Instant Certificate Revocation and Publication using WebDAV

David W Chadwick, Sean Antony, Rune Bjerk

Computing Laboratory, University of Kent, UK

Abstract. There are several problems associated with the current ways that certificates are published and revoked. This paper discusses these problems, and then proposes a solution based on the use of WebDAV, an enhancement to the HTTP protocol. The proposed solution provides instant certificate revocation, minimizes the processing costs of the certificate issuer and relying party, and eases the administrative burden of publishing certificates and certificate revocation lists (CRLs). We describe how WebDAV can be used for X.509 certificate revocation, and describe how we have implemented it in the PERMIS authorization infrastructure.

Keywords: revocation, CRLs, LDAP, HTPP, WebDAV

1. Introduction

The most common and standardized way of publishing X.509 certificates is in corporate LDAP servers. Several technical problems with the use of LDAP directory servers have been widely documented [1], including: the use of *;binary* encodings, the lack of a distributed directory capability, the inability to search for certificates containing specific fields, and the difficulty of retrieving individual certificates when a user has several. Other problems are less technical in nature and instead are operational, but they are nevertheless just as inhibiting to successful deployments. For example, most corporate firewalls do not allow the LDAP protocol to pass through them, therefore certificates or certificate revocation lists (CRLs) cannot be easily accessed by external organizations. Finally, we have the complexity of installing and managing LDAP servers, for example, loading the correct schema and setting the correct access rights, which although not insoluble, nevertheless cause frustration and inconvenience especially to small scale deployments with a lack of specialist staff. For these reasons we wanted to find an alternative mechanism to LDAP for publishing X.509 certificates and CRLs that would not suffer from these problems. We wanted a generic solution that would support both public key certificates (PKCs) and attribute certificates (ACs), and that most (ideally all) organizations would already be familiar with. We wanted to use Apache servers and the HTTP protocol, since these are ubiquitous and most organizations know how to install and configure their own web servers. But HTTP on its own is insufficient for certificate and CRL management, since it does not provide a number of useful features, such as the ability to search for specific content. For this reason we investigated (and subsequently utilised) the WebDAV extensions to HTTP [2].

The most common way of revoking public key certificates is through the use of CRLs. However these suffer from a number of well documented operational problems such as the potential for denial of service attacks from lack of availability, or the consumption of too many resources due to their increasingly large size. We will address and re-appraise the whole issue of certificate revocation in section 2. In section 3 we propose a different approach to revocation based on the Representational State Transfer (REST) principles [19]. Section 4 describes the WebDAV protocol, which is an extension to HTTP, and how it can be used for X.509 certificate and CRL storage and retrieval according to the REST principles. Section 5 describes our implementation of WebDAV in the PERMIS authorization infrastructure, in order to store X.509 attribute certificates used for authorization. This mechanism can similarly be used by PKIs to store public key certificates and CRLs. Section 6 discusses our approach, compares it to other work, and concludes.

2. Reappraising Revocation

There are several different approaches that have been taken to the complex issue of revocation of certificates, and of informing remote relying parties when revocation has taken place. A relying party is any Internet based service that consumes the issued certificate (whether it be a PKC or an AC). The primary objective of revocation is to remove a certificate (and all its copies, if any) from circulation as quickly as possible, so that relying parties are no longer able to use it. If this is not possible, a secondary objective is to inform the relying parties that an existing certificate in circulation has been revoked and should not be used or trusted. The latter can be achieved by requiring either the relying parties to periodically check with the certificate issuer, or the certificate issuer to periodically notify the relying parties. Of these, requiring the relying parties to periodically check with the certificate issuer is preferable for two reasons. Firstly, it places the onus on the relying parties rather than on the issuer, since it is the relying parties who are taking the risk of using revoked certificates. Secondly, an issuer may not know who all the relying parties are, so will have difficulty contacting them all, but the relying parties will always know who the certificate issuer is, since this is contained in the certificate.

The simplest approach to certificate revocation, that used by X.509 proxy public key certificates [6], the Virtual Organisation Membership Service's (VOMS) X.509 attribute certificates [8] and SAML attribute assertions [7] (which are to a first approximation simply XML encoding of X.509 binary ACs), is to never revoke a certificate, and instead to issue short lived certificates that will expire after a short period of time. The certificates are thus effectively and automatically removed from circulation after this fixed period expires. The assumption is that it is unlikely that the certificates will ever need to be revoked immediately after they have been issued and before they have expired due to abuse, therefore the risk to the relying parties is small. Risk (or more precisely risk exposure) is the probability of occurrence multiplied by the loss if the event occurs. Because short lived certificates are only valid for a short period of time, the probability of occurrence is small. Of course, the loss or amount of damage that can be done in a short period of time can be huge, so short lived certificates are not always the best solution where the resulting loss can be high. Consequently SAML attribute assertions have the optional feature of containing a "one time use" field which means that the consuming service can only use the attribute assertion once to grant access, and then it should never be used again. This is designed to minimise the potential loss and minimise the time to effective revocation. A similar standardised extension, called singleUse, has also been defined for X.509 attribute certificates, in the draft amendment [20] to X.509 (2005) [4]. This is due to be published in the 2009 edition of X.509. But short lived certificates and one time use are not ubiquitous solutions since they are not appropriate for every situation.

An advantage of short lived certificates (and one time use) is that they effectively remove a certificate from circulation after a short period of time, and consequently they mandate that users or service providers must frequently contact the certificate issuer in order to obtain new freshly minted certificates.

The main disadvantage of short lived certificates is knowing how long to issue them for. They should be valid for the maximum time that any user is likely to need them for, otherwise one of the later actions that a user performs may fail to be authenticated or authorised which could lead to a session being aborted and all the processing that has been done so far, being lost. This is a current well known problem with the use of X.509 proxy certificates in grid computing. On the other hand, the longer the certificates are valid, the greater their possibility of misuse without any direct way of withdrawing them from circulation. This has caused some researchers to suggest that proxy certificates should be revocable!

A second disadvantage of short lived certificates is that the bulk of the effort is placed on the issuer, who has to keep reissuing new short lived certificates. This could become a bottleneck to performance. A better solution should put the bulk of the processing effort onto the relying parties, since these are the ones who want to use the issued certificates and the ones who need to minimise their risks. Thus it seems appropriate that they should be burdened with more of the costs.

If the primary objective of removing a certificate from circulation cannot be easily achieved, then the secondary objective of notifying the relying parties when a certificate has been revoked can be achieved by issuing certificate revocation lists, trees or hash chains.

