyway then the relying party may as well perform the entire transaction online.

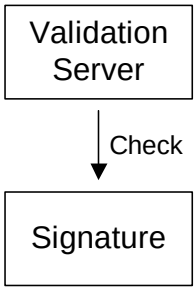

Figure 8: Avoiding certificates altogether

This is similar to another very early certificate model proposed by Davies and Price in the late 1970s in which a CA (or more specifically its predecessors at the time, arbitrators and key registries) provided a dispute resolution mechanism to relying parties by issuing an interactive certificate attesting to the validity of a key in the context of a particular transaction [69]. SPKI's one-time revalidations are another example of this idea. There are already trends back towards this type of model for use with banking and similar settlement-oriented transactions.

A related concept is embodied in the Security Assertion Markup Language (SAML), which provides an XML-based mechanism for describing authorisation mechanisms and authentication events [70][71], or Internet2's Shibboleth [72][73], however these still rely on an (unspecified) external PKI in order to function (although Shibboleth also provides for the use of Kerberos and similar mechanisms). SPKI finally completes the circle by combining the authorisation specification system with a built-in, special-purpose PKI which is designed to both avoid the problems of the traditional X.509 PKI and provide a direct authorisation management system rather than stopping short at identification and leaving the mapping from identity to authorisation as an exercise for the user.

6. PKI Design Recommendations

The various PKI issues covered in the preceding sections can now be condensed into a set of recommendations for working with certificates.

6.1 Identity

Choose a combination of locally meaningful (within a particular domain) and globally unique identity information such as a user name, email address, account or employee number, or similar value and a value derived from the public key (section 2.3). Attempts to do anything meaningful with DNs are more or less doomed to failure. "Locally meaningful" doesn't necessarily mean meaningful to humans, for example if you have an authorisation mechanism keyed off an account number then this is the logical choice for use as a local identifier. On the other hand if you're able to work with objects that are pure tickets, there's no need to bother with identity at all.

6.2 Revocation

If at all possible, design your PKI so that certificate revocation is never directly required. Revocation is an extremely difficult problem, and avoiding the issue entirely is the easiest way to handle it. SET, AADS, ssh, and SSL are examples of this approach (section 3.2).

If it isn't possible to avoid revocation entirely by designing around it, consider the use of a PKI mechanism that allows certificate freshness guarantees, avoiding the need for explicit certificate revocation. A repository that returns only known-good certificates is an example of this approach (section 5.1).

If it isn't possible to avoid explicit revocation, use an online status query mechanism. The best form of mechanism is a direct indication of whether a certificate is valid or not (section 3.4), a slightly less useful one is one that provides a CRL-style response. OCSP is an example of this approach (section 3.3).

For cases where revocation information is of little or no value, use CRLs. Revocation of code-signing and low-assurance email certificates are examples of this approach (section 3.3).

6.3 Application-Specific PKIs

Certificates and PKI types that have been specifically designed to address a particular problem are much easier to work with than a one-size-(mis)fits-all PKI design. For example SPKI certificates bind a public key to an authorisation to perform a particular action (as opposed to X.509, which binds a key to an often meaningless identity that must then be mapped, via some unspecified means, to an authorisation). SPKI is therefore ideal for situations where the goal is to authorise or give permission for a particular action, or grant a capability. X.509 has an equivalent in the form of attribute certificates, but deployment of these is essentially nonexistent and there is no real-world experience in using them, although an earlier attempt at something similar using modified X.509 certificates indicates that they will be extremely problematic [74]. SET is another example of an authorisation certificate, although it's disguised as an X.509 identity certificate.

Similarly, PGP has been designed to handle the problem of secure email communication, and employs a laissez-faire key management model that imposes few restrictions on users. PGP has its own PKI, the web of trust, although it's uncertain how effective this is in the real world [75]. The main PKI-related problem solved by PGP is that of key distribution, for which PGP employs a collection of cooperating, web-based key servers (although other interfaces are available), and direct distribution of keys via email or personal contact.

