Evaluation of Certificate Validation Mechanisms

Tomás Perlines Hormann, Konrad Wrona, Silke Holtmanns

2nd October 2001
Ericsson Research
E-mail: {eedtph,eedkwr,eedshm}@eed.ericsson.se

Abstract

In this article we evaluate different *certificate validation mechanisms* to be possibly used within the *Wireless Public Key Infrastructure (W-PKI)* for the *Wireless Application Protocol (WAP)*. An implementation of a standard compliant *signed content* application offering full *PKI* functionality served as means for evaluating different mechanisms. We compared *short-lived certificates*, *Certificate Revocation Lists (CRL's)*, the *Online Certificate Status Protocol (OCSP)* and the *XML Key Management Specification (XKMS)* with regard to security, interoperability, complexity, on-line vs. off-line and performance in terms of size and scalability.

1 Introduction

The current standardization of the *Wireless Public Key Infrastructure (W-PKI)* for the *Wireless Application Protocol (WAP)* [23] does not yet specify any mechanisms for reliable checking of the *validity status* of public key certificates. Currently, the usage of *short-lived certificates* with a validity period of 48 hours serves as an interim solution.

This article aims at providing a meaningful evaluation of *certificate validation mechanisms* to be possibly included into future revisions of the *W-PKI*. We compared *short-lived certificates* [8, 7] together with *Certificate Revocation Lists (CRL's)*, the *Online Certificate Status Protocol (OCSP)* [16] and the *XML Key Management Specification (XKMS)* [20].

The evaluation was conducted with the help of an application performing *signed content* as well as *certificate validation mechanisms*. It has been implemented using the *Java* programming language and a cryptographic library from the IAIK Institute, TU Graz[1], as well as an interoperability release of *XKMS* offered by *VeriSign*[2].

This article will first present an *evaluation framework*. Next, the aforementioned four *certificate validation mechanisms* will be evaluated and discussed. Finally, a summary will give an overview over the results of the evaluation.

The reader is assumed to be familiar with the *X.509 Authentication Framework* [8, 7] in general and with *certificate validation mechanisms* in particular.

2 Evaluation Framework

Our evaluation framework serves to specify the parameters and criteria used later on during the evaluation. It is based upon the *MITRE Report* [3] and a thesis performed by André Årnes [18].

Since no real-world case-study on usage of certificate validation mechanisms exist so far, we base our study on a set of assumptions, which have also been used in previous studies.

The criteria by which this evaluation will be performed are listed in Table 1.

The assumptions by which the performance evaluation will be performed, as well as the parameters which are going to be evaluated are listed in Table 2. All input parameters are set to upper-case letters, while the output parameters, this being a point of interest,

[1] Institute for Applied Information Processing and Communication, Graz Univerisity of Technology, Austria
[2] http://www.xmltrustcenter.org

Table 1: Evaluation Criteria used in this evaluation

Criterion	Description
Security	Effective protection against malicious attacks.
Interoperability	Standards-compliance and acceptance.
On-line vs. Off-line	Necessity of network connections.
Complexity	Management effort required to handle a particular scheme.
Performance	Performance in terms of data sizes and transmission times. Scalability of the scheme.

are set to lower-case letters.

It is assumed also that the *scalability* is a specific topic within the *performance* criterion.

Within our evaluation framework all *message digests* were performed with the *SHA-1* [6] one-way hash function and the signatures were created with the *RSA* [10] public-key algorithm using a modulus of $K = 1,024$ bits and a public exponent of $e = 17$. It is believed that this keysize offers strong enough security for the purpose being discussed.

In a real-world scenario it can be supposed by a *Certificate Authority (CA)* to take care of a range between $N = 10,000$ up to $100,000$ users. As the purpose of this evaluation is not to study a PKI with distributed CA's, but to evaluate different validation schemes, the choice fell to the setup a system with one CA.

Regarding the validity period T of certificates, an average period of time commonly used by commercial providers is 1 year = 365 days. For *short-lived certificates* it is assumed to have a validity time of only 1 day.

Due to the lack of realistic figures about the number of content providers offering *signed content* a relative figure F describes the percentage out of all users N offering *signed content*. Three use-cases are going to be analyzed for specifying F.

In case the *signed content* providers are offering web services, such as an m-commerce platform, two distinct scenarios may occur. On one hand only a few, say $F = 10\%$, content providers are accessed by many users a day, which means that they will need to get themselves a certificate. This could be the case for a restricted amount of content providers really needing to transmit *signed content*.

On the other hand it may happen that few users access many different content providers. This would happen if the user's behavior is statistically independent. In that case F would increase and is assumed to be 25%.

When observing the *messaging* use-case, it can be assumed that a fairly high amount of the N users will send at least one signed message a day, and consequently needs a new certificate for that purpose. This would lead F to be 50%.

A very interesting issue is the rate of users U out of all (N) who request a status information about the validity of a particular certificate, and from these users U, the amount of status requests Q which they perform on average each day. It is assumed that the number of status requests Q are going to be all new requests and the responses will not be cached. This also means that all queries Q will be for Q different certificates.

Common literature [3, 18] assumed that certificates will be revoked due to distinctive circumstances with a probability P of 10% out of all issued certificates. Since no better and newer estimates exist on this figure we will also assume this value. Nevertheless, regarding the timely fashion of issuance of CRL's R by the respective CA, it is assumed to be done each day, as the number of revoked certificates can reach an average of 2.73 or 27.39, depending on the number of total users (N) involved.

3 Evaluation of Certificate Validation mechanisms

3.1 Security Evaluation

A certificate has the goal to securely bind a particular public-key to the name of the key holder. The *security* evaluation will now determine the strength of the offered security by focusing on the encoding and on the protocols and leaving out any discussion over the cryptographic algorithms used within each proposal. Detailed information regarding this issue can be obtained from [19, 11].

Short-lived Certificates

From a *security* perspective, *short-lived certificates* present a high risk of failing the secure binding. This is caused by an uncertainty of the *optimum validity period* which should be given to such a certificate. No revocation is possible and therefore the validity period should be decided to be as conservative as possible. However, if such an *optimum validity period* is found,

Table 2: Model of a PKI for performance and scalability evaluation

Parameter	Description	Value	Unit
K	Size of RSA modulus	$1,024$	bits
N	Number of Users using the infrastructure	$10,000$; $100,000$	users
C	Number of CA's	1	CA
T	Validity period of a certificate	1; 365	day
F	Percentage of the content providers	10; 25; 50	$\frac{\%}{N}$
U	Percentage of the requesting users	1; 5; 10	$\frac{\%}{N}$
Q	Status requests per day per user	1; 5; 10; 20	$\frac{1}{day \cdot user}$
P	Percentage of revoked certificates	10	$\frac{\%}{N}$
R	Number of CRL's issued per day	1	$\frac{1}{day}$
l_i	Size of the data structure i		bytes
v_i	Data volume of transaction i		bytes
r_i	Request rate of transaction i		$\frac{1}{sec}$

it can be stated that the *timeliness*, expressed in the freshness of the information contained in such a certificate, is favorable.

In order to establish an estimation of the *optimum validity period*, possible reasons for certificate invalidation need to be analyzed. On one side there is a probability of change of any of the personal data included into the certificate, such as a *cessation of operation* or any change of the *Distinguished Name (DN)*. Much more important is the probability of *key compromise*, which means that an attacker has either gathered knowledge of the private-key or manipulated the public-key, so that he can make use of the keypair of any victim.

The probability of appearance of the former reasons can not clearly be determined. They depend too much on the use case and the properties of the users. If the certificate for instance is used in a work environment, it can be supposed that the probability of any change is relatively small within the contract's *term of notice*. Restrictions and policies can be included into the *Certificate Practices Statement (CPS)* in order to restrict any possible risks of invalidation.

The invalidation reason of *key compromise* may reside on the strength of the cryptographic algorithms used or on the revealing of the keypair due to an attack on the storage mechanism, among others. As an X.509 certificate is built in such a way that it is algorithm independent, they are interchangeable with stronger alternatives.

The *WAP Forum* has recommended in [23] a validity period of 48 hours, which is 2 days, requiring an overlapping period of 24 hours in order to avoid misbehavior of unsynchronized clocks at the handheld device. This period of time seems reasonable from a security perspective.

The availability of synchronized clocks, ensured either through a *Timestamping Authority (TSA)* [1] or by the usage of trusted and reliable *network time protocols* [12, 14, 13] can considerably reduce the overlapping period, but adds rountrips.

Additionally, the *short-lived certificates* model is more vulnerable to *Denial-of-Service (DoS)* attacks from malicious users flooding the *Certificate Authority (CA)* with certificate requests than any other use-case. This is mainly due to the high request rate a CA needs to serve on a daily basis, as it is shown in section 3.5.

Certificate Revocation Lists (CRL's)

All contents included into a *CRL* are signed by the CA. Consequently, no possibility is given to an attacker to *add*, *remove* or *manipulate* the content of a CRL, unless the CA's keypair is compromised. In this case the CA should revoke its own certificate and publish it either through an *Authority Revocation List (ARL)* or through CRL's issued by the upper-lying CA.

A CRL is published in a fashion according to a particular policy. The *MITRE report* [3] assumed that 2 such CRL's were published each week. Even when being published on a daily fashion it has been shown in section 2 that for a CA taking care of $N = 100,000$ users on average 27.39 certificates would be revoked

and included into the CRL. The *freshness* of such a daily updated CRL is consequently doubtful.

A requestor of a certificate which has been revoked just after the publication of *CRL 1*, as shown in Figure 1, would not get to know that the requested certificate had been revoked until the publication of *CRL 2* on the next day. That means that he would trust an already revoked certificate for a period ΔT_2 without knowledge of the certificate being revoked.

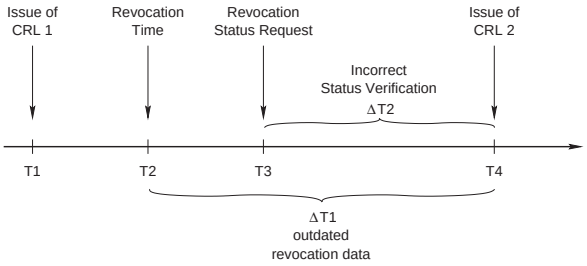


Figure 1: Incorrect certificate status verification through outdated CRL

It could be required that a CRL should be issued after each new entry of a revoked certificate. In that case the *timeliness* could be improved to an approximate hourly fashion, achieving however an increase of the costs of communication and performance.