A CRL is a digitally signed list of revoked certificates, usually signed by the same authority that issued the original certificates. Revocation lists are updated and issued periodically. X.509 CRLs contain their date and time of issue, and have an optional *next update* time which signifies the latest date by which the next CRL will be issued. Relying parties are urged to obtain the next issue of the revocation list from the issuer's repository before the next update time has expired, in order to keep as up to date as possible. If the next update time is not specified in the CRL, then the frequency of update has to be communicated by out of band means from the issuer to the relying parties. Alternatively, the latest CRL can be sent by the subject in his access request along with his certificate, to prove that his certificates has not been revoked. The use of CRLs is standardised in X.509 [4] and the use of LDAP for storing CRLs in RFC 2252 [9]. Revocation lists ensure that relying parties are eventually informed when a certificate has been revoked, no matter how many copies of the certificate there are in circulation, but revocation lists have several big disadvantages. Firstly there is always some delay between a user's certificate being revoked and the next issue of the revocation list appearing. This could be 24 hours or even longer, depending upon the frequency of issuance of the CRLs. Thus, in order to reduce risk to a minimum, a relying party would always need to delay authorising a user's request until it had obtained the latest CRL that was published *after* the user issued his service request, which of course is impractical for most scenarios. If the relying party relies on the current revocation list, then the risk from using a revoked certificate equates, on average, to half that of using a short lived certificate, assuming the validity period of a short lived certificate is equal to the period between successively issued CRLs. This reduced risk comes at an increased processing cost for the relying party and the issuer.

CRLs can put a significant processing load on both the issuer and the relying party. CRLs have to be issued at least once ever time period, regardless of whether any certificates have been revoked or not during that period. In a large system the lists can get inordinately long containing many thousands of revoked certificates. These have to be re-issued every time period, distributed over the network and read in and processed by the relying parties. In Johnson and Johnson's PKI, their CRL was over 1MB large within a year of operation [10]. To alleviate this problem, the X.509 standard defines delta revocation lists, which publish only the changes between the last published list and the current one. But this increases the processing complexity of the client, and few systems appear to support this feature today.

Certificate revocation trees (CRTs) were first specified by Paul Kocher in 1998 [21]. The leaves of the revocation tree comprise pairs of revoked certificate serial numbers, in ascending order, between which all the other certificates have not been revoked. For example, if certificates with serial numbers 2, 12, 18 and 35 are revoked, then the leaves of the hash tree would be ($-\infty$ to 2), (2 to 12), (12 to 18), (18 to 35), and (35 to ∞). The data in each leaf is hashed using a secure hash function such as SHA1. Each adjacent pair of leaves is hashed together to create their parent node in the hash tree, and adjacent parents are hashed together to create grandparent nodes. This continues until a single root node is created. Finally, the root hash is signed by the CA along with its date, time and expiry time of the hash tree. The hash tree and signed root are stored in a publicly accessible repository. When a relying party wishes to find out the status of a particular certificate, the repository responds with the leaf node pertaining to the certificate serial number in question, the hashes that are needed to re-compute the root hash, and the signed root node. The relying party inspects the leaf node to determine if the certificate has been revoked or not, then hashes the leaf node, uses the other hashes to compute the parent, grandparent etc. nodes up to the root node, then checks that the signed root contains the same hash as the one just computed. If so it can deduce that the status of the certificate is trustworthy. The main advantage of CRTs over CRLs is that the size of information returned by the repository soon becomes smaller than for CRLs. The information only increases by $\log_2$ of the number of leaves of the hash tree, whereas with CRLs it increases linearly with the number of revocations. This reduction in size is offset by an increase in computation effort for both the CA and the relying parties, since the hash trees have to be computed as well as the root node signatures. Ideally new CRTs should be computed every time another certificate is revoked, but this could put too much load on the CA. Policies could dictate that new CRTs are only computed periodically, and they will then include all the newly revoked certificates (or none) within each time period, but there is no defined way of distributing this policy information. Consequently they have not been standardised by X.509, and are not widely deployed or used.

The certificate revocation system (CRS) [22], patented by Micali, relies on hash chains created by a CA and stored in a repository. The advantages of the CRS are reduced communications costs between the relying party and the repository, but the disadvantages are increased communications and processing costs by the CA. These disadvantages, coupled with the patent, mean that few (if any) suppliers have implemented it.

A much more successful approach for relying parties is to use the online certificate status protocol (OCSP) [3], standardised by the IETF. Rather than a relying party periodically retrieving the latest revocation list (or CRT or CRS hash number) from the issuer's repository, the OCSP allows a relying party to ask an OCSP responder in real time if a certificate is still valid or not. The response indicates if the certificate is good, or has been revoked, or its status is unknown. Since most OCSP responders base their service on the latest published CRLs, the revocation status information is no more current than if the relying party had consulted the latest revocation list itself, thus the risk to the relying party is not lessened. But what an OCSP responder does do is reduce the amount of communications and processing that a relying party has to undertake in order to validate a user's certificate. This reduced cost to the relying parties is offset by the cost of setting up and running the OCSP service, and the CA having to sign every response (instead of every CRL). Some commercial CAs such as Verisign now employ specialised hardware to sign the OCSP responses in order to gain adequate performance. So that the hardware is not lying idle, or has queues building up during busy periods, OCSP responses are pre-computed for all un-revoked certificates, prior to them actually being requested. Then when a request from a relying party is received, the response can be instantly returned (or if it has not yet been computed it can jump to the top of the signing queue). When a certificate is revoked, the new OCSP response also jumps to the top of the hardware signing queue, so that it is ready for instant return to any relying party that inquiries about this certificate.

Unfortunately none of the above approaches to revocation is ideal. Certificates often have a naturally long validity period, making short lived certificates inappropriate. For example, authentication certificates are typically issued and renewed annually, whilst some authorisation certificates might require an equally long duration e.g. project manager of a 2 year project. Long lived certificates have traditionally necessitated the use of CRLs but these have several disadvantages as highlighted above. OCSP may be an alternative, but it comes at considerable expense and effort to the CA. Alternatively we could continually re-issue short lived session certificates throughout the duration of the natural validity period, for both authentication and authorisation purposes, without needing to issue CRLs as well, but then there is the inherent conflict between making the short lived certificates long enough for the biggest session and short enough to minimise the risk. Thus we propose a new conceptual model that we believe is superior to short lived certificates, CRLs and OCSP servers.

3. A new model for revocation

We believe that the optimum approach to certificate issuing and revocation should have the following features:

- A user's certificate should only need to be issued once (and not continually reissued as with short lived certificates) in order to minimise the effort of the issuer.
- The certificate should be valid for as long as the use case requires, which can be a long (measured in years) or short (measured in minutes) duration. Again, this minimises the effort of the certificate issuer (and the delegator, when attribute certificates are used to delegate authority).
- A certificate should be able to be used many times by many different relying parties, according to the user's wishes, without having to be reissued. Of course, the certificate will need to be validated by each relying party each time it is used. But this mirrors the situation today with our plastic credit cards and other similar types of certificate.
- A certificate should be capable of being revoked at any time, and the revocation should be instantaneous.
- The responsibility for retrieving revocation information should lie with the relying parties, since this is where the risk lies. Relying parties should be able to immediately learn about the current revocation status if they so wish, thereby minimising their risks.
- The distribution of certificates and revocation information should not be inhibited by firewalls and therefore the protocols should be based on HTTP.
- The whole process of checking certificate validity should be as efficient as possible to both the issuer and the relying party.