In many situations no PKI of any kind is necessary, PKI vendor claims to the contrary. This is particularly true when two (or more) parties have some form of established relationship. For example a simple technique for authenticating public keys used for voice encryption is to have one side read out a hash of the public key to the other side over the encrypted link (with optional embellishments such as "Read it backwards in a John Cleese accent") [76]. A MITM attack on this technique would require breaking into the call in real-time and imitating the voice of the caller. Similarly, ssh generally avoids any dependence on a PKI by having the user manually copy the required public key(s) to where they're needed, an approach which is feasible for ssh's application domain. AADS takes advantage of existing business relationships to tie public keys to accounts.

In some cases even PKI-less public-key encryption may not be necessary. Section 5.2 mentions bypassing certificates to perform a direct online signature check, however if it's not faster or easier to ask Citibank to confirm that a particular certificate is still valid than it is to ask Citibank to directly authorize a transaction, then it makes sense to perform the

transaction directly. This has the added benefit that it can be processed using existing transaction-handling mechanisms, a model that has shown itself to be fairly successful to date.

Finally, if you're required to use X.509 because of external constraints, remember that there's nothing that requires you to use it as anything more than a (somewhat complex) bit-bagging scheme. If you have a means of distributing and managing certificates that isn't covered in a formal standard but that fulfils its intended function, go ahead and use it (and publish a paper telling everyone else how you did it, we need more successful PKI implementation experience reports). This gives you the benefits of broad X.509 toolkit and crypto token support from vendors while allowing you to choose a PKI model that works.

7. References

[1] "Advances and Remaining Challenges to Adoption of Public Key Infrastructure Technology", United States General Accounting Office report GAO-01-277, February 2001.

[2] "Solution and Problems: (Why) It's a long Way to Interoperability", Jürgen Schwemmer, *Datenschutz und Datensicherheit*, **No.9**, **2001** (September 2001).

[3] "Prime-Time Player?", Leo Pluswich and Darren Hartman, *Information Security Magazine*, March 2001.

[4] "PKI: An Insider View", Ben Rothke, *Information Security Magazine*, October 2001.

[5] "Strong vs. Weak Approaches to Systems Development", Iris Vessey and Robert Glass, *Communications of the ACM*, **Vol.41**, **No.4** (April 1998), p.99

[6] "Cognitive Fit: An Empirical Study of Information Acquisition", Iris Vessey and Dennis Galletta, *Information Systems Research*, **Vol.2**, **No.1** (March 1991), p.63.

[7] "Cognitive Fit: An Empirical Study of Recursion and Iteration", Atish Sinha and Iris Vessey, *IEEE Transactions on Software Engineering*, **Vol.18**, **No.5** (May 1992), p.368.

[8] "New Directions in Cryptography", Whitfield Diffie and Martin Hellman, *IEEE Transactions on Information Theory*, **Vol.22**, **No.6** (November 1976), p.644.

[9] "Towards a Practical Public-key Cryptosystem", Loren Kohnfelder, MIT Bachelor's thesis, May 1978, available from `http://theses.mit.edu/Dienst/UI/2.0/Composite/0018.mit.theses/1978-29/1`.

[10] "Re: Private key validity period", Denis Pinkas, posting to the osidirectory@az05.bull.com mailing list, message-ID `9610160824.AA24707@emsc.frcl.bull.fr`, 16 October 1996.

[11] "Capabilities and Security", Roger Needham, *Proceedings of the International Workshop on Computer Architectures to Support Security and Persistence of Information*, Springer-Verlag, May 1990, p.3.

[12] "Information Technology — Open Systems Interconnection — The Directory: Authentication Framework", ISO/IEC 9594-8, 1993 (also ITU-T Recommendation X.509, version 2).

