

PKI without Revocation Checking

Karl Scheibelhofer

Institute for Applied Information Processing and Communications
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
Email: karl.scheibelhofer@iaik.tugraz.at

Abstract

Current X.509-based PKIs are typically complex. One of the most critical parts is revocation checking. Providing revocation information is a big effort for CAs, and for clients it is even more costly to get all required revocation data. This work provides a performance calculation of two CA setups with CRLs, delta CRLs and OCSP as revocation checking mechanism. The enhancement of this work is the proposal to avoid revocation checking by issuing certificates on-line and only retroactively. An analysis shows that this approach performs at least as good as OCSP and much better than CRLs. In addition, this solution reduces the complexity of PKI significantly.

Introduction

Today, most computers are connected to a network. Usually, this is a corporate network, a home network or a campus network, and this network is commonly connected to the Internet through firewalls and routers. Pure off-line systems are the exception rather than the rule nowadays. Even if many computers are only on-line from time to time, they attach to a network on a regular basis. This holds for notebooks, PDAs and smart phones. However, with most of these devices, it is easy to go on-line. The Internet becomes ubiquitous: Ethernet at the office, cable modem or DSL at home, WLAN at campus, airports and rail stations, 3G in cities and more to come. Network connectivity is no scarce resource any longer, as well as performance. Even smart phones can create and verify digital signatures in much less

than a second. Certificate based PKIs have been designed based on the assumption that network connectivity is a limited and expensive resource (see III in [13]).

This paper considers current certificate and revocation checking practice in the light of ubiquitous access to the Internet. The focus is on X.509 certificates [1] and current revocation checking techniques like CRLs and OCSP [2]. Based on this analysis, we propose a simple way to make certificate handling and validation easier in a networked environment. The proposed approach is an on-line certification service. The CA issues a new certificate each time the key owner uses the key. We will show that this requires not more effort than OCSP, neither on the CA side nor on the client side. Moreover, it reduces the overall complexity of the system and provides up-to-date status information about the key. This can be seen as a special version of short-lived certificates, which have been proposed several times (e.g. [9]). Unlike short-lived certificates, here, the validity time degrades to an instant in time which lies in the past. Thus, the statement made by the CA through a certificate can be definitive, which makes revocation a less critical issue in many cases. There is no need for revocation checking of such certificates; the certificate already contains the status information.

Certificate Validity

Each X.509 certificate contains a validity period, which is a time interval determined by a start date and an end date. The start date is normally the time at which the CA issued

the certificate. The end time is often calculated as a fixed offset from the start time, e.g. start time plus one year. One year is a typical validity frame for a public-key certificate. The certificate validity is often interpreted as *"this certificate is valid during this time interval unless it is revoked"*. However, PKIX [1] defines it differently: *The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate.* On the first look, the two interpretations seem to be similar, but there are subtle differences. The PKIX definition does not imply any validity information. A certificate can still be valid even beyond its validity period. The CA simply does not guarantee to provide information about the validity of this certificate outside this time frame. The first interpretation does not accept a certificate which has been used outside its validity period, no matter, if other status information is available. In practice, the application of both interpretations ends in the same results. The reason is that most CAs do not provide positive status information for certificates. This means, the CA tells clients what certificates are definitely revoked, but only for certificates which are within their validity period. For other certificates, the clients do not get any revocation information. If a certificate runs out of its validity period, we call it expired. In case a considered certificate expired, clients have no means to find out if the CA no longer provides revocation information or if the certificate has not been revoked. In other words, a client may or may not find out if an expired certificate has been revoked before it expired. The client has no guarantee that such certificates are listed on the CRL.

The situation is similar for OCSP. Since OCSP responders may cache old CRLs, they may provide certificate revocation status beyond certificate expiration. A responder can indicate such support by adding a spe-

cial extension to its response messages, the archive cutoff extension. Thus, OCSP responders may extend the period through which status information about certificates is available. However, it does not really extend the validity period in the sense defined by PKIX because in practice, a certificate owner cannot revoke a certificate which has already expired. In consequence, relying parties can get information about revocation issues which happened within the certificate's validity period, but they cannot get revocation issues beyond this period. The CA does not offer such service for expired certificates.

Revocation Checking

In general, a certificate which is within its validity period can be valid or revoked. A client cannot find out the status of the certificate off-line. Finding out if a certificate is valid requires access to some network resource. This can be the current CRL on a web server or an OCSP responder.

