Information technology —
Open Systems Interconnection —
The Directory: Authentication Framework

Recommendation X.509

ISO/IEC 9594-8

Contents

Foreword

Introduction

This Recommendation | International Standard, together with other Recommendations | International Standards, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of 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 application-entities, people, terminals and distribution lists.

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.

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 B. Virtually all security services are dependent upon the identities of the communicating parties being reliably known, i.e., authentication.

This Recommendation | International Standard 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.

Annex A, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all of the definitions associated with the authentication framework.

Annex B, which is not an integral part of this Recommendation | International Standard, describes security requirements.

Annex C, which is not an integral part of this Recommendation | International Standard, is an introduction to public key cryptography.

Annex D, which is not an integral part of this Recommendation | International Standard, describes the RSA Public Key Cryptosystem.

Annex E, which is not an integral part of this Recommendation | International Standard, describes hash functions.

Annex F, which is not an integral part of this Recommendation | International Standard, describes threats protected against by the strong authentication method.

Annex G, which is not an integral part of this Recommendation | International Standard, describes data confidentiality.

Annex H, which is an integral part of this Recommendation | International Standard, defines object identifiers assigned to authentication and encryption algorithms, in the absence of a formal register.

Annex J, which is not an integral part of this Recommendation | International Standard, lists the amendments and defect reports that have been incorporated to form this edition of this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

Information technology — Open Systems Interconnection — The Directory: Authentication Framework

SECTION 1: GENERAL

1 Scope

This Recommendation | International Standard:

- 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 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.

This Recommendation | International Standard 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.

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.

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 ITU-T Recomendation X.519 | ISO/IEC 9594-5.

The strong authentication method specified in this Recommendation | International Standard 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 C.

In general, the authentication framework is not dependent on the use of a particular cryptographic algorithm, provided it has the properties described in 7.1. Potentially a number of different algorithms may be used. However, two users wishing to authenticate shall 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 D.

Similarly, two users wishing to authenticate shall support the same hash function (see 3.3f) (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 E.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard part. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent editions of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1993, *Information Technology — Open Systems Interconnection — The Directory: Overview of Concepts, Models and Services.*
- ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2:1993, *Information Technology — Open Systems Interconnection —The Directory: The Models*.
- ITU-T Recommendation X.511 (1993) | ISO/IEC 9594-3:1993, *Information Technology — Open Systems Interconnection —The Directory: Abstract Service Definition.*
- ITU-T Recommendation X.518 (1993) | ISO/IEC 9594-4:1993, *Information Technology — Open Systems Interconnection —The Directory: Procedures for Distributed Operation.*
- ITU-T Recommendation X.519 (1993) | ISO/IEC 9594-5:1993, *Information Technology — Open Systems Interconnection —The Directory: Protocol Specifications.*
- ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6:1993, *Information Technology — Open Systems Interconnection —The Directory: Selected Attribute Types.*
- ITU-T Recommendation X.521 (1993) | ISO/IEC 9594-7:1993, *Information Technology — Open Systems Interconnection —The Directory: Selected Object Classes.*
- ITU-T Recommendation X.525 (1993) | ISO/IEC 9594-9:1993, *Information Technology — Open Systems Interconnection — The Directory: Replication*
- ITU-T Recommendation X.680 (1994*) | ISO/IEC 8824-1:1994*, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1): Specification of Basic Notation.*
- ITU-T Recommendation X.681 (1994*) | ISO/IEC 8824-2:1994*, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1): Information Object Specification.*
- ITU-T Recommendation X.682 (1994*) | ISO/IEC 8824-3:1994*, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1): Constraint Specification.*
- ITU-T Recommendation X.683 (1994*) | ISO/IEC 8824-4:1994*, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications.*
- ITU-T Recommendation X.690 (1994*) | ISO/IEC 8824-4:1994*, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1): Basic Encoding Rules.*
- ITU-T Recommendation X.880 (1994*) | ISO/IEC 9072-1:1994*, *Information Technology — Remote Operations: Concepts, Model and Notation.*
- ITU-T Recommendation X.881 (1994*) | ISO/IEC 9072-2:1994*, *Information Technology — Remote Operations — OSI Realization — Remote Operations Service Element (ROSE) Service Definition.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.800 (1991), *Security Architecture for Open Systems Interconnection for CCITT Applications*.

---

* Presently at the stage of draft.

— ISO 7498-2:1987, *Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture*.

3 Definitions

For the purposes of this CCITT Recommendation | International Standard, the following definitions apply.

3.1 OSI Reference Model security architecture definitions

The following terms are defined in CCITT Rec. X.800 | ISO 7498-2:

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 Directory model definitions

The following terms are defined in ITU-T Recommendation X.501 | ISO/IEC 9594-2:

a) *attribute* ;

b) *Directory Information Base* ;

c) *Directory Information Tree* ;

d) *distinguished name* ;

e) *entry* ;

f) *object* ;

g) *root* .

