The drawings contained in this Recommendation have been done in AUTOCAD

Recommendation X.509

THE DIRECTORY - AUTHENTICATION FRAMEWORK [1]

(Melbourne, 1988)

CONTENTS

0 Introduction

1 Scope and field of application

2 References

3 Definitions

4 Notation and abbreviations

SECTION 1 - Simple authentication

5 Simple authentication procedure

SECTION 2 - Strong authentication

6 Basis of strong authentication

7 Obtaining a user's public key

8 Digital signatures

9 Strong authentication procedures

10 Management of keys and certificates

Annex A - Security requirements

Annex B - An introduction to public key cryptography

Annex C - The RSA public key cryptosystem

Annex D - Hash functions

Annex E - Threats protected against by the strong authentication method

Annex F - Data confidentiality

Annex G - Authentication framework in ASN.1

Annex H - Reference definition of algorithm object identifiers

0 Introduction

0.1 This document, together with the others of the series, has been produced to facilitate the interconnection of information processing systems to provide directory services. The set of all such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the Directory. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with or about objects such as OSI application-entities, people, terminals and distribution lists.

---

1) Recomendations X.509 and ISSSO 9594-8, Information Processing Systems - Open Systems Interconnection - The Directory - Authentication Framework, were developed in close collaboration and are technically aligned.

0.2 The Directory plays a significant role in Open Systems Interconnection, whose aim is to allow, with a minimum of technical agreement outside of the interconnection standards themselves, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different ages.

0.3 Many applications have requirements for security to protect against threats to the communication of information. Some commonly known threats, together with the security services and mechanisms that can be used to protect against them, are briefly described in Annex A. Virtually all security services are dependent upon the identities of the communicating parties being reliably known, i.e. authentication.

0.4 This Recommendation defines a framework for the provision of authentication services by the Directory to its users. These users include the Directory itself, as well as other applications and services. The Directory can usefully be involved in meeting their needs for authentication and other security services because it is a natural place from which communicating parties can obtain authentication information of each other: knowledge which is the basis of authentication. The Directory is a natural place because it holds other information which is required for communication and obtained prior to communication taking place. Obtaining the authentication information of a potential communication partner from the Directory is, with this approach, similar to obtaining an address. Owing to the wide reach of the Directory for communications purposes, it is expected that this authentication framework will be widely used by a range of applications.

1 Scope and field of application

1.1 This Recommendation:

- specifies the form of authentication information held by the Directory;
- describes how authentication information may be obtained from the Directory;
- states the assumptions made about how this authentication information is formed and placed in the Directory;
- defines three ways in which applications may use this authentication information to perform authentication and describes how other security services may be supported by authentication.

1.2 This Recommendation describes two levels of authentication: simple authentication, using a password as a verification of claimed identity; and strong authentication, involving credentials formed using cryptographic techniques. While simple authentication offers some limited protection against unauthorized access, only strong authentication should be used as the basis for providing secure services. It is not intended to establish this as a general framework for authentication, but it can be of general use for applications which consider these techniques adequate.

1.3 Authentication (and other security services) can only be provided within the context of a defined security policy. It is a matter for users of an application to define their own security policy which may be constrained by the services provided by a standard.

1.4 It is a matter for standards defining applications which use the authentication framework to specify the protocol exchanges which need to be performed in order to achieve authentication based upon the authentication information obtained from the Directory. The protocol used by applications to obtain credentials from the Directory is the Directory Access Protocol (DAP), specified in Recommendation X.519.

1.5 The strong authentication method specified in this Recommendation is based upon public-key cryptosystems. It is a major advantage of such systems that user certificates may be held within the Directory as attributes, and may be freely communicated within the Directory System and obtained by users of the Directory in the same manner as other Directory information. The user certificates are assumed to be formed by "off-line" means, and placed in the Directory by their creator. The generation of user certificates is performed by some off-line Certification Authority which is completely separate from the DSAs in the Directory. In particular, no special requirements are placed upon Directory providers to store or communicate user certificates in a secure manner.

A brief introduction to public-key cryptography can be found in Annex B.

1.6 In general, the authentication framework is not dependent on the use of a particular cryptographic algorithm, provided it has the properties described in 6.1. Potentially a number of different algorithms may be used. However, two users wishing to authenticate must support the same cryptographic algorithm for authentication to be performed

correctly. Thus, within the context of a set of related applications, the choice of a single algorithm will serve to maximize the community of users able to authenticate and communicate securely. One example of a public key cryptographic algorithm can be found in Annex C.

1.7 Similarly, two users wishing to authenticate must support the same hash function (see 3.3 f)) (used in forming credentials and authentication tokens). Again, in principle, a number of alternative hash functions could be used, at the cost of narrowing the communities of users able to authenticate. A brief introduction to hash functions together with one example hash function can be found in Annex D.

2 References

2.1 ISO 7498-2: Information Processing Systems - Open Systems Interconnection - Security Architecture.

3 Definitions

3.1 This Recommendation makes use of the following general security-related terms defined in Part 2 of the OSI Reference Model on Security:

a) asymmetric (encipherment);

b) authentication exchange;

c) authentication information;

d) confidentiality;

e) credentials;

f) cryptography;

g) data origin authentication;

h) decipherment;

i) encipherment;

j) key;

k) password;

l) peer-entity authentication;

m) symmetric (encipherment).