CRLs

CRLs are a problematic mechanism for revocation checking. For the CAs they provide a simple means with low requirements concerning signature creation and security. For the relying parties who need to verify certificates, CRLs are a pain. They are often quite large, they often contain close to 100% irrelevant information and they usually provide no current revocation information. Consider SSL [4] server certificates from Verisign, Inc.. They contain a CRL distribution point referring to the location of the current CRL. However, this CRL is about 750 kByte. In consequence, most browsers work without revocation checking for server certificates. Downloading the CRL would take considerable time and would delay the HTTP transfer unacceptably long.

CRLs have a validity period similar to certificates. It is determined by the time at which the CRL has been issued, called *thisUpdate*, and the time at which the next CRL will be issued latest, called *nextUpdate*. According to PKIX, a client can cache a CRL until *nextUpdate* and rely on this information. However, it is apparent that a CRL cannot provide status information newer than indicated in its *thisUpdate* field. Caching the CRL is nice and provides a benefit in some special applications, but for many applications, it does not make much sense. System designers using cached CRLs should be aware of the fact that relying on cached CRLs may produce non-deterministic behavior for revocation checking. Consider a web server certificate. The mentioned CRL has a validity period of two weeks. Thus, in case of certificate revocation, clients will notice this revocation up to two weeks later and one week later on average if they cache the CRL as long as possible. Until the cached CRL expires, the client will consider the certificate as not revoked, even if it has already been revoked. This information simply did not yet propagate to the client because the cached CRL has been used. Thus, the client will get a different result for revocation checking the same certificate twice, even if checked with respect to the same time. This non-deterministic behavior can be a show-stopper for certain applications; for instance consider applications where relying parties transfer noticeable volumes of money based on a positive signature verification (including a certificate validation with revocation checking).

Usually, administrators would switch to a new certificate even before they revoke the old certificate. As long as there has not been a key-compromise, revocations may not be that critical anyway. These cases can often be handled by organizational means. For example, the certificate can be updated, or

the server could be replaced by a backup system. In case of a key compromise, however, the administrator would need to revoke the certificate immediately and inform all clients quickly. A delay of several days appears intolerable. CRLs do not provide an efficient solution in this situation. Reducing the caching time of CRLs does not offer a significant improvement either. The client may download the latest CRL for each certificate validation issue, but this is impractical. It wastes bandwidth and introduces delays, which are unacceptable for many applications. Moreover, the server which provides the CRL would break down under the load if all clients would download the current CRL every time, at least in environments with a considerable number of certificates and entries in the CRL. The waste of bandwidth in such configurations would be enormous.

Almost all data in a CRL is of no interest for the client. This is easy to see. The client is usually only interested in the status of a single certificate. The CRL returns a list of all revoked certificates in which the client can look up the concerned certificate. In most cases, the certificate of interest will not be listed in the CRL. Revocation is more an exceptional case than a common case. Common rates for certificate revocation are 5 to 10 percent, i.e. about 5 to 10 percent of all issued certificates get revoked before they expire. This would mean that a client would find the certificate of interest on the CRL in at most 1 of 10 (10 percent) or 1 of 20 (5 percent) cases. *At most* because a revoked signature or authentication certificate will typically only be used after revocation in case of a key compromise. If the certificate owner loses the private key, the certificate is useless and cannot be used any longer. Likewise, the legitimate owner would not use a certificate which has been revoked because of changed owner information. Key compromise will naturally be a

scarce reason for revocation compared to key-loss and change of subject information. As a result, a CRL will contain the concerned certificate even less frequently as calculated before. In effect, during a single certificate validation procedure, a CRL carries only a single bit of information for the client. The client wants to know if the considered certificate is valid (or was valid at a certain time). The answer can only be *yes* or *no* (*unknown* can be considered as a third alternative). Compared to such large CRLs as mentioned before, which contain over 20000 entries, the waste of resources becomes obvious.

The PKIX standards also define so-called Delta CRLs. Their main goal is to reduce bandwidth demand. Subsequent CRLs differ only slightly in their contents compared to their predecessor. Some new entries are added and some old may be removed, but the vast majority of the entries will stay the same between two subsequently issued CRLs. Thus, the idea is to transfer only changes in the CRL most of the time. A delta CRL refers to a complete CRL and just mentions the differences to this CRL, i.e. the added entries and the removed entries. Thus, the client has the same information as if it would have two complete CRLs. Delta CRLs are rarely used in practice. The added complexity is one of the reasons.