3.3 Authentication framework definitions

The following terms are defined in this Recommendation | International Standard:

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

— *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;

— *certification authority* : An authority trusted by one or more users to create and assign certificates. Optionally the certfication authority may create the users' keys;

— *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;

— *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.

- *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;
- *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;
- *public key* : (In a public key cryptosystem) that key of a user’s key pair which is publicly known;
- *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;
- *simple authentication* : Authentication by means of simple password arrangements;
- *security policy* : The set of rules laid down by the security authority governing the use and provision of security services and facilities;
- *strong authentication* : Authentication by means of cryptographically derived credentials;
- *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 shall be certain that it can trust the certification authority to create only valid and reliable certificates;
- *certificate serial number*: An integer value, unique within the issuing CA, which is unambiguously associated with a certificate issued by that CA.

4 Abbreviations

CA	Certification Authority
DIB	Directory Information Base
DIT	Directory Information Tree
PKCS	Public key cryptosystem

5 Conventions

With minor exceptions this Directory Specification has been prepared according to the “Presentation of ITU-TS/ISO/IEC common text” guidelines in the Guide for ITU-TS and ISO/IEC JTC 1 Cooperation, March 1993.

The term “Directory Specification” (as in “this Directory Specification”) shall be taken to mean ITU-T Rec. X.509 | ISO/IEC 9594-8. The term “Directory Specifications” shall be taken to mean the X.500 series of Recommendations and all parts of ISO/IEC 9594.

This Directory Specification uses the term “1988 edition systems” to refer to systems conforming to the previous (1988) edition of the Directory Specifications, i.e., the 1988 edition of the series of CCITT X.500 Recommendations and the ISO/IEC 9594:1990 edition. Systems conforming to the current Directory Specifications are referred to as “1993 edition systems”.

If the items in a list are numbered (as opposed to using “—” or letters), then the items shall be considered steps in a procedure.

The notation used in this Directory Specification is defined in Table 1 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

TABLE 1 — 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».

SECTION 2: SIMPLE AUTHENTICATION

6 Simple authentication procedure

Simple authentication is intended to provide local authorization based upon the distinguished name of a user, a 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 a timestamp, all of which is protected by applying a one-way function.

Notes

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

2 The signaling of procedures for protecting passwords may be a matter for extension to the document.

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.

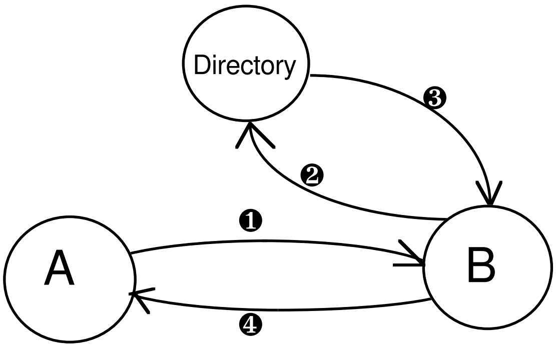


Figure 1— The Unprotected Simple Authentication Procedure

The general procedure for achieving simple authentication is shown in Figure 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 **UserPassword** attribute within the directory entry for A (using the Compare operation of the Directory);

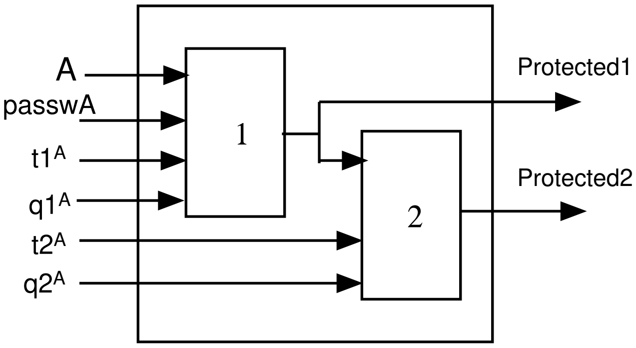
3) the Directory confirms (or denies) to B that the credentials are valid;

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

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.

6.1 Generation of protected identifying information

Figure 2 illustrates two approaches by which protected identifying information may be generated. $f1$ and $f2$ are one-way functions (either identical or different) and the timestamps and random numbers are optional and subject to bilateral agreements.



Legend

A = user's distinguished name
t^A = timestamps
passwA = password of A
q^A = random numbers, optionally with a counter included

Figure 2 — Protected Simple Authentication

6.2 Procedure for protected simple authentication

Figure 3 illustrates the procedure for protected simple authentication.