3.2 The following terms used in this Recommendation are defined in Recommendation X.501:

a) attribute;

b) Directory Information Base;

c) Directory Information Tree;

d) distinguished name;

e) entry;

f) object;

g) root.

3.3 The following specific terms are defined and used in this Recommendation:

a) authentication token (token): information conveyed during a strong authentication exchange, which can be used to authenticate its sender;

b) user certificate (certificate): the public keys of a user, together with some other information, rendered unforgeable by encipherment with the secret key of the certification authority which issued it;

c) certification authority: an authority trusted by one or more users to create and assign certificates. Optionally the certification authority may create the user's keys;

d) certification path: an ordered sequence of certificates of objects in the DIT which, together with the public key of the initial object in the path, can be processed to obtain that of the final object in the path;

e) cryptographic system, cryptosystem: a collection of transformations from plain text into ciphertext and vice versa, the particular transformation(s) to be used being selected by keys. The transformations are normally defined by a mathematical algorithm.

f) hash function: a (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. A "good" hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range;

g) one-way function: a (mathematical) function f which is easy to compute, but which for a general value y in the range, it is computationally difficult to find a value x in the domain such that f(x) = y. There may be a few values y for which finding x is not computationally difficult;

h) public key: (in a public key cryptosystem) that key of a user's key pair which is publicly known;

i) private key (secret key - deprecated): (in a public key cryptosystem) that key of a user's key pair which is known only by that user;

j) simple authentication: authentication by means of simple password arrangements;

k) security policy: the set of rules laid down by the security authority governing the use and provision of security services and facilities;

l) strong authentication: authentication by means of cryptographically derived credentials;

m) trust: generally, an entity can be said to "trust" a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. This trust may apply only for some specific function. The key role of trust in the authentication framework is to describe the relationship between an authenticating entity and a certification authority; an authenticating entity must be certain that it can trust the certification authority to create only valid and reliable certificates;

n) certificate serial number: an integer value, unique within the issuing CA, which is unambiguously associated with a certificate issued by that CA.

4 Notation and abbreviations

4.1 The notation used in this Recommendation is defined in Table 1/X.509 below.

Note - In the table, the symbols X, X_1, X_2 etc. occur in place of the names of users, while the symbol I occurs in place of arbitrary information.

4.2 The following abbreviations are used in this Recommendation:

CA Certification Authority

DIB Directory Information Base

DIT Directory Information Tree

PKCS Public key cryptosystem.

TABLE 1/X.509

Notation

NOTATION	MEANING
Xp	Public key of a user X.
Xs	Secret key of X.
Xp[I]	Encipherment of some information, I, using the public key of X.
Xs[I]	Encipherment of I using the secret key of X
X {I}	The signing of I by user X. It consists of I with an enciphered summary appended.
CA(X)	A certification authority of user X.
$CA^n(X)$	(where n > 1): CA(CA(...n times...(X))).
$X_1<<X_2>>$	The certificate of user X_2 issued by certification authority X_1.
$X_1<<X_2>>X_2<<X_3>>$	A chain of certificates (can be of arbitrary length), where each item is the certificate for the certification authority which produced the next. It is functionally equivalent to the following certificate $X_1<<X_{n+1}>>$. For example, possession of A<<B>>B<<C>> provides the same capability as A<<C>>, namely the ability to find out Cp given Ap.
$X_{1p}.X_1<<X_2>>$	The operation of unwrapping a certificate (or certificate chain) to extract a public key. It is an infix operator, whose left operand is the public key of a certification authority, and whose right operand is a certificate issued by that certification authority. The outcome is the public key of the user whose certificate is the right operand. For example: Ap·A<<B>>B<<C>> denotes the operation of using the public key of A to obtain B's public key, Bp, from its certificate, followed by using Bp to unwrap C's certificate. The outcome of the operation is the public key of C, Cp.
$A \rightarrow B$	A certification path from A to B, formed of a chain of certificates starting with $CA(A)<<CA^2(A)>>$ and ending with CA(B)<<B>>.

5 Simple authentication procedure

5.1 Simple authentication is intended to provide local authorization based upon a Distinguished Name of a user, bilaterally agreed (optional) password and a bilateral understanding of the means of using and handling this password within a single domain. Utilization of Simple Authentication is primarily intended for local use only, i.e. for peer entity authentication between one DUA and one DSA or between one DSA and one DSA. Simple authentication may be achieved by several means:

a) the transfer of the user's distinguished name and (optional) password in the clear (non- protected) to the recipient for evaluation;

b) the transfer of the user's distinguished name, password, and a random number and/or a timestamp, all of which are protected by applying a one-way function;

c) the transfer of the protected information described in b) together with a random number and/or timestamp, all of which is protected by applying a one-way function.

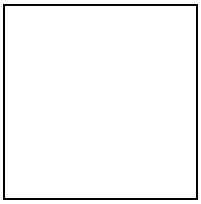
Note 1 - There is no requirement that the one-way functions applied be different.

Note 2 - The signalling of procedures for protecting passwords may be a matter for extension to the Document.

5.2 Where passwords are not protected, a minimal degree of security is provided for preventing unauthorized access. It should not be considered a basis for secure services. Protecting the user's distinguished name and password provides greater degrees of security. The algorithms to be used for the protection mechanism are typically non-enciphering one-way functions that are very simple to implement.

