VALIDATION OF LONG-TERM SIGNATURES

About Revocation Checking of Certificates in the Context of long-term Signatures

Karl Scheibelhofer
Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a
A-8010 Graz
Email: Karl.Scheibelhofer@iaik.at

Abstract: The current practice of digital signature creation is simple. However, signature verification is much harder. This especially holds for long-term signatures, signatures that should remain verifiable over years. After the signing certificate expired, it is hard to find out if the certificate was valid at the time the signature was created. Current revocation checking mechanisms, like CRLs and OCSP, may not provide the status of certificates which are no longer valid. This is one reason why many of the current signature verification systems cannot verify signatures after the signing certificate expired. There are several approaches for coping with these problems: attach all data that is required for validation to the signature right after signature creation, let the verification software collect and archive all validation data that it needs, or use advanced services for certificate status checking. Currently, there are hardly any advanced services available. However, this paper shows that it is not hard to design such services.

Key words: digital signatures, long-term signatures, revocation checking, certificate status checking, advanced electronic signatures, CRL, OCSP, DPV

1. INTRODUCTION

Not only since the European Union published the Directive on Electronic Signatures [1] digital signatures have been a big issue. Many European countries have enacted laws to implement this directive [2]. There are already CAs (Certification Authorities) which provide solutions that are

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35612-9_23

B. Jerman-Blaži et al. (eds.), *Advanced Communications and Multimedia Security*

compliant with the laws. Such products often use a smart card as signature creation device, a smart card reader, software for signing and verification, and a qualified certificate.

Everyone who works with signed email knows how easy it is to sign a document. Insert the smart card into the reader, enter the PIN (Personal Identification Number) and then the smart card calculates the signature value over the data to be signed. For the process of signing, most systems do not need to access any online service. If the user created the signature using products compliant with law, the resulting signature will be a legally valid electronic signature. However, this is just half of the story. Having a legally valid signature is nice, but we also need to be able to verify the signature.

Verifying the signature right after it was created can also be harder than one might think. This is the case with systems that use CRLs (Certificate Revocation Lists) [5]. Because CAs are not required to issue a new CRL each time a certificate gets revoked, the client will not notice a certificate's revocation before the CA issues the next CRL. From the most recent CRL the client can only get reliable status information for times before the CRL was issued. This problem also occurs with OCSP (Online Certificate Status Protocol) [6] responders which are implemented based on CRLs. Unfortunately this is very often the case.

When the CA issued the next CRL after the signature creation, CRLs and OCSP responders will provide the necessary information to verify the signer's certificate. To verify a signature, it is required to verify the certificate with respect to the time when the signature was created. Current mechanisms may not provide this information directly. This makes the verification of signatures hard, especially if one needs to verify them after the certificate expired. In this case, current revocation checking mechanisms may not provide any information about the certificate. Because of this, current software products that use digital signatures are unable to verify signatures after some time. For instance, most of the prominent email clients have this problem, but most of all other software products that verify signatures have the same problem. However, there are ways to cope with this problem. This paper will show some of them. One approach tries to solve this problem at creation time of the signature. It attaches all validation data that the verification process needs to verify the signature. Another approach requires new revocation checking mechanisms. Such new mechanisms can ease signature validation dramatically. This paper proposes a simple but effective revocation checking service.

2. VERIFICATION OF LONG-TERM SIGNATURES

It is easy to sign a document using current technology. All you need is signing software, a signature key, and a certificate for this key. You can get all these things easily. When a user signs a document he uses his signing key to create a signature value. The signed version of the document will consist of several parts: the original document, a signature value and the signer's certificate (maybe not the complete certificate, but at least an unambiguous identifier of it). Moreover, the signed document may contain other attributes like the signing time or an identifier of the policy under which the user signed the document. Figure 1 shows the contents of such a signature. The signature value covers all those elements which are highlighted grey. But is all this information enough to be able to verify the signature later on?

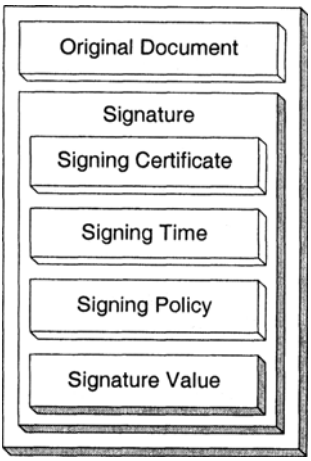

Figure 1. The contents of a simple signed document.