All the above features, including instant revocation, can be achieved in the following way. The model is based on the Representational State Transfer (REST) principles [19], which were first documented in the PhD thesis of Roy Fielding, an author of HTTPv1.1 [15]. REST assumes a client server interaction model, in which the server is stateless. Any state information must be modeled as a web resource, addressable by a URL. The web itself is therefore the state transition machine for a system. When applied to certificates, if a certificate does not exist it has no state and therefore no web page. When a certificate exists, it has a unique web page (which we call the certificate URL) at which it must be found. When a certificate is revoked, it has a unique web page (which we call the revocation URL) at which the revocation information should be found[1]. The revocation information can in principle be any

[1] Note that we say a revoked certificate should be found at the revocation URL and not must be found there. This is because some applications may treat a revoked certificate as being the same as a certificate that never existed, in which case a revoked certificate wont have a URL. Other applications may need cryptographic evidence that a certificate did exist but has now been revoked, in which case it will have a revocation URL.

information, such as the reason and time of revocation, but in order to have cryptographically secure revocation information we suggest it should be a conventional CRL of length one, containing the serial number of the revoked certificate. Obviously both the certificate URL and revocation URL pages must not exist simultaneously on the web, since a certificate cannot be in both states simultaneously. Normally just one of these pages will exist for the entire validity period of the certificate. Optionally the revocation URL may exist after the validity period of a certificate has expired so that relying parties can still discover, after a certificate has expired, that it was revoked sometime during its validity period.

The life cycle of a certificate is therefore as follows. The issuer issues a certificate, giving it its natural validity period, and stores the certificate in a HTTP based repository which is under the control of the issuer. We choose HTTP since this is the protocol of the web, it can penetrate firewalls and provide read access to external relying parties. Each certificate is given its own unique URL (the certificate URL) which points to its location in the repository. This URL is stored in the certificate in a standard extension, in order to strongly bind the repository location to the certificate, so that relying parties know where to go to check if the certificate is still valid. When a certificate comes to the natural end of its validity period, it is simply removed from the repository. If a certificate needs to be revoked during its validity period, the issuer simply deletes the certificate from its repository and optionally stores a CRL of length 1 at a different location in the repository (the revocation URL), and this location is also stored in a standard extension in the certificate. The absence of a certificate at its published certificate URL may be sufficient for some applications to know that it is no longer valid. Other applications may require cryptographic proof that the certificate has been revoked, in which case the issuer will simultaneously issue a CRL of length 1 and store this at the revocation URL. Relying parties are now able to instantaneously find out the current status of a certificate by contacting the issuer’s repository, using either of the URLs embedded in the certificate.

The frequency and method by which a relying party contacts the issuer’s repository is determined by its risk mitigation strategy and the optional presence of the revocation URL. The frequency can vary per application or per user request, and is set by the relying party as appropriate, and not by the issuer, which is putting the responsibility and risk where it belongs, with the relying party. In order to minimise risk, a relying party should contact the issuer’s repository when a certificate is first validated, and then periodically during the life of the user’s session according to its own risk assessment.

Relying parties can operate in either certificate push or certificate pull mode. In certificate push mode, the user pushes his certificate(s) to the relying party along with his request. In certificate pull mode, the user issues a request and the relying party pulls the necessary certificates from configured repositories (see section 4.2). In certificate pull mode, the relying party must contact the repository in order to obtain the user’s certificates, and therefore the current revocation status of the certificates is automatically obtained for free. If the relying party is operating in certificate push mode, contacting the repository is not necessary in order to obtain the user’s certificates; therefore it should be done as determined from a risk perspective.

We propose the following procedure for determining the revocation status of a certificate that is already in the possession of the relying party. This procedure will be carried out periodically during the certificate's validity period, initially when the certificate is first validated, and then at risk determined intervals during the user's session. The relying party issues a HTTP or HTTPS GET command to the certificate URL. We will discuss the choice between HTTP and HTTPS later. If the HTTP status code 404 Not Found is returned, the relying party may assume that the certificate has been revoked, and permanently record this in its internal cache along with the time of the request. If the certificate is returned, a simple bitwise comparison of the initial validated certificate with the subsequently retrieved copy of the certificate is all that is needed by the relying party to ensure that the certificate is still identical to the one originally validated. Certificate signature verification is therefore not needed for revocation status checking. The relying party may optionally cache the time of last retrieval. This procedure is designed to minimise the processing effort of both the

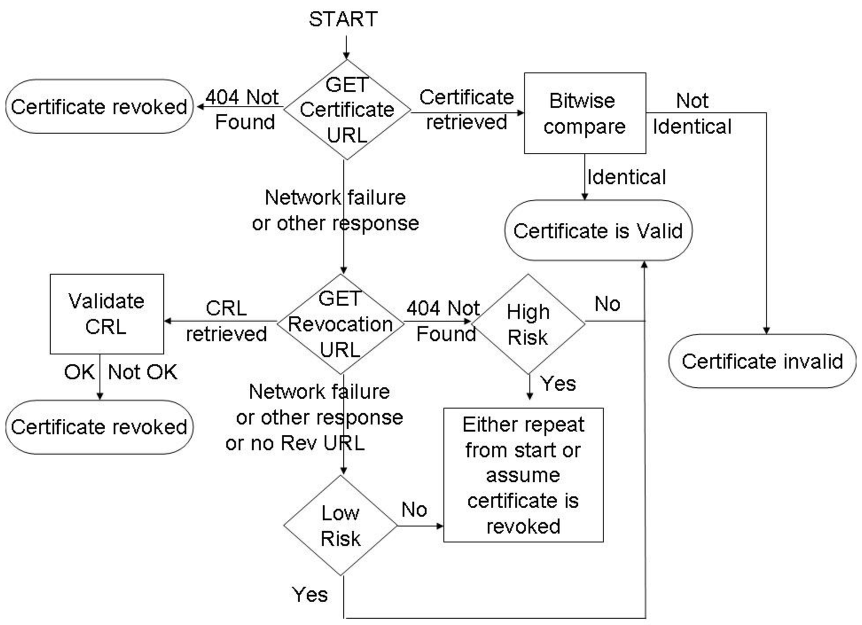


Figure 1. The revocation procedure

issuers and the relying parties, whilst maximising the freshness of the revocation information. Issuers do not need to continually mint new certificates or CRLs, and relying parties do not need to process potentially large revocation lists or perform expensive cryptographic operations for the vast majority of their revocation checks.