5.3 The general procedure for achieving simple authentication is shown in Figure 1/X.509.

FIGURE 1/X.509 - 0704380



5.3.1 The following steps are involved:

1) an originating user A sends its distinguished name and password to a recipient user B;

2) B sends the purported distinguished name and password of A to the Directory, where the password is checked against that held as the User Password attribute within the directory entry for A (using the Compare operation of the Directory);

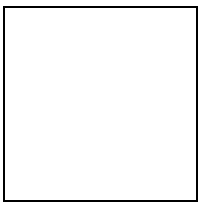
3) the Directory confirms (or denies) to B that the credentials are valid;

4) the success (or failure) of authentication may be conveyed to A.

5.3.2 The most basic form of simple authentication involves only step 1) and after B has checked the distinguished name and password, may include step 4).

5.4 Figure 2/X.509 illustrates two approaches by which protected identifying information may be generated. f_1 and f_2 are one-way functions (either identical or different) and the timestamps and random numbers are optional and subject to bilateral agreements.

FIGURE 2/X.509 - T0704390-88



A = user's distinguished name

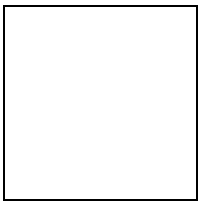
t^A = timestamps

$passw^A$ = password of A

q^A = random numbers, optionally with a counter included

5.4.1 Figure 3/X.509 illustrates the procedure for protected simple authentication.

FIGURE 3/X.509 - T0704400-88



The following steps are involved (initially using f_1 only):

1) An originating user, User A, sends its protected identifying information (Authenticator1) to User B. Protection is achieved by applying the one-way function (f_1) of Figure 2/X.509, where the timestamp and/or random number (when used) is to minimize replay and to conceal the password.

   The protection of A's password is of the form:

   $$Protected1 = f_1 (t1^A, q1^A, passwA).$$

   The information conveyed to B is of the form:

   $$Authenticator1 = t1^A, q1^A, A, Protected1.$$

   B verifies the protected identifying information offered by A by generating (using the timestamp, distinguished name and, optionally, additional timestamp and/or random number provided by A, together with a local copy of A's password) a local protected copy of A's password (of the form of Protected1). B compares (for equality) the purported identifying information (Protected1) with the locally generated value).

2) B confirms (or denies) to A the verification of the protected identifying information.

5.4.2 The procedure of 5.4.1 can be modified to afford greater protection (using f_1 and f_2).

The main differences are as follows:

1) A sends its (additionally) protected identifying information (Authenticator2) to B. Additional protection is achieved by applying a further one-way function, f_2, as illustrated in Figure 2/X.509. The further protection is of the form:

   $$Protected2 = f_2 (t2^A, q2^A, Protected1).$$

   The information conveyed to B is of the form:

   $$Authenticator2 = t1^A, t2^A, q1^A, q2^A, A, Protected2.$$

For comparison, B generates a local value of A's additionally protected password and compares it (for equality) with that of Protected2. (Similar in principle to step 1) of 5.4.1.)

2) B confirms (or denies) to A the verification of the protected identifying information.

Note - The procedures defined in this are specified in terms of A and B. As applied to the Directory (specified in Recommendation X.511 and X.518), A could be a DUA binding to a DSA, B; alternatively A could be a DSA binding to another DSA, B.

5.5 A User Password attribute type contains the password of an object. An attribute value for the user password is a string specified by the object.

```
UserPassword ::=    ATTRIBUTE
                WITH ATTRIBUTE-SYNTAX
                     OCTET STRING (SIZE (0..ub-user-password))
                     MATCHES FOR EQUALITY
```

5.6 The following ASN.1 macro may be used to define the data type arising from applying a one-way function to a given other data type.

PROTECTED MACRO ::= SIGNATURE

SECTION 2 - Strong authentication

6 Basis of strong authentication

6.1 The approach to strong authentication taken in this Recommendation makes use of the properties of a family of cryptographic systems, known as public-key cryptosystems (PKCS). These cryptosystems, also described as asymmetric, involve a pair of keys, one secret and one public, rather than a single key as in conventional cryptographic systems. Annex B gives a brief introduction to these cryptosystems and the properties which make them useful in authentication. For a PKCS to be usable in this authentication framework, at the present time, it must have the property that both keys in the key pair can be used for encipherment with the secret key being used to decipher if the public key was used, and the public key being used to decipher if the secret key was used. In other words $X_p . X_s = X_s . X_p$ where X_p/X_s are encipherment/decipherment functions using the public/secret keys of user X.

Note- lternative types of PKCS, i.e., ones which do not require the property of permutability and that can be supported without great modification to this Recommendation, are a possible future extension.

6.2 This authentication framework does not mandate a particular cryptosystem for use. It is intended that the framework will be applicable to any suitable public key cryptosystem, and will thus support changes to the methods used as a result of future advances in cryptography, mathematical techniques or computational capabilities. However, two users wishing to authenticate must support the same cryptographic algorithm for authentication to be performed correctly. Thus, within the context of a set of related applications, the choice of a single algorithm will serve to maximize the community of users able to authenticate and communicate securely. One example of a cryptographic algorithm can be found in Annex C.

6.3 Authentication relies on each user possessing a unique distinguished name. The allocation of distinguished names is the responsibility of the Naming Authorities. Each user must therefore trust the Naming Authorities not to issue duplicate distinguished names.

6.4 Each user is identified by its possession of its secret key. A second user is able to determine if a communication partner is in possession of the secret key, and can use this to corroborate that the communication partner is in fact the user. The validity of this corroboration depends on the secret key remaining confidential to the user.

