Modifying LDAP to Support X.509-basedPKIs

D.W.Chadwick, E. Ball, M.V. Sahalayev, University of Salford, Salford, M5 4WT
Email: D.W.Chadwick@salford.ac.uk M.Sahalayev@pgr.salford.ac.uk
E.Ball@salford.ac.uk

Abstract

One of the impediments to a successful roll out of X.509-based public key infrastructures (PKIs), is that LDAP directories do not fully support PKIs. In particular, it is not possible to search for X.509 attributes (certificates or CRLs) that match user defined criteria. This paper describes the various approaches that have been suggested for enabling users to search for X.509 attributes, namely component matching and attribute extraction. The implementation of attribute extraction in the OpenLDAP product is then described.

Keywords

X.509, public key certificates, attribute certificates, CRLs, PKI, LDAP, Search

Introduction

This paper initially describes the problem space concerning LDAP being used to support X.509-based PKIs. It then describes two alternate solutions to the problem space that are being defined by the IETF PKIX working group. These solutions are called Component Matching and Attribute Extraction. We describe these solutions in detail, and then describe how we attempted to implement both approaches in the open source OpenLDAP product. We describe the problems encountered in implementing the Component Matching solution, and why in the end we settled for the Attribute Extraction approach. Our implementation, called the X.509 Parsing Server (XPS) is then described along with some of its main configuration options. We conclude by comparing the two approaches, and report on how the IETF has decided to proceed with the standardisation of the two approaches.

The Problem

The Lightweight Directory Access Protocol (LDAP) [1] was originally conceived as a simplified version of the X.500 Directory Access Protocol (DAP) [4]. X.500 was designed to be a general purpose repository for storing (mainly communications related) attributes of entities, including security attributes such as public key certificates (PKCs) and certificate revocation lists (CRLs). However, during the simplification process of designing LDAP, many features of the DAP were lost, including the ability to fully support public key infrastructures (PKIs). In particular, whilst X.500 DAP could support the searching for and retrieval of public key certificates based on their contents, LDAP could not. For example, a Search such as find the encryption public key (i.e. keyUsage extension is set to keyEncipherment) for the user whose email address (i.e. the SubjectAltName rfc822Name) is D.W.Chadwick@salford.ac.uk, cannot be supported by LDAP. LDAP is unable to perform this Search because it does not know how to carry assertion values for components of fields in the protocol (in this case the protocol field is an X.509 certificate, and a component is the SubjectAltName of the certificate). Worse still, the original LDAPv2 did not have a correct way of encoding public key certificates or CRLs for storage or retrieval. So regardless of the fact that an LDAP directory might be built on top of a relational database that

does have good search facilities, LDAP is unable to utilise these as it is unable to transfer the user's search requests to the back end database.

The original LDAP protocol was based on encoding fields of the DAP ASN.1 [9] protocol as ASCII strings [2] for carrying over TCP/IP. However, only selected fields from the DAP were chosen for encoding in LDAP. (This is why LDAP is called Lightweight.). Whilst a string encoding of certificates was defined, it was erroneous (see [11] for more details). DAP on the other hand uses the Basic Encoding Rules (BER) [10] for encoding its protocol elements for transfer, so it is not necessary to specify how any particular field is encoded. BER automatically specifies this for all fields Thus X.500 DAP automatically knows how to encode and transfer public key certificates and their assertion values that are defined in X.509 [8], since they are specified in ASN.1. X.500 DAP inherently knows the structure, contents and transfer encodings of all current and future PKI attributes and extensions that are or may be defined in X.509 (as long as they continue to be defined in ASN.1).

Additions to version 3 of LDAP [7] rectified many of the shortcomings of the original LDAP, and allowed LDAP servers to correctly store and retrieve X.509 attributes, but searching for them was still impossible. This is because the protocol fields, i.e. the X.509 attributes, are simply transferred and stored as binary blobs by LDAPv3, with the server having no knowledge about their structure and contents. Further, no string encodings were specified for the assertion values, so that it still remains impossible to Search for X.509 attributes containing specific fields. A fuller description of the limitations of LDAP when used to support PKIs is given in [11].