Online Certificate Status Protocol (OCSP)

OCSP offers a possibility of gathering knowledge over a particular status of a certificate without the need to acquire a CRL. As the *OCSP response* is signed by the *OCSP Responder* the client needs to have a possibility of verifying the correct identity of the signer, in order to trust the information included. This means that the client needs to have a certificate of the responder, which he can trust upon.

It can be stated that the solutions offered by [16] are not satisfactory, as it provides no clear way for a client to get a proof of the correctness and validity of the *OCSP Responder's* certificate.

The variant of a *Trusted Responder* states that it is simply *assumed* to be trusted. This is definitely not a practicable approach. In the *Authorized Responder* option the CA includes a specific flag[3] in the responder's certificate, in order to grant it with privileges for signing *OCSP Responses*. It can be considered to have a comparable sense to the `keyUsage` extension in a certificate, stating the purpose of usage.

The *CA-Responder* alternative, with the CA acting directly as the *OCSP Responder*, is the only one suitable for a real use-case. The CA is already entitled to sign certificates and CRL's and for simplicity and security purposes it should also publish status responses. The reason is that the trust an end-user puts into its CA can be used to implicitly trust the *OCSP Responder*. No further trust assertions need to be ensured. Anyhow, there must be a dedicated communication channel from the CA to the *OCSP Responder*, as the revocation requests are addressed to the CA. This CA would then have to communicate the *OCSP Responder* the certificates which have been revoked. Therefore, and in order to avoid possible leaks of information on such a channel, this service should be directly merged into the CA. Taking into account *WAP Forum's* effort of the *W-PKI* specification [23], this scenario would perfectly fit into the *PKI Portal* and therefore it is strongly recommended.

In case the *OCSP Responder* is impersonated or any of its assertions has changed, all certificate status responses would be invalidated. The usage of the extension `id-pkix-ocsp-nocheck` as proposed in [16] is *not* recommended, as a compromise of the *OCSP Responder's* certificate could have the same effect for instance as a compromised CA's certificate signing CRL's. The only alternative which is recommended in case of invalidation is the inclusion into a CRL or even better into an *Authority Revocation List (ARL)*, if using the *CA-Responder* mode. Although this involves again the publication of such lists, and does not solve the *CRL problem*, but merely reduces their size.

As OCSP does not prescribe any use of secure communication channels, a *man-in-the-middle* attack can be performed in order to avoid that the requestor gains knowledge about the certificate status. Its request, as well as a produced response, can both be retained from reaching its specified destination. The attacker can easily produce an unsigned response specifying a `responseStatus` which may give the requestor a false sense of the reality. For instance an attacker can impersonate an *OCSP Responder* by intercepting a request and respond with `unauthorized`, in spite of the requestor being authorized. In such a situation the requestor would either be able to verify the certificate or have to proceed to search for a possibly outdated CRL. Another possibility could be to simply delay the response, in order to cause the client to time out and skip the status verification.

The reason for such attacks is the unsigned `responseStatus` field. In order to avoid such attacks, a revision of the OCSP protocol should include this field into the data structure to be signed. However, this would increase the processing time in case the *OCSP Server* is suffering any kind of malfunction and a fast message should be conveyed to the requestor for information.

The major benefit of OCSP lies in the *timeliness* of the validation process. Whereas CRL's are issued

[3] `extKeyUsage`

periodically, an OCSP request can determine the status of a certificate with almost perfect accuracy, independently of the availability of a valid CRL. It is the responsibility of the OCSP responder to get the most recent revocation information, either through CRL's or by any other means.

An additional *security* feature of the OCSP protocol avoiding *response-replay* attacks, is the optional inclusion of a `nonce` to bind a particular request to a response. If such a value is used, the requestor includes a randomly chosen `nonce` into its request and the responder will extract it upon receipt and parse it into its response. When the requestor receives the response, it can then check if this is the response to its request by verifying the value of the `nonce`.

Problems with avoiding the *response-replay* attacks may arise in case of *response pre-production* being used. This feature allows faster service completion, as the signature creation is the computationally slowest operation, as will be shown in section 3.5. If an attacker gains access to the repository of the pre-produced response, it can deliberately change the entries if they are not efficiently secured. It can also take a previous pre-produced response and send it to the requestor, even though a more updated response should be produced. This has a direct effect on the *timeliness*.

It also has to be mentioned that an important task of verification of a certificate's validity is performed by an entity being outside of the verifier's domain. This entity has to be fully trusted by all end-entities. As the process is centralized, a *Denial-of-Service (DoS)* by flooding the responder with queries can effectively block all verification processes within one domain and lead to catastrophic consequences. As the requests do not specify to which responder they are directed, this attack can be target multiple responders by replaying the request flooding.

A possibility to avoid such an attack could be to accept only *signed* requests, as they can easily be monitored. All unsigned requests should be then discarded as potentially being harmful attacks. Nevertheless, this solution introduces an additional processing load in a mobile device. A further possibility would be to setup a specific policy for the choice of the `nonce`, being used not only for request-response binding, but also for *authorization* of the service. An example of such a policy could be the setup of one-time passwords used for `nonces`. The responder would then first verify the correct `nonce` before proceeding to create a response.

Finally it has to be remarked that a secure usage of *OCSP* requires trusted and reliable timestamps, as the validity periods of such transactions have to be short. The significance of synchronized clocks between the *OCSP Responder* and the requesting client is critical.

XML Key Management Specification (XKMS)

XKMS is comparable in the *security* considerations to OCSP in that it offers similar services together with some additions, such as *validation* of a full certificate path or *key recovery*.

The main disadvantage though is, that currently *no* mechanism is provided for a client to make a trust decision about the *XKMS services's* public-key used for verifying the signature over the responses sent to a request. In case this key is compromised, no possibility is offered for a trusted replacement or revocation.

It is strongly recommended that this mechanism should at least offer a possibility to establish a trust relationship to any CA's certificate, such as the *CA-Responder* mode in the *OCSP* protocol. The upoincoming *XML Trust Assertion Service Specification (XTASS)* [21] will address this problem.

The possibility of impersonating the *XKMS service* through a *man-in-the-middle attack* is successfully prevented as all the fields contained within the response are signed. Even in case of failure, the *XKMS service* would respond with a signed message, indicating in its `<ResultCode>` field with a value of `Failure`.

Further on, *replay-attacks* are avoided by including a `<TransactionID>`, which is comparable to the `nonce` in the *OCSP* variant. This `<TransactionID>` binds a particular request with the correspondent response.

In order to secure the service against *Denial-Of-Service (DoS)* attacks, the possibility of implementing a *limited-use shared secret* indicating a `<PassPhraseAuth>` and/or a `<KeyBindingAuth>` element allows for previously negotiated authorization of the service. These features should nevertheless only be used in conjunction with secured communication channels [20], as they do not provide efficient security.

In case the *XKMS service* is used for *key recovery* purposes special care needs to be taken of the implementation and its policy to not allow for *repudiation* or for unintended *accountability*.

The major benefit of *XKMS* is that it provides good *timeliness*, as the *XKMS service* will have dedicated communications channels to communicate with each PKI and get *fresh* status responses. Nevertheless, the synchronization of timestamps has to be ensured either by the inclusion of a *Timestamping Authority (TSA)* [1] or by secure *network time protocols* [12, 14, 13] as it is highly critical.

3.2 Interoperability Evaluation

The *interoperability* of a certificate validation mechanism is important since multiple vendors may implement such a service. Clients retrieving the data should be able to interoperate without any problems. The *interoperability* can be ensured through a standardization process.

Short-lived Certificates

Short-lived certificates are standardized X.509 certificates with a restricted validity period. The *International Telecommunication Union - Telecommunication Standardization Sector (ITU-T)* first specified such data structures in *Recommendation X.509* [8] and then it was adopted by the *Internet Engineering Task Force (IETF)* as RFC 2459 [7].

For quite some years this has been the most widely deployed *PKI* and therefore it can be considered to be fully interoperable. Not only the data structures have been standardized, but also full certificate management protocols [2, 17] have been specified in order to enable a smooth exchange of certificate related information.

As mentioned earlier, certificates issued according to ITU-T's X.509 recommendation are encoded according to the standardized ASN.1 syntax using the *Distinguished Encoding Rules (DER)* [9].

Regarding the cryptographic algorithms used for generation and verification of such certificates, it can be stated that the way X.509 was specified, enabled a full algorithm independence. A specific set of proposed algorithms is listed in [8, 7]. The only requirement is to acquire an ASN.1 `OBJECT IDENTIFIER` value, in order to be properly recognized by all receiving parties.

Certificate Revocation Lists (CRL's)

CRL's have been standardized into the same specifications [8, 7] as the aforementioned *short-lived certificates*. Therefore they can be considered equally interoperable between different implementations as well as between different parts of the encoding.

Online Certificate Status Protocol (OCSP)

The specification of *OCSP* [16] was driven by IETF's *PKIX Working Group* of the *Security Area*, which has also been working on the adaptation of X.509 certificate profiles.

It is currently being revised [15] in order to add a new set of features into the protocol.

Several implementations from different vendors, such as the libraries available from Baltimore and IAIK, have proven to be fully interoperable. The fact that data structures are encoded in ASN.1 DER notation, also facilitates interoperability.

The cryptographic algorithms chosen in OCSP to handle the signature of requests and responses, is performed in an algorithm independent manner. A set of algorithms is proposed to be used for hashing and signing, but an addition of other algorithms is possible.

XML Key Management Specification (XKMS)

XKMS is a very recent specification driven by *VeriSign*, *Microsoft* and *webMethods*. It has been published to the *World Wide Web Consortium (W3C)* as a *Technical Note*. Therefore it has to be considered as not yet standardized.

Both *VeriSign* and *Entrust* offer separate interoperability implementations.

XKMS is encoded using standardized XML syntax, and therefore it should grant interoperability to all XML-related protocols and methods. An example of it can be considered to be the usage of the *XML Digital Signature* [5] method used for signing all requests and responses within the XKMS protocol.