6.5 For a user to determine that a communication partner is in possession of another user's secret key, it must itself be in possession of that user's public key. Whilst obtaining the value of this public key from the user's entry in the Directory is straightforward, verifying its correctness is more problematic. There are many possible ways for doing this: 7 describes a process whereby a user's public key can be checked by reference to the Directory. This process can only operate if there is an unbroken chain of trusted points in the Directory between the users requiring to authenticate. Such a chain can be constructed by identifying a common point of trust. This common point of trust must be linked to each user by an unbroken chain of trusted points.

7 Obtaining a user's public key

7.1 In order for a user to trust the authentication procedure, it must obtain the other user's public key from a source that it trusts. Such a source, called a certification authority (CA), uses the public key algorithm to certify the public key, producing a certificate. The certificate, the form of which is specified in 7.2 has the following properties:

- any user with access to the public key of the certification authority can recover the public key which was certified;
- no party other than the certification authority can modify the certificate without this being detected (certificates are unforgeable).

Because certificates are unforgeable, they can be published by being placed in the Directory, without the need for the latter to make special efforts to protect them.

Note - Although the CAs are unambiguously defined by a distinguished name in the DIT, this does not imply that there is any relationship between the organization of the CAs and the DIT.

7.2 A certification authority produces the certificate of a user by signing (see 8) a collection of information, including the user's distinguished name and public key. Specifically, the certificate of a user with distinguished name A, produced by the certification authority CA, has the following form:

$$CA\langle\langle A\rangle\rangle = CA\ \{SN, AI, CA, A, Ap, T^A\}$$

where SN is the serial number of the certificate, AI is the identifier of the algorithm used to sign the certificate, and T^A indicates the period of validity of the certificate, and consists of two dates, the first and last on which the certificate is valid. Since T^A is assumed to be changed in periods not less than 24 hours, it is expected that systems would use Coordinated Universal Time as a reference time base. The signature in the certificate can be checked for validity by any user with knowledge of CAP. The following ASN.1 data type can be used to represent certificates.

```
Certificate ::=  SIGNED SEQUENCE{
     version                 [0]Version DEFAULT 1988,
     serialNumber            SerialNumber,
     signature               Algorithmidentifier
     issuer                  Name
     validity                Validity,
     subject                 Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo}

Version        ::=       INTEGER { 1988(0)}
SerialNumber   ::=       INTEGER

Validity       ::=
     SEQUENCE{
          notBefore                UTCTime,
          notAfter                 UTCTime}

SubjectPublicKeyInfo   ::=
     SEQUENCE{
          algorithm                AlgorithmIdentifier
          subjectKey               BIT STRING}

AlgorithmIdentifier    ::=
     SEQUENCE{
          algorithm                OBJECT IDENTIFIER
          parameters               ANY DEFINED BY algorithm
                                   OPTIONAL}
```

7.3 The directory entry of each user, A, who is participating in strong authentication, contains the certificate(s) of A. Such a certificate is generated by a Certification Authority of A which is an entity in the DIT. A Certification Authority of A, which may not be unique, is denoted CA(A), or simply CA if A is understood. The public key of A can thus be discovered by any user knowing the public key of CA. Discovering public keys is thus recursive.

7.4 If user A, trying to obtain the public key of user B, has already obtained the public key of CA(B), then the process is complete. In order to enable A to obtain the public key of CA(B), the directory entry of each Certification Authority, X, contains a number of certificates. These certificates are of two types. First there are forward certificates of X generated by other Certification Authorities. Second there are reverse certificates generated by X itself which are the certified public keys of other certification authorities. The existence of these certificates enables users to construct certification paths from one point to another.

7.5 A list of certificates needed to allow a particular user to obtain the public key of another, is known as a certification path. Each item in the list is a certificate of the certification authority of the next item in the list. A certification path from A to B (denoted A -> B):

- starts with a certificate produced by CA(A), namely $CA(A)\langle\langle X^1\rangle\rangle$ for some entity X^1;
- continues with further certificates $X^i\langle\langle X^{i+1}\rangle\rangle$;
- ends with the certificate of B.

A certification path logically forms an unbroken chain of trusted points in the Directory Information Tree between two users wishing to authenticate. The precise method employed by users A and B to obtain certification paths A→B and B→ A may vary. One way to facilitate this is to arrange a hierarchy of CAs, which may or may not coincide with all or part of the DIT hierarchy. The benefit of this is that users who have CAs in the hierarchy may establish a certification path between them using the Directory without any prior information. In order to allow for this each CA may store one certificate and one reverse certificate designated as corresponding to its superior CA.

7.6 Certificates are held within directory entries as attributes of type **UserCertificate**, **CACertificate** and **CrossCertificatePair**. These attribute types are known to the Directory. These attributes can be operated on using the same protocol operations as other attributes. The definition of these types may be found in 3.3 of this Recommendation, the specification of these attribute types is as follows:

```
UserCertificate ::=        ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX Certificate

CACertificate   ::=        ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX Certificate

CrossCertificatePair    ::=        ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificatePair

CertificatePair ::=
    SEQUENCE{
        forward [0] Certificate OPTIONAL
        reverse [1] Certificate OPTIONAL
            - -at least one must be present --}
```

certificate bears the name of the Certification Authority which issued it.