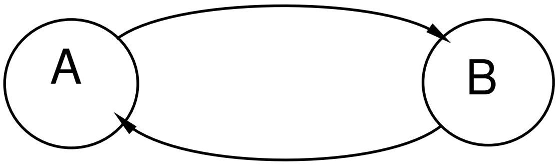


Figure 3 Ä The Protected Simple Authentication Procedure

The following steps are involved (initially using $f1$ 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 ($f1$) of Figure 2, where the timestamp and/or random number (when used) is used to minimize replay and to conceal the password.

   The protection of A s password is of the form:

   Protected1 = $f1$ ($t1^A$, $q1^A$, 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 distinguished name and optional 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 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.

The procedure can be modified to afford greater protection using $f1$ and $f2$. 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, $f2$, as illustrated in Figure 2. The further protection is of the form:

   $$\text{Protected2} = f2\ (t2^A, q2^A, \text{Protected1})$$

   The information conveyed to B is of the form:

   $$\text{Authenticator2} = t1^A, t2^A, q1^A, q2^A, A, \text{Protected2}$$

   For comparison, B generates a local value of A s additionally protected password and compares it for equality with that of Protected2 (this is similar in principle to step 1 of clause 6.4.1).

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

Note The procedures defined in these clauses are specified in terms of A and B. As applied to the Directory (specified in ITU-T Rec. X.511 | ISO/IEC 9594-3 and ITU-T Rec. X.518 | ISO/IEC 9594-4), A could be a DUA binding to a DSA, B; alternatively, A could be a DSA binding to another DSA, B.

6.3 User Password attribute type

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 SYNTAX                 OCTET STRING (SIZE (0..ub-user-password))
        EQUALITY MATCHING RULE      octetStringMatch
        ID                          id-at-userPassword }
```

SECTION 3: STRONG AUTHENTICATION

7 Basis of strong authentication

The approach to strong authentication taken in this Directory Specification 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 C 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 this present time, it shall 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/private keys of user X.

> **Note** Alternative types of PKCS, i.e., ones which do not require the property of permutability and that can be supported without great modification to this Directory Specification, are a possible future extension.

This authentication framework does not mandate a particular cryptosystem for use. It is intended that the framework shall be applicable to any suitable public key cryptosystem, and shall 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 shall 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 shall serve to maximize the community of users able to authenticate and communicate securely. One example of a cryptographic algorithm can be found in Annex D.

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

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.

For a user to determine that a communication partner is in possession of another user's secret key, it shall 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: clause 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 shall be linked to each user by an unbroken chain of trusted points.

8 Obtaining a user's public key

In order for a user to trust the authentication procedure, it shall 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 later in this clause, 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.

A certification authority produces the certificate of a user by signing (see Clause 8) a collection of information, including the user's distinguished name and public key, as well as an optional *unique identifier* containing additional information about the user. The exact form of the unique identifier contents is unspecified here and left to the certification authority and might be, for example, an object identifier, a certificate, a date, or some other form of certification on the validity of the distinguished name. Specifically, the certificate of a user with distinguished name A and unique identifier UA, produced by the certification authority with name CA and unique identifier UCA, has the following form:

$$CA<<A>> = CA\{V,SN,AI,CA,UCA,A,UA,Ap,T^{A}\}$$

where V is the version of the certificate, SN is the serial number of the certificate, AI is the identifier of the algorithm used to sign the certificate, UCA is the optional unique identifier of the CA , UA is the optional unique identifier of the user A, 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 v1,
        serialNumber                    CertificateSerialNumber,
        signature                       AlgorithmIdentifier,
        issuer                          Name,
        validity                        Validity,
        subject                         Name,
        subjectPublicKeyInfo            SubjectPublicKeyInfo,
        issuerUniqueIdentifier   [1]    IMPLICIT UniqueIdentifier OPTIONAL,
                                                -- if present, version must be v2
        subjectUniqueIdentifier  [2]    IMPLICIT UniqueIdentifier OPTIONAL
                                                -- if present, version must be v2 -- }}

Version                   ::=    INTEGER { v1(0), v2(1) }

CertificateSerialNumber   ::=    INTEGER

AlgorithmIdentifier       ::=    SEQUENCE {
        algorithm          ALGORITHM.&id ({SupportedAlgorithms}),
        parameters         ALGORIGHM.&Type ({SupportedAlgorithms}{ @algorithm}) OPTIONAL }
–       Definition of the following information object set is deferred, perhaps to standardized
–       profiles or to protocol implementation conformance statements. The set is required to
–       specify a table constraint on the parameters component of AlgorithmIdentifier.
–       SupportedAlgorithms      ALGORITHM   ::=     { ... | ... }

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

SubjectPublicKeyInfo      ::=    SEQUENCE {
        algorithm          AlgorithmIdentifier,
        subjectPublicKey   BIT STRING }
```

Note In situations where a distinguished name might be reassigned to a different user by the Naming Authority, CAs can use the unique identifier to distinguish between reused instances. However, if the same user is provided certificates by multiple CAs, it is recommended that the CAs coordinate on the assignment of unique identifiers as part of their user registration procedures.

