Authenticating Downloaded Content

Alex Deacon, Ram Moskovitz
VeriSign Inc.

DRAFT Version 0.9(Nov 13, 2002)

Abstract

The ability to download content via the Internet has become second nature to many users over the past several years. The practice however is dangerous and exposes the user who downloaded the content to many forms of attack. The details and history of several popular content security mechanisms currently deployed are described. Although these mechanisms have been designed to protect the user from malicious content, they are far from perfect. This paper outlines the security and deployment issues inherent in these mechanisms and proposes a new mechanism that addresses them.

October 24, 2002

1 Introduction

In order to protect Internet users from downloading malicious content there must be mechanisms in place to ensure the authenticity and integrity of the downloaded content. Without proof of authenticity the user has no way to determine if the content came from a known and reputable provider. Without proof of integrity the user has no way of telling if the content has been tampered with on its travels between the content provider and the end users system.

Content downloaded over the Internet manifests itself in several forms, ranging from executable content such as software objects, firmware images, Java applications, device drivers, macros and scripts to static content such as virus updates and configuration files. Although the format of this content is quite diverse, the mechanisms used to protect it are the same.

2 Cryptography and Trust

Public Key cryptography permits secure communication to be established between any parties provided only that each has trustworthy knowledge of the public key of the other. The means by which that trustworthy knowledge is obtained is known as Public Key Infrastructure (PKI).

The mechanisms currently in use to secure downloaded content, as described in this paper, are based on PKI and thus it is assumed that the reader has a basic understanding of both Public Key cryptography and PKI concepts.

At a minimum there are three parties involved in the creation and validation of secure content.

1. Content-Publisher - The individual or organization that wishes to make content available for download.

2. Public Key Infrastructure (PKI) Provider (PKIP) - The organization that authenticates the identity of the content provider and manages the identifying credentials in the form of a certificate.

3. Content-User - The individual that wishes to access content from the content publisher.

We will see later that additional parties may get involved, providing additional functionality and security.

3 First Generation Solutions

3.1 Details

The first generation authenticated content services are based on a standard digital signature over the content using a private key belonging to the content provider.

Step 1. The content-publisher generates a public and private key pair.

Step 2. The content-publisher sends the public key and associated naming and authentication information to a PKIP for certification.

Step 3. The PKIP authenticates and verifies the content publisher and issues an X.509 certificate, binding the content publisher's name to the public key. The certificate is returned to the content-publisher.

Step 4. The content-publisher uses the private key to generate a signature over the content to be distributed. The content, signature and certificate are packaged into a standard PKCS#7 signed data message. This package is made available for download

Step 5. The content-user downloads the signed content to their system. The signature on the content is verified to ensure it has not been altered in transit. The certificate used to sign the content is also verified, proving the authenticity of the content. If any of the verifications fail, the content cannot be considered secure.[1]

[1] It is assumed that certificate and certificate chain validation procedures as defined in RFC 3280 are adhered to.

3.2 Issues

The signature created by the content-publisher over the content binds the content to a particular organization, as identified in the certificate. However, the signature makes no statements as to the correctness of the content. Ultimately the decision to render the content is left to the user who is typically aided by a dialog box during download that states that the signature is valid and was signed by a particular organization. In the end it's up to the user to make the judgment as to the safety of the content based on the reputation of the content-publisher.

There are two issues inherent in this solution that may cause a content-user to accept potentially malicious content. The first stems from the fact that time the content was signed is not part of the signature and second that the certificate used to create the signature is not verified.

3.2.1 Signing Time Not Included

Although the signature of the content proves authenticity and integrity, it does not specify when the content was actually signed. The ramifications of this are twofold.

First, the validity period of the signed content is effectively bound to the validity period of the certificate. In practice this means that the validity of the content will expire at the same time the certificate expires. Content signing certificates generally have a lifetime of one year, which forces the content-publisher to resign and redistribute all of their content after they have renewed their certificate. Although this may not be a issue for organizations that have signed very little content, it can be quite a burden on those who have a large library of content. It is also a burden on users who then must find, download, and install the content again.