[13] "Binder, a Logic-Based Security Language", John DeTreville, *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002, p.105.

[14] "Information Technology — Open Systems Interconnection — The Directory: Models", ISO/IEC 9594-2, 1993 (also ITU-T Recommendation X.501).

[15] "SDSI — A Simple Distributed Security Infrastructure", Ron Rivest and Butler Lampson, 17 September 1996, `http://theory.lcs.mit.edu/~rivest/sdsi10.html`.

[16] "SPKI Requirements", RFC 2692, Carl Ellison, September 1999.

[17] "SPKI Certificate Theory", RFC 2693, Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylönen, September 1999.

[18] "Certificate Revocation: Why You Should Do It and Why You Don't", Paco Hope, *;login*, **Vol.26**, **No.8** (December 2001), p.36.

[19] "Re: Light-weight certificate revocation lists?", Carl Ellison, posting to the spki@c2.net mailing list, message-ID `3.0.1.32.19970402003040.00c756e8@cybercash.com`, 2 April 1997.

[20] "Certificate Management as Transaction Processing", Peter Gutmann, to appear.

[21] "Can We Eliminate Certificate Revocation Lists", Ronald Rivest, *Proceedings of the 2nd International Conference on Financial Cryptography (FC'98)*, Springer-Verlag Lecture Notes in Computer Science No.1465, February 1998, p.178.

[22] "UNCITRAL Model Law on Electronic Signatures", United Nations Commission on International Trade Law, March 2001.

[23] "PKI: First Contact [Early PKI Experiences at Boeing]", Steve Whitlock, *The Open Group: Trust and Confidence in the Global Infrastructure*, 25 October 1999.

[24] "PKI Experiences at JP Morgan", Charles Blauner, *The Open Group: Trust and Confidence in the Global Infrastructure*, 25 October 1999.

[25] "Making Netscape Compatible with Fortezza — Lessons Learned", George Ryan, *Proceedings of the 22nd National Information Systems Security Conference* (formerly the National Computer Security Conference), October 1999, CDROM distribution.

[26] "On the Complexity of Public-Key Certificate Validation", Diana Berbecaru, Antonio Lioy, and Marius Marian, *Proceedings of the 4th International Conference on Information Security (ISC'01)*, Springer-Verlag Lecture Notes in Computer Science No.2200, October 2001, p.183.

[27] "Deploying and Using Public Key Technology: Lessons Learned in Real Life", Richard Guida, Robert Stahl, Thomas Bunt, Gary Secrest, and Joseph Moorcones, *IEEE Security and Privacy*, **Vol.2**, **No.4** (July/August 2004), p.67.

[28] Anders Rundgren, private communications.

[29] "Public Key Infrastructure Study: Final Report", Shimshon Berkovits, Santosh Chokhani, Judith Furlong, Jisoo Geiter, and Jonathan Guild, Produced by the MITRE Corporation for NIST, April 1994.

[30] "QPKI: A QoS-Based Architecture for Public-Key Infrastructure (PKI)", *Proceedings of the 3rd International Conference on Cryptology in India (IndoCrypt'02)*, Springer-Verlag Lecture Notes in Computer Science No.2551, December 2002, p.108.

[31] "A Model of Certificate Revocation", David Cooper, *Proceedings of the 15th Annual Computer Security Applications Conference (COMPSAC'99)*, December 1999, p.256.

[32] "A Closer Look at Revocation and Key Compromise in Public Key Infrastructures", David Cooper, *Proceedings of the 22nd National Information Systems Security Conference* (formerly the National Computer Security Conference), October 1999, CDROM distribution.

[33] "PKI Account Authority Digital Signature Infrastructure", Anne Wheeler and Lynn Wheeler, `draft-wheeler-ipki-aads-01.txt`, 16 November 1998.

[34] "Account-Based Secure Payment Objects", ANSI X9.59 draft, 28 September 1999.

