By David Chadwick

DEFICIENCIES IN LDAP WHEN USED TO SUPPORT PKI

Problems arise when a protocol initially developed to simplify access to a distributed directory failed to take into account all the uses the directory was originally intended for.

The lightweight directory access protocol (LDAP) is the Internet standard way of accessing directory services that conform to the X.500 data model. It is very widely supported by all the leading software vendors and is part of Windows 2000 Active Directory. There are two versions of LDAPv2, the original lightweight variation of the X.500 Directory Access Protocol (DAP); and LDAPv3, the heavyweight version [10]. While the DAP was designed from its inception to support public-key infrastructures (PKIs), being part of the same X.500 family of standards as X.509, LDAP was not designed with this support in mind. LDAP has, however, become the predominant protocol in support of PKIs accessing directory services for certificates and certificate revocation lists (CRLs), but because of its lineage, it has some deficiencies.

The deficiencies in both the LDAPv2 and v3 protocols are described here, along with the solutions that have been and are being standardized within the IETF to rectify them. The deficiencies are documented initially for a centralized directory service, in which a single standalone LDAP server is used to support a single PKI, and secondly for a distributed directory service, in which there are many LDAP servers that must cooperate in order to support a network of interconnected PKIs.

Centralized LDAP Deficiencies

Certificate and CRL Retrieval. The first attempt to support certificate retrieval using LDAPv2 was documented in [6]. Unfortunately, it did not work because the LDAP native encoding of a certificate converted it from its ASN.1 BER (basic encoding rules) type, length, value binary encoding, into a simple LDAP ASCII string, and this is an irreversible process for distinguished names. It is a many-to-one encoding as relative distinguished names (RDNs) can be of type teletex, printable, or universal string. Consequently when the LDAP client tries to reconstitute the ASN.1 binary from the ASCII string (in order to validate the signature on the certificate) it proves impossible without producing possibly hundreds of different ASN.1 BER encodings. Checking the signature against each is not practical. A more sophisticated LDAP native encoding for RDNs, for example by encoding the type as well as the value, could have solved the problem for certificates, but at the expense of complicating all LDAP operations, since RDNs are a predominant data type. The most obvious solution is to deprecate the LDAP native encoding in [6] and to transfer the certificate or CRL in its binary format so that the ASN.1 BER encoding is preserved. This solution was adopted for LDAPv2 and is documented in [1]. The solution works in most cases, and is used by PKI LDAPv2 clients successfully. However some LDAP clients, such as some versions of Netscape Communicator, still do not interpret the certificate properly when it is retrieved. They attempt to display it as a single stream of binary characters in the browser window, rather than decoding the ASN.1 into its constituent parts.

When LDAPv3 was specified, it overcame the encoding bug of LDAPv2 by introducing the concept of transfer encodings as part of the new attribute descriptions. Attribute descriptions are used to optionally qualify attribute type definitions, primarily to indicate attribute subtypes. However, one special option, the *;binary* transfer option, was built into the base LDAPv3 specification [10]. The ;binary option was

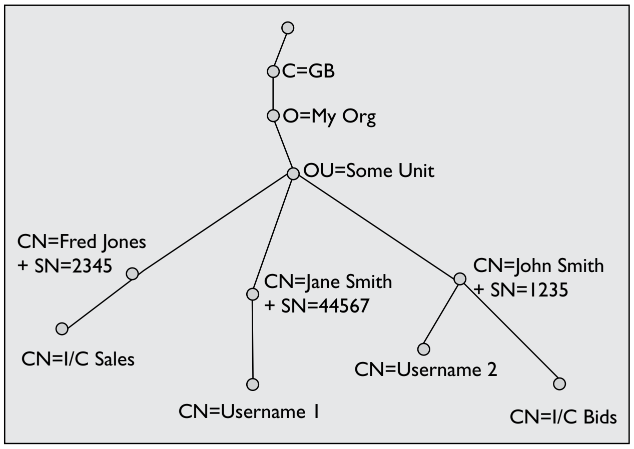


Figure 1(a). Child entries.