3.3 On-line vs. Off-line Evaluation

The *on-line vs. off-line* criterion is relevant not only for the *performance* and *scalability* of the scheme, but also for the *security* analysis, as particular vulnerabilities, being dependent on this requirement, may lead to various exploits.

Short-lived Certificates

The certificate verification process of *short-lived certificates* works purely *off-line*. No network connection needs to be established in order to verify the validity status of such a certificate once all required certificates for path validation are available. However, the collection of such certificates can be performed on a network query, but may also be pre-loaded into the verification entity on manufacturing, as exemplified by *WAP Forum's* specifications [23, 24].

Certificate Revocation Lists (CRL's)

CRL's can be considered half-way *on-line*, as it is recommended to download an updated copy of such a data structure before any certificate validation. The procedure should start first by verifying the `nextUpdate` field, in order to evaluate the need of querying for a new CRL. If the variant of *over-issuing* certificates is being used, it makes sense to directly query the CA for a new CRL. Once the most updated CRL is available, the process of verification occurs *off-line* directly on the client.

Online Certificate Status Protocol (OCSP)

As the name of this protocol states itself, it is an *on-line* status verification protocol. This means that upon each certificate verification request, an *on-line* connection to the *OCSP Responder* needs to take place in order to determine the actual validity status of the requested certificate.

The path validation and signature verification occurs then *off-line* on the client's side. Up to date, OCSP merely gives a response regarding revocation status of the requested certificate. However, it is planned to incorporate full path validation into OCSP [15] in the future. In that case, the protocol would turn into fully *on-line*, as almost no trust assertion would be performed *off-line* on the client's side.

XML Key Management Specification (XKMS)

XKMS offers a full range of possibilities, varying from registration over location up to full path validation.

In case the client decides to use the *XKMS service* only for location of public-keys, any further verification would take place on the client itself and thus be *off-line*. Instead, if it decides to let the service perform a full validation of a submitted public-key, the service would be considered *on-line*.

3.4 Complexity Evaluation

The *complexity* of each of the proposed certificate validation mechanism is dependent on the amount of processing, as well as on the required user interaction. The latter includes both the user of a client as well as the administrator of the service.

Independently of the chosen scheme, all of them require cryptographic processing facilities of *message digest* algorithms and public-key algorithms for creating the hash functions and the signature respectively.

Short-lived Certificates

The *short-lived certificates* require an ASN.1 encoder. As previously mentioned, the most important decision to be made, is the *validity period*. This should be defined in the *Certificate Practices Statement (CPS)* along with the inclusion into the validity period fields of a certificate.

The *complexity*, in terms of management required to handle this proposal, is relative to the *validity period*. On one side only one data structure is used to convey trust assertions, the certificate itself. On the other side, the fact of re-issuing such data structure, even in case of absence of changes or of key compromise leads to an enormous management overhead for both parties, the client and the CA. The client is forced to re-send a PKCS#10 formatted certificate request which the CA takes as input to re-issue a certificate.

This re-issuing should not be automated but reviewed on each iteration, as changes may have occurred, which can not be noticed by an automated service.

Depending on the policy applied to the client side, it may be recommended to use a pre-signed and stored PKCS#10 certificate request, in case of absence of any changes occurred to the personal information of the user and of usage of the same keypair. If any of both data changes, the client should create a new certificate request, sign it and send it to the CA. This proposal would slightly reduce the management overhead, even though security considerations should be taken into account, such as possible attacks on the used keypair.

Certificate Revocation Lists (CRL's)

The usage of *CRL's* requires an ASN.1 encoder. In that aspect there is no difference to the previous scheme.

From a management perspective, this scheme relies upon two different data structures. On one side there is the certificate. On the other side a revocation list informs about all invalid certificates. This means that the client is required to gather the most updated CRL, before being able to verify the correctness of the certificate itself.

The CA needs to maintain a revocation service, enabling all end-users to communicate their invalidation request. Once such a request is received by the CA, it needs to include the revoked certificate into the *CRL*, sign it and publish it to the public repository.

Online Certificate Status Protocol (OCSP)

OCSP requires, as the two previously mentioned schemes, ASN.1 capabilities.

From the management point of view, this proposal requires the client to maintain online access to the *OCSP Service* in contrast to the other both schemes. Whenever a certificate needs to be verified for trustworthiness, an *OCSP Request* needs to be sent, indicating the certificate to be checked. It may happen that this request needs to be signed, which means that it requires specific user interaction.

The *OCSP Server* will need to be a dedicated service, as it will have to handle on one side the requests and issue proper responses, while on the other side it will have to interact constantly with the CA or any other repository, in order to gain *fresh* information about any certificate's status within its domain. The OCSP specification [16] only defines a protocol for

communication between the requesting client and the *OCSP server* and the management which has to be performed on that side, but leaves open the means by which it is supposed to gather certificate status information.

In order to reduce the management on the server-side, *pre-computed* responses may be produced upon receipt of a CRL or any other status information list. These *pre-produced* responses are then stored in a secure database and sent to any requesting party on request. The main reason for pre-computation of *OCSP responses* is mainly due to the computationally expensive signature creation. However, it needs to be mentioned that this pre-production has serious security weaknesses, as described in the section 3.3.

XML Key Management Specification (XKMS)

XKMS requires support for XML encoding. Additionally, it is dependant on the *XML Digital Signature*. All signatures performed over requests and responses need to be performed using this latter scheme.

The *XKMS* protocol has a management overhead comparable to the OCSP variant, in that a client needs to interact continuously with the *XKMS service* if any trust decision is to be done. Additionally, the *XKMS* service is required to obtain constantly *fresh* information about the current status of the keys, if used in conjunction with the PKI as described in [8, 7].

The client using such a service needs to determine either by itself or by a given policy to what extent it will use the services provided by XKMS. It may be desirable in particular use-cases to simply *locate* the keys of another end-user or a content provider and to perform the validation on the own client. In other use-cases, it may be possible to off-load such a verification to the service and to simply get a final result. This decision needs to be clearly defined either in a usage policy or explicitly made by the user being aware of security considerations.

The major advantage of *XKMS* is that one infrastructure offers a full range of independent services. The user decides the degree of trust assertion he wants his client to off-load to the *XKMS* service.

3.5 Performance Evaluation

The *performance* evaluation of *certificate validation* mechanisms will be divided into three parts. First, a *size evaluation* of message sizes l_i of the required data structures will be performed.

Additionally, a *scalability evaluation* of such mechanisms into real-world scenarios will be performed. This will be measured by the *data volume* v_i transferred using a pre-established model.

Table 3: Specific environment parameters for the performance evaluation

Parameter	Description	Unit
$l_{PKCS\#10}$	Size of a *PKCS#10* Certificate Request	bytes
l_{Cert}	Size of an *X.509v3 Certificate*	bytes
l_{CRL}	Size of an *X.509 CRL*	bytes
$l_{OCSPReq}$	Size of an *OCSP Request*	bytes
$l_{OCSPResp}$	Size of an *OCSP Response*	bytes
$l_{XKMSReq}$	Size of an *XKMS Locate Request*	bytes
$l_{XKMSResp}$	Size of an *XKMS Locate Response*	bytes
$v_{sh.-liv.}$	Data volume for the *short-lived certificates* variant	bytes
v_{CRL}	Data volume for the *CRL* variant	bytes
v_{OCSP}	Data volume for the *OCSP* variant	bytes
v_{XKMS}	Data volume for the *XKMS* variant	bytes

Table 3 lists the parameters which are going to be measured.

Size Evaluation

The *sizes* of the data structures required for the certificate management according to [8, 7] have been measured with a BER viewer[4] for all ASN.1 data structures and with a text editor for the XML-based files.

Figure 2 illustrates a PKCS#10 Certificate Request message, an X.509v3 certificate as well as a CRL.

It has to be mentioned that the `Issuer` and `Subject` fields have freely been chosen with common names, such as:

- `Issuer: EMail=ca@SignedContent.com, CN=CA for SignedContent, OU=EED/R/A, O=Ericsson, STREET=K-Street 9, ST=NRW, L=Aachen, C=DE`
- `Subject: EMail=xxxxxx@eed.ericsson.se, CN=User 01, OU=EED/R/A, O=Ericsson, STREET=K-Street 9, ST=NRW, L=Aachen, C=DE`

In order to request a certificate, a *PKCS#10 Certificate Request* needs to be created and sent to the CA.

[4] Aram's BERViewer v2.1.1

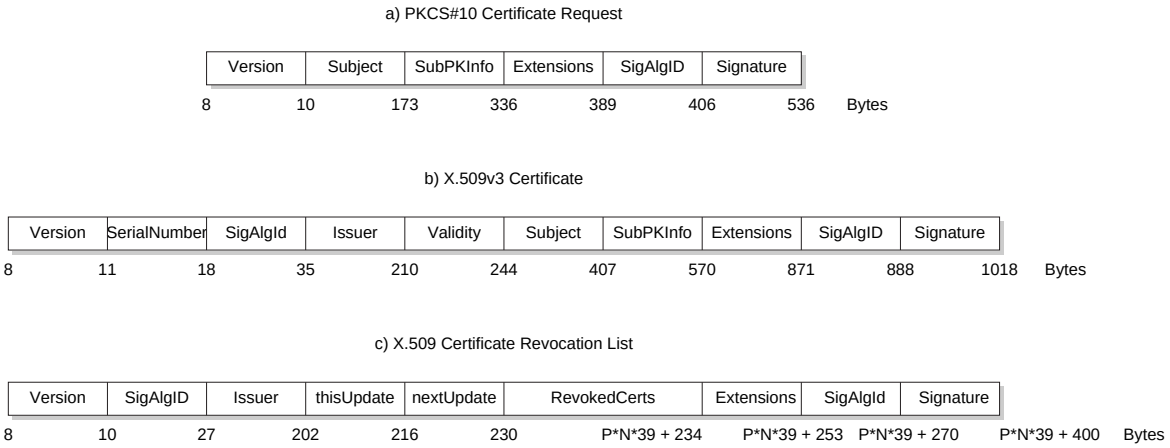


Figure 2: Sizes of a) PKCS#10 Certificate Request, b) X.509v3 Certificate and c) X.509 Certificate Revocation List (CRL)

This will be very important later on for the *scalability* evaluation of *short-lived certificates*. This PKCS#10 message incorporates a `Subject` field and the `SubjectPublicKeyInfo` with an `OBJECT IDENTIFIER` indicating that the public-key included is an RSA key (`rsaEncryption`) with a size of $K = 1024$ bits. The `extension` field carries an additional `subjectAltName` for including the `email` of the subject. This message is finally signed and the signature appended with an `OBJECT IDENTIFIER` of `sha1withRSAEncryption`.