Second, a malicious content-publisher could turn back the system clock on the system they use to sign content. This allows a content-publisher to use an expired, and thus invalid, certificate. The content-user has no way of determining if this has happened and may then accept content that was signed by an expired certificate.

It should be noted that the use of the `signingTime` attribute in the PKCS#7 message can't be used to address these issues. Because the value of this attribute is generated by the content-publisher and not authenticated by a third party, it is still possible to trick to user into accepting content that has been signed with an expired certificate.

3.2.2 No Certificate Revocation Check

Although first generation content signing solutions don't explicitly disallow certificate validation, most of the currently deployed solutions do not check the revocation status of the certificate used to sign the content either during signature creation or signature validation.

During signature creation, it would be ideal if the software that is used to create the signature on the content checked the status of the content-publishers certificate. Without this revocation check, the possessor of the private key is able to sign content even when the key has been revoked.

Similarly, and perhaps more importantly, the client side software responsible for verifying the signature over the content must check the revocation status of the certificate used to create the signature. Without doing so the client will continue to validate signed content even after it has been reported that a content-publishers signature has been compromised.

3.2.3 Revocation Granularity

Although many first generation solutions do not validate the content-publishers certificate, adding the ability to perform this check is not very difficult. However doing so raises two interesting issues.

First, similar to the certificate expiration issue described in Section 3.2.1, all content signed using this certificate will become immediately invalid once revoked. While this is expected behavior, it places additional burden on the content-publisher as they must resign and redistribute all of their content using their new (and valid) key pair. Again the ultimate consumer is inconvenienced with having to find and reinstall any revoked content.

Second, this solution does not allow for fine-grained revocation possibilities at the signed content level. If, for example, a content-publisher signs and distributes content that shortly afterwards is found to be incorrect, there is no way to "retract" the validity of this single piece of content. The only way to ensure content-users never validate this content is by revoking the content-publishers certificate thus invalidating all content signed by this content-publisher

4 Second Generation Solutions

4.1 Details

The second-generation authenticated content services attempted to improve upon its predecessor by addressing the signing time issues as described in Section 3.2.1. The solution adds a signed time-stamp from a trusted third party in addition to the signature from the content-publisher.

The addition of the time stamp proves that the content existed and was valid at the time the content was signed and thus a content-user can not only validate the integrity and authenticity of the content, but also can verify that the content existed at a certain point in time.

Step 1. The content-publisher generates a public and private key pair.

Step 2. The content-publisher sends the public key and associated naming and authentication information to a PKIP for certification.

Step 3. The PKIP authenticates and verifies the content publisher and issues an X.509 certificate, binding the content publisher's name to the public key. The certificate is returned to the content -publisher.

Step 4. The content-publisher uses the private key to generate a signature over the content to be distributed.

Step 5. The content-publisher creates a time stamping request for the content to be distributed and sends it to a third party time stamping authority (TSA).

Step 6: The TSA creates a time stamp based on the request and returns it to the content-publisher.

Step 7: The content, signature, time stamp and certificate are packaged into a standard PKCS#7 signed data message. This package is made available for download.

Step 8. The content-user downloads the signed content to their system. The signature on the content is verified to ensure it has not been altered in transit. The certificate used to sign the content is also verified, proving the authenticity of the content. The signature on the time stamp is validated, proving when the content was signed. Finally, if the content-publishers certificate was revoked or expired the time contained in the time stamp is compared with the revocation time from the CRL or OCSP response to assert the content was signed whilst the publisher's credential was in good standing. If any of the verifications fail, the content cannot be considered secure.

4.2 Improvements over G1

The addition of the time stamp removes the close binding between the validity period of the content-publishers certificate and the content itself. This allows the content to continue to be validated past the expiration or revocation time on the content-providers certificate. The client simply checks that the time in the time stamp falls between the validity start and end dates in the content-publishers certificate.