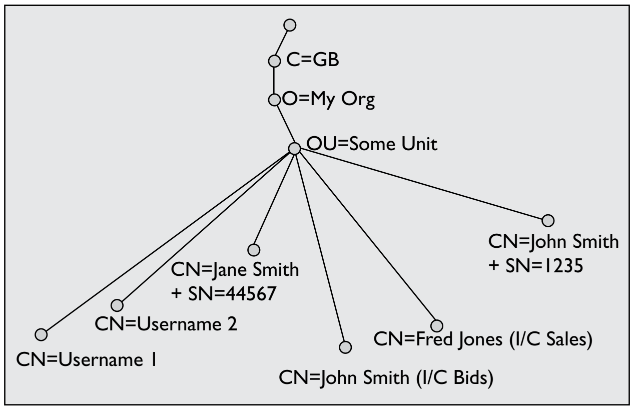


Figure 1(b). Sibling entries.

used to indicate a special type of transfer encoding (ASN.1 BER) rather than an attribute subtype. When the ;binary option, for example, userCertificate;binary, is used by a client to describe an attribute it wants to retrieve, it mandates the LDAPv3 server to return the attribute values encoded using the ASN.1 BER rather than in their LDAP-defined native encodings. In this way, when used to retrieve signed attributes such as certificates and CRLs, the client can validate the signatures against the binary values.

Many LDAPv3 clients and servers correctly support the ;binary transfer option as defined in the proposed standard [10]. However, some do not, and further, some LDAP servers do not recognize that a userCertificate attribute stored by an LDAPv2 client is the same attribute that an LDAPv3 client is trying to retrieve as a userCertificate;binary attribute. Finally, if LDAPv3 were to define a new transfer encoding, say *;xml*, then implementations that did not recognize this would consider it an unrecognized attribute subtype rather than a new transfer encoding (which only goes to confirm the folly of using the same protocol feature to indicate two different concepts). For these reasons the current revision of LDAPv3 [9] has removed the concept of ;binary from its specification and the PKIX Internet Draft (ID) [4] has said that the native LDAP encoding for PKI attributes is ASN.1 BER rather than ASCII, thus bringing equality to LDAPv2 and LDAPv3. Unfortunately, this will cause some interoperability problems in the short-term as products migrate from the old specification of LDAPv3 to the proposed new one. How this migration will be managed is still being actively debated in the IETF.

LDAPv3 Complexity. LDAPv3 is a complex protocol. It has many sophisticated features built into it, so it can incorporate all the richness needed of a general-purpose directory access protocol (just like the original X.500 DAP). Many of these features are not needed for simple PKI certificate and CRL retrieval, and so an LDAPv3 profile [3] is being specified by the PKIX working group. This profile specifies the features of LDAPv3 mandatory to support a PKI, those features that are optional and those that are not needed for a PKI. This ID was about to go to Last-Call in the PKIX working group (Last-Call is the penultimate stage to becoming a proposed standard RFC in the IETF standardization process) when one controversial item arose. The profile states that LDAPv3 servers must support multi-valued RDNs as this allows RDNs to be both user-friendly and unique (for example, CN=David Chadwick+SerialNumber=12345), but Microsoft's Active Directory does not support this feature and has no current plans to. So this issue currently remains unresolved.

Searching for the Correct Certificate or CRL. Simple PKIs usually only store single certificates in each user's directory entry, and only one CRL in each distribution point. But how does an LDAP client know which entry to retrieve? The client needs to be able to specify filtering criteria that tell the LDAP server which entry or entries to select. For example, "Find the entry for the person named Carly Simon and return her userCertificate attribute," or "Find the userCertificate attribute containing the email address carly.simon@someorg.com." Unfortunately no certificate or CRL matching rules have been standardized for LDAP (although they are for X.500 DAP). Thus the latter search cannot be requested without some work-around procedure.

PKI administrators cope with this problem today by creating one or more new attributes in the user's entry, in parallel with the userCertificate attribute. These new attributes each contain the contents of one of the fields of the user's certificate that are to be searched for, for example, the mail attribute holds the contents of the rfc822 value from the subjectAltName certificate extension, or the serialNumber attribute holds the certificate serial number. The LDAP client can then search for the entry containing these new attributes and ask for the accompanying userCertificate attribute to be returned. Hopefully the certificate will be the one that matches the user's search criteria.