Since LDAP has become the primary way of accessing X.500 based directories, and is supported by all the major vendors, e.g. Microsoft, Netscape, IBM, Novell etc., this lack of support for PKIs is a serious impediment to the roll out of X.509-based PKIs. Indeed, the US Federal PKI pilot reported that the lack of a fully functional inter-operating X.500 directory was one of the inhibitors of PKI rollout [12]. Similarly, the final report [13] of the recently completed EC PKI Challenge project (see http://www.eema.org/pki-challenge) states that PKIs are being held back by the lack of support for them in related technologies such as LDAP/X.500 directories and S/MIME clients. Consequently, the authors set out to rectify the primary deficiency in LDAP, i.e. its inability to Search for X.509 related attributes, by firstly defining a standard LDAP encoding for the protocol fields containing X.509 attributes and their assertion values (i.e. a standard LDAP schema) and secondly implementing this schema and its associated matching rules in the OpenLDAP product (see http://www.openldap.org). The schema will allow LDAP servers to know the structure and contents of the X.509 attributes, and will allow assertion values in Search filters to identify specific fields in the attributes. This paper describes the results of this research.

String Matching

The first approach to solving the problem was to follow the traditional LDAP route. This takes the existing X.500 matching rules and defines how all the ASN.1 type fields of the X.509 attribute value assertions can be carried as ASCII strings in the LDAP protocol [14]. For example, the notBefore Validity time of a PKC might be transferred in a certificate assertion value as yyyymmddhhmmssZ, and might follow the third $ separator character (where preceding $ separated strings might hold the certificate serial number, the issuer's name, and the subject

key identifier). Where there is a CHOICE in the equivalent ASN.1 type, for example as in the GeneralNames construct, keywords are defined and these precede the value, so that it is obvious which option has been chosen, e.g. dns myorg.com or ip 123.4.5.6. An extension to the LDAP standard schema would define which ASN.1 type fields of assertion values are to be encoded for transmission in the LDAP protocol (part of the simplification of LDAP is that not every field in the DAP assertion need be transferred). The individual ASN.1 fields of an X.509 attribute value assertion can then be passed as ASCII strings in matching rule assertion values in the LDAP protocol, by LDAP clients wishing to search for X.509 attributes containing particular field values.

When an X.509 attribute e.g. a PKC, is passed to an LDAP server, it should parse the attribute, pull out the individual ASN.1 values, and store these in its indexes, in the same way that it might store and index email address attribute values. The LDAP server will need to have a lookup table, or equivalent, that knows the X.509 matching rules and the order of occurrence of each field in the assertion values presented by the user, so that it can compare them to the correct indexes that it is storing.

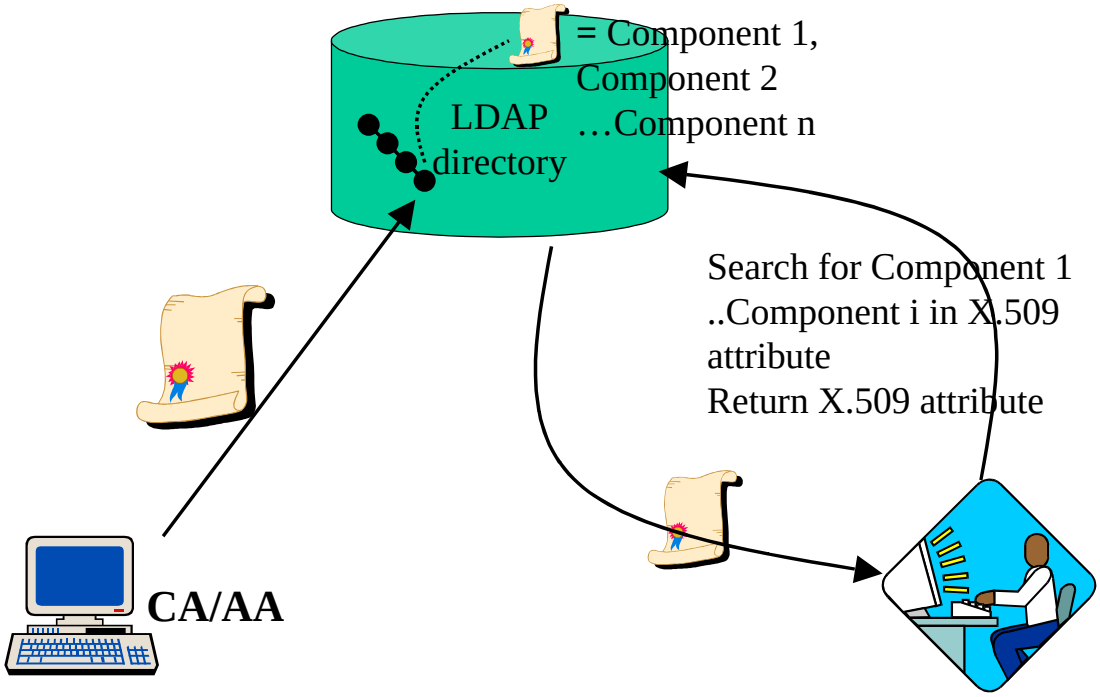
Whilst this approach is relatively easy to understand, and follows the traditional LDAP method, it lacks the ability to automatically cater for new ASN.1 type fields, new certificate extensions and new matching rules. The reason is that the string encoding of each field in the assertion values, and any new keywords, are specific to each X.509 syntax. As new X.509 attributes, and new certificate and CRL extensions are defined along with their new matching rules, new LDAP string encodings (and possibly new keywords) will need to be defined for each new assertion value. Consequently new Internet standards will need to be written (or existing ones updated) each time this happens. It would be better if a more automated approach could be defined. Component Matching is such an automated approach, and this is described below.