The time stamp prevents a malicious content-publisher from turning back their system clock when they sign the content. Because the time in the time stamp is assumed to be accurate the content-user can be confident that the content was signed at a particular point in time when the content-publishers certificate was still valid.

4.3 Issues

Although the addition of the time stamp solves some issues in the first generation solution, it actually introduces more problems than it solves.

4.3.1 Additional Complexity

The addition of the time stamp adds an additional signature to the verification process on the client. Although this may not be an issue for "fat" clients that run on desktop and server class systems, it may be an issue for clients that run on constrained platforms such as mobile devices and PDA's. Finally, the client must perform two separate revocation checks on the certificate that signed the content and the certificate associated with the time stamping server. All of this adds additional burden on the client, especially those that are constrained.

4.3.2 Time Stamp Request not AuthenticatedCurrent solutions provide no mechanism to authenticate the requestor of the time stamp. This allows any one to request a time stamp from the time stamping server. Because of this the time stamping server will still create time stamps for content-publishers who's identity key has been revoked or expired. This is particularly problematic in the case when a content-providers key pair has been compromised and is no longer in control of the real owner.

4.3.3 Revocation Issues

The addition of the time stamp does not address any the revocation issues described in Section 3.2.2. It does however exacerbate the problem because clients must now also check the revocation status of the time stamping authorities certificate. Clients that do not check the status of these certificates continue to be vulnerable unlimited exposure.

4.3.4 CRL Maintenance Issues

In order to keep the size of CRL's reasonably small, CRL entries associated with expired certificates are routinely removed from the CRL list. This process is called CRL pruning and is a standard practice by PKI providers. The pruning of CRL's ensure that the size of CRL's do not continue to grow unbounded as the number of certificates issued by the CA grows.

However as described in Section 4.2 , the addition of the time stamp effectively allows the content to have an unlimited lifetime. This feature plus the practice of CRL pruning by the PKI provider will "un-revoke" a previously revoked content-publisher certificate. This unwanted side effect is described by the following timeline:

Time 0: Content publisher certificate issued

Time 1: Content publisher signs content and requests a time stamp.

Time 2: Content publisher key is reported as compromised and revoked by the PKI provider.

Time 3: PKI provider updates the CRL, thereby adding the serial number of the revoked content provider certificate.

Time 4: Content-user attempts to validate the content. This validation will correctly fail because the content-publishers certificate is on the CRL.

Time 5: Content-provider's certificate expires.

Time 6: PKI provider prunes the CRL and removes the entry for the content-providers certificate as it has expired.

Time 7: A content user attempts to validate the content. Because the content-publishers certificate is no longer on the CRL the content is considered valid and trusted, even though the certificate has been revoked.

This issue forces the PKI provider to not prune CRL's for CA's that issue certificate to content-publishers. As described earlier, this causes the CRL to grow larger as time goes on. Large CRL's introduce many potential problems, including network latency and client side processing issues.

There are a few solution to this problem including the use of an online certificate status protocol such as OCSP or XKMS-Validate.

These protocols ensure the correct status of a certificate is returned even after the certificate in question has expired. Another solution would be to retire the CA after a limited time period, thus limiting the number of certificates that it has issued.

5 Issue Summary

This section summarizes the issues inherent in the first generations of content signing solutions and lists the requirements that the next generation solution should address

Problem 1. The signature over the content makes no statements as to the correctness of the content.

Problem 2. The signature of content becomes invalid when the content-publisher certificate expires or is revoked.

Problem 3. There is no mechanism to revoke a single piece of signed content.

Problem 4. Checking multiple signatures may be a burden for some constrained clients.

Problem 5. A malicious content-publisher can continue to sign content after their certificate has expired or has been revoked. .

Problem 6. Software used by content-users does not properly check the revocation status of certificates.

Problem 7. Time stamp requests are not authenticated by the time stamping server.

Problem 8. Pruning of the CRL will "un-revoke" a previously revoked content-publisher certificate.

Problems 1 through 5 could be considered operational and/or usage problems. However, problems 6 through 8 are security issues and may enable an attacker to trick a content-user into accepting malicious content.