But this ad-hoc approach places a large burden on the directory/PKI administrator, who must ensure the new attributes are kept synchronized with the contents of the user's certificate, and there is no agreed standard schema for these new attributes. Furthermore, this solution has security implications. The underlying security model for X.509 is that the directory service is not trusted; only the Certification Authority (CA) is trusted. The directory server may be open to attack, and wrong attributes inserted by the attacker. Consequently, CRLs and certificates are digitally signed by the CA to prevent unauthorized tampering, but the new attributes inserted by the directory/PKI administrator are not signed, and thus can be tampered with without it being immediately obvious. (The client would have to check that the contents of the retrieved certificate truly did match his filtering criteria and the directory/PKI administrator would continually have to check that the new attributes matched the contents of their peer certificate.) Finally, this solution does not work when the user has two or more certificates, since there is no way of pairing the new attribute values with particular userCertificate values. This is because attribute values are held as an unordered set and not in any defined order. Clearly a better solution is required.

A relatively new ID [7] has started to standardize the schema for these new attributes (but currently only for userCertificates), and also recommends that the each certificate and its corresponding attributes should be stored in a separate entry subordinate to that of the certificate holder (see Figure 1a). This pragmatic approach solves many, though certainly not all, of the certificate searching problems.

The IETF PKIX group has published a more comprehensive though complex solution. An ID [4] specifies LDAP encodings for matching rules and assertion values for both certificates and CRLs, based on those in the X.500 standard. This allows the client to specify its filtering criteria, and the server will match directly on the contents of the certificate or CRL, rather than on the contents of some peer attributes. The client can then be assured the returned certificate or CRL matches the one(s) requested. The simplest of these matching rules, certificateExactMatch, has already been implemented in OpenLDAP release 2.1. And at least one vendor of X.500-based LDAP servers also supports a subset of these matching rules in their server. However, clients will need to be updated in order to support these new matching rules and assertion values.

Users with Multiple Certificates. As PKIs become more advanced, users will start to be issued with several certificates. For example, a user might have a certificate for creating digitally signed messages, a certificate for receiving encrypted S/MIME email, and a certificate to be used for nonrepudiable e-commerce transactions. If all these certificates are stored in the user's entry—which is the natural place to store them—then LDAP clients have an additional problem (this is assuming that the matching rules mentioned previously have been implemented so that a client can search for a particular certificate, for example, find the S/MIME encryption certificate for A.Person@e-commerce.site.com).

The problem is there is no standard way in LDAP to ask the directory server to only return one certificate from the user's entry. Both LDAP protocols are only capable of returning all the attribute values from a particular attribute, or none of them. It is not possible to ask for a subset of the attribute values. This causes a problem for the client software if several certificates are returned when only one was being sought. The client software will have to implement complex matching rules to parse each certificate and find the correct one. This is both time-consuming and unnecessary. If the LDAP server has already implemented the certificate matching rules mentioned previously in order to find the correct certificate, it should surely be able to return just the desired certificate to the client, rather than all or none of them.

DAP already has a solution to this problem, a Boolean variable that asks for only matched values to be returned. The IETF LDAPEXT working group initially decided to add this Boolean to LDAPv3, but upon rigorous examination found a simple Boolean to be deficient when complex filters were being specified by the user. For example, in a multiple OR filter, where only one filter item matches an attribute value, should the other filter items that did not match on any attribute values cause none of those values to be returned (the Boolean is obeyed), or all values to be returned (the Boolean is ignored). The matched values ID [5] specifies an extension to the LDAPv3 protocol that allows the user to say precisely which value or values should be returned. This ID has now been through Last-Call and should soon be published as an RFC. Furthermore, it has recently been implemented in OpenLDAP release 2.1. Unfortunately, it is not possible to specify this extension in LDAPv2, as LDAPv2 is a fixed protocol with no mechanism for extending it.

How do most PKI implementers cope with this problem today? Basically they have to ensure each certificate attribute only holds a single certificate attribute value. This can be done in one of two ways, either by creating new certificate attributes for each type of certificate (for example, smimeEncCert, smimeNRcert), or altering the structure of the directory information tree (DIT) to fit the PKI applications. The former method is problematical, as some PKIs are not capable of changing the attribute type used for publishing certificates—they always assume the X.509 standard attribute type (userCertificate) is used. The latter method is sure to work with all PKI implementations, and is usually executed in one of three ways.