Component Matching

An improvement on the original string matching approach was first suggested by Legg, who produced an Internet Draft called Component Matching [15]. This took a more fundamental approach to the problem. Instead of defining LDAP string encodings for complex ASN.1 types, it defined UTF8 string encodings at the lowest level of ASN.1 granularity - the built-in types. Thus string encodings were defined for each ASN.1 built-in type such as INTEGER, BOOLEAN, OCTET STRING, SEQUENCE, CHOICE etc. The encoding of more complex ASN.1 types would then be built up automatically from their component parts. (This part of the document was subsequently released as a separate Internet Draft called the Generic String Encoding Rules for ASN.1 Types [16].) Legg also defined how string assertions could be specified about the components of ASN.1 values, through a ComponentAssertion construct. This defines how each component can be referenced, as well as specifying the assertion value and the matching rule to be used when performing the comparison. Referencing components can be via an identifier, or by its position, for example, a user can assert if the n^{th} value in a SEQUENCE OF INTEGER is greater than or equal to some assertion value.

Working with Legg, we produced an Internet Draft [17] describing how the generic string encoding rules can be applied to the assertion values for the X.509 attribute matching rules defined in [14]. We then started to implement this using the open source code of OpenLDAP.

Component Matching



This implementation proved difficult for a number of reasons. Firstly it meant that all existing LDAP clients would need to be modified to support LDAP extensible matching rules, and the generic string encoding rules, so that individual components within an X.509 attribute could be specified by the client and passed to the server. Secondly all existing servers would need to be modified to support the new matching rules, and the generic string encoding rules for assertion values. The matching process is complex. Incoming X.509 attributes have to be parsed, their individual fields have to be located within the DER byte stream, and pointers to the bytes stored in the indexes of the LDAP database, along with their corresponding keywords taken from the definition of their assertion values. When a client sends an assertion value (as an ASCII string), the server has to match the presented keyword against the correct index, then use the associated pointer to pull the value out of the ASN.1 DER byte stream, covert it into an ASCII string, and then compare this with the presented ASCII value using the appropriate matching rule (e.g. equality match or substring match). This continual conversion from ASCII strings to DER bytes and back again is not very efficient, but is unavoidable since LDAP is based on ASCII strings and X.509 attributes are based on DER bytes.