Revocation checking does only look that simple on the first look. If one implements the complete PKIX specification and considers all (or at least all practical) use-cases, revocation checking based on CRLs can get complex. Here are some reasons:

- There could be a different CRL for each revocation reason. In consequence, the client would have to check each of these CRLs.

- The issuer of the CRL could be different to the issuer of the certificate. It may be difficult to make a trust decision and no user wants to do this manually (most users do not even have the required background knowledge to do it).
- If the application has to verify a certificate (which may already be expired) with respect to a time in the past, it would need an old CRL. There is no standardized way to get an old CRL. The standards only specify how to get the latest CRL.

This list can be extended by various bullets. These cases do not frequently appear in practice, but they are perfectly valid according to the underlying standards. If a developer has to implement a system, which supports this standard, the system needs to be able to handle these special cases as well, even though they might never appear in practice. As a result, developers may end up with a final system where many parts of the certificate validation system are never really used.

For completeness, we should mention that there are use-cases where CRLs are sufficient and the required bandwidth is not that critical. If the CRL does not contain many entries, the lavished bandwidth is not that significant either. The size of the CRL can be decreased using different approaches. A simple one is to use partitioned CRLs. Using this method, the CA splits the set of all issued certificates into groups of similar size. The CA will issue an individual CRL for each group of certificates. Thus, the certificates, even though issued by the same CA, will point to a different CRL location, depending on the group they belong to. For CAs which issue only a small number of certificates, the CRL will generally stay small enough without segmentation.

OCSP

CRLs have been designed as an off-line mechanism for revocation checking. An off-line mechanism cannot meet certain requirements for timeliness and efficiency. So what about on-line protocols? Does OCSP perform better? It may. At least, it does not waste that much bandwidth in most cases.

The basic idea of OCSP is simple. It is an on-line service which provides revocation information about certificates to clients. When the client needs to validate a certificate, it can send a request to an OCSP responder. The OCSP server answers with information about the revocation state of the certificate. Unfortunately, there are some details which can make OCSP harder to use than necessary. First, the client needs the issuer certificate of the certificate of interest because the hash of the issuer certificate's public key is required in the OCSP request message. Second, the response of the server may not be as concrete as the client would need it to be. A normal OCSP response does not provide positive certificate status information. OCSP responses have several properties which degrades their effectiveness in many use-cases:

- The response does not say if the certificate is valid, it merely says that the certificate is not revoked. However, this can even mean that the certificate has never been issued.¹
- The revocation information provided by an OCSP server may not be up to date.

At a minimum, a client cannot expect to get more information from an OCSP responder, as it can get from the latest CRL.

¹ The German ISIS-MTT [12] profile defines the CertHash extension for OCSP responses. A response containing this extension can provide a positive status information.

This is nice for OCSP service providers because it allows very simple implementations of an OCSP responder. In this case, the OCSP service is nothing more than a front-end for the latest CRL. Consequently, an OCSP service implemented this way inherits the problems inherent to CRLs except the bandwidth consumption. Remarkably, the bandwidth consumption of CRLs is the biggest cost factor for big CAs as documented in [11].

Revocation Checking in Practice

In practice, many systems do not perform revocation checking at all. This has various reasons. Most web browsers do not check server certificates for revocation. The main reason seems to be the potentially long delay due to downloading the CRL. Many email clients often perform no checking per default either. The long delay seems to be the reason once again. However, most applications at least offer options to enable revocation checking. Some of them, for instance Microsoft Outlook 2002 or later, perform revocation checking per default. It is easy to identify such email clients with enabled revocation checking because there often is a long delay if the user opens a signed or encrypted email. Downloading the CRL takes most of this time.

Only a few systems support OCSP. Mozilla is one of these few exceptions. Most web browsers and email clients do not support it, at least not without additional third-party tools. On the CA side, the situation is similar—not many of them run OCSP servers. Those who run an OCSP server often implement them based on CRLs rather than to base them on current status information. Third-party OCSP responders usually have no other means than to base their response information on CRLs. Third-party OCSP responders have the advantage that

they can provide revocation information for certificates from various CAs. On the other hand, users must configure their clients manually to use external responders. In sensitive environments, it might be impossible to rely on another external party; additional contracts have to be prepared, liability issues have to be clarified, and so on.

OCSP is capable of providing a better service to clients than CRLs can do. For the provider side, OCSP requires additional resources. Since OCSP is an on-line service, it requires a reliable server system with a high level of security. The server has to sign its responses with a private key. This key requires the same security level as the key for CRL signing does because it serves to protect the same kind of information—revocation information for the same certificates. Some simple investigations and calculations show that OCSP shifts the demands from network bandwidth to processing power.