We can find this out by having a closer look at software that tries to verify the signature. Let us assume that a relying person tries to verify the signed document one year later. In addition, we assume that the certificate that the signer used is expired meanwhile. The software can use the certificate, which is contained in the signed document, to cryptographically verify the signature value, even though, this is not enough. To verify the signature, the software must also find out, if the signer's certificate was valid at the time the signature was created. If the certificate is valid now, at the time of verification, is irrelevant. The signing time is normally part of the signed document. Maybe, the user can specify the signing time when verifying the signature, if the signed document does not contain the signing

time. Having the signing time, it must find out, if the certificate was valid at this time. The following sections show different methods for that.

2.1 Using CRLs for Revocation Checking

The software may try to use CRLs to get the desired information. Almost all CAs provide CRLs for their issued certificates. In principle, CRLs are lists of certificates which have been revoked. The list does not contain the complete certificates but the serial numbers of the certificates. The issuer of the CRL is also given in the CRL. The combination of issuer and serial number identifies a certificate uniquely in X.509 based systems. Thus, the software checks if the serial number of the concerned certificate is in the CRL. If the certificate is within its validity period and the serial number is not in the CRL, the certificate has not been revoked and can be considered valid. A CA issues CRLs regularly. Each CRL contains the date when it was issued and the date when the next CRL will be issued latest. With this information, the client can see, if the CRL he has is still valid. The CA may issue a new CRL even earlier than the date given in the last CRL. This may be applicable, if some certificate has been revoked meanwhile and the CA wants to provide current revocation information. Thus, a client may decide to try to get always the latest CRL. Figure 2 shows the most important elements of a CRL. The extensions allow conveying additional information, for example, an identifier of the signing key of the authority or the number of the CRL, which is a monotonically increasing sequence number.

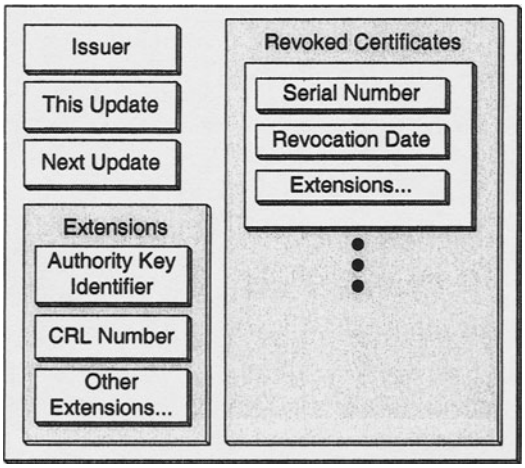

Figure 2. The most important contents of a CRL.

To find out where to get the CRL, the software can investigate the certificate itself. Inside a certificate, there is an entry that specifies where the software can find the CRL for this certificate - this is called the CRL distribution point. In practice, this is often an URL (Uniform Resource Locator) to a file on an HTTP (Hypertext Transfer Protocol) sever which contains the most recent CRL. There we can already see a problem. Under this location, the client only gets the current CRL. The current CRL normally contains only information about certificates which are still within their validity period. However, we need a CRL, which contains information about the status of the certificate at the signing time – for instance, one year ago. There is no standard mechanism to get old CRLs. So what can the software do in this case? Instruct the user to contact the CA for getting an old CRL that was valid at the signing time, download it manually and configure the software to use this? – Most users will say something like that: "What the hell is a CRL? And why is my software not able to do that for me automatically?" And they are right. This is unsuitable for practical purposes. The only chance for the software to verify the certificate is to have all CRLs available locally - all CRLs of all CAs that the user may ever try to verify a certificate for. This is practically infeasible.