If on the rare occasion a client is unable to contact the certificate URL and retrieve either a certificate or a Not Found response, it may attempt to contact the revocation URL, providing there is one is in the certificate. If the HTTP status code 404 Not Found is returned from the revocation URL, and the transaction is low risk, the relying party may assume that the certificate is still valid. In high risk cases, the relying party should assume that an attacker is blocking access to the certificate URL and spoofing the revocation URL, and should either start the procedure again or assume that the certificate is revoked. If a CRL of length 1 is returned, the signature and serial number are validated and the relying party caches the result permanently to ensure the certificate cannot be used again and no further retrieves need be made. Intermediate caching of the CRL is supported and encouraged, so that if a certificate has been revoked and the CRL successfully retrieved, intermediate web servers can cache the CRL to speed up subsequent queries. Finally, if the relying party is unable to make a connection to the revocation URL, or one does not exist, then the relying party can check its cache to see if a previous request to either URL has succeeded or not. If neither URLs are available, the relying party should use its local risk assessment procedure to decide what to do when there are network problems. If the transaction is low risk, it may decide to treat the certificate as valid. If the transaction is high risk, it may decide to try contacting the URLs again, or alternatively to treat the certificate as revoked. The flow chart in figure 1 summarizes this procedure.

Clients may use either HTTPS or HTTP depending upon their and the issuer's security requirements. HTTP presents a number of security weaknesses compared to HTTPS. Firstly HTTP provides public access to the certificate, which may violate the privacy of the certificate subject. (There is no equivalent privacy leakage for a CRL.) Furthermore intermediate Web servers may cache copies of frequently accessed web pages to improve performance, but this would negate the proposed revocation service. To counteract this, the issuer's Web server must use the no-cache cache-response-directive [15] in the HTTP response of successful certificate requests and Not Found CRL requests, to prevent intermediate servers from caching these responses. This will ensure that all subsequent queries are directed to the authoritative source of the information and that stale cached responses are not received. Finally HTTP is susceptible to redirection, substitution and man in the middle attacks. Consequently, if the certificates are not meant to be publicly available or stronger security is required, then secure access should be provided using HTTP with TLS [5]. This will stop network redirection, substitution attacks and intermediate caching. TLS can also provide confidentiality of the retrieved certificates during transfer, in cases where privacy protection of sensitive certificates is required by the issuer. TLS can also provide strong client side authentication, which will allow access controls to be placed on the HTTP repository, further protecting the privacy of the subjects' certificates. The privacy of CRLs is less important, and it enhances security if more copies of these are publicly available.

3.1. X.509 Certificate Extensions

We define two new access methods for the AuthorityInformationAccess (AIA) extension defined in RFC 3280 [14]. The AIA extension is designed to point to services of the issuer of the certificate in question. One of the standard uses of this extension is to point to the OCSP service provided by the issuer. Since the REST service can be used as an alternative to the OCSP service, it seems appropriate to use the AIA extension to point to this. We copy below the ASN.1 of the AIA extension, taken from RFC 3280 for the convenience of the reader:

```
AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

 AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }
```

The two new accessMethods, certificateURL and revocationURL, are defined as follows:

```
certificateURL OBJECT IDENTIFIER ::= { 1.2.826.0.1.3344810.10.2 }

revocationURL OBJECT IDENTIFIER ::= { 1.2.826.0.1.3344810.10.3 }
```

When the AIA accessMethod is certificateURL or revocationURL, then the accessLocation must be a URL pointing to the repository where the certificate or CRL (of length 1) can be found. The URL must point to the exact location of the certificate or CRL in the server so that relying parties can download the certificate or the CRL.

Note that the revocationURL may only be present in a certificate if the certificateURL is present.

When a certificate has been issued containing the certificateURL extension, and it has not been revoked, then it must be present at the certificateURL location specified in this extension. When a certificate has been issued and revoked, the certificate must not be available at the certificateURL location.

If the revocationURL extension was present in the certificate prior to its revocation, then a CRL of length 1 containing the serial number of the revoked certificate, should be present at the revocationURL location immediately the certificate is revoked.

4. The WebDAV protocol and its use with X.509

WebDAV [2] is an Internet RFC that specifies extensions to the HTTP/1.1 protocol so that web content can be managed remotely. WebDAV provides users with the ability to create, remove and query information about web pages, including their contents and properties, such as their creation dates, expiry dates, authors etc. In the context of X.509, a web page will be a single X.509 certificate (either public key or attribute) or a CRL containing a single entry, and their properties can be any fields of the certificate or CRL. WebDAV also provides the ability to create sets of related web pages, called collections, and to retrieve hierarchical membership listings of them. In the context of X.509, a certificate subject can represent a collection, and his/her certificates can be the collection membership listing. The set of CRLs issued by an issuer can also be a collection membership listing. WebDAV is widely supported, several open source implementations are available including one for Apache, and there is an active community working with it (see http://www.webdav.org/).

WebDAV resources are named by URLs, where the hierarchical names are delimited with the "/" character. The name of a collection ends with /. When a relying party does not have a certificate to hand, it needs a standard way of locating a subject's certificate in a WebDAV store. If we model our X.509 certificate store in the same way as an X.500/LDAP directory tree, and name it using the subject distinguished names (DNs) to represent collections, this provides us with a standard way of retrieving a listing of all the certificates that are owned by a single subject. We use the rules of RFC 4514 [24] to convert the DNs into strings, with the exceptions that we replace the comma "," separator between RDNs with the "/" character which is the WebDAV separator, and replace spaces with %20. For example, a collection belonging to the subject whose Distinguished Name is, in RFC 4514 syntax, "cn=David Chadwick, o=University of Kent, c=gb", will be named in a WebDAV repository with the URL:

https://server.dns.name/c=gb/o=University%20of%20Kent/cn=David%20Chadwick/

A GET request to retrieve all the certificates of David Chadwick would use the URL of the collection, viz:

```
GET /c=GB/o=University%20of%20Kent/cn=David%20Chadwick/ HTTP/1.1
Host: server.dns.name
```

A GET request to retrieve a specific certificate of a subject will use the URL specified in the certificateURL access location.

We can similarly model a CRL store as a collection of CRLs under its issuer, using the collection name cn=CRLs/, where each CRL contains a single CRL entry and is named with the serial number of the certificate that it revokes. This provides us with the ability to retrieve a listing of all the CRLs that have been issued by a single issuer, and consequently a listing of all certificates that have been revoked.

For example, if David Chadwick is an attribute authority who delegates attribute certificates to people, and subsequently revokes some of them, then a GET request to retrieve all the CRLs issued by David Chadwick would use the URL of the collection, viz:

```
GET /c=GB/o=University%20of%20Kent/cn=David%20Chadwick/cn=CRLs/ HTTP/1.1
Host: server.dns.name
```

A GET request to retrieve a specific CRL of a certificate will use the URL specified in the revocationURL access location of the certificate.