Calculations can show some values for the performance estimation of CRLs and OCSP for revocation checking. First, we sketch rough values for the required bandwidth in certain environments. In addition, we consider the number of required signature creations on the server, which differs significantly between CRLs and OCSP. On the client side, CRLs and OCSP have also others impacts. If a client uses CRLs, it needs to cache CRLs. Without caching, CRLs are a real performance and resource killer because the client would download the complete CRL for each revocation checking. Some implementations do this intentionally to ensure that they have the latest available revocation information. This practice cannot be recommended in general. If there is a demand to get always the latest revocation information, OCSP is a better choice. In addition, processing a CRL is usually more costly than processing an OCSP response,

unless the CRL is very small which is typically not the case. Verifying the signature of the status information—the signature on the CRL or on the OCSP response—is effectively the same effort for the client. The timeliness of the status information may also differ between CRLs and OCSP. An OCSP responder may provide real-time revocation information, while a CRL always provides aged information, though, many OCSP servers do not provide more current information than the CRLs do.

	Small CA	Big CA
Certificates	1 000	1 000 000
Certificate Users	1 000	1 000 000
Certificate Validity [years]	1	1
Revocation Probability	10%	10%
Certificate Validations per Certificate [day ⁻¹]	10	10

Table 1: Properties of the considered configurations

Table 1 shows the properties of two CA configurations which we will use as a basis for performance estimations. There are two configurations—a small CA configuration with thousand issued certificates, and a big CA with one million issued certificates. We set the validity period of the issued certificates to one year, which is a typical value. In addition, we assume that the number of certificate users (users who validate certificates from this CA) is roughly the same as the number of issued certificates. However, these two numbers can vary significantly in practice; a CA issuing only server certificates for a few big web servers like Amazon, eBay and Google would have a huge number of users validating the certificates while the number of issued certificates is small. For the probability of certificate revocation, we take 10%, i.e. the probability that a certificate will be revoked before it expires. This is

also a quite common value in practice, but it may even be much lower, e.g. one percent. We assume, that a user validates about 10 certificates per day, for example, verify 5 signed emails and encrypt 5 emails per day. These numbers set the frame for our short investigations. They are typical values, even though there are configurations for which these values would look very dissimilar and would result in quite different performance estimations.

	Small CA	Big CA
CRL Validity [hours]	24	24
Signature Creations [day ⁻¹]	4	3 014
CRL Size [Byte]	2 250	2 000 250
CRL Downloads [day ⁻¹]	1 000	1 000 000
Bandwidth [MB/day]	2,1	1 900 000

Table 2: Expected CRL Performance

Table 2 shows the expected performance data for the two CAs using CRLs for revocation checking. We took a CRL validity of 24 hours. This is the time between the *thisUpdate* and *nextUpdate* values in the CRL. The number of signature creations required on the CA side is comprised of certificate and CRL issuing—one signature per each certificate and each CRL. For instance, the small CA issues 3 certificates per day on average and 1 CRL, which requires 4 signature creations in total per day. Here, the certificates account for most of these signatures because the CA issues CRLs infrequently. For calculating the CRL size, we took 40 Byte as the size of each CRL entry². Despite the entries of the revoked certificates, each CRL contains additional data like the signature, *thisUpdate*, *nextUpdate* and the signer

² This consists of at least 13 Byte for the revocation date, plus the serial number (e.g. a SHA-1 hash value), plus an overhead for the DER encoding of at least 6 Byte. In addition, there may be extensions for the reason code or invalidity date.

name. For this additional data, we assumed 250 Byte. If we take the small CA, for example, the CRL would contain 50 entries on average³, which will result in a size of 2250 Byte (50.40 + 250). The total number of certificates and the probability of certificate revocation determine the number of entries. Taking the 10 certificate revocation checks per user and per day, every user downloads each issued CRL. This ends up in a network transfer volume of 2,1 MB per day for the small CA and 1900 GB per day (about 22 MB per second) for the big CA. This is a noteworthy volume. Only big companies can afford such a network bandwidth, and often merely in a local network and hardly via the Internet. It is also a matter of costs. Would you afford this for revocation checking only if there are cheaper alternatives?

Moreover, these numbers can easily increase. The current numbers are based on an issuing interval of one day. In the worst case, clients get one-day-old revocation information. If the managers of a system decide to increase the CRL issuing frequency to get a better timeliness, the increase of network transfer is inverse proportional. Reducing the CRL validity to the half (i.e. to 12 hours) doubles the transfer volume. The peaks of bandwidth consumption may be even much higher because we assumed evenly distributed download requests over time. In practice, peaks are very likely, for instance, in the morning when people start working there will be a peak, while the download rate will drop during the night.