Do delta CRLs relieve the situation? Delta CRLs are a special form of CRLs. They aim at keeping the amount of transferred data small. A delta CRL just contains the changes with respect to a complete CRL. To be useful for the client, the client needs this base CRL and the delta CRL. The base CRL may contain hundreds of revoked certificates. For instance, after this base CRL has been issued, but before the next complete CRL gets issued, the CA revokes another two certificates. To avoid issuing a new complete CRL, the CA can issue a delta CRL which refers to the base CRL and contains the two additional certificates. Consequently, a client, which already has the base CRL, can download the latest delta CRL. This delta CRL will be relatively small compared to the complete CRL. Having the base and the delta CRL, the client has all the latest revocation information. The CA may issue delta CRLs even more frequently than CRLs. However, a delta CRL must always refer to a complete CRL; it can never refer to another delta CRL as its base. Similarly to complete CRLs, the certificate itself contains the information where the client can find a delta CRL. Now we can see, that delta CRLs do not provide any information that we cannot get from the complete CRL. Thus, we have not made a real progress with respect to our verification problem.

There is another type of CRL called indirect CRL. Indirect in this context means that the issuer of the CRL is not the same as the CA that issued the certificates that are in scope of this CRL. A CA can use indirect CRLs to outsource CRL issuing. In this case, the CA will issue a certificate for the

other organization. This organization uses this certificate to issue and sign CRLs for the CA. Another situation where indirect CRLs are necessary is when a CA ceases operation. Then a different organization may continue to provide CRLs for this CA. In this case, the other organization will also use a different key for signing the CRL – it will be an indirect CRL. All in all, there is not much difference between normal CRLs and indirect CRLs, at least not regarding the information they provide for the client.

2.2 Using OCSP for Revocation Checking

Let us assume that we have a more sophisticated software. A software that is able to use OCSP services. OCSP is an online service that normally uses HTTP as transport protocol. If the certificate contains the address of the OCSP responder, the software can use it. The client sends a request to the OCSP server and gets a response. With such a request, the client can ask for the status of a certain certificate. The answer of the server tells the client one of three possible states: the certificate has been revoked with the revocation time, the status is good or the status is unknown. If the certificate has been revoked, the meaning of the answer is clear. The meaning of unknown status is also clear, though not very helpful. What does "good" mean? It does not mean that the certificate is valid. OCSP defines "good" as that the certificate has not been revoked. However, this does not imply that the certificate is valid, is within its validity period or has ever been issued.

Now, all is fine, the software will be able to verify the certificate, won't it? Unfortunately no. OCSP, as it is now, may not provide any additional information to CRLs. Depending on the implementation, an OCSP responder may not provide any more information than the most recent CRL. But doesn't it tell me the status of certificate at the signing time? No, not directly. OCSP does only provide the current certificate status. Seen realistically, OCSP is no big advance over CRLs. However, it is possible to implement an OCSP responder in a manner that it provides more information than CRLs do. It can provide the current status of a certificate; this is the status of the certificate when the OCSP server creates the response. Moreover, it can provide the certificate's status as long as it is required, long after the certificate expired. Many current implementations do not provide any more information than the recent CRLs, because the OCSP standard does not require it. Using such an implementation, OCSP only saves bandwidth in some cases. As we can see, software that is based on current revocation checking mechanisms may be unable to check the certificate's status in the depicted use-case – validation of long-term signatures without having the necessary validation data in the signed document.

2.3 Adding Validation Data to the Signature

The only solution that can be implemented using the current standards is adding all necessary data to the signature at signing time (or short after that). This means, attaching the complete certificate chain and all corresponding CRLs to the signed document. Remind that we need the first CRLs that were issued right after signature creation. The certificate chain may require all certificates up to a self-signed CA certificate, but in certain scenarios a shorter certificate chain may be sufficient. In general, the signer cannot anticipate in advance which CA certificate in the chain a verifier trusts. Additionally, the signing software may add a trusted timestamp to the signed document. Such a timestamp can provide evidence that the signature was created before the indicated time. This will support the verification software in determining the signature time. Instead of or in addition to CRLs, the signing software could also attach current OCSP responses. Figure 3 shows an example for such a signed document with attached validation data.

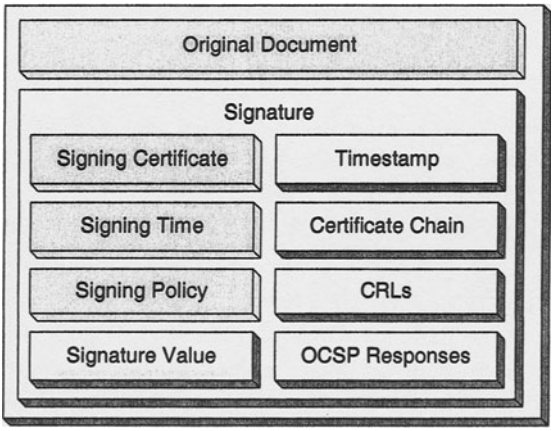

Figure 3. A signed document with attached validation data.