Such data structure, when being encoded using the DER encoding rules, has a typical size of $l_{PKCS\#10} = 536$ bytes.

The certificate, which the CA then issues to the requesting end-user includes a `SerialNumber`, a `SignatureAlgorithmIdentifier` which according to the received PKCS#10 request is `sha1withRSAEncryption`, the `Issuer`'s *Distinguished Name (DN)*, a validity period, indicating `notBefore` and `notAfter`, the `Subject`'s DN, its `SubjectPublicKeyInfo`, including the `OBJECT IDENTIFIER` together with the public-key value, and finally some specific `extensions`, as mandated in [22]. This sequence is then signed and the signature appended to the certificate together with the `OBJECT IDENTIFIER` specifying that this signature is created by first applying the *SHA-1* hash algorithm and the signing with the *RSA* algorithm.

The X.509v3 certificate is then shown to have a total size of $l_{Cert} = 1018$ bytes. The size of the certificate of the CA has been measured to have $l_{Cert,CA} = 787$ bytes.

The *CRL* is constructed by the `SignatureAlgorithmIdentifier` set to `sha1withRSAEncryption`, the issuer's DN, a `thisUpdate` field as well as a `nextUpdate` field together with the list of `RevokedCertificates` and some mandatory extensions. This sequence is then signed and the signature appended to the CRL.

As the list of `RevokedCertificates` can be huge, the size of a CRL is dependent on this list. Each entry of a revoked certificate takes 39 bytes, including a 6 byte long `SerialNumber` of the revoked certificate, a 13 byte long `RevocationDate`, a 12 byte long `ReasonCode` and 2 bytes of indication for each type of the data structure. The size of the fixed fields has been measured to be 400 bytes. Taking into account the total number of certificate holders N within the scope of one CA and a revocation rate of $P = 10$ %, the following equation 1 calculates the actual size of such a CRL:

$$l_{CRL}(P, N) = P \cdot N \cdot 39 + 400 bytes \qquad (1)$$

With $N = 10,000$ the size of the CRL is $l_{CRL} = 39,400$ bytes. In case of $N = 100,000$ the CRL's size would increase to $l_{CRL} = 390,400$ bytes. It has to be admitted that these sizes are only approximate, as expired certificates will be removed from the CRL and a stationary use case involves much more complex calculations about the number of included certificates. However, these calculated sizes can be determined realistic when observing the commercially available CA's and the CRL's they publish.

The sizes of *OCSP* requests and responses are shown in Figure 3. Several fields are optional, as well as the content of the specified fields can vary upon usage. Therefore any size evaluation can only be considered relative to each other, and not absolute.

An *OCSP Request* includes the *Relative Distinguished Name (RDN)* of the requestor. The `RequestList` includes the `CertID` of the certificate to be verified. This `CertID` consists of an hash algorithm identifier, the hash of the issuer's DN, the hash of the issuer's public-key and the `Serial Number` of the re-

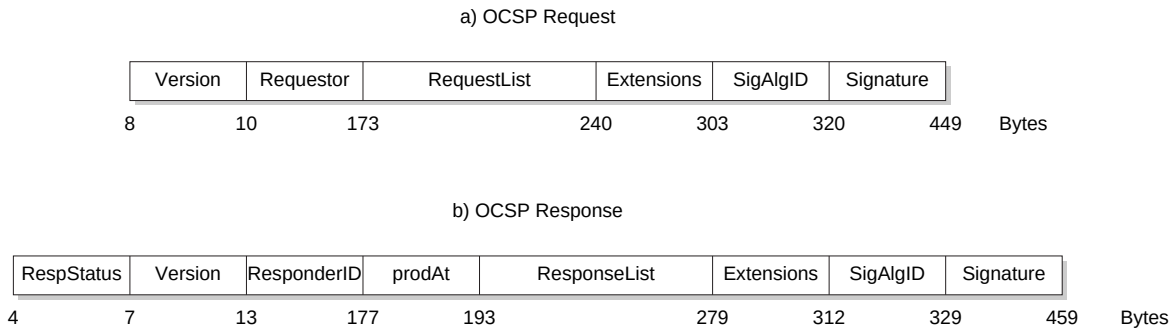


Figure 3: Sizes of a) OCSP Request, b) OCSP Response

quested certificate. The `requestExtensions` includes two fields. First, `AcceptableResponses` with a value of `id-pkix-ocsp-basic`, which means that the *OCSP Responder* should reply with a `BasicOCSPResponse`. Second, a `nonce` of 16 bytes random data is included for binding the responses to the requests.

Finally, this *OCSP Request* has been signed, even though it is not mandatory. The reason for this is to enable the objective comparison of sizes of *OCSP* and *XKMS* messages with enhanced capabilities.

Such an *OCSP Request* has then a final data size of $l_{OCSPReq} = 449$ bytes. The unsigned request would have a size of $l_{OCSPReq} = 303$ bytes.

The *OCSP Response* answers to this request with a `ResponseStatus` of `successful`. The `ResponderID` is given by the *OCSP Responder's Relative Distinguished Name (RDN)* for full expressiveness. The `producedAt` field indicates the date of production of this response message. The `RequestList` gives the status response to the certificate in question, as well as return the `CertID`, the `CertStatus` and the date of `thisUpdate`, being the same as the one included into the `producedAt`. This is not necessarily true in case of response pre-production. The `nextUpdate` field has been explicitly discarded, since it is optional, and does not providing further information. It is assumed that whenever a status verification is desired, an *OCSP Request* should be issued, avoiding caching of responses at any place.

In this particular case, the status was determined to be `good`, and therefore no additional `revokedInfo` needed to be appended to the `CertStatus`. The `nonce` obtained from the request has been included into the `responseExtensions` for message-binding reasons.

This `BasicOCSPResponse` has finally been signed by the *OCSP Responder*, leading to a total size of $l_{OCSPResp} = 459$ bytes.

In order to maintain consistency with the previous *OCSP* messages, it has been an explicit intention to supply the *XKMS* messages with consistent data, trying to establish a comparison wherever possible. It has also to be mentioned that even though *XKMS* is meant to be encoded using the *Simple Object Access Protocol (SOAP)*, the measurements have been performed analyzing *raw* XML structures. The usage of *SOAP* for transportation would increase the sizes by 109 bytes.

An *XKMS Locate* message, as illustrated in Figure 4, has the purpose of locating a public-key to a requested name, or vice-versa. It is not intended to give any kind of status information. Nevertheless it has been included into this evaluation, as being very interesting for public-key exchange. An *XKMS Locate Request* includes a `TransactionID`, serving as a `nonce` in the OCSP protocol. The `Query` indicates either the name or the key, where the missing element is the requested one. The `Respond` field states the desired response information. This is typically set to `KeyName`, `KeyValue` and `SignedResult`. The latter is meant for the *XKMS Server* to sign the response message.

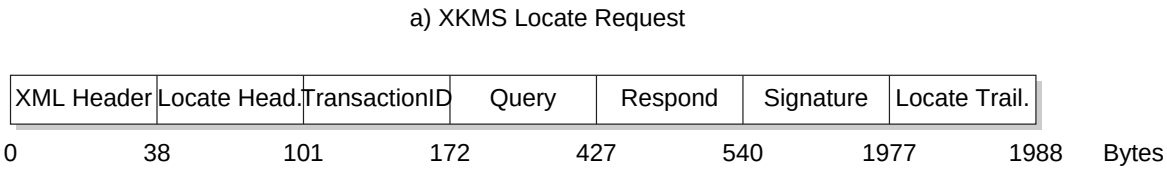
The total size of such an *XKMS Locate Request* sums up to $l_{XKMSReq} = 1988$ bytes.

The *XKMS Locate Response* message, which is sent by the *XKMS service* contains a `ResultCode`, comparable to the `responseStatus` of the *OCSP Response*. The `TransactionID` included by the requestor is returned unchanged. The `Answer` of such a *Locate* message includes the requested `KeyValue` as well as the `KeyName`. An *XML Digital Signature* over the whole message prevents any attacker from manipulating the response.

The total size of this *XKMS Locate Response* is $l_{XKMSResp} = 2364$ bytes.

The *XKMS Validate Request* can either serve to get a revocation status response, as in the previous *OCSP* protocol, but it can also request the *XKMS service* to perform a full path validation of a requested public-key or name. An example of such a message pair is shown in Figure 5.

Exactly as in the *XKMS Locate Request*, the *XKMS Validate Request* contains a `TransactionID`. However, it is not yet clear to the specification committee as to include it into the `Query` element, or leave it outside,



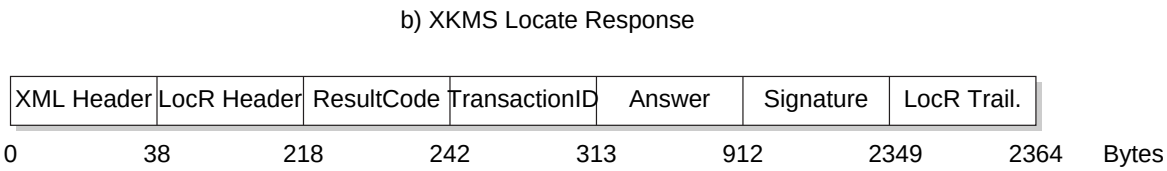
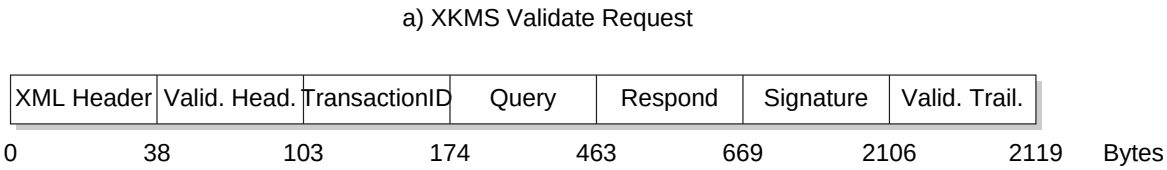


Figure 4: Sizes of a) XKMS Locate Request, b) XKMS Locate Response



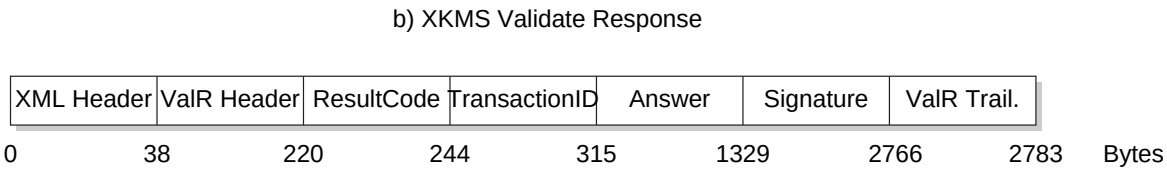


Figure 5: Sizes of a) XKMS Validate Request, b) XKMS Validate Response

as in the previous message type. For simplicity reasons this validation request will keep it outside. Additionally, the `Query` gives the responder additional information to one the requestor already has. It can be an `Indeterminate` status as well as the `KeyName` or the `KeyValue`. The `Respond` element defines the elements which are included into the *XKMS Validate Response*. This are typically going to be the missing counterpart of `KeyName` or `KeyValue`, the `ValidityInterval`, the `KeyUsage`, the `Status` and a `SignedResult`.