³ We get 100 entries in the worst case, which occurs if the CA issues all certificates at the same time (e.g. at the beginning of the year). If it issues the certificates continuously over the year, we could expect 50 entries in each CRL, because on average a certificate would be revoked after half of its validity period.

	Small CA		Big CA	
Base CRL Validity [days]	7	7	7	7
Delta CRL Validity [hours]	24	2	24	2
Signature Creations [day ⁻¹]	4	15	3015	3025
Base CRL Size [Byte]	2250	2250	2000250	2000250
Avg. Delta CRL Size [Byte]	288	288	38606	38606
Base CRL Downloads [day ⁻¹]	143	143	142857	142857
Delta CRL Downloads ⁴ [day ⁻¹]	1000	10000	1000000	10000000
Bandwidth [MB/day]	0,58	3,1	310 000	640 000

Table 3: Expected Delta CRL Performance

Delta CRLs can reduce the bandwidth consumption significantly. Table 3 shows that the CA can issue delta CRLs even more frequently while the bandwidth consumption is still much lower than for full CRLs. To get comparable results, we included a delta CRLs case with the same timeliness constraints, which we had for the full CRL case; i.e. there is one column for the small CA and for the big CA for which the delta CRL validity is 24 hours—the same value as the CRL validity in Table 2. To get an impression how the results would change due to a reduction of the delta CRL validity, we added another column for each CA with smaller delta CRL validity—two hours in this case. In contrast to the full CRL case, with delta CRLs, the CA has to create a similar number of signatures. Only a few

⁴ Note that the number of delta CRL downloads per day will not be higher than the overall number of certificate validations per day, which is the product of certificate validations per day (per certificate) and the number of certificates.

delta CRL signatures are required additionally, while the number of signatures for base CRLs decreases at the same time because the CA issues them only once a week.

As a result, we can see that the bandwidth consumption decreased significantly. In the case where delta CRLs are issued once a day—which provides the same timeliness as the full CRL case considered before—the transferred data volume dropped to 0,58 MB per day for the small CA and to 310 GB per day for the big CA. This is a reduction of about 73% and 84% respectively. Even if we reduce the validity period of the delta CRLs (and increasing the timeliness of the information at the same time) in the Big CA setup, the load on the network remains lower than with full CRLs. Only for the small CA, the network load is higher than in the full CRL case, 3,1 MB per day compared to 2,1 MB per day.

	Small CA	Big CA
Signature Creations [day ⁻¹]	10 003	10 003 014
Request Size [Byte]	250	250
Response Size [Byte]	1 250	1 250
Bandwidth [MB/day]	14,3	14 305

Table 4: Expected OCSP Performance

Table 4 gives a performance estimation for an OCSP responder for our two CA configurations. This table is simpler because there is nothing such as a validity period. Even if the OCSP responder gets its status information from CRLs, it makes no performance difference, which would reflect in the table. If the OCSP responder gets its status information from CA database directly instead of from CRLs, the CA could abandon CRLs completely. In addition, it would enable the OCSP responder to provide real-time status information and positive status responses (this requires a non-standard extension like the CertHash extension in [12]). In contrary to the CRL case, for OCSP the bandwidth consumption

increases linearly with the number of certificate validations. With CRLs, the client downloads a CRL and does all revocation checking based on it until it expires. With OCSP, the client has to get a separate OCSP response for each certificate validation. The caching of responses gives no benefit unless the client always validates the same certificates.

As we can see in the table, the bandwidth consumption is quite affordable. While it is a little higher for the small CA, it is smaller by an order of magnitude for the big CA. For the small CA, the OCSP responder produces 14,3 MB traffic per day compared to 2,1 MB with full CRLs and 0,58 MB and 3,1 MB with delta CRLs. In large environments, the difference is tremendous. Here, the OCSP responder causes about 14 GB network traffic per day compared to 1900 GB with full CRLs and 310 GB and 640 GB with delta CRLs. In addition, an OCSP responder can give real-time status information—CRLs cannot. These 14 GB per day (0,17 MB per second) are an affordable bandwidth, not only for a corporate network but also for Internet connections.

These simple calculations already give a clear impression of the scalability of the different revocation mechanisms used in practice. OCSP scales much better than CRLs. As well, OCSP can provide more current revocation information than CRLs. For small CAs, CRLs may cause slightly less network traffic, but also provide less timely status information. OCSP can show its advantages in large environments, where it can reduce the network traffic massively.