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.

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.

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)⌋X^{1}≅ for some entity X^{1};

continues with further certificates X^{i}⌋X^{i+1}≅;

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.

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 can be found in clause 3.3 of this Directory Specification; the specification of these attribute types is as follows:

```
userCertificate             ATTRIBUTE   ::=     {
        WITH SYNTAX         Certificate
        ID                  id-at-userCertificate}

cACertificate               ATTRIBUTE   ::=     {
        WITH SYNTAX         Certificate
        ID                  id-at-cAcertificate }

crossCertificatePair        ATTRIBUTE   ::=     {
        WITH SYNTAX         CertificatePair
        ID                  id-at-crossCertificatePair }

CertificatePair             ::=     SEQUENCE {
        forward        [0]  Certificate OPTIONAL,
        reverse        [1]  Certificate OPTIONAL
                            -- at least one of the pair shall be present -- }
```

A user may obtain one or more certificates from one or more Certification Authorities. Each certificate bears the name of the Certification Authority which issued it. The following ASN.1 data types can be used to represent certificates and a certification path:

```
Certificates                ::=     SEQUENCE {
        userCertificate             Certificate,
        certificationPath           ForwardCertificationPath OPTIONAL}

CertificationPath           ::=     SEQUENCE {
        userCertificate             Certificate,
        theCACertificates           SEQUENCE OF CertificatePair OPTIONAL}
```

In addition, the following ASN.1 data type can be used to represent the forward certification path. This component contains the certification path which can point back to the originator.

```
ForwardCertificationPath    ::=     SEQUENCE OF CrossCertificates

CrossCertificates           ::=     SET OF Certificate
```

8.1 Optimization of the amount of information obtained from the Directory

In the general case, before users can mutually authenticate, the Directory shall supply the complete certification and return certification paths. However, in practice, the amount of information which shall 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 shall check the validity of the received certificates.

8.2 Example

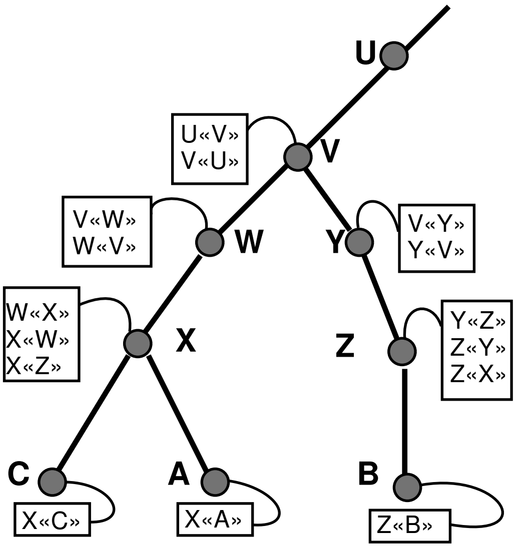


Figure 4.ÄCA Hierarchy @ A Hypothetical Example

Figure 4 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.

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:

XJW≅, WJV≅, VJY≅, YJZ≅, ZJB≅

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 ÅXJW≅ WJV≅ VJY≅ YJZ≅ ZJB≅

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

ZJY≅, YJV≅, VJW≅, WJX≅, XJA≅.

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 ÅZJY≅ YJV≅ VJW≅ WJX≅ XJA≅

Applying the optimizations of 8.1:

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 ÅX⌋C≅

and unwrapping the return certification Path reduces to:

Ap = Xp ÅX⌋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). 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 ÅX⌋C≅

and

Ap = Xp Å X⌋A≅

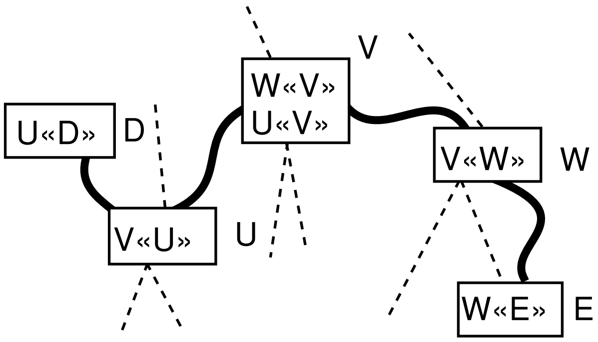


Figure 5 Ä Non-hierarchical Certification Path @ An Example

In the more general case the Certification Authorities do not relate in a hierarchical manner. Referring to the hypothetical example in Figure 5, suppose a user D, certified by U, wishes to authenticate to user E, certified by W. The Directory entry of user D shall hold the certificate U⌊D≅ and the entry of user E shall 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.

9 Digital signatures

This clause 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.

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). Thus

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

Note Ä 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.