[35] "CONSEPP: Convenient and Secure Electronic Payment Protocol Based on X9.59", Albert Levi and Çetin Kaya Koç, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.286.

[36] "The SSH (Secure Shell) Remote Login Protocol", Tatu Ylönen, `draft-ylonen-ssh-protocol-00.txt`, 15 November 1995.

[37] "SSH — Secure Login Connections over the Internet", Tatu Ylönen, *Proceedings of the 1996 Usenix Security Symposium*, July 1996, p.37.

[38] "lifetime of certs now in circulation", Dan Geer, posting to the cryptography@c2.net mailing list, message-ID `199901251953.AA09739@world.std.com`, 25 January 1999.

[39] "Lessons Learned in Implementing and Deploying Crypto Software", Peter Gutmann, *Proceedings of the 11th Usenix Security Symposium*, August 2002, to appear.

[40] "Online Certificate Status Checking in Financial Transactions: The Case for Re-Issuance", Barbara Fox and Brian LaMacchia, *Proceedings of Financial Cryptography 1999 (FC'99)*, Springer-Verlag Lecture Notes in Computer Science No.1648, February 1999, p.104.

[41] "Revocation: Options and Challenges", Michael Myers, *Proceedings of Financial Cryptography 1998 (FC'98)*, Springer-Verlag Lecture Notes in Computer Science No.1465, February 1998, p.165.

[42] "PKI Account Authority Digital Signature Infrastructure", Anne Wheeler and Lynn Wheeler, `draft-wheeler-ipki-aads-01.txt`, 16 November 1998.

[43] "Online Certificate Status Protocol — OCSP", RFC 2560, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams, June 1999.

[44] "Certificate Revocation and Certificate Update", Moni Naor and Kobbi Nissim, *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[45] "Fast Checking of Individual Certificate Revocation on Small Systems", Selwyn Russell, *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)*, December 1999, p.249.

[46] "Evaluating Certificate Status Mechanisms", John Iliadis, Diomidis Spinellis, Sokratis Katsikas, Dimitris Gritzalis, and Bart Preneel, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, November 2000, p.1.

[47] "Simple Certificate Validation Protocol (SCVP)", Ambarish Malpani and Paul Hoffman, `draft-ietf-pkix-scvp-03.txt`, 12 June 2000.

[48] "Internet X.509 Public Key Infrastructure: Data Validation and Certification Server Protocols", RFC 3029, Carlisle Adams, Peter Sylvester, Michael Zolotarev, Robert Zuccherato, February 2001.

[49] "Web-based Integrated CA services Protocol, ICAP", Mine Sakurai, Hiroaki Kikuchi, Hiroyuki Hattori, Yoshiki Sameshima, and Hitoshi Kumagai, `draft-sakurai-pkix-icap-01.txt`, 31 January 1999.

[50] "Real Time Certificate Status Protocol — RCSP", Ambarish Malpani, Carlisle Adams, Rich Ankney, and Slava Galperin, `draft-malpani-rcsp-00.txt`, March 1998.

[51] "Web based Certificate Access Protocol — WebCAP/1.0", Surendra Reddy, `draft-ietf-pkix-webcap-00.txt`, April 1998.

[52] "Peter's Active Revocation Protocol (PARP)", Peter Gutmann, posting to the ietf-pkix@lists.tandem.com mailing list, message-ID `89045037024866@cs26.cs.auckland.ac.nz`, 1 April 1998.

[53] "Open CRL Distribution Process (OpenCDP)", Phillip Hallam-Baker and Warwick Ford, `draft-ietf-pkix-ocdp-00.txt`, 21 April 1998.

[54] "Directory Supported Certificate Status Options", Alan Lloyd, `draft-ietf-pkix-dir-cert-stat-01.txt`, 24 August 1998.

[55] "Roadmap issues", Bob Jueneman, posting to the ietf-pkix@imc.org mailing list, message-ID `s79348e7.059@prv-mail20.provo.novell.com`, 19 July 1999.