However, all these calculations are valid only for two simple but reasonable setups. The results can vary depending on various factors of the environment, including the number of issued certificates, the number of certificate users, the number of certificate

validations and the timeliness requirements for revocation information. Up to now, timeliness requirements seem to have a low priority for CAs because there are many CAs which do not offer OCSP responders and those of which who implement their servers to provide no more current revocation information than the latest CRL does. It is unclear if there is not enough demand from the customers for up-to-date status information, or if the CAs are reluctant to implement better OCSP servers.

Proposal for an on-line PKI

Certification Authorities issue certificates. Users who receive such a certificate need to find out, if the data in the certificate is valid with respect to a certain time. For digital signatures, this is the time of signature creation. In case of encryption, it is usually the current time. The application performs revocation checking to find out if the certificate is valid. Other steps are also required, but applications can typically process them with locally available data. Certificate chain building, for example, is often rather easy because most protocols like secure email and SSL/TLS recommend sending the complete certificate chain anyway. Fortunately, most implementations stick to this recommendation.

Is revocation checking always required? If we assume that the start of the validity period of a certificate corresponds to the time when the certificate has been issued, an application would not need to check a certificate for revocation with respect to this time. In fact, a certificate does not say anything about its validity. Even the validity period is only defined to specify the time interval during which the issuing CA guarantees to provide status information. Currently, there are new standards for certificate validation on the way. One of the most prominent ones is the Simple Certificate

Validation Protocol, short SCVP. The PKIX working group develops this standard. It should enable clients to off-load the complete task of certificate validation. The client sends a request to the service for validation of a certain certificate. As a result, it gets the validation result.

Developers with no PKI knowledge may ask, if we need certificates at all. This is a legitimate question. The client has to trust the validation server anyway to provide correct results, and it does not do any processing of the certificate itself any longer despite picking the public key and the identifier of the key owner. In many cases, it would be sufficient to add the signer's name and the signature verification key to a signature. The verifier could then ask the validation server if this signer was the legitimate owner of the verification key at the time of signature creation. The validation server could still employ certificates for its internal processing and select the appropriate ones. However, if the CA runs the validation server, there seems to be no point in using certificates in the traditional way. Accessing databases seems to be simpler and more efficient. The signer could add a link to the certification authority to the signature to support the client in selecting the right one for validation. It may even make sense to have more than one authority at the same time. The recipient can pick a trusted one or may validate with several or all of them to increase the trust in the result. It would also be possible that the signer already gets a signed validation result from the authority at the time of signature creation. The validation result is actually nothing else than a certificate. The recipient can still do its own validation on demand. Thus, the same format can be used. All we need is an on-line service for certificate issuing from the CA. The requirements for such a service would be pretty much the same as for an OCSP service. The CA would issue a new certificate for each signature, or at least for

each time instant at which the signer creates a signature. An important difference is that the validity period of such certificates would always be in the past (with respect to the issuing time) or would at most extend to the current time.

This approach has several advantages. First, there is no need for additional revocation checking. The issued certificate would already transport all information to the recipient—the signer was the legitimate owner of this key at the signing time. For encryption, the time of interest is the current time, and the sender would go for a certificate for the intended recipient.

One apparent disadvantage is the required on-line access for the client. Having a closer look, the requirements are the same as if the user would have to perform revocation checking with up-to-date status information. This is also only possible with on-line access. Users and applications can still use old certificates if they do not have such high security demands. Here, the decision is up to the users if they accept out-dated information about the binding between alleged key owner and key or if they go for reliable and definitive information.

An additional improvement is the reduced complexity. Only one format is required for certificates, no additional format for CRLs or OCSP or any other validation service. The existing certificate format could even serve as the request format for this on-line certification service. The request would not need a signature though. The response would be just a certificate. The validity period of this certificate would be just a single time instant or maybe a period in the past. For the past, the certification authority would be able to provide definitive answers. Thus, revocation may not be required at all, in the sense it is used today.

Business cases would also be simpler and more flexible with this approach. A commercial CA would be able to charge on a per-use basis, for example, per issued signing certificate. Since the CA does not need to run an additional revocation checking service, the resource planning is also easier and more predictable. Just imagine that Microsoft enables certificate revocation checking using CRLs per default in their Internet Explorer. Commercial CAs would face tough times providing their large CRLs for millions of clients in the world.