Requirement 1. It shall be possible to assess the correctness of the content to be signed.

Requirement 2. It shall be possible for content to continue to be valid even after the content-publishers certificate expires or is revoked.

Requirement 3. It shall only be possible for content-publishers in good standing to create signatures on content.

Requirement 4. It shall be possible to revoke a single piece of content.

Requirement 5. The solution should not be a burden to implement on constrained devices.

Requirement 6. To the greatest extent possible, the solution should be backwards compatible with currently deployed clients and agents.

6 Third Generation Solutions - Authenticated Content Signing

The third generation Authenticated Content Signing (ACS) solution provides many improvements over previous models and does so while retaining backwards compatibility with deployed first and second-generation compatible software and hardware. The solution described in this section attempts to address the requirements as outlined in Section 5 .

6.1 Details

Similar to the first and second-generation solutions, ACS services are based on a standard digital signature over the content using a private key identifying the content-publisher. However unlike the first two solutions, there is a clean separation between certificates used to identify the content-publisher themselves and certificates used to identify a particular piece of authenticated content. In this section we refer to the certificate that identifies the content-publisher as the *identity certificate* and the certificate used to identify content as the *attestation certificate*. These certificates are issued from an *identity CA* and an *attestation CA* respectively.

Step 1. The content-publisher generates a public and private key pair.

Step 2. The content-publisher sends the public key and associated naming and authentication information to a PKI provider for certification.

Step 3. The PKI provider authenticates and verifies the content-publishers identity and issues an X.509 identity certificate, binding the content-publisher's identity to the public key. The identity certificate is returned to the content publisher.

Step 4. The content-publisher uses the private key associated with their identity certificate to generate a signature over some content to be published. The signature and certificate are packaged into a standard PKCS#7 signed data message and sent to the PKI provider. A suggested validity period and the name and version of the content to be authenticated can also be included in this request.

Step 5. The PKI provider verifies the signature of the PKCS#7 message as well as the identity certificate, including its current revocation status and validity. A failure in any of these checks will result in an error and will halt all further processing.

Step 6. The PKI provider then generates a temporary attestation key pair. The public key is signed by the attestation CA yielding the attestation certificate that is bound to the content. The subject of the attestation certificate includes the name of the content-publisher and also uniquely identifies the content.

Step 7. The PKI provider uses the attestation private key to create a signature over the hash of the submitted content. The temporary attestation private-key is then irrecoverably destroyed. The signature, and certificate are packaged into a standard PKCS#7 signed data message and returned to the content publisher for distribution. This paper will refer to this signed package as a *Trust Mark*.

Step 8. The content-user downloads the content and Trust Mark to their system. The signature on the content is verified to ensure it has not been altered in transit. The certificate used to sign the content is also verified, proving the authenticity of the content. If any of the verifications fail, the content cannot be considered secure.

Note that the PKIP need not witness the content being published as a hash of the content is submitted to the PKIP thus keeping the content confidential. In Section 6.3 below we will describe a scenarios where the content is sent in the message to allow for quality control of the content.

6.2 Identity and Attestation Domains

The separation of certificates used to identify the content-publisher from the certificates used to identify a particular piece of content signature addresses many of the issues of the previous generations of solutions. This solution creates two separate trust domains: The Identity Domain and the Attestation Domain. A distinct root certificate is used to identify each domain.

6.3 Identity Domain - Identifying the Content Publisher

Identity certificates issued to content-publishers are issued from a domain called the Identity Domain. The identity of the content publisher is verified to the extent that the PKI Provider's practices dictate when processing the initial identity certificate request (see steps 2 and 3). The PKI provider needs to be able to verify the signature on the identity certificate and thus must trust the root of the Identity Domain. Only when the identity certificate is found to be in good standing does the PKI provider create an attestation certificate.

6.3.1 Attestation Domain - Validating the Authenticated Content

Attestation certificates are issued from a separate domain called the Attestation Domain. Certificates issued from this domain are bound to and identify a particular piece of content.