7.7 In the general case, before users can mutually authenticate, the Directory must supply the complete certification and return certification paths. However, in practice, the amount of information which must be obtained from the Directory can be reduced for a particular instance of authentication by:

a) if the two users that want to authenticate are served by the same certification authority, then the certification path becomes trivial, and the users unwrap each other's certificates directly;

b) if the CAs of the users are arranged in a hierarchy, a user could store the public keys, certificates and reverse certificates of all certification authorities between the user and the root of the DIT. Typically, this would involve the user in knowing the public keys and certificates of only three or four certification authorities. The user would then only require to obtain the certification paths from the common point of trust;

c) if a user frequently communicates with users certified by a particular other CA, that user could learn the certification path to that CA and the return certification path from that CA, making it necessary only to obtain the certificate of the other user itself from the directory;

d) certification authorities can cross-certify one another by bilateral agreement. The result is to shorten the certification path;

e) if two users have communicated before and have learned one another's certificates, they are able to authenticate without any recourse to the Directory.

In any case, having learned each other's certificates from the certification path, the users must check the validity of the received certificates.

7.8 (Example). Figure 4/X.509 illustrates a hypothetical example of a DIT fragment, where the CAs form a hierarchy. Besides the information shown at the CAs, we assume that each user knows the public key of its certification authority, and its own public and secret keys.

7.8.1 If the CAs of the users are arranged in a hierarchy, A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>>, W<<V>>, V<<Y>>, Y<<Z>>, Z<<B>>

When A has obtained these certificates, it can unwrap the certification path in sequence to yield the contents of the certificate of B, including Bp:

Bp = Xp · X<<W>> W<<V>> V<<Y>> Y<<Z>> Z<<B>>

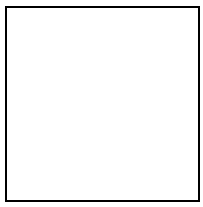
In general, A also has to acquire the following certificates from the directory to establish the return certification path from B to A:

Z<<Y>>, Y<<V>>, V<<W>>, W<<X>>, X<<A>>.

When B receives these certificates from A, it can unwrap the return certification path in sequence to yield the contents of the certificate of A, including Ap:

Ap = Zp · Z<<Y>> Y<<V>> V<<W>> W<<X>> X<<A>>.

FIGURE 4/X.509 - T0704410-88



7.8.2 Applying the optimizations of 7.7:

a) taking A and C, for example: both know Xp, so that A simply has to directly acquire the certificate of C. Unwrapping the certification path reduces to:

Cp = Xp · X<<C>>

and unwrapping the return certification Path reduces to:

Ap = Xp · X<<A>>

b) assuming that A would thus know W<<X>>, Wp, V<<W>>, Vp, U<<V>>, Up, etc., reduces the information which A has to obtain from the directory to form the certification path to:

V<<Y>>, Y<<Z>>, Z<<B>>

and the information which A has to obtain from the directory to form the return certification path to:

Z<<Y>>, Y<<V>>

c) assuming that A frequently communicates with users certified by Z, it can learn (in addition to the public keys learned in b) above) V<<Y>>, Y<<V>>, Y<<Z>>, and Z<<Y>>. To communicate with B, it need therefore only obtain Z<<B>> from the directory;

d) assuming that users certified by X and Z frequently communicate, then X<<Z>> would be held in the directory entry for X, and vice versa (this is shown in Figure 4/X.509). If A wants to authenticate to B, A need only obtain:

X<<Z>>, Z<<B>>

to form the certification path, and

Z<<X>>

to form the return certification path;

e) assuming users A and C have communicated before and have learned one another's certificates, they may use each other's public key directly, i.e.

Cp = Xp · X<<C>>

and

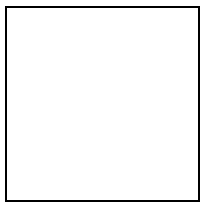
Ap = Xp · X<<A>>.

7.8.3 In the more general case the Certification Authorities do not relate in a hierarchical manner. Referring to the hypothetical example in Figure 5/X.509, suppose a user D, certified by U, wishes to authenticate to user E, certified by W. The directory entry of user D will hold the certificate U<<D>> and the entry of user E will hold the certificate W<<E>>.

Let V be a CA with whom CAs U and W have at some previous time exchanged public keys in a trusted way. As a result, certificates U<<V>>, V<<U>>, W<<V>> and V<<W>> have been generated and stored in the Directory. Assume U<<V>> and W<<V>> are stored in the entry of V, V<<U>> is stored in U's entry, and V<<W>> is stored in W's entry.

User D must find a certification path to E. Various strategies could be used. One such strategy would be to regard the users and CAs as nodes, and the certificates as arcs in a directed graph, in these terms, D has to perform a search in the graph to find a path from U to E, one such being U<<V>>, V<<W>>, W<<E>>. When this path has been discovered, the reverse path W<<V>>, V<<U>>, U<<D>> can also be constructed.

FIGURE 5/X.509 - T0704420



8 Digital signatures

This section is not intended to specify a standard for digital signatures in general, but to specify the means by which the tokens are signed in the Directory.

8.1 Information (info) is signed by appending to it an enciphered summary of the information. The summary is produced by means of a one-way hash function, while the enciphering is carried out using the secret key of the signer (see Figure 6/X.509). Thus

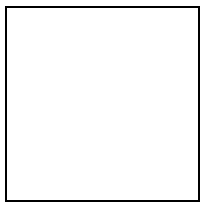
X{Info} = Info,Xs[h (Info)]

Note - The encipherment using the secret key ensures that the signature cannot be forged. The one-way nature of the hash function ensures that false information, generated so as to have the same hash result (and thus signature), cannot be substituted.