The workflow for setting up a relation or contract between the user and the CA can remain the same. The CA can issue a first certificate during this initial phase. Even though the client does not really need this certificate, it may help ensuring compatibility with existing applications. The client could even use just a self-signed certificate instead because it also includes the required public key and name. Moreover, a self-signed certificate can serve as a proof of possession during the registration. If the client registers its public key at several CAs, it would not need to have several certificates. There is nothing wrong with the registration of the same user-to-key binding at several CAs. It would only cause different CAs to vouch for the same statement. In fact, this is an increase of security because it may be possible that a single CA makes mistakes, but it is less likely that several independent CAs make the same mistake.

The client actually needs just its identifier (something like the subject name) and its key-pair. For further communication with the CA, it can use this signature key and the identifier to authenticate the requests to the CA. A request can be for lengthening the contract with the CA or for revocation of a key-to-identifier binding. Note that there is no revocation of a single certificate any longer because there is no certificate to re-

voke. After revocation of the binding, the CA would simply no longer issue certificates via its on-line certification service for this combination of key and user. There is no need to revoke any old certificates it issued before because the certificate statements refer to a time in the past before the revocation took place.

What is the performance of such a simplified on-line certification PKI? The bandwidth consumption is actually the same as for OCSP, assuming that the requests and responses are roughly of comparable size. This seems to be a reasonable assumption.

	Small CA	Big CA
Signature Creations [day ⁻¹]	10 000	10 000 000
Request Size [Byte]	500	500
Response Size [Byte]	1 000	1 000
Bandwidth [MB/day]	14,3	14 305

Table 5: Expected performance for on-line certification

Table 5 shows the expected performance values at the CA side. The number of required signature creations is even slightly lower than for OCSP because for OCSP the CA has to issue certificates and OCSP responses. Here, we only have one signature for each certificate, the total number of which is the same as for OCSP responses. In practice, the situation may be even better because clients typically do not cache OCSP responses. Thus, they get an OCSP response each time they validate a certificate, even if they validate the same certificate with respect to the same time. In the case of on-line certification, when the signer already attaches the certificate to the signature, the verifier may not ever need to go for a new certificate.

For the big CA, the on-line certification service needs to perform 10 million signature creations per day, which are about 116 signature creations per second. This is an amount, which modern server systems can easily process with pure software implementations of cryptographic operations⁵. An on-line certification system would use secure crypto hardware anyway for sheer security requirements. Modern hardware security modules (HSMs) are typically able to perform several hundred operations per second. There are some HSMs available which can create several thousand signatures per second,⁶ and there are crypto processors which can perform several ten-thousands per second⁷. Thus, even for large CAs, an on-line certification service seems to be feasible with off-the-shelf hardware.

Compared with CRLs and delta CRLs, on-line certification is a clear tradeoff between network bandwidth and processing power. On small-scale PKIs, on-line certification can require even more bandwidth than CRLs, but in large-scale PKIs, it performs equally well as OCSP. At the same time, on-line certification can provide the latest key status information, which an OCSP responder can also do, but CRLs and delta CRLs cannot.

On the client side, on-line certification can bring a significant reduction of complexity because additional revocation checking is no longer required. Moreover, clients need to handle less different data formats.

⁵ For example, OpenSSL 0.9.7d compiled with assembler optimizations on an AMD Opteron 146 with 2 GHz performs about 900 signature creations per second using RSA 1024 bit keys (using CRT).

⁶ The Sun Crypto Accelerator 4000 PCI Card can create 4300 RSA signatures per second with 1024 bit keys using CRT.

⁷ A NITROX Security Processor from Cavium Networks can perform up to 40 000 RSA signatures per second with 1024 bit keys using CRT.

They only need to handle the certificate format, which can remain the same as already used in current systems. As a result, existing protocols like S/MIME [5] and SSL/TLS [4] can run without modification. Some other protocols can even be simplified because there is no need to support CRLs, OCSP and other revocation checking formats and protocols any longer.

The service's signature key is an attractive target for attacks. This is a drawback, even though it is manageable with dedicated security devices. If an adversary can get the signature key of the service, it can issue certificates at will. Current CAs often issue certificates off-line and just issue revocation information on-line. Getting the signature key of the revocation service is not sufficient for issuing certificates, though the ability to issue wrong revocation information may allow attacks with similar severe effects.