When the PKI provider generates the attestation certificate the subject name of the content publisher's identity certificate is used in the subject name of the attestation certificate as well, thereby pushing forward the verified identity of the content provider into the attestation certificate. In addition, the original request signed by the content publisher can be archived by the PKI provider indefinitely and made available for auditing purposes.

6.3.2 Domain Flexibility

The solution does not mandate a particular root for either domain. Depending on the application and requirements of the content-publishers and their customer base, the solution can be customized accordingly. For example certificates rooted in standard public hierarchies, such as VeriSign Trust Network™ roots, may make sense in open environments where broad interoperability is required. Where additional control is needed domains identified by private roots may be tailored to ensure maximum control and oversight.

6.4 Ensuring Content Correctness

Unlike the previous content signing solutions that do not make any assertions regarding the correctness of the content, this solution can be augmented to include a correctness check. Various aspects of quality can be assured to increasing degrees by the use of testing and validation procedures. The organization providing these services is called the Content Correctness Service Provider. It may be PKI provider or a separate organization all together.

6.4.1 Quality Control

A Content CorrectnessService Provider may offer the service of performing an Anti-Virus scans on all content presented to the ACS service. These tests may catch infection of the content that occurred after the production of the content but before the content was signed by the content-publisher. Providers of this service may also offer ongoing scanning of published content when updated definitions or techniques for evaluation are available.

6.4.2 Quality Assurance

Some environments require a much stricter validation of distributed content. The details and the rigor of the validation is defined by the organization running the service and may range from a set of automated tests to a full blown complex evaluation schemes.

The ACS 3G code-signing model can be used to allow one or more validation service providers to participate in the release and approval process.

In this case the steps as outlined in Section 6.1 are augmented slightly. Changes to the flow are specified in **bold**:

Step 1. The content-publisher generates a public and private key pair.

Step 2. The content-publisher sends the public key and associated naming and authentication information to a PKI provider for certification.

Step 3. The PKI provider authenticates and verifies the content-publishers identity and issues an X.509 identity certificate, binding the content-publishers identity to the public key. The identity certificate is returned to the content publisher.

Step 4. The content-publisher uses the private key associated with their identity certificate to generate a signature over some content to be published. The **content**[2], signature and certificate are packaged into a standard PKCS#7 signed data message and sent to a PKI provider. A suggested validity period and the name and version of the content to be authenticated can also be included in this request.

Step 5. The PKI Provider verifies the signature of the PKCS#7 message as well as the identity certificate, including its current revocation status and validity. A failure in any of these checks will result in an error and will halt all further processing.

Step 5a: The content is sent to a content correctness service provider which performs the evaluation of the content based on established criteria.

Step 6. **Once the required tests are completed**

If the **content correctness service provider** is not the PKIP then they must use their identity certificate and associated private-key to generate

[2] Unlike the scenario outlined in Section 4.1 where only the hash of the content is sent. This scenario requires that original content be conveyed to the validation service provider.

a signature across the evaluated content which is then submitted back to the PKIP. This creates a secured audit trail verifying the explicit approval of the content as having met the evaluation criteria of quality as indicated by the content correctness service provider.

Step 5. The PKI provider verifies the signature of the PKCS#7 message as well as the identity certificate, including its current revocation status and validity. A failure in any of these checks will result in an error and will halt all further processing.

Step 6. The PKI provider then generates a temporary attestation key pair. The public key is signed by the attestation CA yielding the attestation certificate that is bound to the content. The subject of the attestation certificate includes the name of the content-publisher and also uniquely identifies the content.

Step 7. The PKI provider uses the attestation private key to create a signature over the hash of the submitted content. The temporary attestation private-key is then irrecoverably destroyed. The signature, and certificate are packaged into a standard PKCS#7 signed data message and returned to the content publisher for distribution. This paper will refer to this signed package as a *Trust Mark.*

Step 8. The content-user downloads the content and Trust Mark to their system. The signature on the content is verified to ensure it has not been altered in transit. The certificate used to sign the content is also verified, proving the authenticity of the content. If any of the verifications fail, the content cannot be considered secure.