8.2 The recipient of signed information verifies the signature by:

- applying the one-way hash function to the information;
- comparing the result with that obtained by deciphering the signature using the public key of the signer.

FIGURE 6/X.509 - -0704430-88



8.3 This authentication framework does not mandate a single one-way hash function for use in signing. It is intended that the framework will be applicable to any suitable hash function, and will thus support changes to the methods used as a result of future advances in cryptography, mathematical techniques or computational capabilities. However, two users wishing to authenticate must support the same hash function for authentication to be performed correctly. Thus, within the context of a set of related applications, the choice of a single function will serve to maximize the community of users able to authenticate and communicate securely. An example hash function is specified in Annex D.

The signed information includes indicators that identify the hashing algorithm and the encryption algorithm used to compute the digital signature.

8.4 The encipherment of some data items may be described using the following ASN.1 MACRO:

```
ENCRYPTED MACRO          ::=
BEGIN

TYPE NOTATION     ::=    type (ToBeEnciphered)
VALUE NOTATION    ::=    value (VALUE BIT STRING)
END
```

The value of the bit string is generated by taking the octets which form the complete encoding (using the ASN.1 Basic Encoding Rules) of the value of the ToBeEnciphered type and applying an encipherment procedure to those octets.

Note 1 - The encryption procedure requires agreement on the algorithm to be applied, including any parameters of the algorithm, such as any necessary keys, initialization values, and padding instructions. It is the responsibility of the encryption procedures to specify the means by which synchronization of the sender and receiver of data is achieved, which may include information in the bits to be transmitted.

Note 2 - The encryption procedure is required to take as input a string of octets and to generate a single string of bits as its result.

Note 3 - Mechanisms for secure agreement on the encryption algorithm and its parameters by the sender and receiver of data are outside the scope of this Recommendation.

8.5 In the case where a signature must be appended to a data type, the following ASN.1 macro may be used to define the data type resulting from applying a signature to the given data type.

```
SIGNED MACRO      ::=

BEGIN

TYPE NOTATION     ::=    type (ToBeSigned)

VALUE NOTATION    ::=    value (VALUE

    SEQUENCE{
        ToBeSigned,
        AlgorithmIdentifier,
        -- of the algorithm used to compute
        -- the signature
        ENCRYPTED OCTET STRING
        - -re the octet string is the result
        - -the hashing of the value of
        - -'ToBeSigned' --}

END - - of SIGNED.                    )
```

8.6 In the case where only the signature is required, the following ASN.1 macro may be used to define the data type resulting from applying a signature to the given data type.

```
SIGNATURE MACRO ::=

BEGIN  TYPE NOTATION          ::=    type (OfSignature)

VALUE NOTATION          ::=    value (VALUE

    SEQUENCE{
        AlgorithmIdentifier,
        -- of the algorithm used to compute
        -- the signature
        ENCRYPTED OCTET STRING
        -- where the octet string is a function (e.g. a
        -- compressed or hashed version) of the
        -- value 'OfSignature', which may include
        -- the identifier of the algorithm used to
        -- compute the signature --}

END -- of SIGNATURE.                  )
```

8.7 In order to enable the validation of SIGNED and SIGNATURE types in a distributed environment, a distinguished encoding is required. A distinguished encoding of a SIGNED or SIGNATURE data value shall be obtained by applying the Basic Encoding Rules defined in Recommendation X.209 with the following restrictions:

a) the definite form of length encoding shall be used, encoded in the minimum number of octets;

b) for string types, the constructed form of encoding shall not be used;

c) if the value of a type is its default value, it shall be absent;

d) the components of a Set type shall be encoded in ascending order of their tag value;

e) the components of a Set-of type shall be encoded in ascending order of their octet value;

f) if the value of a Boolean type is true, the encoding shall have its contents octet set to $'FF'_{16}$;

g) each unused bits in the final octet of the encoding of a BitString value, if there are any, shall be set to zero;

h) the encoding of a Real type shall be such that bases 8, 10 and 16 shall not be used, and the binary scaling factor shall be zero.

9 Strong authentication procedures

9.1 Overview

9.1.1 The basic approach to authentication has been outlined above, namely the corroboration of identity by demonstrating possession of a secret key. However, many authentication procedures employing this approach are possible. In general it is the business of a specific application to determine the appropriate procedures, so as to meet the security policy of the application. This clause describes three particular authentication procedures, which may be found useful across a range of applications.

Note - This Recommendation does not specify the procedures to the detail required for implementation. However, additional standards could be envisaged which would do so, either in an application-specific or in a general-purpose way.

9.1.2 The three procedures involve different numbers of exchanges of authentication information, and consequently provide different types of assurance to their participants. Specifically,

a) one way authentication, described in 9.2, involves a single transfer of information from one user (A) intended for another (B), and establishes the following:

- the identity of A, and that the authentication token actually was generated by A;
- the identity of B, and that the authentication token actually was intended to be sent to B;
- the integrity and "originality" (the property of not having been sent two or more times) of the authentication token being transferred.

The latter properties can also be established for arbitrary additional data accompanying the transfer;

b) two-way authentication, described in 9.3, involves, in addition, a reply from B to A.