Adding the CRLs or OCSP responses would result in a considerable amount of extra data. Normally, the verifier needs to validate a complete certificate chain, which means that he needs the CRLs of all involved certificates. Moreover, if there is any indirect CRL involved, the verifier must construct and validate a separate certificate chain for the signer of this CRL. Attaching several CRLs and/or OCSP responses to each signature is not very convenient. In some use-cases, the required amount of additional memory for this validation data may exceed the original document's size. There are even standards from the ETSI that specify how to encode such types of electronic signatures, signatures that include all the validation data.

The ETSI TS 101 733 [3] document defines such structures for encoding signatures based on CMS (Cryptographic Message Syntax) [8] and the ETSI TS 101 903 [4] document defines similar structures for signatures based on the XML Signature [9] standard. Both standards are rarely used in practice yet.

2.4 Advanced Certificate Status Checking Services

What would solve such problems is a service that can answer the simple question "Was this certificate valid at this specific time?" Currently, there is no such service and it is unlikely that we will see such services in the near future. The PKIX (Public-Key Infrastructure X.509) working group is currently working on successor of OCSP versions one. This successor may include a service called Delegated Path Validation (DPV). This service could validate a complete certificate chain with respect to a certain time. The service will use the chain validation algorithm as specified by the PKIX working group. Such a service will come much closer to what we need to verify a signature, or more precisely to verify the certificate that was used for signing. The service must have access to the complete history of all certificates in its scope; otherwise it would not be able to answer our request.

This service would have to tell if the certificate chain, which ends with the signer's certificate, was valid one year ago, at the time the signature was created. If the service has only access to the current CRL, it would not be able to answer this request either. In this case, the service would respond that it does not know if the certificate chain was valid at the concerned time. We see, it is not sufficient to define such new protocols. The providers of such services must have databases that contain the required history of the certificate states, or they must have access to a service that provides the same information. If these services are implemented solely based on mechanisms like CRLs, they will not provide any additional information. Moreover, the service providers need to maintain these databases for many years; at least as long as the signatures should be verifiable. If a signature should be verifiable for at least thirty years, which may apply to advanced electronic signatures, the service provider needs to maintain the certificate's status history at least thirty years after the certificate expired. Seen realistically, this is not an extraordinary requirement. The laws for electronic signatures often require archiving the revocation information for such long periods. In addition, services that provide delegated path construction and validation may be suitable for enterprises but not for CAs. A CA may not want to provide such service or it may want to outsource it. In this case, the service providers must have means to get current information about certificate states. Services like delegated path validation will not solve the

problem of getting the status of a certificate, they only move the problem to the service providers.

A problem with advanced services like DPV is trust. The client has to decide, if he trusts in the correct operation of that service. A user will not delegate such a relatively critical task to a service that he does not trust. Therefore, it is likely that each enterprise runs its own service, because trust relations are easier to manage in a closed environment. A public service will return a signed response. In consequence, the client has to verify the signature of the response. This, once again, requires verification of the certificate. The responsibility can never be transferred completely to a service; the final trust decision is always up to the client. In practice that means that the client needs a trust anchor. A trust anchor is most likely a certificate which the client trusts, normally a CA certificate. Careful users check the hash of trusted certificates, before they configure their client to trust this certificate. To check the hash, they use a trusted channel, which may be via telephone or even face to face.

2.5 Signed Document Store

Another solution for this problem would be a dedicated server which accepts signed documents and takes care about the collection of all necessary validation data. This approach would have several advantages over the straight forward one, which attaches all this validation data to the signed document. The server would collect each CRL just once and would store it locally; it can use the same CRL for other signed documents as well. Contrary, for the all-in-the-document approach, each client that signs a document collects the validation data at least once. Moreover, a server could parse incoming CRLs and convert the status information into a more efficient form. This may be a database that keeps track of the history of all concerned certificates. Such a server could also easily keep track of resigning. Resigning of signed document is required, if algorithms or keys become relatively weaker over time, because technology and science advance and it becomes easier to break the keys or the algorithms. Therefore, the server could resign the signed document (more precisely the signature of the document) with a stronger algorithm or key before the signature can be forged. If a signature has been resigned early enough with a stronger algorithm or key, it does not matter, if the inner signature is forgeable afterwards. Of course, after resigning we must obtain a timestamp that provides evidence that the signature was resigned before it could have been forged. Confidentiality of documents would not raise a problem, because it would not be required to send the original document to the server. The server does not need it to maintain the signature. The original document

is only required for signature verification. Moreover, the original document could be sent encrypted.