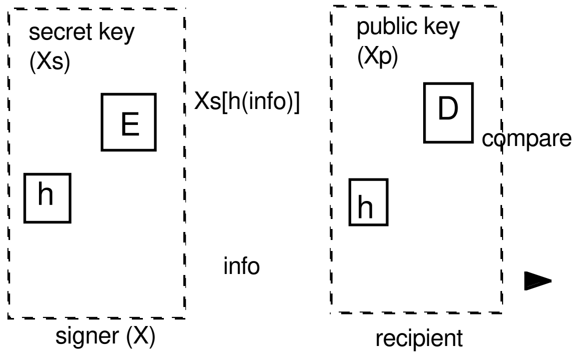


Figure 6 Ä Digital Signatures

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.

This authentication framework does not mandate a single one-way hash function for use in signing. It is intended that the framework shall be applicable to any suitable hash function, and shall 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 shall 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 shall serve to maximize the community of users able to authenticate and communicate securely. An example hash function is specified in Annex E.

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

The encipherment of some data item may be described using the following ASN.1:

```
ENCRYPTED { ToBeEnciphered }  ::=       BIT STRING ( CONSTRAINED BY {
        -- must be the result of applying an encipherment procedure --
        -- to the BER-encoded octets of a value of -- ToBeEnciphered})
```

The value of the bit string is generated by taking the octets which form the complete encoding (using the ASN.1 Basic Encoding Rules ITU-T Rec. X.690 | ISO/IEC 8825-1) of the value of the **ToBeEnciphered** type and applying an encipherment procedure to those octets.

Notes

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.

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.

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 Directory Specification.

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

```
SIGNED { ToBeSigned }          ::=     SEQUENCE {
        toBeSigned             ToBeSigned,
        COMPONENTS OF          SIGNATURE { ToBeSigned }}
```

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 { OfSignature }      ::=     SEQUENCE {
        algorithmIdentifier    AlgorithmIdentifier,
        encrypted              ENCRYPTED { HASHED { OfSignature }}}
```

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 ISO 8825, 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 Bit String 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.

10 Strong authentication procedures

10.1 Overview

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 Directory Specification 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.

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 10.2, 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 10.3, 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 10.4, 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 clause 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 clauses 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.

10.2 One-way authentication

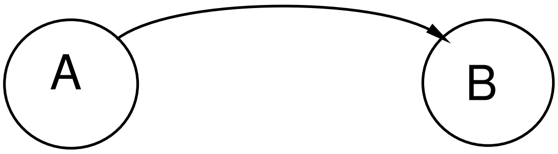


Figure 7 Ä One-way Authentication

The following steps are involved, as depicted in Figure 7:

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 expiry 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 shall 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 expiry date indicated by t^A. r^A is always accompanied by a sequential part, which indicates that A shall 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.

10.3 Two-way authentication

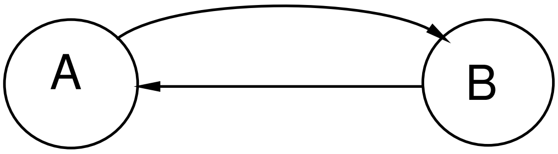


Figure 8 Ä Two-way Authentication

The following steps are involved, as depicted in Figure 8:

1) as for 10.2.

2) as for 10.2.

3) as for 10.2

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 shall 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 10.2 step 3d).

10.4 Three-way authentication

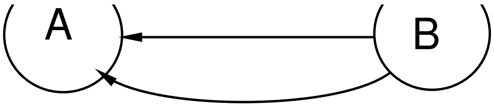


Figure 9 Ä Three-way Authentication

The following steps are involved, as depicted in Figure 9:

1) As for 10.3.

2) As for 10.3. Timestamp t_A may be zero.

3) As for 10.3, except that the timestamp need not be checked.

4) As for 10.3.

5) As for 10.3. Timestamp t_B may be zero.

6) As for 10.3, 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:

   $A\{r_B,B\}$.

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.

11 Management of keys and certificates

11.1 Generation of key pairs

The overall security management policy of an implementation shall define the lifecycle of key pairs, and is thus outside 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 shall 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 shall 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 Directory Specification.

Three ways in which a user's key pair may be produced are:

a) 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 D.

b) The key pair is generated by a third party. The third party shall 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 shall be employed to ensure that the third party and the data operations are free from tampering.

c) The key pair is generated by the CA. This is a special case of b), and the considerations there apply.

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

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

11.2 Management of certificates

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

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

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

The production of a certificate occurs offline and shall 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.

It is important that the transfer of information to the certification authority is not compromised, and suitable physical security measures shall 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 11.1 c) is employed, no secure transfer is needed.

c) If the means of generation of key pairs of 11.1 a) or of 11.1. b) 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.

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 shall 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 shall be given suitable access rights.

Certificates shall 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. Two related points are:

> 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.
>
> 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.

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. Four related points are:

> 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.
>
> 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 know to the CA, certified by the CA.
>
> Both certified lists shall exist, even if empty.
>
> 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 shall then be used to authenticate the user to the Directory.
>
> 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 SYNTAX             CertificateList
        ID                      id-at-certificateRevocationList }

authorityRevocationList     ATTRIBUTE   ::=     {
        WITH SYNTAX             CertificateList
        ID                      id-at-authorityRevocationList }

CertificateList             ::=     SIGNED { SEQUENCE {
        signature                   AlgorithmIdentifier,
        issuer                      Name,
        thisUpdate                  UTCTime,
        nextUpdate                  UTCTime OPTIONAL,
        revokedCertificates         SEQUENCE OF SEQUENCE {
                userCertificate             CertificateSerialNumber,
                revocationDate              UTCTime } OPTIONAL}}
```