4.1. Deriving Unique Names for Certificates and CRLs

Because PKCs and ACs may be updated or re-issued by their issuers, it is important to have unique names (certificate URLs) for each of them. Furthermore, each certificate MUST have its unique certificate URL and optional revocation URL embedded in it so that relying parties can retrieve the contents of either URL to check the current state of the certificate.

Whilst the primary contents of a PKC are fixed (i.e. public key and subject names) the contents of an AC are very varied, and can be any attribute, including authorisation policies. We therefore propose a fixed naming scheme for PKCs and CRLs, but two different naming schemes for ACs, depending upon whether the attribute contains a role/privilege, or a policy.

We have defined the following algorithm to create the unique names for certificates and CRLs:

- each AC has the file suffix .ace,
- each PKC has the file suffix .p7c (if in CMS format) or .cer (if DER or Base64 encoded)
- each CRL has the file suffix .crl

We use the rules described in RFC 4514 [24] to create strings from distinguished names (DNs), in particular, using the comma "," to separate the RDNs in a DN, a plus sign "+" to separate the attribute value assertions in an RDN, and an equals sign "=" to separate attribute types and values.

- the name of a PKC file is created from the issuer DN and certificate serial number concatenated with a plus sign e.g. "cn=CSCA,o=university of kent, c=gb+SN=123445.p7c"
- the name of a role AC file is created from the contents of its attribute values plus the serial number of the certificate, concatenated with a plus sign E.g. a role AC with the embedded attribute type PermisRole with attribute value Project Manager, and certificate serial number of 12345 would create the filename "PermisRole=Project Manager+SN=123456.ace". The serial number provides the uniqueness, whilst the attribute type and value provides user friendliness when the issuer wants to browse a WebDAV repository and retrieve an AC for editing.
- the name of a policy AC file is created from the unique name of the policy embedded in the policy attribute value, E.g. a policy with the name "AstroGridUsers" would produce the filename "Policy=AstroGridUsers.ace". Note that it is a policy language issue how the policy name is encoded in the policy attribute value.
- the name of a CRL file is created from the serial number of the certificate that it revokes. E.g. a CRL that revokes a certificate with serial number 1234 would produce the filename "serialNumber=1234.crl".

4.2. Location of the WebDAV servers

The certificate issuer is free to choose whichever WebDAV server(s) it wishes for publishing the state of its CRLs and certificates. Note that the certificate and revocation URLs for a given certificate don't need to point to the same WebDAV server, since conceptually they are independent URLs. However, in order for relying parties to be able to independently pull certificates from the WebDAV store(s), relying parties need to be able to make an informed guess where to look for them. As RFC 4387 [25] points out, there are 4 possible ways for relying parties to determine where a particular certificate might be located:

- Information contained in the certificate (this is easy for relying parties that have a copy of the certificate to hand, since the information is now immediately available)
- Use of DNS SRV records
- Use of a "well-known" location
- Manual configuration of the relying party software

Without the certificate to hand, relying parties will need a certain amount of manual configuration information in their software.

4.2.1. Use of DNS SRV records

DNS SRV records are a facility for specifying the location of the server(s) for a specific protocol and domain [23]. For the WebDAV certificateURL and revocationURL, new DNS SRV symbolic names for the protocol need to be defined. We propose to use "webdavcerts" and "webdavcrls" respectively.

However this still does not tell the relying party which domain name to use in the SRV lookup, or whether the certificate issuer is using DNS SRV records. This can only be obtained by manual configuration of the relying party's software

4.2.2. Use of a "well-known" location

The WebDAV certificate and revocation URLs may be located at a "well-known" location constructed from the issuer's domain name. In this case the URL may be constructed by pre-pending the type of information to be retrieved ("webdavcerts." or "webdavcrls.") to the domain name to obtain the net_loc portion of the URL. The URL form of the "well-known" location therefore becomes:

```
webdavcerts.<issuers_domain_name>/
webdavcrls.<issuers_domain_name>/
```

However this still does not tell the relying party which domain name to use with the well-known location or whether the issuer is using well-known locations. The appropriate domain name to use can only be obtained by manual configuration of the relying party's software. If a certificate issuer does not wish to publish the certificate state information at a "well-known" location, it is recommended that the issuer should put a redirection page at the well-known location to inform relying parties precisely where the certificate state information can be found.

4.3. Using the WebDAV protocol to manage X.509 repositories

In order to create a new collection, WebDAV specifies the MKCOL method. The difference between this method and HTTP PUT or POST, is that the latter are allowed to overwrite existing content at the specified URL, whereas MKCOL will fail if there is any existing content at the specified URL. In the context of X.509, this ensures that a certificate issuer cannot unwittingly overwrite existing certificates when creating a new collection for a subject. This is an important concern when there are several certificate issuers for the same subject (either attribute certificate issuers and/or public key certificate issuers). It is important to ensure that no issuer deletes the certificates issued by another issuer.

In order to create a certificate or CRL or update an existing certificate in an existing collection, the PUT method is used. It is essential that every certificate and CRL has a unique name within a collection, so that updates can overwrite the same certificate and new certificates and CRLs cannot overwrite existing ones. The onus for creating the unique names is with the issuer. We have defined one algorithm for this is section 4.1 above.

In order to revoke a certificate, the HTTP DELETE command is used by the issuer. This removes the certificate and its properties from the WebDAV server. Simultaneously with this, if the revoked certificate contains the revocationURL access location, the issuer must use the HTTP PUT method to create a new CRL containing the serial number of the certificate that has just been revoked. The revocationDate and thisUpdate fields of the CRL should be set to the current time, and the nextUpdate field should be set to sometime after the certificate was due to expire and the date after which the CRL will be removed from the WebDAV store. This ensures that:

i) the CRL never needs to be reissued or updated, and

ii) relying parties know the minimum duration that the CRL will exist for on the web (should they wish to prove sometime after the certificate has expired that it was revoked prior to expiring).

4.4 Searching for particular certificates or CRLs

Document properties are specified in WebDAV as XML name/value pairs. Property names must be globally unique and are specified using XML namespaces [11]. Property values should be human readable (in any appropriate character set). Properties can be flagged as live or dead, where live means that the server validates that the values are syntactically correct, and the server may actually set the values of some system known properties, whereas dead means that the client is responsible for validating the syntax and semantics of the property values. For X.509 use, we initially intended to use live properties, set by the certificate issuer, to represent fields of the certificate. We anticipated this would allow easy searching of the certificate store to find certificates with certain properties, for example, find the AC of David Chadwick that has a *manager* role value. The WebDAV protocol does support the PROPFIND method, in which the properties of a resource can be retrieved, but it not possible to specify which property value you require. Only the property types can be specified. Consequently, if we perform a PROPFIND for the "Role" property, then the web server will return an XML encoded message containing all the ACs that contained a property named Role along with their values. Clearly this is not a viable solution. Work on the WebDAV searching and locating capability (DASL) started in 1998, but the work was never completed and the IETF closed the DASL working group some years later. The latest version of the WebDAV Searching and Locating protocol is very recent [17] and several implementations are said to exist, but we were unable to find a usable one. Consequently we have left the search feature for future work. Instead we have implemented a browsing capability in our user agents which allows a user to tree walk through a certificate store and select the certificate that he is looking for. The browse capability is fully scalable, user friendly and meets all the requirements of our use cases. See section 5 and figure 3 for more details.