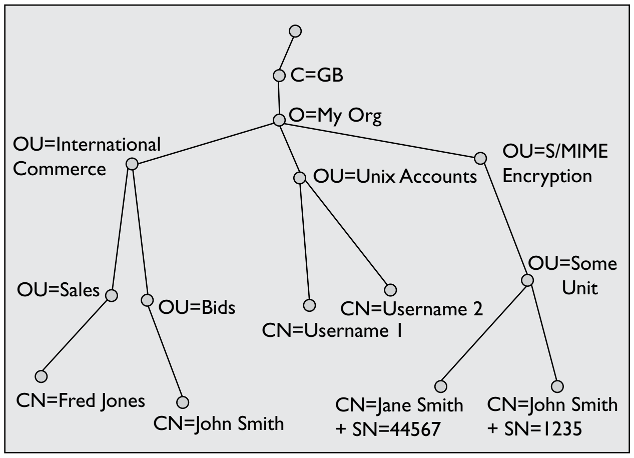


Figure 2. Application-based subtrees.

The first and second ways keep the existing DIT structure intact, but add extra entries for each user, as new certificates are issued. This ensures that only a single certificate is ever stored in each entry. The new entries are created either subordinate to the user's existing entry (as depicted in Figure 1a) or as siblings of the existing entries (as depicted in Figure 1b). The entries are typically given names that indicate the type of certificate stored within them. The third way creates a new DIT subtree for each PKI application, and users have separate entries in each parallel application tree (see Figure 2). Note that whether the organization uses DC or X.521 naming is immaterial to the examples.

A disadvantage of the first way is that some LDAP browsers expect common name (CN) entries to be leaf entries, and cannot cope with them being non-leaves. A disadvantage of the second way is that a user's name is prefixed with the type of certificate contained within the entry. A disadvantage of the third way is that new tree structures have to be built for each application (although this can be an advantage if the application is to be the sole user of the new subtree).

Distributed LDAP Deficiencies

Once PKIs start to link together, via either CA hierarchies, cross certification, or bridge CAs, then PKI users will need to have access to the certificates and CRLs contained in each of the PKI directory services. Thus, ideally we need a distributed directory service made up of all the individual PKI LDAP directories. Experience to date, for example with the U.S. Federal PKI [2], suggests that an X.500-based distributed directory with chaining, using LDAP for client access, provides the only workable solution. LDAP-only servers are currently not up to the task of building distributed directories. Why is this?

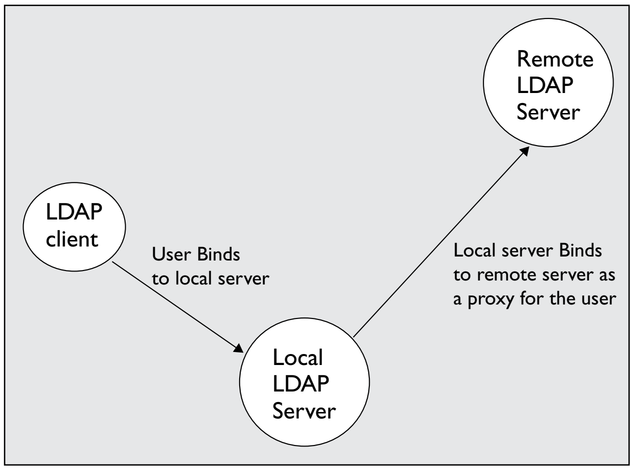
LDAP was initially conceived as a lightweight front-end protocol to a distributed X.500 directory service. Consequently, LDAPv2 had no need to support distribution, as the X.500 servers used the Directory System Protocol (DSP) for chaining. The types of features that a distributed directory service needs are:

- The ability of the servers to know about other servers (in X.500 these are called knowledge references, in the DNS these are the NS and A resource records) and how to interact with them; otherwise the client has to know the whereabouts of all the LDAP servers it wishes to contact, and has to choose the correct one for each request that it issues;
- The ability of the servers to pass client requests between themselves (in X.500 these mechanisms are called chaining and referrals, in the DNS these features are called the recursive and iterative modes respectively), otherwise the client has to contact each server separately and track its own progress,
- The ability of the servers to perform distributed authentication, otherwise the client has to have separate authentication credentials (typically passwords) for each LDAP server it wishes to contact. The DNS does not suffer from this problem, as its look-ups are unauthenticated.