Notes

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

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

Annex A Ä Authentication Framework in ASN.1

(This Annex forms an integral part of this Recommendation | International Standard)

This annex includes all of the ASN.1 type, value and information object class definitions contained in this Directory Specification, in the form of the ASN.1 module, ϶**AuthenticationFramework** .

```
AuthenticationFramework {joint-iso-ccitt ds(5) module(1) authenticationFramework(7) 2}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        id-at, informationFramework, upperBounds selectedAttributeTypes, basicAccessControl
                FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}

        Name, ATTRIBUTE
                FROM InformationFramework informationFramework

        ub-user-password
                FROM UpperBounds upperBounds

        AuthenticationLevel
                FROM BasicAccessControl basicAccessControl

        UniqueIdentifier
                FROM SelectedAttributeTypes selectedAttributeTypes      ;

-- types --

Certificate                     ::=     SIGNED { SEQUENCE {
        version                         [0]     Version DEFAULT v1,
        serialNumber                            CertificateSerialNumber,
        signature                               AlgorithmIdentifier,
        issuer                                  Name,
        validity                                Validity,
        subject                                 Name,
        subjectPublicKeyInfo                    SubjectPublicKeyInfo,
        issuerUniqueIdentifier          [1]     IMPLICIT UniqueIdentifier OPTIONAL,
                                                        -- if present, version must be v2
        subjectUniqueIdentifier         [2]     IMPLICIT UniqueIdentifier OPTIONAL
                                                        -- if present, version must be v2 -- }}

Version                         ::=     INTEGER { v1(0), v2(1) }

CertificateSerialNumber         ::=     INTEGER

AlgorithmIdentifier             ::=     SEQUENCE {
        algorithm               ALGORITHM.&id ({SupportedAlgorithms}),
        parameters              ALGORIGHM.&Type ({SupportedAlgorithms}{ @algorithm}) OPTIONAL }
–       Definition of the following information object set is deferred, perhaps to standardized
–       profiles or to protocol implementation conformance statements. The set is required to
–       specify a table constraint on the parameters component of AlgorithmIdentifier.
–       SupportedAlgorithms             ALGORITHM   ::=       { ... | ... }

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

```
SubjectPublicKeyInfo        ::=     SEQUENCE {
        algorithm           AlgorithmIdentifier,
        subjectPublicKey    BIT STRING }

Certificates                ::=     SEQUENCE {
        userCertificate             Certificate,
        certificationPath           ForwardCertificationPath OPTIONAL}

ForwardCertificationPath    ::=     SEQUENCE OF CrossCertificates

CertificationPath           ::=     SEQUENCE {
        userCertificate             Certificate,
        theCACertificates           SEQUENCE OF CertificatePair OPTIONAL}

CrossCertificates           ::=     SET OF Certificate

CertificateList             ::=     SIGNED { SEQUENCE {
        signature                   AlgorithmIdentifier,
        issuer                      Name,
        thisUpdate                  UTCTime,
        nextUpdate                  UTCTime OPTIONAL,
        revokedCertificates         SEQUENCE OF SEQUENCE {
                userCertificate             CertificateSerialNumber,
                revocationDate              UTCTime } OPTIONAL}}

CertificatePair             ::=     SEQUENCE {
        forward         [0] Certificate OPTIONAL,
        reverse         [1] Certificate OPTIONAL
                            -- at least one of the pair shall be present -- }

-- attribute types --

userPassword                ATTRIBUTE   ::=     {
        WITH SYNTAX                     OCTET STRING (SIZE (0..ub-user-password))
        EQUALITY MATCHING RULE          octetStringMatch
        ID                              id-at-userPassword }

userCertificate             ATTRIBUTE   ::=     {
        WITH SYNTAX         Certificate
        ID                  id-at-userCertificate}

cACertificate               ATTRIBUTE   ::=     {
        WITH SYNTAX         Certificate
        ID                  id-at-cAcertificate }

authorityRevocationList     ATTRIBUTE   ::=     {
        WITH SYNTAX         CertificateList
        ID                  id-at-authorityRevocationList }

certificateRevocationList   ATTRIBUTE   ::=     {
        WITH SYNTAX         CertificateList
        ID                  id-at-certificateRevocationList }

crossCertificatePair        ATTRIBUTE   ::=     {
        WITH SYNTAX         CertificatePair
        ID                  id-at-crossCertificatePair }