WebDAV also provides other features that we do not need for X.509 use, such as the ability to copy and move web documents between servers, and the ability to write lock the certificate store when an issuer is performing updates. Consequently, these wont be discussed further.

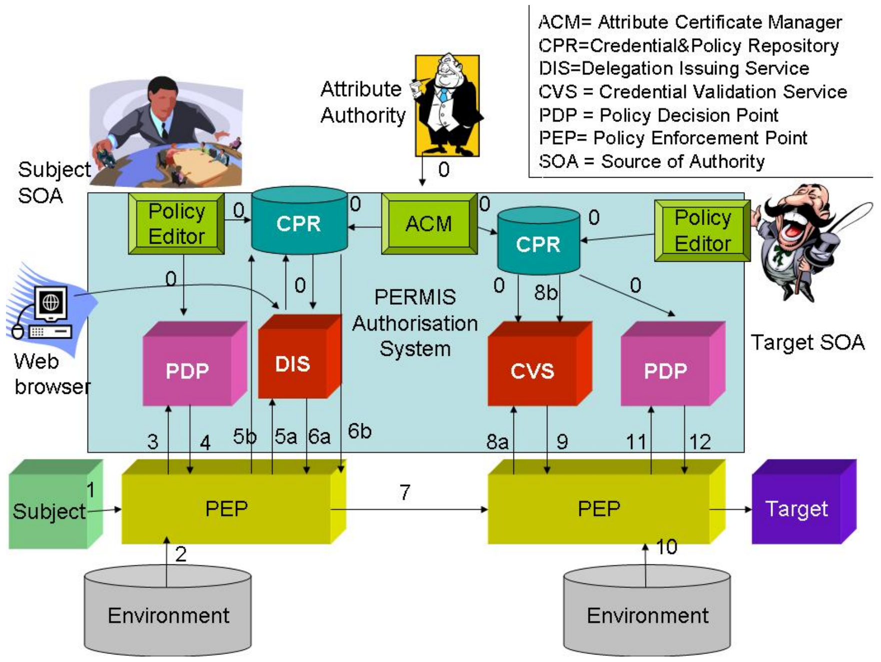
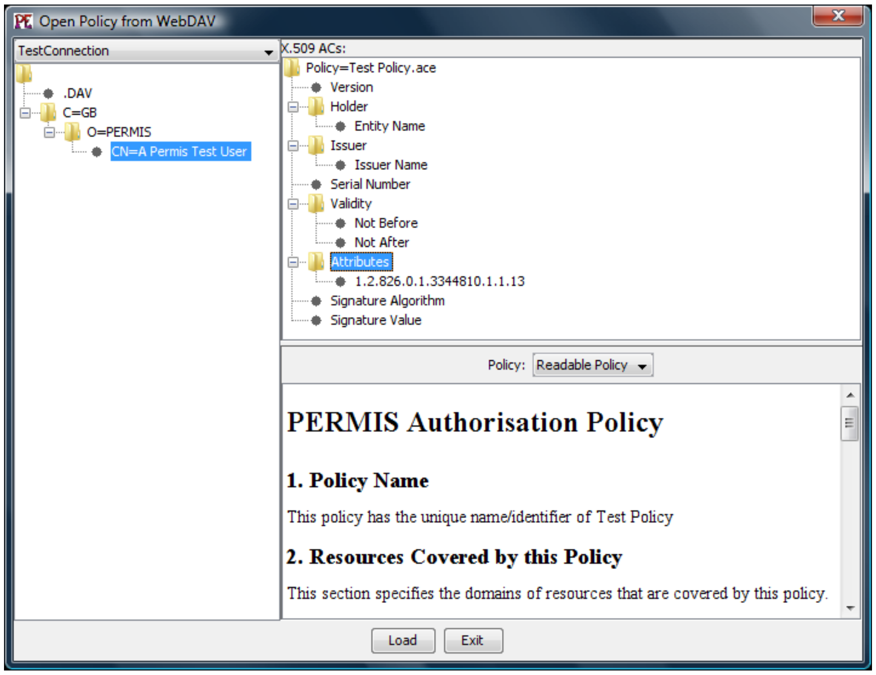

ACM= Attribute Certificate Manager
CPR=Credential&Policy Repository
DIS=Delegation Issuing Service
CVS = Credential Validation Service
PDP = Policy Decision Point
PEP= Policy Enforcement Point
SOA = Source of Authority
Attribute Authority
Subject SOA
Policy Editor
CPR
ACM
CPR
Policy Editor
Web browser
PDP
DIS
PERMIS Authorisation System
CVS
PDP
Target SOA
0
3
4
5b
5a
6a
6b
8a
9
11
12
8b
1
Subject
PEP
7
PEP
Target
2
10
Environment
Environment

Figure 2. The PERMIS Authorisation Infrastructure

5. Using WebDAV in PERMIS

PERMIS [12, 13] is an application independent privilege management infrastructure that comprises credential issuing and credential validation functionality as well as policy creation and policy decision making functionality. The components that are important to the current discussion are the Attribute Certificate Manager (ACM) and the Delegation Issuing Service (DIS), which both issue X.509 role ACs to holders, and the Credential Validation Service (CVS) which validates the issued role ACs (see Figure 2). In addition, the Policy Editor (PE) creates XML policies to control the behaviour of the DIS, CVS and PDP, and each policy can be embedded as a policy attribute in an X.509 AC and digitally signed by its issuer. The policy AC can then be stored in the issuer's collection of ACs, along with his role ACs. The primary difference between a role AC and a policy AC is the content of the attribute that is embedded in the AC, although the holder and issuer of a policy AC is always the same DN, whereas they are different in role ACs. In the original implementation of PERMIS, all the issued X.509 ACs were stored in LDAP directories, in the attributeCertificateAttribute of the entries of their holders. A major disadvantage of this, is that it is impossible to retrieve a single AC (policy or role) of a holder. Instead the entire set of role and policy ACs held by a holder has to retrieved as a set. With the addition of WebDAV repositories to the latest PERMIS implementation it is now possible to store and retrieve individual ACs in which each AC is uniquely identified as described in section 4.1.