2.6 Proposal for a Simple Certificate Status Checking Service

We saw that the current mechanisms for certificate status checking, CRLs and OCSP, are not optimal. However, a simple service could provide all information we need. First, we need to find out what information such a service should provide. The question that the client needs an answer for is "Was this certificate valid at this time?" Thus, the minimal information that the client must send to the service is: the certificate (or an unambiguous identifier of it) and the concerned time. The answer could be one of three: the certificate was valid at this time, the certificate was invalid at this time or the service does not know the answer. If the certificate was invalid or the service does not know the answer, the service may provide additional information about the reason. Such a service is relatively simple. Figure 4 shows the contents of possible requests and responses. The figure does not show it, but the response will contain a signature and the certificate of the responder. Based on the responder's certificate, the client can make its trust decisions. The trust model can be the same as for other certificates.

Upon setup of a new responder, we can get all status information about the past from CRLs. Of course, we would need all CRLs that a CA issued. In practice, the service would convert the information contained in the CRLs into a more convenient form; for instance, it would store the information in a database. As already mentioned, there is no automatic mechanism for getting old CRLs. Thus, the administrator needs to do this manually. This is not a very big drawback, because it is needed just once at setup time. After setup, the service gets the states directly from the database. The CA has to ensure that its database always contains the current states.

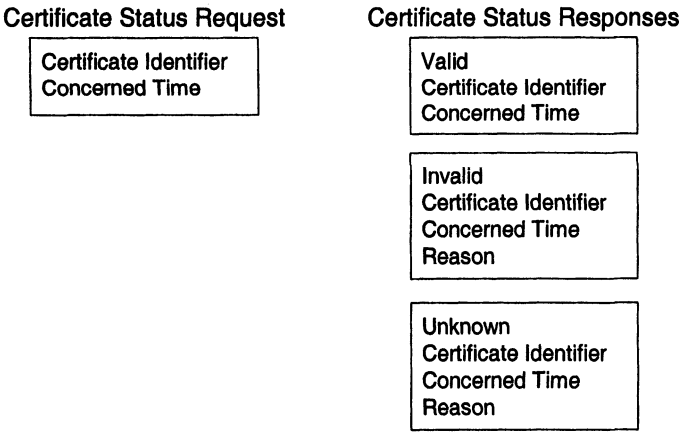

Figure 4. The contents of requests and responses of a simple certificate status service.

It is easy to extend this service to provide complete status histories of certificates. The only thing that we need to change in the request is the time field. We would provide a time interval instead of a single time value. The answer that this request would express would then change to: "What was the status history of the given certificate in the given time interval?" The response is a list of status information records. The first entry in the list is the status at the beginning of the time interval. There would be one additional entry in this list for each status change. Each of these entries would have a time value, which tells, when the status changed to the new state. In Figure 5 we can see the structure of such requests and responses. If we just want to know "Was this certificate valid at this time?", it is still easy to use. In such a case, the client sends a time interval of zero length; it starts at the concerned time and ends at the concerned time. The response of the server can only contain one entry in the list, because a certificate can only have one state at a single point in time. As in the simpler version, the responses will be signed and will contain the responder's certificate.