It establishes, in addition, the following:

- that the authentication token generated in the reply actually was generated by B and was intended to be sent to A;
- the integrity and originality of the authentication token sent in the reply;
- (optionally) the mutual secrecy of part of the tokens;

c) three-way authentication, described in 9.4, involves, in addition, a further transfer from A to B. It establishes the same properties as the two-way authentication, but does so without the need for association time stamp checking.

In each case where Strong Authentication is to take place, A must obtain the public key of B, and the return certification path from B to A, prior to any exchange of information. This may involve access to the Directory, as described in 7 above. Any such access is not mentioned again in the description of the procedures below.

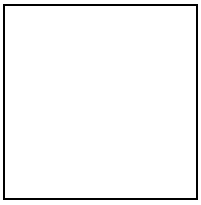
The checking of timestamps as mentioned in the following sections only applies when either synchronized clocks are used in a local environment, or if clocks are logically synchronized by bilateral agreements. In any case, it is recommended that Coordinated Universal Time be used.

For each of the three authentication procedures described below, it is assumed that party A has checked the validity of all of the certificates in the certification path.

9.2 One-way authentication

The following steps are involved, as depicted in Figure 7/X.509.

FIGURE 7/X.509 - T0704440-88



1) A generates r^A, a non-repeating number, which is used to detect replay attacks and to prevent forgery.
2) A sends the following message to B:

   B A, A{t^A, r^A, B}

   where t^A is a timestamp. t^A consists of one or two dates: the generation time of the token (which is optional) and the expire date. Alternatively, if data origin authentication of "sgnData" is to be provided by the digital signature:

   B A, A{t^A, r^A, B, sgnData}

   In cases where information is to be conveyed which will subsequently be used as a secret key (this information is referred to as "encData"):

   B A, A{t^A, r^A, B, sgnData, Bp[encData]}.

   The use of "encData" as a secret key implies that it must be chosen carefully, e.g. to be a strong key for whatever cryptosystem is used as indicated in the "sgnData" field of the token.
3) B carries out the following actions:
   a) obtains Ap from B A, checking that A's certificate has not expired;
   b) verifies the signature, and thus the integrity of the signed information;
   c) checks that B itself is the intended recipient;
   d) checks that the timestamp is "current";
   e) optionally, checks that r^A has not been replayed. This could, for example, be achieved by having r^A include a sequential part that is checked by a local implementation for its value uniqueness.

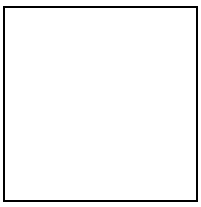
r^A is valid until the expire date indicated by t^A, r^A is always accompanied by a sequential part, which indicates that A will not repeat the token during the timerange t^A and therefore that checking of the value of r^A itself is not required.

In any case it is reasonable for party B to store the sequential part together with timestamp t^A in the clear and together with the hashed part of the token during timerange t^A.

9.3 Two-way authentication

The following steps are involved, as depicted in Figure 8/X.509.

FIGURE 8/X.509 - T0704450-88



1) as for 9.2.
2) as for 9.2.
3) as for 9.2.
4) B generates r^B, a non-repeating number, used for similar purpose(s) to r^A.
5) B sends the following authentication token to A:

   B $\{t^B, r^B, A, r^A\}$

   where t^B is a timestamp defined in the same way as t^A.

   Alternatively, if data origin authentication of "sgnData" is to be provided by the digital signature:

   B $\{t^B, r^B, A, r^A,$ sgnData$\}$

   In cases where information is to be conveyed which will subsequently be used as a secret key (this information is referred to as "encData"):

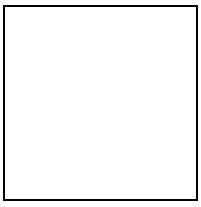
   B $\{t^B, r^B, A, r^A,$ sgnData, Ap[encData]$\}$.

   The use of "encData" as a secret key implies that it must be chosen carefully, e.g. to be a strong key for whatever cryptosystem is used as indicated in the "sgnData" field of the token.
6) A carries out the following actions:
   a) verifies the signature, and thus the integrity of the signed information;
   b) checks that A is the intended recipient;
   c) checks that the timestamp t^B is "current";
   d) optionally, checks that r^B has not been replayed [see 9.2 step 3) e)].

9.4 Three-way authentication

The following steps are involved, as depicted in Figure 9/X.509.

FIGURE 9/X.509 - T0704460-88



1) as for 9.3.
2) as for 9.3. Timestamp t^A may be zero.
3) as for 9.3, except that the timestamp need not be checked.
4) as for 9.3.
5) as for 9.3. Timestamp t^B may be zero.

6) as for 9.3, except that the timestamp need not be checked.

7) A checks that the received r^A is identical to the r^A which was sent.

8) A sends the following authentication token to B:
$A\{r^B\}$.

9) B carries out the following actions:

   a) checks the signature and thus the integrity of the signed information;

   b) checks that the received r^B is identical to the r^B which was sent by B.

10 Management of keys and certificates

10.1 Generation of key pairs

10.1.1 The overall security management policy of an implementation will define the lifecycle of key pairs, and is thus outstide the scope of the authentication framework. However, it is vital to the overall security that all secret keys remain known only to the user to whom they belong.