Chaining and Referrals. LDAPv3 made a start on providing a distributed directory service by adding referrals to the LDAPv3 protocol [10]. Some LDAP server suppliers, mainly those that have built their

LDAP service around their X.500 servers, also provide LDAP chaining. However, referrals on their own are insufficient, since servers need a standard way of using them.

Knowledge References. X.500 defines five types of knowledge reference (superior, immediate superior, subordinate, cross and nonspecific subordinate) plus two for use in replication (consumer and supplier). There have been several attempts over the last five years to standardize equivalent LDAP knowledge references. Why these failed to progress past Internet draft stage is difficult to assess. However, during 2000 there was a resurgence of interest in this topic, and finally a RFC [12] has been published. Although this only standardizes one type of LDAP knowledge reference—subordinate—this is undoubtedly one of the most important types for distributed directories, and perhaps once this is fully implemented then standardization of other types may follow.



Remote LDAP Server
LDAP client
User Binds to local server
Local server Binds to remote server as a proxy for the user
Local LDAP Server

Figure 3. LDAP server proxying.

PKIX Certificate Extensions. Because LDAP directories were typically standalone directory islands that did not have knowledge of each other, the IETF PKIX group specified several certificate extensions designed to allow PKI clients to find the correct LDAP directory for their needs. This was effectively building LDAP knowledge references into certificates. First, they specified the precise contents of the X.509 CRL Distribution Points extension. This extension informs the client where CRLs may be found, and when CRLs are located in an LDAP directory, PKIX mandates that the distribution point name must be a LDAP Uniform Resource Identifier (URI), for example, ldap://directory.ja.net/cn=distribution point, c=gb.

Secondly, for hierarchical PKIs, subordinate CAs may include in their issued certificates a list of their superior CAs up to the trusted root of the hierarchy, along with the LDAP URIs of their directory services, for example, ldap://superior.directory.com/o=superiorCA, dc=myorg, dc=com. This information is held in a newly defined Authority Information Access extension.

Finally, for cross-certified CAs, the cross certificate may hold the location of the certified CA's LDAP directory in a newly defined Subject Information Access extension for example, ldap://ldap.otherCA.com/o=crossCA, dc=orgname, dc=com.

Unfortunately, at this point in time, the PKI-defined extensions are not widely implemented. Certainly if the CRL Distribution Points extension had been widely supported, there would not have been the huge problem experienced by Microsoft when in March 2001 it was publicized that Verisign had been duped into issuing two code-signing certificates in the name of Microsoft, to a non-Microsoft employee (CERT Advisory Note CA-2001-04). Once Verisign had discovered its error and revoked the certificates, PKI clients could have automatically retrieved the CRLs, instead of Microsoft having to hastily release a Windows Security Update.

Distributed Authentication. X.500 defines three mechanisms for distributed authentication. Two rely on the directory servers having a trust relationship between themselves, and all rely on them knowing each others' credentials. In the first mechanism, the user binds to its local server and is authenticated by it. The level of authentication could be none, password- or digital-signature based. The user then sends its request to the local server. If the local server is unable to fulfill the request, it binds to the remote server and chains the user's request to the latter. It tells the remote server the name of the user and the level of authentication that it has performed. The remote server then allows the user to have access to the appropriate amount of directory data, as directed by its access control policy.

In the second mechanism, which is only designed for password-based authentication, the user binds to a remote server using the same credentials it would normally use for its local directory server. The remote server binds to the local server, and after establishing a trusted link, issues a Compare operation to the local server, passing the user's credentials. The local server is then in a position to check these credentials and pass a True/False reply back to the remote server. (Note that this latter method can only work securely if the link is secure, or the credentials are encrypted prior to transfer.)

The third mechanism relies on the user sending digitally signed requests to its local server, which in the case of nonfulfillment, are chained to the remote server. The remote server must then validate the digital signature on the request and fetch any CRLs from the local server as appropriate.