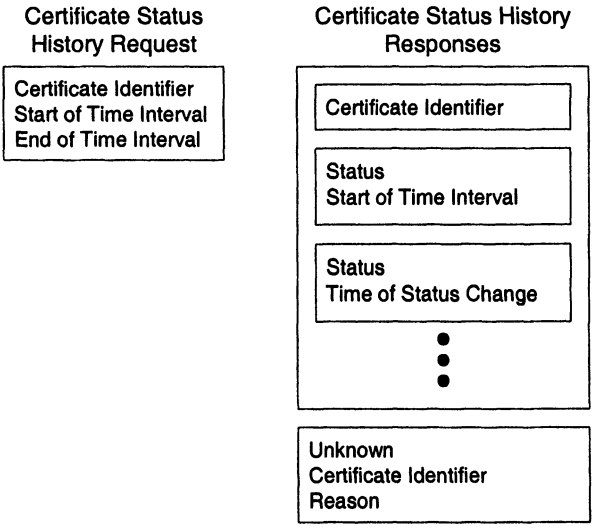

Figure 5. The contents of requests and responses of an advanced certificate status history service.

What advantages would this advanced version have? A client can get the complete history of a certificate with one request/response. For example, if a client has a certificate that is already expired, it can request the complete history. It stores this response and can use it later on to verify any signature that was created using this certificate. The client will never need to request any status information for this certificate again. If we operate a server that stores signed documents, it can easily get the status history of certificates. Most certificates will have a simple life-cycle. They will be issued and used until they expire. The server can store this information efficiently with respect to memory consumption and performance.

3. CONCLUSION

As this paper shows, it is much harder to verify signatures as to create them. Simple signed documents without the complete validation data cannot be automatically verified after some time, especially after the signing certificate expired. Using current revocation checking mechanisms, like CRLs and OCSP services, it may be impossible to get the status of an expired certificate. To verify a signature however, it is required to find out, if the used signing certificate was valid at the time the signature was created. Moreover, CRLs do not provide the current revocation status; even OCSP responders may not provide this information.

In principle, there are two ways to cope with this problem, if there are only OCSP or CRLs available for revocation checking. First, we can solve the problem at the time of signature creation. We can simply attach all data to the signature that is required to verify it. This includes the complete certificate chain and all corresponding CRLs or OCSP responses to verify the certificate with respect to the signing time. A big drawback of this solution is the relatively big amount of extra data that we must attach to a signature. Second, we can implement the verification software to collect all revocation information that it might need some time. This solution is not suitable for simple signature verification software; it is rather suitable for dedicated servers that store signed documents. For such servers it would be feasible to collect such big amounts of revocation information and archive them.

Nevertheless, advanced services for certificate status checking could make the verification of signatures much easier. It is not hard to design a service that provides the answer to the simple question: "Was this certificate valid at this time?" The real work for establishing such a service would be to setup a database that contains the status history for all certificates in scope. Moreover, it would be easy to design such a service in a manner that it can provide the complete history of a certificate's status.

For the future it would be desirable to have such advanced services which provide the information that the client needs to verify a certificate. While providing the required information, new services should refrain from supporting optional features which provide no real benefit. Nowadays, most of the work for revocation checking is delegated to the client, even if a server could do this work more efficiently. Today, the client has to download and archive CRLs and OCSP responses. It is really time for software developers and service providers to realize that there are serious deficiencies in current PKI (Public Key Infrastructure) technologies. Many of the current problems can be solved much better with solutions that are simpler than the current ones.

REFERENCES

[1] European Directive on Electronic Signature. The European Parliament and the Council, Brussels, December 1999, available online at <http://europa.eu.int/ISPO/ecommerce/legal/digital.html>

[2] The Austrian Signature Law. The National Council of Austria, Vienna, August 1999, available online at <http://www.a-sit.at/>

[3] Electronic Signature Formats. Electronic Telecommunications Standards Institute, TS 101 733 V.1.3.1, France, February 2002, available online at <http://www.etsi.org/SEC/el-sign.htm>

[4] XML Advanced Electronic Signatures (XAdES). Electronic Telecommunications Standards Institute, TS 101 903, France, February 2002, available online at <http://www.etsi.org/SEC/el-sign.htm>

[5] 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>

[6] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP, RFC 2560, June 1999, available online at <http://www.ietf.org/rfc/rfc2560.txt>

[7] Adams, Carlisle, Lloyd, Steve, "Understanding the Public-Key Infrastructure", New Riders Publishing, ISBN: 157870166X

[8] Housley, R.: Cryptographic Message Syntax. The IETF, RFC 2630, June 1999, available online at <http://www.ietf.org/rfc/rfc2630.txt>

[9] XML Signature. The W3C and the IETF, Recommendation, 12 February 2002, available online at <http://www.w3.org/Signature/>