This request message has also been signed for consistency purposes. The total size measured of this *XKMS Validate Request* message is $l_{XKMSReq} = 2119$ bytes.

The *XKMS Validate Response* message contains besides the `ResultCode` the copied value of the `TransactionID` from the *XKMS Validate Request*. The `Answer` contains a `KeyBinding` sub-element, in which all relevant and requested status assertions are inserted. These are `KeyID`, `KeyName`, `KeyValue`, `ValidityInterval` and `KeyUsage`. This response is also signed using the *XML Digital Signature* method and has a size of $l_{XKMSResp} = 2783$ bytes.

In order to summarize the *size evaluation*, Table 4 compares the presented and evaluated data structures.

As it can be observed, the size of a CRL is determined by the amount of revoked certificates within one domain. This makes it very hard to estimate a proper size, as the parameter P of the amount of revoked certificates has not been identified by a detailed study, but has been taken from previous analyses.

It is clearly visible that the data size of an *OCSP Request* is considerably smaller than the size of an *XKMS Locate Request* and an *XKMS Validate Request*, when comparing signed requests. Whenever unsigned requests are sent, this difference decreases, but still exists. This is mainly due to the contribution in the size of the *XML Digital Signature*. The *OCSP Response* also shows a significantly smaller size than both *XKMS Response* variants.

Table 4: Sizes of different Certificate management data structures (in bytes)

Type	Parameter	Value
PKCS#10 Cert Request	$l_{PKCS\#10}$	536
X.509v3 Certificate (CA)	l_{Cert}	787
X.509v3 Certificate (End-User)		1,018
CRL (with $N = 10,000$)	l_{CRL}	39,400
CRL (with $N = 100,000$)		390,400
OCSP Request (unsigned)	$l_{OCSPReq}$	303
OCSP Request (signed)		449
OCSP Response	$l_{OCSPResp}$	459
XKMS Locate Request (unsigned)	$l_{XKMSReq}$	551
XKMS Locate Request (signed)		1,988
XKMS Locate Response	$l_{XKMSResp}$	2,364
XKMS Validate Request (unsigned)	$l_{XKMSReq}$	682
XKMS Validate Request (signed)		2,119
XKMS Validate Response	$l_{XKMSResp}$	2,783

Scalability Evaluation

The *scalability evaluation* will determine how well each of the four proposed certificate validation mechanisms scales in the described scenario of section 2.

In order to perform such an analysis, the main parameter is decided to be the total data volume v_i transferred on 1 day assuming there is only 1 CA serving all users N for each mechanism.

The *short-lived certificates* scenario is described in Figure 6a. Consequently, the data volume $v_{sh.-liv.}$ generated is relative to the amount of content providers F out of N offering *signed content* as well as to the rate of users U requesting *signed content* from the content providers. Both parties generate a specific amount of traffic.

The traffic generated by the content providers are the requests to the CA to get a certificate by sending a PKCS#10 formatted certificate request as well as the certificate itself sent back. Additionally, each user will request the certificate from the content provider for each request Q. Equation 2 intends to show the data volume $v_{sh.-liv.}$ generated using this model during 1 day:

$$v_{sh.-liv.} = ((l_{PKCS\#10}+l_{Cert})\cdot F+l_{Cert}\cdot Q\cdot U)\cdot N \quad (2)$$

The *CRL* model, as illustrated in Figure 6b, requires for each *content provider* to get a valid certificate by sending a PKCS#10 certificate request and receive an X.509 certificate from its CA. As the validity of such a certificate will be $T = 365$ days, it can be neglected in the studied timeframe. The data volume transferred during 1 day will be dependent on the amount U of users N interested in a status verification. It does not depend in any way on how many status requests Q they will perform on this timeframe, as the CRL refresh rate R is assumed to be 1. Equation 3 will serve to determine the transferred data volume v_{CRL} during this 1 day:

$$v_{CRL} = (P \cdot N \cdot 39 + 400) \cdot U \cdot N \quad (3)$$

The scenario of the *OCSP* validation mechanism is depicted in Figure 7a. As it can be observed, each content provider, as well as the *OCSP Server*, need first to get a certificate by sending a PKCS#10 formatted certificate request and receiving then an X.509 certificate. As both certificates have a validity period of $T = 365$ days, their transmission can be neglected in the focused timeframe of 1 day. Interesting parameters to be studied are the amount of data needed for each requesting user Q assuming that U out of N users perform such a status request. Consequently, Equation 4 describes the data volume v_{OCSP} generated during 1 day.

$$v_{OCSP} = (l_{OCSPReq} + l_{OCSPResp}) \cdot Q \cdot U \cdot N \quad (4)$$

As it can be seen in Figure 7b the *XKMS* model does not differ at all from the *OCSP* model. The only differences are the data sizes of *XKMS Validate Requests* and their corresponding *XKMS Validate Responses*. Therefore, all assumptions done in the previous scheme apply also for this one. Equation 5 gives a mathematical overview of the data volume v_{XKMS} generated using the *XKMS* model during 1 day.

$$v_{XKMS} = (l_{XKMSReq} + l_{XKMSResp}) \cdot Q \cdot U \cdot N \quad (5)$$

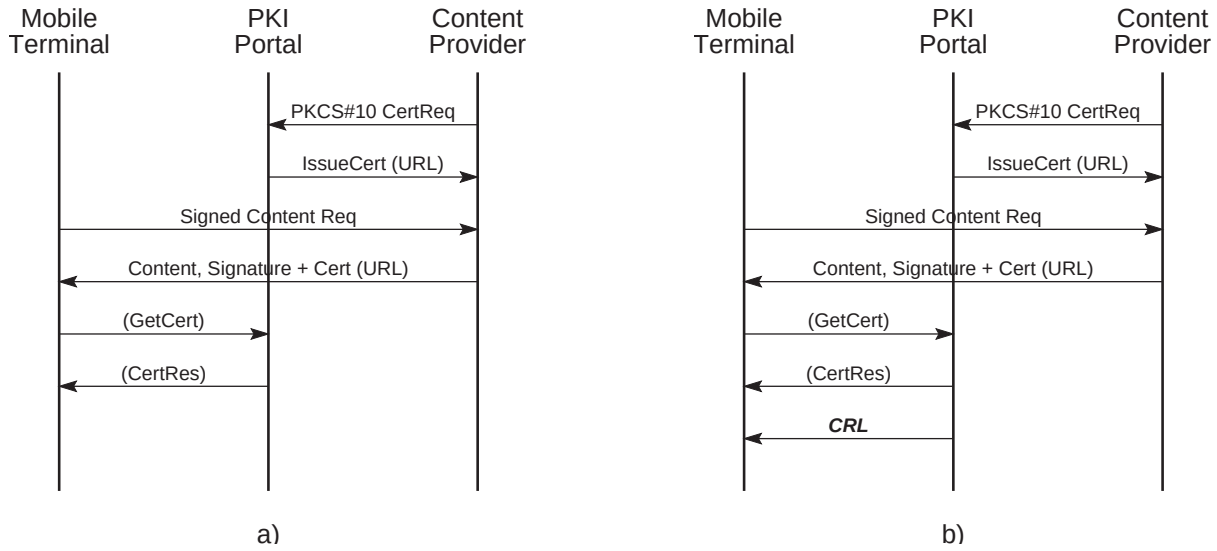


Figure 6: Message Diagram of the a) *short-lived certificates* and b) *CRL* scenarios

The evaluation will be focused on different combinations of F, U and Q for 2 different populations N.

Figures 8, 9, 10 show the data volume v_i transferred during 1 day for $F = 10; 25; 50$ % and a population of $N = 10,000$ respectively. Figures 11, 12, 13 show the data volume v_i transferred during 1 day for $F = 10; 25; 50$ % for a population of $N = 100,000$ respectively. The way they are read, is the smallest bar being the most cost-effective alternative for the specific parameters.

In a population of $N = 10,000$ users, the *CRL* model tends to perform worst in most of the cases. The *short-lived certificates* scenario performs slightly better than the *CRL* model, except for $F = 50\%$ and $U = 1\%$. In those cases it is the less cost-effective model, as the volume v_i is lower than in the other three analyzed models.

The *XKMS* model shows only slightly better results than the *shot-lived certificates* in less than half of the studied use-cases. For $Q = 20$ requests per user each day, it even shows the worst behavior, which means that the usage of *XKMS* creates the biggest data volume.

OCSP on the contrary always shows the best performance in data volume v_{OCSP}, beating all other three approaches. This advantage is specially noticeable for a small number of requestors U and in case these perform only a few status queries Q.

For a population of $N = 100,000$, the *CRL* is clearly the less cost-efficient alternative. The *short-lived certificates* model performs slightly better, but as previously mentioned, it is worse than the *XKMS* model in case only a few queries being issued a day. For values of at least $Q = 10$ queries per user each day, *XKMS* behaves worse than the *short-lived certificates* use case.