LDAP does not have a standardized way of performing distributed authentication. It does not support chaining, which rules out the first mechanism, nor signed operations, which rules out the third mechanism. LDAP servers could be built to support the second mechanism, but it has to be said that this is the weakest of the three X.500 schemes. However, proxy authorization is a possibility. Proxy authorization is a feature of the Simple Authentication and Security Layer (SASL) [8] that is used in LDAPv3 Bind operations. In a SASL Bind, a client may specify an authorization identity that is different from its authentication identity. This allows a server, acting as an LDAP client, to authenticate to an LDAP server as itself but then to perform an action on behalf of its user. If the local LDAP server acts as the client, then it can perform an operation on a remote LDAP server on behalf of its user (see Figure 3). This mechanism relies on a trust relationship existing between the two LDAP servers.

Release 2.1 of OpenLDAP supports this proxying feature to the extent that it can act in the remote server role, but not in the local server/client role. In order for this to work in a controlled manner, the OpenLDAP implementation holds the following information:

- Authentication information for the local server; and
- Policy information that controls the user authorization identities the local server is allowed to assert (this minimizes the trust relationship the remote server must have).

Note that there are a couple of drawbacks to this approach. The first is that compromise of the local server can result in unauthorized access to the remote server. The second is that the local server must have separate sessions to the remote server for each user for which it is concurrently acting as a proxy. This disadvantage is not suffered by the X.500 distributed authentication schemes one and three, as the local server does not act as a proxy for the user, but rather chains the user's request via the DSP. The DSP carries either the user's DN in its chaining arguments (mechanism one) or the user's signed request (mechanism three). To solve this problem, LDAP needs to mirror the DN carrying feature of the DSP, by adding a control to each LDAPv3 chained operation to present the client's authorization identity. Such a control is specified in [11]. This will allow LDAP operations from multiple clients to share a common connection between the local and remote servers. While OpenLDAP plans to implement this feature in a subsequent release, again it will only be in the remote server mode, and not the local server mode. Thus, much more effort is required in order to make LDAP servers capable of building a distributed directory service.

Conclusion

The many deficiencies in the LDAP protocols when they are used to access PKI-related information have been highlighted here. These are mainly because LDAP was initially conceived as a protocol to access a distributed X.500 directory. However, the IETF has been steadily addressing these deficiencies over the past few years, with the consequence that by the end of 2003 it is anticipated that standard solutions should be available to address most of the identified deficiencies. Unfortunately, specifying standards on its own is not sufficient to build a distributed LDAP directory service. Products will still be needed that implement them. C

References

1. Boeyen, S., Howes, T., and Richard, P. *Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2. RFC 2559*, April 1999.
2. Electronic Messaging Association Challenge 2000. *Report of Federal Bridge Certification Authority Initiative and Demonstration*. Draft 101500, August 2000; csrc.nist.gov/pki/documents/emareport_20001015.pdf.
3. Chadwick, D.W. *Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv3*. <draft-ietf-pkix-ldap-v3-05.txt>, Jan. 2002.
4. Chadwick, D.W. and Legg, S. *Internet X.509 Public Key Infrastructure, LDAP Schema and Syntaxes for PKIs*. <draft-pkix-ldap-pki-schema-00.txt> June 2002.
5. Chadwick, D.W. and Mullan, S. Returning Matched Values with LDAPv3. Internet Draft <draft-ldapext-matchedval-07.txt>, Sept. 2002.
6. Howes, T., Kille, S., Yeong, W. and Robbins, C. *The String Representation of Standard Attribute Syntaxes*. RFC 1778, Mar. 1995.
7. Klasen, N. and Gietz, P. *An LDAPv3 Schema for X.509 Certificates*. <draft-klasen-ldap-x509certificate-schema-00.txt>, Feb. 2002.
8. Myers, J. *Simple Authentication and Security Layer (SASL)*. RFC 2222, Oct. 1997.
9. Sermersheim, J. *LDAP: The Protocol*. <draft-ietf-ldapbis-protocol-08.txt>, June 2002.
10. Wahl, M., Howes, T., and Kille, S. *Lightweight Directory Access Protocol (v3)*. Dec. 1997, RFC 2251.
11. Weltman, R. *LDAP Proxied Authorization Control*. <draft-weltman-ldapv3-proxy-11.txt>, May 2002.
12. Zeilenga, K. *Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories*. RFC 3296, July 2002.

David Chadwick (d.w.chadwick@salford.ac.uk) is a professor of information systems security at the University of Salford, U.K.