Conclusion and Further Work

During this work, we analyzed a few typical PKI setups. Not surprisingly ([11]), it turned out that CRL-based revocation checking consumes a lot of bandwidth, especially for large CAs with many certificates and many users. For the proposed type of on-line certification, the requirements for the signature creation performance are higher than for CRLs, but they are at most as high as for an OCSP responder. Unlike CRLs and certain OCSP responders, on-line certification provides always the latest status information. This approach also reduces the complexity of PKI systems because it eliminates the need for additional revocation checking, and in consequence, there is no need to support additional formats and protocols like CRLs and OCSP. In summary, the system requirements regarding bandwidth and processing performance are comparable to OCSP. Thus, existing hardware and software

components are sufficient to implement an on-line certification system.

On-line certification has some neat properties, which makes it appealing to certain business cases. A pay-per-use scheme seems to be easy to implement. Its behavior is also better predictable than that of current systems, which include separate revocation checking. Overall, the introduction of an on-line certification service is more of a technical nature than an organizational one. Thus, the existing organizational environment can often remain unchanged. Even many parts of the technical equipment can remain unaffected or may require only small adaptations. On the client side, the reduction in complexity is even more evident. This can make PKI attractive even for such environments where current PKI systems would entail an unacceptable increase of complexity.

The next step towards an on-line certification service would be the more detailed specification of a protocol. Such a service would have to be as simple as possible. A prototype implementation as a proof-of-concept would also help to get a clearer impression of the advantages and disadvantages of this approach. It would also allow benchmarking in different environments and use-cases. Measured values from real implementations are more convincing than theoretical calculations.

Two important issues have not been considered yet. The first is the setup of trusted root keys. Even though, the same techniques as for the setup of trusted root certificates are applicable, simpler and more elegant mechanisms are desirable. The second is the process of revocation itself, i.e. the actions the key owner can take to notify the CA about revocation. Current systems solve this issue unsatisfactorily.

While recent work often increases the overall complexity of PKI systems and thus diminishes its attractiveness, on-line certification can offer a real reduction of complexity while improving utility.

References

- [1] Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure, Certificate and CRL Profile. The IETF, RFC 3280, April 2002, available online at <http://www.ietf.org/rfc/rfc3280.txt>
- [2] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP. The IETF, RFC 2560, June 1999, available online at <http://www.ietf.org/rfc/rfc2560.txt>
- [3] Adams, Carlisle, Lloyd, Steve, “Understanding the Public-Key Infrastructure”, New Riders Publishing, ISBN: 157870166X
- [4] Dierks, T., Allen, C.: The TLS Protocol, Version 1.0. The IETF, RFC 2246, January 1999, available online at <http://www.ietf.org/rfc/rfc2246.txt>
- [5] Ramsdell, B. (Editor): S/MIME Version 3 Message Specification. The IETF, RFC 2633, June 1999, available online at <http://www.ietf.org/rfc/rfc2633.txt>
- [6] Doyle, P., Hanna, S.: Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage. The OASIS Public Key Infrastructure (PKI) Technical Committee (TC), Version 1.0, 8 August 2003, available online at <http://www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf>
- [7] Myers, M.: Revocation: Options and Challenges. Financial Cryptography 1998, Springer-Verlag Berlin Heidelberg 1998, LNCS 1465, pp. 165-171.
- [8] Cooper, D.: A Model of Certificate Revocation. Proceedings of the Fifteenth Annual Computer Security Applications

- Conference (ACSAC), pages 256-264, December 1999.
- [9] Rivest, R.: Can We Eliminate Certificate Revocation Lists? *Financial Cryptography 1998*, Springer-Verlag Berlin Heidelberg 1998, LNCS 1465, pp. 178-183.
 - [10] Fox, B., LaMacchia, B.: Online Certificate Status Checking in Financial Transactions: The Case for Re-issuance. *Financial Cryptography 1999*, Springer-Verlag Berlin Heidelberg 1999, LNCS 1648, pp. 104-117.
 - [11] Berkovits, S., Chokhani, S., Furlong, J., Geiter, J., Guild, J.: Public Key Infrastructure Study, Final Report. Produced by the MITRE Corporation for NIST, McLean, Virginia, April 1994.
 - [12] Common ISIS-MTT Specifications for Interoperable PKI Applications from T7 & TeleTrusT. Version 1.1, 16 March 2004, available online at <http://www.isis-mtt.org>
 - [13] Kohnfelder, L.: Towards a Practical Public-Key Cryptosystem. Bachelor Thesis, MIT, May 1978.
 - [14] Ellison, C.: Naming and Certificates. *Proceedings of the tenth conference on Computers, freedom and privacy*, pp. 213-217. Toronto, Ontario, Canada, April 2000.