As it is shown in Figures 11, 12 and 13, the parameter F specifying the amount of content providers does not take any effect over the *short-lived certificates* scheme, even though it is important to take it into account.

OCSP shows again the best cost-effectiveness, due to its small message sizes. Similar to the previous use-case of a lower population, this advantage decreases for increasing values of Q and U, but the advantage is still considerable.

An additional and interesting parameter for evaluating the scalability of the certificate validation mechanisms is the data volume v_i generated by each requesting party. It is the intention to compare all of the schemes, in order to evaluate the number of *short-lived certificates* which can be requested on one side and the number of status requests which can be performed using either the *OCSP* or the *XKMS* message on the other side, compared to the download of a daily CRL. This is performed for a population of $N = 10,000$ and $N = 100,000$.

Therefore equations 2 to 5 are modified as follows:

$$v_{sh.-liv.} = l_{Cert} \cdot Q \tag{6}$$

$$v_{CRL} = (P \cdot N \cdot 39 + 400) \tag{7}$$

$$v_{OCSP} = (l_{OCSPReq} + l_{OCSPResp}) \cdot Q \tag{8}$$

$$v_{XKMS} = (l_{XKMSReq} + l_{XKMSResp}) \cdot Q \tag{9}$$

Figure 14a shows that for a population of $N = 10,000$ users only 11 requests can be performed using

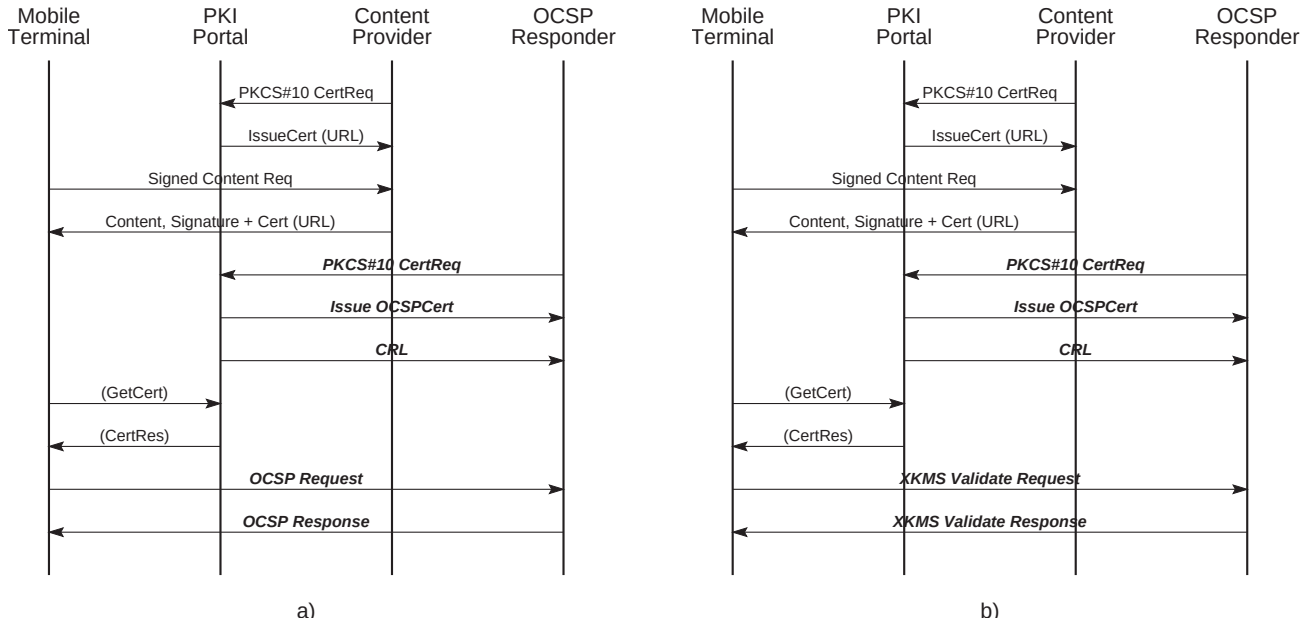


Figure 7: Message Diagram of the a) *OCSP* and b) *XKMS* scenarios

XKMS, in order to not produce more data traffic as the daily download of a *CRL*. In order to achieve a similar traffic, up to 38 *short-lived certificates* can be downloaded or even 51 *OCSP* roundtrips can be performed each day.

For a population of $N = 100,000$, Figure 14b shows that 112 *XKMS* status validations can be done with *XKMS*, 383 with the *short-lived certificates* scheme and 512 using *OCSP*.

This proves that for a large scale infrastructure the *CRL* scheme is not suitable at all. Additionally, the *XKMS* scheme has proven to perform much worse than the *short-lived certificate* model and *OCSP*. It can clearly be stated that *OCSP* scales best from the data volume's (v_i) point of view.

Besides the data volume v_i, another interesting topic to be studied is the request rate r_i for each of the four certificate validation mechanisms. Cooper [4] studied this topic for CRL's and some variants over them.

The first assumption stated that a user will only perform one CRL request within the validity of it, as the `nextUpdate` field gives a concrete date for the next CRL issuance. This assumption is extended in this presented model to the *short-lived certificates* and to *OCSP* and *XKMS*, as the studied timeframe is 1 day and it can be assumed that in case a user receives several *signed* messages from one particular entity, it will not query for additional status requests once it has been confirmed to be valid.

Further on, it was assumed [4] that the population being served by one CA is reasonably large for the requests being considered to be queried in an independent fashion. Therefore an exponential inter-arrival probability density will be used for modeling the timings of status queries. The probability that a user will query for such a validation within the interval $[t \dots t+dt]$, for $dt \to 0$ is

$$P(t) = Q \cdot exp^{-Q \cdot t} dt \tag{10}$$

with Q being the aforementioned queries per user each day. The total number of queries can then be determined to be

$$T(t) = U \cdot N \cdot P(t) \tag{11}$$

The request rate r_i at time t for all users N will consequently be

$$r_i(t) = \frac{T(t)}{dt} = U \cdot N \cdot Q \cdot exp^{-Q \cdot t} \tag{12}$$

The studied timeframe will start at $t_0 = 0$ with the issuance of a new CRL. Therefore the maximum request rate will be experienced at this point of time and can easily be calculated by

$$r_{i_{max}} = r(0) = U \cdot N \cdot Q \tag{13}$$

Figures 15a and 15b show the request rates for the CRL model, as specified by Cooper [4] with a rate of requesting users $u = 10\%$ and a population of $N = 10,000$ and $N = 100,000$ respectively.

For the *short-lived certificates* model, as well as for *OCSP* and *XKMS*, the exponentially distributed inter-arrival time will also take place, as the arrival rates for