Attribute Extraction

Other researchers in Germany were also working on the same problem of how to search for X.509 attributes in LDAP directories. Klasen and Gietz were dubious about the approach of Chadwick and Legg. Although it was elegant and easily extensible, they did not believe that all the LDAP suppliers would implement this mechanism in their products. In fact Legg had already implemented component matching in the LDAP server product of his company whilst the Internet Draft was being refined. But this had been made much easier by the fact that their back

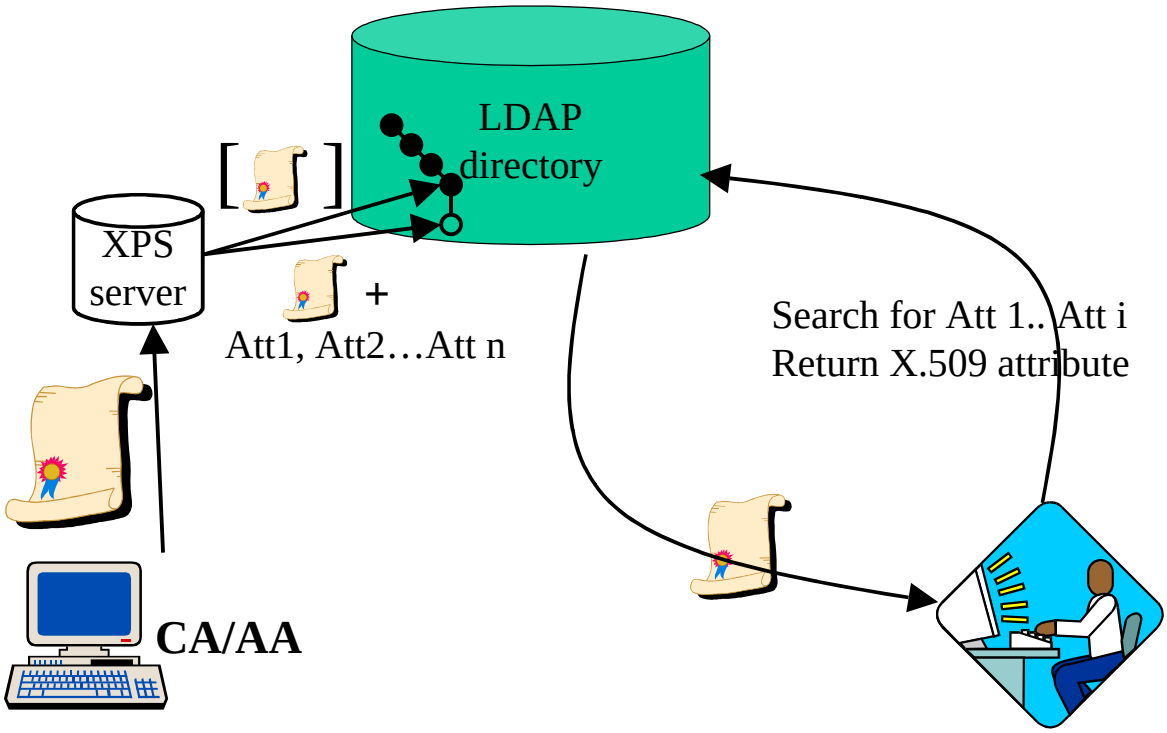
end database was based on X.500 rather than on LDAP, so every component of every attribute was stored and processed in its DER format by the database. This meant that no conversion had to take place when the server received X.509 attributes for storage, and when clients presented LDAP string assertion values, these were converted in their DER equivalents, and passed to the back end database.

Klasen and Gietz proposed an alternative solution, based on the work-around that PKI administrators are taking today. This is to extract a component from an X.509 attribute and store it as a simple, searchable attribute, in the same entry in the LDAP database. For example, in order to support S/MIME, the PKI administrator extracts the user's email address from the Subject Alt Name field of the public key certificate, and stores this in an email address attribute in the user's LDAP entry. The sending user then searches the LDAP directory for the needed email address, and asks the LDAP server to return the public key certificate attribute from the entry matching the email address. The assumption is that no-one will have tampered with the LDAP directory, and that the email address attribute and the subject alt name field in the certificate will be identical.

Klasen and Gietz extended this model, by defining a set of ~30 attributes for public key certificates, where each attribute holds a field from a public key certificate [18]. For example, the x509keyUsage attribute holds the key usage field, whilst the x509authorityKeyIdentifier attribute holds the authority key identifier field (this is the identifier of the public key used to sign this certificate or CRL). If a front end X.509 attribute Parsing Server (XPS) is used to automatically parse X.509 attributes and extract the fields into these attributes, it means that existing LDAP servers will not need to be modified. No new matching rules or assertion values need to be defined, since the 30 new attributes use existing matching rules and assertion values. The simplicity of this approach is that most existing (i.e. configurable) LDAP clients will not need to be modified either. They can use their existing assertion value creation code, and simply need to be configured with new attribute types to be used with it.