Key data is not easy for a human user to remember, so a suitable method for storing it in a convenient transportable manner must be employed. One possible mechanism would be to use a "Smart Card". This would hold the secret and (optionally) public keys of the user, the user's certificate, and a copy of the certification authority's public key. The use of this card must additionally be secured by e.g. at least use of a PIN (Personal Identification Number), increasing the security of the system by requiring the user to possess the card and to know how to access it. The exact method chosen for storing such data, however, is beyond the scope of this Recommendation.

10.1.2 There are three ways in which a user's key pair may be produced, as described in 10.1.2.1 to 10.1.2.3.

10.1.2.1 The user generates its own key pair. This method has the advantage that a user's secret key is never released to another entity, but requires a certain level of competence by the user as described in Annex C.

10.1.2.2 The key pair is generated by a third party. The third party must release the secret key to the user in a physically secure manner, then actively destroy all information relating to the creation of the key pair plus the keys themselves. Suitable physical security measures must be employed to ensure that the third party and the data operations are free from tampering.

10.1.2.3 The key pair is generated by the CA. This is a special case of 10.1.2.2, and the considerations there apply.

Note - The certification authority already exhibits trusted functionality with respect to the user, and will be subject to the necessary physical security measures. This method has the advantage of not requiring secure data transfer to the CA for certification.

10.1.2.4 The cryptosystem in use imposes particular (technical) constraints on key generation.

10.2 Management of certificates

10.2.1 A certificate associates the public key and unique distinguished name of the user it describes. Thus:

a) a certification authority must be satisfied of the identity of a user before creating a certificate for it;

b) a certification authority must not issue certificates for two users with the same name.

10.2.2 The production of a certificate occurs off-line and must not be performed with an automatic query/response mechanism. The advantage of this certification is that because the secret key of the certification authority, CAs, is never known except in the isolated and physically secure CA, the CA secret key may then only be learnt by an attack on CA itself, making compromise unlikely.

10.2.3 It is important that the transfer of information to the certification authority is not compromised, and suitable physical security measures must be taken. In this regard:

a) it would be a serious breach of security if the CA issued a certificate for a user with a public key that had been tampered with;

b) if the means of generation of key pairs of 10.1.2.3 is employed, no secure transfer is needed;

c) if the means of generation of key pairs of 10.1.2.1 or of 10.1.2.2 is employed, the user may use different methods (on-line or off-line) to communicate its public key to the CA in a secure manner. On-

line methods may provide some additional flexibility for remote operations performed between the user and the CA.

10.2.4 A certificate is a publicly available piece of information, and no specific security measures need to be employed with respect to its transportation to the Directory. As it is produced by an off- line certification authority on behalf of a user who will be given a copy of it, the user need only store this information in its directory entry on a subsequent access to the Directory. Alternatively the CA could lodge the certificate for the user, in which case this agent must be given suitable access rights.

10.2.5 Certificates will have a lifetime associated with them, at the end of which they expire. In order to provide continuity of service, the CA shall ensure timely availability of replacement certificates to supersede expired/expiring certificates. This has a number of aspects, as described in 10.2.5.1 and 10.2.5.2.

10.2.5.1 Validity of certificates may be designed so that each becomes valid at the time of expiry of its predecessor, or an overlap may be allowed. The latter prevents the CA from having to install and distribute a large number of certificates that may run out at the same expiration date.

10.2.5.2 Expired certificates will normally be removed from the Directory. It is a matter for the security policy and responsibility of the CA to keep old certificates for a period of time if a non repudiation of data service is provided.

10.2.6 Certificates may be revoked prior to their expiration time, e.g. if the user's secret key is assumed to be compromised, or the user is no longer to be certified by the CA, or if the CA's certificate is assumed to be compromised. This has a number of aspects, as described in 10.2.6.1-10.2.6.4.

10.2.6.1 The revocation of a user certificate or CA certificate shall be made known by the CA, and a new certificate shall be made available, if appropriate. The CA may then inform the owner of the certificate about its revocation by some off-line procedure.

10.2.6.2 The CA shall maintain:

a) a time-stamped list of the certificates it issued which have been revoked;

b) a time-stamped list of revoked certificates of all CAs known to the CA, certified by the CA.

Both certified lists shall exist, even if empty.

10.2.6.3 The maintenance of Directory entries affected by the CA's revocation lists is the responsibility of the Directory and its users, acting in accordance with the security policy. For example, the user may modify its object entry by replacing the old certificate with a new one. The latter will then be used to authenticate the user to the Directory.

10.2.6.4 The revocation lists ("black-lists") are held within entries as attributes of types "CertificateRevocationList" and "AuthorityRevocationList". These attributes can be operated on using the same operations as other attributes. These attribute types are defined as follows:

```
CertificateRevocationList       ::=     ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificateList

AuthorityRevocationList                 ::=     ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificateList
CertificateList                         ::=     SIGNED SEQUENCE{
    signature AlgorithmIdentifier,
    issuer Name,
    lastUpdate UTCTime,
    revokedCertificates

        SIGNED SEQUENCE OF SEQUENCE{
            signature AlgorithmIdentifier,
            issuer Name, CertificateSerialNumber subject,
            revocationDate UTCTime}
                    OPTIONAL}
```

Note 1 - The checking of the entire list of certificates is a local matter.

Note 2 - If a non-repudiation of data service is dependent on keys provided by the CA the service must ensure that all relevant keys of the CA (revoked or expired) and the timestamped revocation lists are archived and certified by a current authority.