-- information object classes --

ALGORITHM           ::=     TYPE-IDENTIFIER

-- parameterized types --

HASHED { ToBeHashed }           ::=     OCTET STRING ( CONSTRAINED-BY {
        -- must be the result of applying a hashing procedure to the --
        -- DER-encoded (see 8.7) octets --
        -- of a value of -- ToBeHashed })

ENCRYPTED { ToBeEnciphered }  ::=       BIT STRING ( CONSTRAINED BY {
        -- must be the result of applying an encipherment procedure --
        -- to the BER-encoded octets of a value of -- ToBeEnciphered})
```

```
SIGNED { ToBeSigned }              ::=        SEQUENCE {
          toBeSigned               ToBeSigned,
          COMPONENTS OF            SIGNATURE { ToBeSigned }}

SIGNATURE { OfSignature }          ::=        SEQUENCE {
          algorithmIdentifier      AlgorithmIdentifier,
          encrypted                ENCRYPTED { HASHED { OfSignature }}}

-- object identifier assignments --

id-at-userPassword                 OBJECT IDENTIFIER          ::=       {id-at 35}
id-at-userCertificate              OBJECT IDENTIFIER          ::=       {id-at 36}
id-at-cAcertificate                OBJECT IDENTIFIER          ::=       {id-at 37}
id-at-authorityRevocationList      OBJECT IDENTIFIER          ::=       {id-at 38}
id-at-certificateRevocationList    OBJECT IDENTIFIER          ::=       {id-at 39}
id-at-crossCertificatePair         OBJECT IDENTIFIER          ::=       {id-at 40}