We decided to alter our modifications to OpenLDAP by adopting Klasen and Gietz's approach. We added XPS functionality to OpenLDAP, so that OpenLDAP can act as either a stand alone XPS server, or a combined XPS and LDAP server. We also wrote two new Internet Drafts to compliment [18], one that defines new LDAP attributes for CRLs [19], the other for attribute certificates [20].

Attribute Extraction



XPS Implementation

The Directory Information Tree (DIT) Design

X.500 and LDAP use a hierarchical (tree) data structure to hold the directory information. There is a single root node to the tree, and this can have any number of subordinate nodes, and the nodes can be nested to any depth. A node is termed a directory entry, and it holds any number of attributes that describe the entry e.g. a telephone number, a name, an address etc. The conventional X.500 design is that X.509 certificates are typically held in the entry of the user to whom they belong, as a multi-valued attribute e.g. a certificate for digital signature verification, and a certificate for encryption. In the XPS design, each certificate is held in a separate subordinate entry of the existing user's entry. This certificate entry holds the set of attributes defined in [18] for public key certificates or in [20] for attribute certificates, along with the complete unmodified certificate. In addition, the XPS server has a configuration option that allows the certificates to be still held in the user's entry as a multi-valued attribute. This is useful for example if a client wishes to retrieve all the attributes of a particular user. The subordinate entries allow LDAP clients to search for fields of the certificate, by using the corresponding attributes and their assertion values. When the attribute(s) is(are) matched, the corresponding certificate in the same entry can be retrieved. The subordinate entries can be named in one of three ways, by using:

i) a combination of the certificate serial number and issuer name (which are guaranteed to be unique), as a multi-valued RDN, or
ii) the certificate serial number and issuer name concatenated into a single string (to be used by those LDAP servers e.g. MS Active Directory) that do not support the standard LDAP feature of multi-valued RDNs, or
iii) the serial number on its own (this is only suitable for LDAP servers that will only ever store certificates from a single CA)

The OpenLDAP XPS server has a configuration option allowing the administrator to choose the naming scheme most suitable for him. Another XPS configuration option allows the administrator to choose to store the certificate entries either subordinate to the user's entry, or in a completely separate subtree whose root node is pointed to by a configuration option.

CRLs are stored in a slightly different way to certificates. They are typically either stored in the CA's entry, or in any entry that is pointed to by the CRL Distribution Point extension in a certificate. When using the XPS server, each CRL is stored in a subordinate entry of the previous entry, but this time each CRL may create a subtree of entries. The immediately subordinate entry holds all the attributes defined in [19], except for the revocation time and CRL entry extensions (such as revocation reason). The latter are stored in subordinate entries of the former, there being one subordinate entry per revoked certificate. The reason that a subtree of entries had to be created, rather than a single subordinate entry, is that LDAP attributes are a set of values, rather than a sequence of values. Therefore it is not possible to relate multi-valued attributes together in any meaningful way. Since each revoked certificate has its own revocation time and set of extensions, it would not make sense to store a set of revocation times and certificate serial numbers in two separate attributes of the subordinate entry, and expect a client to be able to determine which certificate was revoked at which time. By placing each revoked certificate in a separate child entry, it is possible to search for certificates revoked at a certain time, or with a certain revocation reason etc.

Configuration Options

OpenLDAP has been modified so that it can be an X.509 attribute Parsing Server (XPS) and operate in either standalone mode, or as a front end to an existing LDAP server. Those sites that are already using OpenLDAP as their LDAP server, will typically operate in standalone mode, by upgrading to the appropriate OpenLDAP release. Those sites that use any other make of LDAP server, can simply place the OpenLDAP XPS server as a front end to their existing LDAP server as in Figure 2. The Certification Authority is then configured to talk to XPS rather than its existing LDAP server. XPS parses the incoming request, modifies it, and passes the resulting requests to the existing LDAP server

Discussion and Conclusion

We are currently in the testing phase of our XPS server, and the plan is to release this as part of the OpenLDAP release in 4th quarter 2003. We therefore currently do not have any experimental data about the performance, scalability or usability of the attribute extraction approach. We anticipate that these results will be available about a year from now.