[56] "Building Certifications Paths: Forward vs. Reverse", Yassir Elley, Anne Anderson, Steve Hanna, Sean Mullan, Radia Perlman, and Seth Proctor, *Proceedings of the Network and Distributed System Security Symposium (NDSS'01)*, 2001.

[57] "Restricting Access with Certificate Attributes in Multiple Root Environments — A Recipe for Certificate Masquerading", Capt.James M.Hayes, *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, December 2001, p.386.

[58] "RE: IP: SSL Certificate "Monopoly" Bears Financial Fruit", Lucky Green, posting to the cryptography@wasabisystems.com mailing list, message-ID `002901c228b4$14522fb0$-6501a8c0@LUCKYVAIO`, 11 July 2002.

[59] "Compliance Defects in Public-Key Cryptography", Don Davis, *Proceedings of the 6th Usenix Security Symposium*, 1996, p.171.

[60] "Re: Computation of issuerKeyHash in OCSP", Peter Gutmann, posting to the ietf-pkix@imc.org mailing list, message-ID `98445050727773@kahu.cs.auckland.ac.nz`, 13 March 2001.

[61] "Online Certificate Status Protocol, version 2", `draft-ietf-pkix-ocspv2-02.txt`, Michael Myers, Rich Ankney, Carlisle Adams, Stephen Farrell, and Carlin Covey, March 2001.

[62] "Delegated Path Validation and Delegated Path Discovery: Protocol Requirements", Denis Pinkas and Russ Housley, `draft-ietf-pkix-dpv-dpd-req-05.txt`, May 2002.

[63] "Zentrale PKI-Lösungen and deren Anwendung in Trust Services", Tilo Schürer, *Datenschutz und Datensicherheit*, **No.10**, **2000** (October 2000).

[64] "Formal Treatment of Certificate Revocation Under Communal Access Control", Xuhiu Ao, Naftaly Minsky, and Victoria Ungureanu, *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, p.116.

[65] "Cryptography: A New Dimension in Computer Data Security", Carl Meyer and Stephen Matyas, John Wiley & Sons, 1982.

[66] "Accountable Certificate Management using Undeniable Attestations", Ahto Buldas, Peeter Land, and Helger Lipmaa, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, November 2000, p.9.

[67] "Hearsay Exceptions — Availability of Declarant Immaterial", Federal Rule of Evidence 803(6), "Business Records".

[68] "Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy", Stefan Brands, MIT Press, August 2000.

[69] "Security for Computer Networks : An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer", Donald Davies and W.Price, John Wiley and Sons, 1984.

[70] "Security Assertions Markup Language: Core Assertion Architecture", draft 0.9, Phillip Hallam-Baker, Tim Moses, Bob Morgan, Carlisle Adams, Charles Knouse, David Orchard, Eve Maler, Irving Reid, Jeff Hodges, Marlena Erdos, Nigel Edwards, and Prateek Mishra, 20 June 2001.

[71] "Oasis Security Services Use Cases And Requirements", draft 1, Darren Platt and Evan Prodromou, 30 May 2001.

[72] "Shibboleth Working Group Overview and Requirements Document", draft 1, Steven Camody, 20 February 2001.

[73] "Shibboleth Working Group Specification Document", draft 1, 25 May 2001.

[74] "Re: Comments on draft-ietf-pkix-ac509prof-05.txt", Al Arsenault, posting to the ietf-pkix@imc.org mailing list, message-ID `39EF4022.A1401779@home.com`, 19 October 2000.

[75] "Reflecting on PGP, keyservers, and the Web of Trust", Greg Rose, posting to the cryptography@c2.net mailing list, message-ID `4.3.1.0.20000901145546.00c55100@127.0.0.1`, 1 September 2000.

[76] "Nautilus Secure Phone Home Page", `http://www.lila.com/nautilus/`.