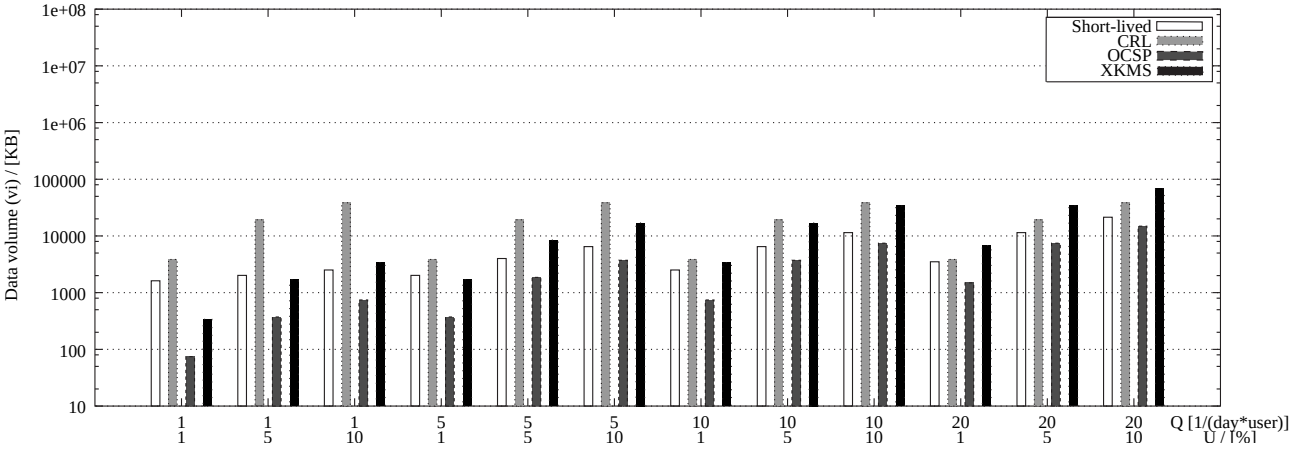


Figure 8: Data volume v_i for $F = 10\%$ and $N = 10,000$

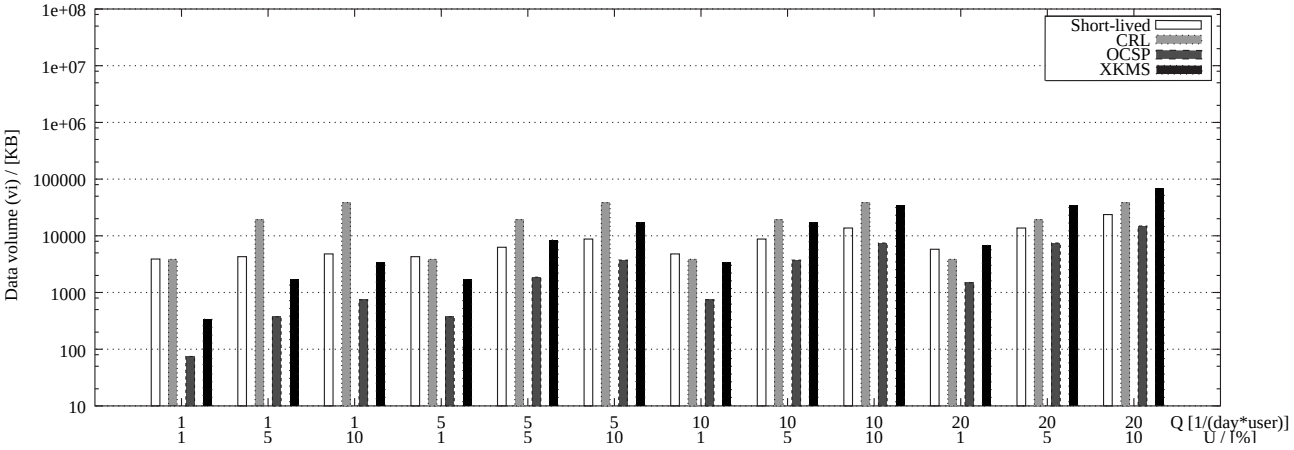


Figure 9: Data volume v_i for $F = 25\%$ and $N = 10,000$

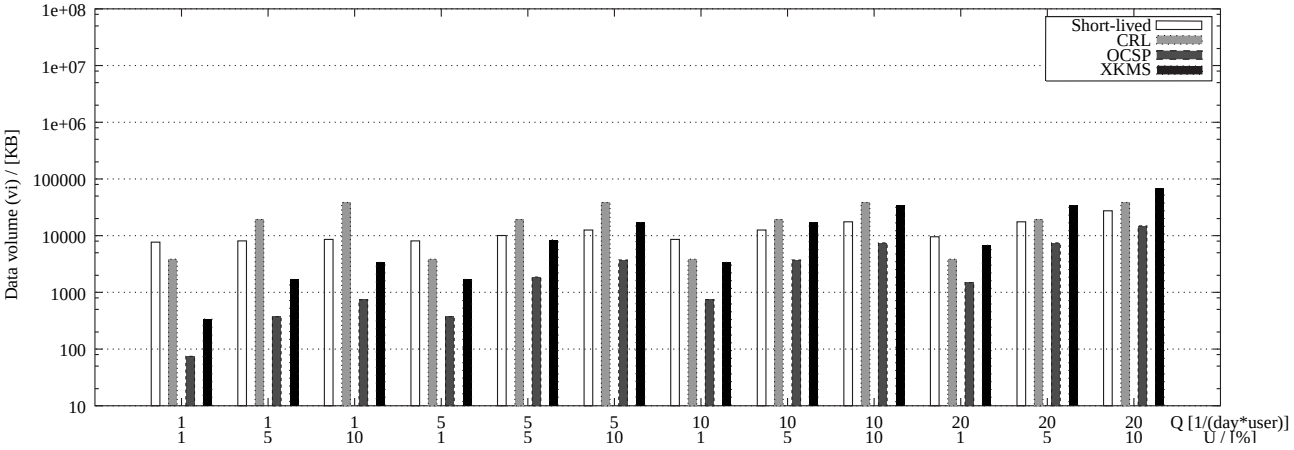


Figure 10: Data volume v_i for $F = 50\%$ and $N = 10,000$

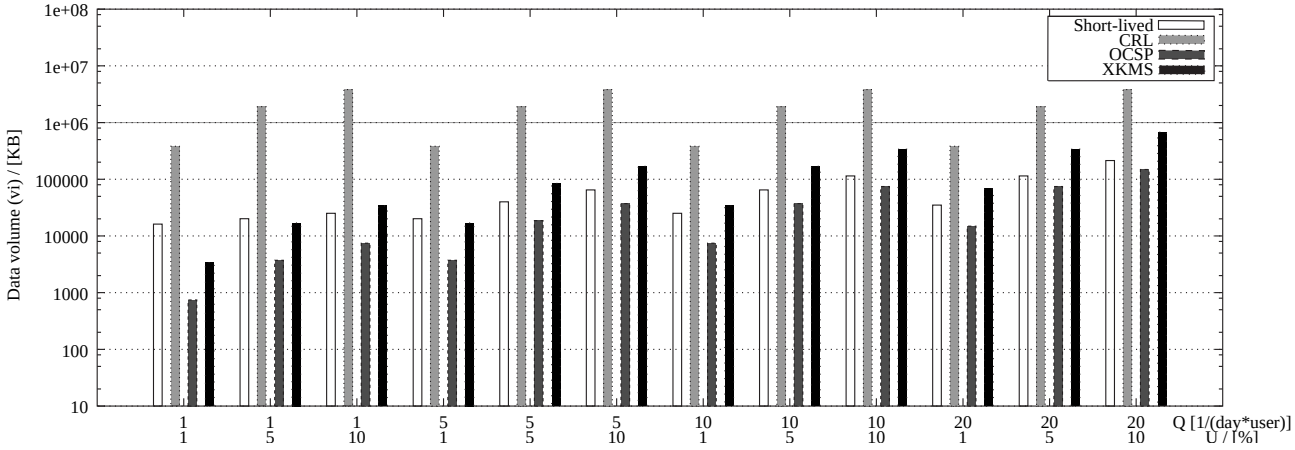


Figure 11: Data volume v_i for $F = 10\%$ and $N = 100,000$

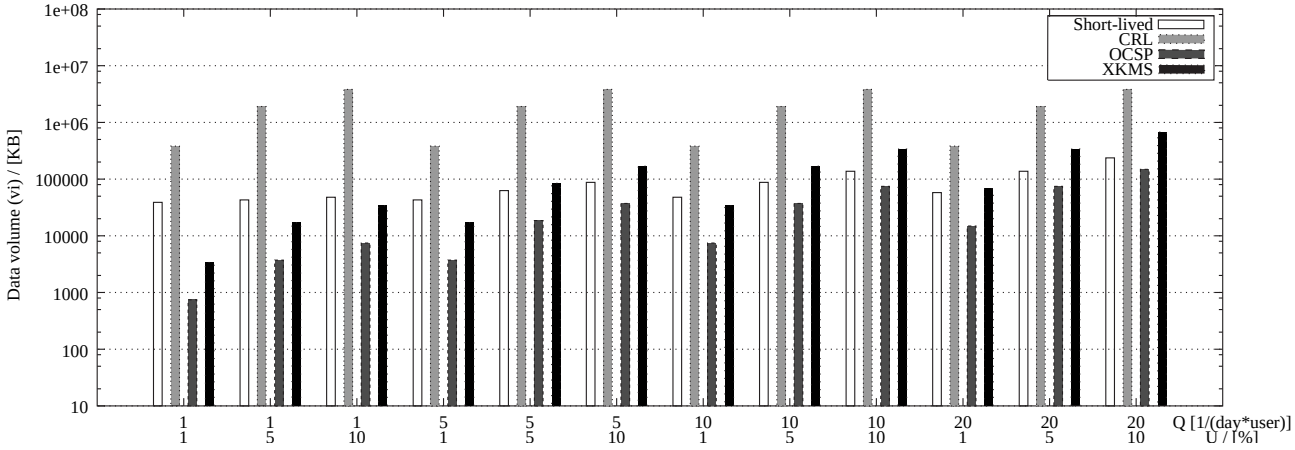


Figure 12: Data volume v_i for $F = 25\%$ and $N = 100,000$

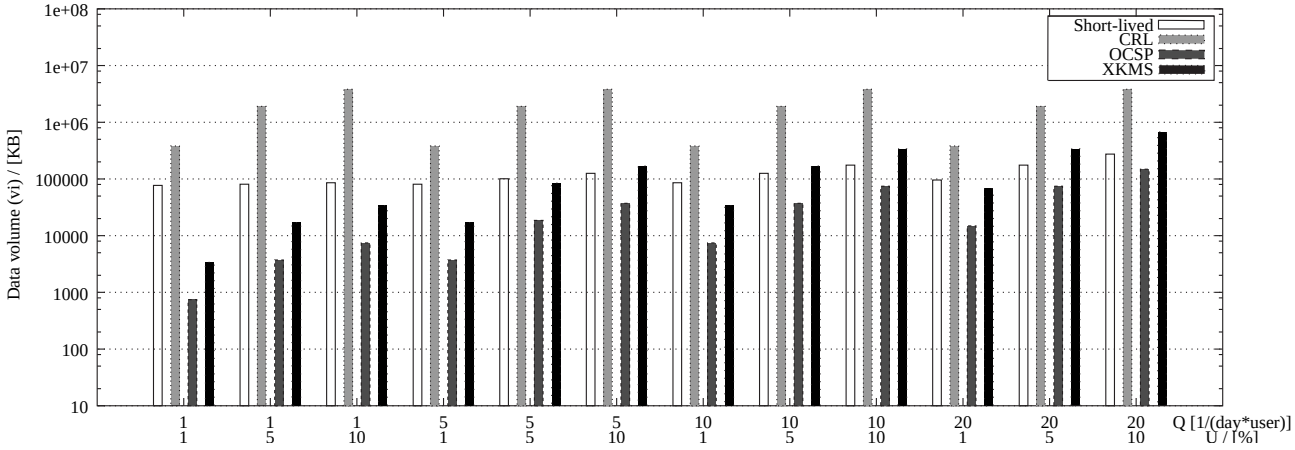


Figure 13: Data volume v_i for $F = 50\%$ and $N = 100,000$

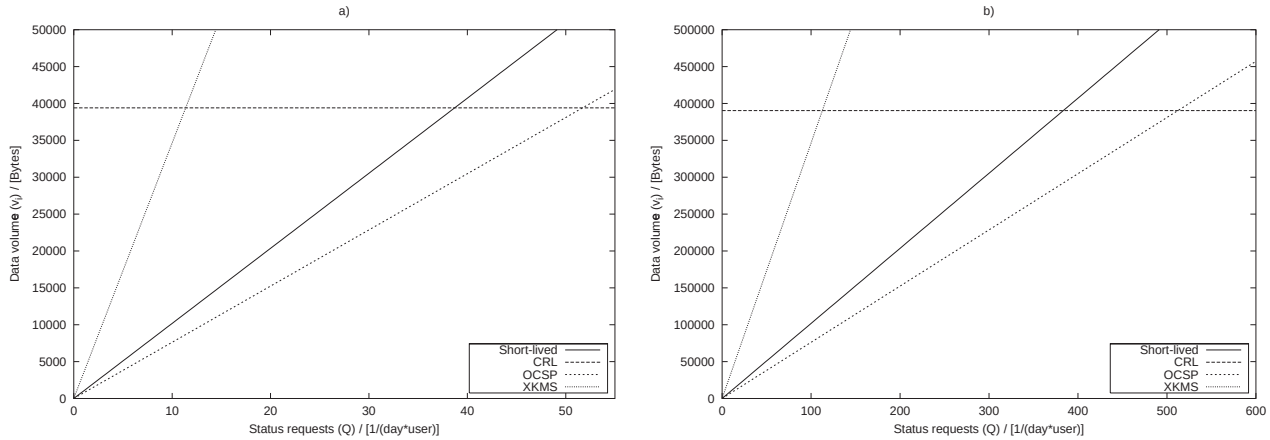


Figure 14: Data volume v_i for each user upon Q requests for a) $N = 10,000$ and b) $N = 100,000$ users