Figure 3 shows a screen shot from the PERMIS Policy Editor where the user is browsing through the WebDAV store to locate a policy AC in order to retrieve and edit it. The left hand window shows the hierarchical tree structure of the WebDAV repository, and the user can open and close the various directories (collections) by clicking on the + and – signs. The top right hand window displays each of the certificates in a collection (in this case it is displaying a policy AC of the user whose DN is C=GB/O=PERMIS/CN=A Permis Test User). Individual certificates can be opened or closed and the various fields expanded or contracted by clicking on the + or - sign. The bottom right hand window displays the value of the field that is highlighted in the top right hand window. In figure 3 it is the value of the attributes field that is being displayed. A novel feature of the PERMIS Policy Editor, is that the PMI XML Policy Attribute value, which contains the PERMIS Authorisation Policy in XML, can be displayed as either raw XML or as a readable policy in natural language. The latter is achieved by passing the XML through a stylesheet (XLST), as shown in Figure 3.



PERMIS Authorisation Policy

1. Policy Name

This policy has the unique name/identifier of Test Policy

2. Resources Covered by this Policy

This section specifies the domains of resources that are covered by this policy.

Figure 3. Retrieving a Policy AC using WebDAV

Modified implementations of the ACM and PE algorithmically create the filenames in which the ACs will be stored, according to the rules in section 4.1. The new certificate and revocation URL extensions specified in section 3.1 are also automatically inserted into role ACs created by an enhanced ACM, whenever the user chooses WebDAV as the repository. ACM configuration options allow the user to determine the DNS name and port to use when these extensions are to be created. Currently we do not store CRLs of length 1 at the revocation URL, and will accept any page contents there, plus the deletion of the certificate from the certificate URL, as proof that the certificate has been revoked. In future we plan to add the ability to create CRLs of length 1 and store these at the revocation URLs, as cryptographic proof of revocation.

The PERMIS CVS/PDP have been modified to periodically check if the user's ACs have been revoked or not. In pull mode, at construction time, the Policy Enforcement Point (PEP) passes to the PERMIS CVS the details of the repositories which are to be used to retrieve the users ACs when each user requests access to the protected resources. When a user issues an access request, the PEP calls the PERMIS CVS getCreds method, and this fetches the user's ACs from the configured repositories. Processing now continues as in push mode. In push mode, the user's ACs are passed to the CVS when the user makes an access request. The ACs have their signatures checked, and if valid, are parsed to see if they are revocable or not. X.509 ACs may be short lived and have a noRevAvail extension [4] in them, in which case no revocation information will be made available. If however an AC is revocable, the new certificate and revocation URL extensions, if present, are extracted from the certificate. The PERMIS CVS can be configured to contact the certificate URL first (default) or the revocation URL first. The appropriate URL is contacted, and the flow proceeds as in the appropriate half of figure 1. If the AC has not been revoked, the valid attributes (according to the CVS's policy) are extracted, merged with the valid ones from the user's other ACs, and returned to the PEP. The PEP then calls the decision method of the PERMIS PDP, passing it the user's valid attributes and the user's access request. If the request can be granted according to the access control policy, the PDP then calls the callback method assessRisk, passing it the user's access request. This design allows the PEP to decided if the user's ACs should still be considered valid or not and to choose the frequency at which the user's AC should be revalidated, depending upon the risk associated with the particular user request. How assessRisk is implemented is an application dependent issue, but it must return one of three responses: OK, revalidate or terminate user session. OK will cause granted to be returned to the PEP. Revalidate will return an exception to the PEP, causing it to call getCreds again. Terminate will also return an exception to the PEP, but this time will cause the user's session to be terminated and no further requests be made. We have built two exemplar assessRisk methods as proof of concept: simpleCounter and simplePeriodic. The former counts the number of granted responses that are being returned to the user, before returning revalidate to the PDP after a configuration number N is reached. The latter records the time of the first call to assessRisk and after a configuration number N of seconds has passed, it returns revalidate to the PDP. Both of these cause a user's certificates to be revalidated midway through a session.

The performance of WebDAV instant revocation was measured. We measured the time taken for getCreds to return the valid attributes of a user, in both push and pull mode, with and without revocation being enabled. The ACs were stored in a remote WebDAV server, connected to the PERMIS CVS via a high speed LAN, and accessed via HTTP. Each set of performance tests were carried out 100 times and the average and standard deviation performance figures were calculated. Spurious results (average + (3 x standard deviation)) were removed from the averages. The spurious results can be due to Java garbage collection operating at random intervals or different loads on the network. We found that switching on revocation checking added 35% to the performance of pull mode when the user had 1 AC, and 200% to the performance of push mode. The latter is to be expected, since in push mode without revocation checking, getCreds only needs to validate against the local policy the AC that is given to it. When revocation checking is switched on, getCreds has in addition to pull the AC from the remote WebDAV server and then compare it to the pushed AC.

5. Discussion and Conclusions

The OASIS SAML specification has the concept of an artifact that can be obtained from a remote server using an ArtifactResolve message to request the artifact and an ArtifactResponse message to return it [16]. The artifact could be a SAML Attribute Assertion, which is similar in concept to an X.509 attribute certificate, except that it is designed to be short lived and never revoked. The SAML artifact messages are carried over HTTP, therefore will pass transparently through firewalls in the same way as our WebDAV protocol. But there the similarity between the two schemes ends. There are fundamental conceptual differences between the artifact concept in SAML and the certificate publishing and revocation concepts in this paper. Firstly a SAML artifact can only be used once. The SAML specification states “The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent request with the same artifact by any requester results in an empty response” [16]. Secondly SAML artifacts are meant to be short lived, quote “The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and return it in an <ArtifactResolve> message to the issuer” [16]. In our design, certificates are assumed to be as long lived as required, and used as many times as needed by as many different recipients as the subject desires. We thus believe our system is more flexible and requires less processing resources on the part of both the issuer and relying party.

Our scheme has some similarities with the Netscape Navigator certificate revocation mechanism [18]. In the Netscape scheme, an X.509 extension **netscape-revocation-url** is used to refer to a web location where information about a certificate’s status can be found. The actual URL that a relying party should use comprises this extension concatenated with the certificate’s serial number (encoded as ASCII hexadecimal digits) e.g. https://www.certs-r-us.com/cgi-bin/check-rev.cgi?02a56c. The document that is retrieved from this URL contains a single ASCII digit, '1' if the certificate is not currently valid, and '0' if it is currently valid.

The differences with our scheme are immediately obvious. The revocation URL document always exists, and its content is not digitally signed by the issuer. In comparison our certificate only exists whilst it is has not been revoked and it is a standard certificate digitally signed by the issuer. Optionally our CRL only exists if the certificate has been revoked, and it is a standard CRL containing a single entry signed by the issuer. Our scheme also optionally allows a relying party to find out all the certificates that have been revoked by a particular issuer, by retrieving the WebDAV CRL collection (cn=CRLs/) under the issuer's WebDAV entry.