6.5 Resilience to Attack

This section discusses potential attack and misuse scenarios and how the ACS solution is designed to correctly deal with them.

6.5.1 Attacks on the Content Publisher

Many things can go wrong at the content publisher link in the chain.

First, a content publisher's identity key may be compromised. Thanks to the separation of hierarchies (identity and attestation) these cases are mitigated substantially.

The key to this scenario, and other scenarios where a key has been compromised, is to ensure the content publisher's certificate is immediately revoked. The audit trail maintained by the PKI Provider can be referenced to list all uses of the publisher's revoked certificate and private-key to request a trust mark. This list can then be examined to see if any fraudulent use has occurred by comparing the time of use with the reported time of compromise. Trust marks issued to fraudulent requests are immediately revoked. A compliant consumer of ACS signed content can be immediately aware of the discovery of compromise.

Second, if a content publisher errantly signs and submits the wrong content (e.g. older version with known bug) or discovers that the content has been infected with a virus or some other evil-ware or mal-ware component, the attestation certificate (trust mark) is revoked immediately and further distribution problems are mitigated.

Third, A disgruntled employee who once was authorized to publish content may begin to submit errant or dangerous content\. As soon as this is suspected their identity certificate is revoked and rendered impotent. The cryptographic audit log maintained by the PKIP may be used to check for inappropriate use and if any is uncovered those attestation certificates can be revoked thereby mitigating further problems.
Attacks on the Attestation Service

If the attestation service errantly issues an attestation certificate the errant certificate is immediately revoked and the service may be halted until the root cause of errant issuance is discovered. By examining the cryptographic audit log the entire set of errantly issued attestations certificates can be discovered and revoked mitigating further problems. Once the fault is repaired the service can be brought back into operational mode. Note also that as the log of signed requests is available locally at the PKI Provider site a watchdog process may audit this , continuously correlating and evaluating signed requests with issued trust marks – possibly while evaluating previously signed published content against newer anti-virus databases.

6.5.2 Compromise of Identity and Attestation Hierarchies

6.5.2.1 Identity CA

If an identity CA has been compromised the CA must immediately be revoked. All subordinate certificates must also be revoked. A new key-pair is generated and a new certificate issued to the new identity CA by the identity root. The new CA must then re-issue identity certificates.

Note that certificates from the compromised hierarchy have no potential to attack the system, as service requests will not be honored from revoked certificates. Trust Marks issued from requests made by identity certificates issued after the compromise time must be revoked.

Note also that there is no impact on existing trust marks. These trust marks are built on attestation certificates which are signed by the attestation CA and not the identity CA.

6.5.2.2 Identity Root

If the identity root CA has been compromised it must be revoked immediately. The identity CA and identity certificates issued under the compromised root must also be revoked. A new key-pair is generated and a new root certificate created. The identity CA must also be re-issued by the new identity root CA. Finally the new identity CA must then re-issue identity certificates.

Note that certificates from the compromised hierarchy have no potential to attack the system, as service requests will not be honored from revoked certificates. Trust Marks issued from requests made by identity certificates issued after the compromise time must be revoked.

Note also that there is no impact on existing trust marks. These trust marks are built on attestation certificates which are signed by the attestation CA and not the identity CA nor identity root CA.

6.5.2.3 Attestation CA

If the attestation CA is compromised the CA must be revoked immediately.

Any attestations made by the previous CA must be revoked. The risk is if the attestation CA's private key is stolen or discovered the entity having the compromised key can then use it to sign new attestations. Like any other system critical component the attestation CA must be safeguarded from abuse. Reputable CAs use tamperproof hardware to ensure theft of the key is not possible without detection but this is not sufficient as a key may be discovered via brute force attack and flaws may exist in the cryptographic hardware such that a theft of the private-key material may be possible without detection.

The design of ACS is such that all legitimate requests are stored in a cryptographically tamper proofed format (i.e. signed by the requestor) and so it is a trivial matter for the attestation CA to generate new attestations to previous legitimate requests. However these new attestation certificates and corresponding trust marks must replace the previously distributed trust marks.