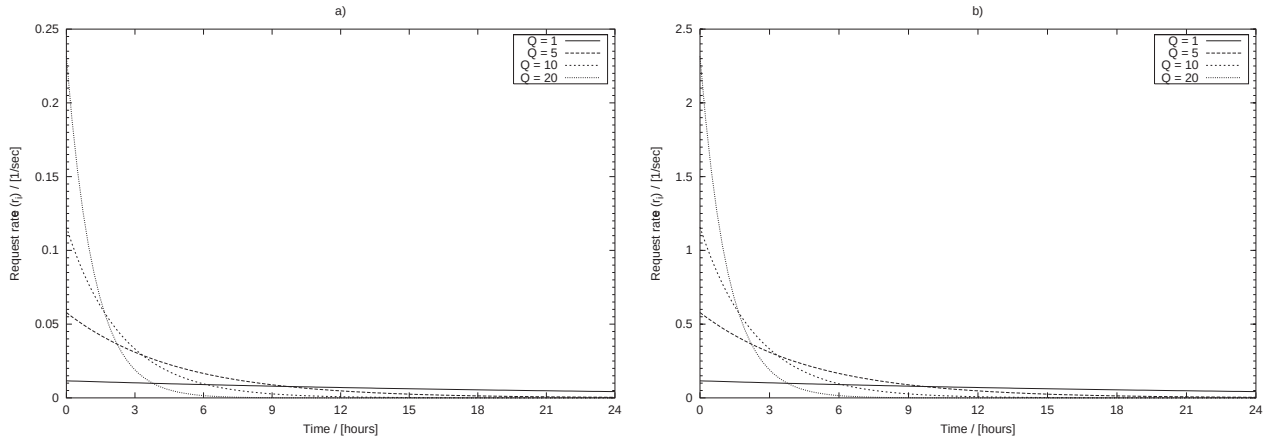


Figure 15: Request rates r_{CRL} for CRL's with Q requests per user per day, $U = 10\%$ and a) $N = 10,000$ b) $N = 100,000$

each user are statistically independent. Though, the difference is, that whenever *signed content* is received a new online status request will be issued, unless a previously obtained status response over the same identity is still valid[5]. This would be in place if caching of responses is allowed and performed at the client's side.

The maximum request rate $r_{i_{max}}$ will occur if all users query for a status response at the same time. Equation 14 describes this in the following way:

$$r_{i_{max}} = U \cdot N \cdot Q \tag{14}$$

This leads to the same maximum request rate estimated for the *CRL* model, and thus proves that all four schemes will notice an identical maximum request rate $r_{i_{max}}$.

[5]indicated by an optional extension of `nextUpdate` in OCSP and `ValidityInterval/NotAfter` in XKMS

Nevertheless, the average request rate $r_{i_{mean}}$ for online validation models as well as for the *short-lived certificates* model, is estimated to be much lower than in the *CRL* case. It is also considered to be constant, varying only in *busy hours*.

4 Summary

This section gives a brief summary of the evaluation results. Table 5 illustrates the rating which is given to each of the analyzed models. The rating has been decided to be divided into four steps, ranging from [−−] meaning *not satisfactory* up to [++] meaning *very satisfactory*.

The *security* evaluation has concluded that the *short-lived certificates* introduce a high risk of invalidating the security requirements due to an unproper *validity period*. It is strongly recommended to use secure clock synchronization, either by *Timestamping Authorities*

Table 5: Comparison of Certificate Validation mechanisms

Encoding scheme	Security	Interop.	On-line	Complexity	Performance	
					Size	Scal.
Short-lived Certs	−	++	+	−	+	+
CRL's	+	++	++	++	−−	−−
OCSP	++	++	−	+	++	++
XKMS	−−	+	−	+	−	−

or secure *network time protocols*.

The usage of *CRL's* present a risk of showing the requestor an outdated revocation information. Apart from that, this model has been observed to present the least possibilities of compromise, as the signature over the list is performed off-line and simply uploaded to a public repository.

Both *OCSP* and *XKMS* may suffer from *Denial-of-Service* attacks, due to flooding of queries. Several solutions for avoiding this have been presented, as well as solutions for avoiding other types of attacks. However, both models present the best *timeliness* if they are directly linked to the CA.

It has to be clearly pointed out that the *security* of both online schemes needs to be strongly revised, due to the key role they have acquired within a full PKI.

Regarding the *interoperability* of all four schemes, it has to be stated that apart from *XKMS* they are all standardized and already implemented by different vendors. However, *XKMS* is in the process of standardization and will soon be fully interoperable by different vendors.

Both *short-lived certificates* and *CRL's* can be considered as *off-line*, in that the status verification is performed without the need of any network connection, besides the acquiring of a new certificate or an updated *CRL*. The *OCSP* and *XKMS* variants on the contrary require a continuous network availability for querying the corresponding service for status validation.

The *complexity evaluation* has focused on the implementation effort and ease-of-use. It has been observed that besides the *XKMS* model all others require ASN.1 processing facilities, while *XKMS* requires XML path processing facilities.

Apart from that, the management required by an end-entity to handle the *short-lived certificates* scenario is much higher than in the other three cases. The *CRL* model requires the download of an updated CRL, according to the information listed in the `nextUpdate` field. Once it has this data structure, all management involved is done at its own place. Both online schemes offer a user to query a particular service upon certificate status verification. However, *XKMS* offers the user a set of possibilities to be performed, which should be defined by the service provider of the user.

The management of the service provider is not yet defined by any of both online status protocols. The provision of *fresh* revocation information is left to the provider's decision.

The most exhaustive evaluation has dealt with the *performance* of each of the four schemes. This has been divided into two subparts, the *size evaluation* and the *scalability evaluation*.

The *size evaluation* showed that even though several informations are of free choice to be included into the data structured, it has been a priority to include the same information into them whenever possible. The *CRL* has shown to be linearly dependent on the number of revoked certificates included in its list. The comparison between *OCSP* and *XKMS* has shown that the data sizes of both *OCSP Requests* and *OCSP Responses* are much smaller than their *XKMS* counterparts. The relation of difference is about 1 : 4 and mainly due to the *XML Digital Signature*.

The *scalability evaluation* clearly shows that *OCSP* is the most cost-effective model. The *short-lived certificates* variant and *XKMS* perform similarly, depending on the studied scenario. The *CRL* has shown to generate the highest data volume within the studied timeframe. Focusing on the data generated by one user on the studied timeframe, it can be observed that *XKMS* performs worse than the *short-lived certificates* model.

References

[1] C. Adams, P. Cain, D. Pinkas, R. Zuccherato. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP) (RFC 3161)*. IETF (http://www.ietf.org), 2001.

[2] C. Adams, S. Farrell. *Internet Public Key Infrastructure, Certificate Management Protocols (CMP) (RFC2510)*. IETF (http://www.ietf.org), 1999.

[3] S. Berkovits, S. Chokhani, J. Furlong, J. Geiter,

J. Guild. *Public Key Infrastructure Study: Final Report*. The MITRE Corporation, 1994.

[4] D. Cooper. *A Model of Certificate Revocation*. In *Proceedings of the Fifteenth Annual Computer Security Applications Conference*, pp. 256–264, 1999.

[5] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing (RFC3075)*. IETF (http://www.ietf.org), 2001.

[6] D. FIPS Pub 180-1. *Secure Hash Standard (SHA-1)*. National Institute of Standards and Technology, 1995.

[7] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC2459)*. IETF (http://www.ietf.org), 1999.

[8] ITU-T. *Recommendation X.509: Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), 1997.

[9] ITU-T. *Recommendation X.690: Specification of Abstract Syntax One (ASN.1)*. International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), 1988.

[10] RSA Laboratories. *PKCS#1: RSA Cryptography Standard*. RSA Data Security Inc., 2001.

[11] A. Menezes, P. van Oorschot, S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Cleveland, Ohio, 1996.

[12] D. Mills. *Network Time Protocol (NTP) version 3: Specification, Implementation and Analysis (RFC1305)*. IETF (http://www.ietf.org), 1992.

[13] D. Mills. *Public-Key Cryptography for the Network Time Protocol (NTP) version 1 (DRAFT Version)*. IETF (http://www.ietf.org), 2001.

[14] D. Mills. *Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6 and OSI (RFC 2030)*. IETF (http://www.ietf.org), 1996.

[15] M. Myers, R. Ankney, C. Adams, S. Farrell, C. Covey. *Online Certificate Status Protocol, version 2 (DRAFT Version)*. IETF (http://www.ietf.org), 2001.

[16] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams. *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol - OCSP (RFC2560)*. IETF (http://www.ietf.org), 1999.

[17] M. Myers, X. Liu, J. Schaad, J. Weinstein. *Certificate Management Messages over CMS (CMC) (RFC2797)*. IETF (http://www.ietf.org), 2000.

[18] A. Årnes. *Public Key Certificate Revocation Schemes*. Masters thesis, Queen's University, Kingston, Ontario, Canada, February 2000.

[19] B. Schneier. *Applied Cryptography; Protocols, Algorithms and Source Code in C*. John Wiley and Sons, Inc., New York, NJ, 1996.

[20] VeriSign, Microsoft, webMethods. *XML Key Management Specification (DRAFT Version)*. VeriSign, April 2001.

[21] VeriSign, Microsoft, webMethods. *XML Trust Assertion Service Specification (DRAFT Version)*. VeriSign, 2001.

[22] WAPForum. *Wireless Application Protocol: Certificate and CRL Profiles Specification*. http://www.wapforum.org, 2001.

[23] WAPForum. *Wireless Application Protocol: Public Key Infrastructure Definition (W-PKI)*. http://www.wapforum.org, 2001.

[24] WAPForum. *Wireless Application Protocol: Wireless Identity Module Specification (WIM), Part: Security*. http://www.wapforum.org, 2001.