The one security weakness in our scheme is that it is vulnerable to denial of service attacks, in that if the WebDAV server is not available, relying parties will not be able to tell if a certificate has been revoked or not. But other schemes such as OCSP servers and published CRLs are also equally vulnerable to DOS attacks, and so our scheme is no different in this respect. However, published CRLs do have one advantage in that an old CRL retrieved sometime in the past might still be available to the relying party, and this is better than having no CRL at all, since it does contain some revoked certificates. If this is seen to be a significant benefit, then our optional CRL publishing mechanism is equivalent to it, in that a CRL collection can be downloaded at any time, just like a conventional CRL. The CRL collection can also be replicated and cached to improve availability. Other well known DOS protection methods, such as overcapacity and server clustering will have to be employed in order to be fully protected against DOS and DDOS attacks, but these are fairly standard techniques that are employed by DNS servers and commercial web sites such as Amazon. Consequently we do not believe that DOS attacks are any more of a significant security threat to our scheme than to existing ones.

The one performance weakness of our scheme is that the latency of certificate validation increases due to network overheads, as compared to that of traditional CRLs, but not to that of OCSP servers. This may be a critical factor to some relying parties such as central servers which process thousands of certificates per second. Central servers benefit from downloading traditional CRLs when they are not busy and storing the results in a local cache for fast lookups when they are validating certificates. The disadvantage of this approach is that the central server still has a vulnerability period between the date and time the latest CRL was published and now, during which all recently revoked certificates will not yet have been incorporated into the latest CRL. This provides an attack window for the holders of the recently revoked certificates. If on the other hand the certificate issuer supports our WebDAV certificate publishing scheme alongside its traditional CRL publishing (or our WebDAV single CRLs scheme) then the central server may check the current status of certificates. It can use the WebDAV GET operation for the certificates of high risk or high value transactions, whilst continuing to use its CRL cache for the certificates of lower risk transactions. In this way the latency penalty of WebDAV lookups is only incurred for a few high risk certificate validations.

In contrast, low throughput relying parties which only process a certificate every few minutes, and where the subject base is very large (and hence so are the traditional CRLs) will benefit greatly from our WebDAV approach of contacting the certificate URL at the time of each certificate validation. This is not too dissimilar from contacting an OCSP server, in terms of processing overheads (HTTPS overheads vs. signed OCSP responses) and latency. The advantages of our WebDAV scheme are that HTTPS and web servers are more ubiquitous than OCSP servers, and where OCSP servers compute their responses on published CRLs and therefore are out of date, WebDAV responses are based on the latest up to date certificate status information.

Finally, comparing our single CRL per certificate scheme against traditional CRLs, we see that there is a trade off between the currency of the revocation information and the overhead of signature creation and validation. A traditional CRL only requires one signature creation per revocation period and one signature validation per relying party per period, whereas our scheme requires one signature creation per revoked certificate and one signature validation per relying party per revocation. The more certificates are revoked per revocation period, the more the processing overhead increases, but so does the risk. Consequently the increased processing overhead has to be offset against the risk reduction of instant revocation, but this tradeoff can only be determined on a per-application basis.

To conclude, we have described a new way of publishing and revoking X.509 certificates based on the ubiquitous WebDAV protocol that has a number of distinct advantages over current schemes. We have implemented this in our PERMIS privilege management infrastructure and performed initial validation testing. It has recently been released as open source software along with the existing PERMIS source code, and we will then expect to obtain operational experiences from users.

References

1. D.W.Chadwick. “Deficiencies in LDAP when used to support a Public Key Infrastructure”, Communications of the ACM, March 2003/Vol 46, No. 3 pp. 99-104.
2. Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen. “HTTP Extensions for Distributed Authoring – WEBDAV”. RFC 2518, February 1999
3. Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (1999). “X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol – OCSP”, RFC 2560
4. ITU-T (2005). “The Directory: Public-key and attribute certificate frameworks” ISO 9594-8 (2005) /ITU-T Rec. X.509
5. Dierks, T., Allen, C. "The TLS Protocol Version 1.0", RFC 2246, January 1999
6. S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”. RFC3820, June 2004.
7. OASIS. “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 March 2005
8. Alfieri, R., Cecchini, R., Ciaschini, V., Dell'Agnello, L., Frohner, A., Lorentey, K., Spataro, F., “From gridmap-file to VOMS: managing authorization in a Grid environment”. Future Generation Computer Systems. Vol. 21, no. 4, pp. 549-558. Apr. 2005
9. Wahl, M., Coulbeck, A., Howes, T., Kille, S. “Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions”. RFC 2252. December 1997.
10. Guida, R.; Stahl, R.; Bunt, T.; Secrest, G.; Moorcones, J. “Deploying and using public key technology: lessons learned in real life”. IEEE Security and Privacy, Volume: 2 Issue: 4, July-Aug. 2004. Page(s): 67- 71
11. T. Bray, D. Hollander, A. Layman, "Namespaces in XML". World Wide Web Consortium Recommendation REC-xml-names-19990114. See http://www.w3.org/TR/1999/REC-xml-names-19990114/

12. D.W.Chadwick, A. Otenko, E.Ball. "Role-based access control with X.509 attribute certificates", IEEE Internet Computing, March-April 2003, pp. 62-69
13. David Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, Tuan Anh Nguyen. "Building a Modular Authorization Infrastructure", UK All Hands Meeting, Nottingham, Sept 2006. Available from http://www.allhands.org.uk/2006/proceedings/papers/677.pdf
14. Housley, R., Ford, W., Polk, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280, April 2002
15. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. "Hypertext Transfer Protocol -- HTTP/1.1.". RFC 2616, June 1999
16. OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005
17. J. Reschke et al. "Web Distributed Authoring and Versioning (WebDAV) SEARCH". <draft-reschke-webdav-search-14>, 15 Nov 2007.
18. Netscape Certificate Extensions, Navigator 3.0 Version, Available from http://wp.netscape.com/eng/security/cert-exts.html
19. See http://roy.gbiv.com/pubs/dissertation/top.htm
20. ISO-ITU-T "Final Proposed Draft Amendment 2 to ITU Rec. X.509 (2005) | ISO/IEC 9594-8: 2005", Geneva, Sept 2007
21. P. Kocher. "On Certificate Revocation and Validation." Proc. of Financial Cryptography 1998, pp. 172-177.
22. S. Micali. "Efficient Certificate Revocation". Technical Memo MIT/LCS/TM-542b, 1996
23. Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
24. K. Zeilenga. "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names". RFC 4514, June 2006.
25. P. Gutmann, Ed. "Internet X.509 Public Key Infrastructure Operational Protocols: Certificate Store Access via HTTP". RFC 4387. February 2006.