It is for this reason that it is recommended that one uses the optional dual signature trust mark package. This package differs from the base package by including the original signed request as an extension field in the attestation certificate. The requirement to have the original signed request block, which was signed by the developer or publishing agent using a credentials issued by the identity CA, ensures that both the publisher's private key and the attestation CA's private key must be compromised before the system is compromised. When using the dual signed ACS mode there is little if any impact on existing trust marks as any exposure would require that a publisher's credentials have been compromised already.

6.5.2.4 Attestation Root

If the attestation root is compromised the root and CA and all attestation certificates issued there under must be revoked immediately.

Again as reputable CAs use tamperproof cryptographic hardware to store sensitive private-keys and so theft is likely to result in detection. Like the attestation CA compromise scenario it is relatively easy to generate the new attestations given the existence of signed requests in the PKIP database.

Again if the optional dual signature ACS model there is a very small probability of a compromise to the root having any affect on existing applications. However recovery is a bit more complex as a new root cannot simply be deployed. However several models for handling this scenario are available as well.

6.6 Issues

The PKI Provider must retain the original request as provided and signed by the content provider for use as cryptographic evidence and must be available for verification.

PKI Providers may timestamp their logs and audit date to extend the period upon which they may be relied.

7 Meeting the Requirements

This section summarizes how the ACS solution meets the requirements outlined in Section 5

Requirement 1. *It shall be possible to assess the correctness of the content to be signed.*

As described in Section 6.4

Requirement 2. *It shall be possible for content to continue to be valid even after the content-publishers identity certificate expires or is revoked.*

Because there is a clean separation between the key used to identify the content-publisher and the key used to identify the content, the expiration and/or revocation of the identity certificate does not effect the status of the attestation certificate or the signature on content itself.

Requirement 3. *It shall only be possible for content-publishers in good standing to create signatures on content.*

Before a PKI provider creates an attestation certificate, the validity of the identity certificate is checked. This ensures that signatures on content are only created when the content-publisher's certificate is valid.

Requirement 4. *It shall be possible to revoke a single piece of content.*

Because there is a one to one mapping between attestations certificates and a particular piece of content it is possible to invalidate a single piece of content by revoking the associated attestation certificate. Standard CRL and/or OCSP mechanisms can be used to accomplish this.

Requirement 5. *The solution should not be a burden to implement on constrained devices.*

At a minimum the client need only have the ability to validate the attestation certificate and the signature on the associated content.

Requirement 6. *To the greatest extent possible, the solution should be backwards compatible with currently deployed clients and agents.*

The ACS service will respond in kind to the format of request it receives. If a simple signed hash is delivered to the service the same hash will be signed and returned by the service. If a structured object whose structure is known by the service is submitted the service will parse and structure and replace the signature before returning the signed object.

8 Conclusions

The third generation Authenticated Content signing solution provides many improvements over the previous generations. Improvement include cryptographic auditability of all events, unit granular revocation control over signed content, the ability to retract a signature on a particular piece of content, enabling the addition of a third party validation lab, and keeping backwards compatibility with existing content-signing

9 References

[OCSP] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, *X.509 Internet Public*

Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 June 1999.

[WAPCert] WAP Certificate profile Specification, http://www1.wapforum.org/tech/terms.asp?doc=WAP-211-WAPCert-20010522-a.pdf

[SCont] WAP Signed Content Specification

[Profile] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. RFC 2459, January 1999.

[XKMS]W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp, XML Key Management Specification (XKMS), W3C Note 30 March 2001, http://www.w3.org/TR/xkms/

[XML-SIG] D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. *XML-Signature Syntax and Processing*, World Wide Web Consortium. http://www.w3.org/TR/xmldsig-core/

[Authenticode] Authenticode Overviews and Tutorials, http://msdn.microsoft.com/library/default.asp?url=/workshop/security/authcode/authenticode_ovw_entry.asp

[PKCS#7] PKCS#7 standard, RSA.