The component matching approach is a more elegant solution, since the X.509 attributes are only stored once in the LDAP directory, and matching is performed directly on their contents. Once Component Matching is implemented in LDAP servers, it will be easy to match on new ASN.1 types by simply adding them to the configuration files. Attribute extraction on the other hand, doubles or trebles the storage requirements of the LDAP server, and matching is performed on the attributes associated with the X.509 attribute, rather than on the attribute itself.

However, attribute extraction is somewhat easier to implement than component matching, especially on the client side. Furthermore, by using a stand alone XPS server, existing LDAP servers do not need to be modified. Given the past reluctance of LDAP vendors to have solved the problem of searching for PKI related attributes during the last 5 years, we have some doubts if they are likely to implement component matching anytime soon.

The IETF has decided that component matching should become the standardised way of searching for X.509 attributes and the current Internet Drafts {15,16,17] will be published as standard track RFCs once they are fully stable. This should encourage LDAP suppliers to implement this approach and should provide a long term elegant solution. The pragmatic attribute extraction approach as documented in [18,19,20] will become informational RFCs.

We therefore conclude that attribute extraction is the best pragmatic solution to quickly solve the current problem of PKI users who need to search for X.509 attributes, but that component matching should become the long term elegant method of solving the problem.

References

[1] W. Yeong, T. Howes, S. Kille. “X.500 Lightweight Directory Access Protocol.” RFC 1487, July 1993.
[2] Howes, T., Kille, S., Yeong, W., Robbins, C. “The String Representation of Standard Attribute Syntaxes”. RFC 1778, March 1995
[3] Wahl, M., Coulbeck, A., Howes, T., Kille, S. “Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions”. RFC 2252. December 1997.
[4] ISO 9594-3/ITU-T Rec. X.511(1988) The Directory: Abstract Service Definition
[5] Wahl, M., Howes, T., Kille, S. “Lightweight Directory Access Protocol (v3)”, RFC 2251, Dec. 1997
[6] ITU, "Information Technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, March 2000.
[7] Wahl, M., Howes, T., Kille, S. “Lightweight Directory Access Protocol (v3)”, RFC 2251, Dec. 1997
[8] ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Authentication Framework
[9] ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation
[10] ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1,2,3:1998 Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
[11] 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.

[12] Electronic Messaging Association Challenge 2000 “Report of Federal Bridge Certification Authority Initiative and Demonstration” DRAFT 101500, August 2000 (Available from http://csrc.nist.gov/pki/documents/emareport_20001015.pdf)
[13] EEMA/ECAF PKI Challenge Project “D8.1 – Final Report”, IST-2000-25012, March 2003
[14] D.W.Chadwick.“Internet X.509 Public Key Infrastructure - Additional LDAP Schema for PKIs and PMIs”. <draft-pkix-ldap-schema-00.txt>, July 2000
[15] S. Legg. “LDAP & X.500 Component Matching Rules”, <draft-legg-ldapext-component-matching-08.txt>, April 2002
[16] Legg, S., “Generic String Encoding Rules for ASN.1 Types”, <draft-legg-ldap-gser-00.txt>, March 2002.
[17] Chadwick, D.W., Legg, S.“Internet X.509 Public Key Infrastructure - LDAP Schema and Syntaxes for PKIs and PMIs”, <draft-pkix-ldap-schema-02.txt>, November 2001.
[18] Klasen, N., Gietz, P. "An LDAPv3 Schema for X.509 Certificates",<draft-klasen-ldap-x509certificate-schema-00.txt>, February, 2002
[19] Chadwick, D.W., Sahalayev, M. V. "Internet X.509 Public Key Infrastructure LDAP Schema for X.509 CRLs", <draft-ietf-pkix-ldap-crl-schema-00.txt>, February 2003
[20] Chadwick, D.W., Sahalayev, M. V. "Internet X.509 Public Key Infrastructure LDAP Schema for X.509 Attribute Certificates", <draft-ietf-sahalayev-pkix-ldap-ac-schema-00.txt>, February 2003