END
```

Annex B – Security requirements [1]

(This annex does not form an integral part of this Recommendation | International Standard)

Many OSI applications, CCITT-defined services and non-CCITT-defined services will have requirements for security. Such requirements derive from the need to protect the transfer of information from a range of potential threats.

B.1 Threats

Some commonly known threats are:

a) *Identity Interception* : the identity of one or more of the users involved in a communication is observed for misuse.

b) *Masquerade* : the pretense by a user to be a different user in order to gain access to information or to acquire additional privileges.

c) *Replay* : the recording and subsequent replay of a communication at some later date.

d) *Data Interception* : the observation of user data during a communication by an unauthorized user.

e) *Manipulation* : the replacement, insertion, deletion or misordering of user data during a communication by an unauthorized user

f) *Repudiation* : the denial by a user of having participated in part or all of a communication.

g) *Denial of Service* : the prevention or interruption of a communication or the delay of time-critical operations

Note This security threat is a more general one and depends on the individual application or on the intention of the unauthorized disruption and is therefore not explicitly within the scope of the authentication framework.

h) *Mis-routing* : the mis-routing of a communication path intended for one user to another

Note Mis-routing will naturally occur in OSI layers 1 through 3. Therefore mis-routing is outside of the scope of the authentication framework. However, it may be possible to avoid the consequences of mis-routing by using appropriate security services as provided within the authentication framework.

i) *Traffic Analysis* : the observation of information about a communication between users (e.g. absence/presence, frequency, direction, sequence, type, amount, etc.).

Note - Traffic analysis threats are naturally not restricted to a certain OSI layer. Therefore Traffic analysis is generally outside the scope of the authentication framework. However, traffic analysis can be partially protected against by generating additional unintelligible traffic (traffic padding), using enciphered or random data.

B.2 Security services

In order to protect against perceived threats, various security services need to be provided. Security services as provided by the authentication framework are performed by means of the security mechanisms described in clause B.3 of this annex.

a) *peer entity authentication* : This service provides corroboration that a user in a certain instance of communication is the one claimed. Two different peer entity authentication services may be requested:

single entity authentication (either *data origin* entity authentication or *data recipient* entity authentication);

mutual authentication, where both users communicating authenticate each other.

When requesting a peer entity authentication service, the two users agree whether their identities shall be protected or not.

The peer entity authentication service is supported by the authentication framework. It can be used to protect against masquerade and replay, concerning the users' identities.

---

[1] For further information see ISO 7498-2

b) *access control* : This service can be used to protect against the unauthorized use of resources. The access control service is provided by the Directory or another application and is therefore not a concern of the authentication framework.

c) *data confidentiality* : This service can be used to provide for protection of data from unauthorized disclosure. The data confidentiality service is supported by the authentication framework. It can be used to protect against data interception.

d) *data integrity* : This service provides proof of the integrity of data in a communication. The data integrity service is supported by the authentication framework. It can be used to detect and protect against manipulation.

e) *non-repudiation* : This service provides proof of the integrity and origin of data both in an unforgeable relationship which can be verified by any third party at any time.

B.3 Security mechanisms

The security mechanisms outline here perform the security services described in clause B.2.

a) *authentication exchange* : There are two grades of authentication mechanism provided by the authentication framework:

simple authentication: relies on the originator supplying its name and password, which are checked by the recipient;

strong authentication: relies on the use of cryptographic techniques to protect the exchange of validating information. In the authentication framework, strong authentication is based upon an asymmetric scheme.

The authentication exchange mechanism is used to support the peer entity authentication service.

b) *encipherment* : The authentication framework envisages the encipherment of data during transfer. Either asymmetric or symmetric schemes may be used. The necessary key exchange for either case is performed either within a preceding authentication exchange or off-line any time before the intended communication. The latter case is outside the scope of the authentication framework. The encipherment mechanism supports the data confidentiality service.

c) *data integrity* : This mechanism involves the encipherment of a compressed string of the relevant data to be transferred. Together with the plain data, this message is sent to the recipient. The recipient repeats the compressing and subsequent encipherment of the plain data and compares the result with that created by the originator to prove integrity.

The data integrity mechanism can be provided by encipherment of the compressed plain data by either an asymmetric scheme or a symmetric scheme. (With the symmetric scheme, compression and encipherment of data might be processed simultaneously). The mechanism is not explicitly provided by the authentication framework. However it is fully provided as a part of the digital signature mechanism (see below) using an asymmetric scheme.

The data integrity mechanism supports the data integrity service. It also partially supports the non-repudiation service (that service also needs the digital signature mechanism for its requirements to be fully met).

d) *digital signature* : This mechanism involves the encipherment, by the originator s secret key, of a compressed string of the relevant data to be transferred. The digital signature together with the plain data is sent to the recipient. Similarly to the case of the data integrity mechanism, this message is processed by the recipient to prove integrity. The digital signature mechanism also proves the authenticity of the originator and the unambiguous relationship between the originator and the data that was transferred.

The authentication framework supports the digital signature mechanism using an asymmetric scheme.

The digital signature mechanism supports the data integrity service and also supports the non-repudiation service.

B.4 Threats protected against by the security services

Table B-1 indicates the security threats which each security service can protect against. The presence of an bullet (•) indicates that a certain security service affords protection against a certain threat.

B.5 Negotiation of security services and mechanisms

The provision of security features during an instance of communication requires the negotiation of the context in which security services are required. This entails agreement on the type of security mechanisms and security parameters that are necessary to provide such security services. The procedures required for negotiating mechanisms and parameters can either be carried out as an integral part of the normal connection establishment procedure or as a separate process. The precise details of these procedures for negotiation are not specified in this annex.

Table B-1 Threats and protection

THREATS	SERVICES			
	Entity Authentication	**Data Confidentiality**	**Data Integrity**	**Non– Repudiation**
Identity Interception	•(if req'd)			
Data Interception		•		
Masquerade	•			
Replay	•(identity)		•(data)	•
Manipulation			•	
Repudiation				•

Annex C Ä An introduction to public key cryptography [1]

(This annex does not form an integral part of this Recommendation | International Standard)

In conventional cryptographic systems, the key used to encipher information by the originator of a secret message is the same as that used to decipher the message by the legitimate recipient.

In public key cryptosystems (PKCS), however, keys come in pairs, one key of which is used for enciphering and the other for deciphering. Each key pair is associated with a particular user X. One of the keys, known as the public key (Xp) is publicly known, and can be used by any user to encipher data. Only X, who possesses the complementary secret key (Xs) may decipher the data. (This is represented notationally by $D = Xs[Xp[D]]$). It is computationally infeasible to derive the secret key from knowledge of the public key. Any user can thus communicate a piece of information which only X can find out, by enciphering it under Xp. By extension, two users can communicate in secret, by using each other's public key to encipher the data, as shown in Figure C.1.

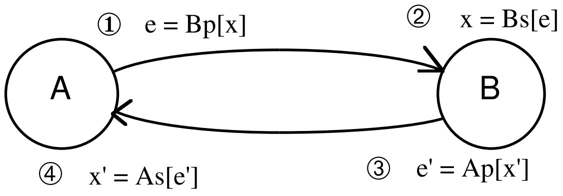


Figure C.1 Ä Use of a PKCS to Exchange Secret Information

User A has public key Ap and secret key As, and user B has another set of keys, Bp and Bs. A and B both know the public keys of each other, but are unaware of the secret key of the other party. A and B may therefore exchange secret information with one another using the following steps (illustrated in Figure B.1).

① A wishes to send some secret information x to B. A therefore enciphers x under B's enciphering key and sends the enciphered information e to B. This is represented by:

$e = Bp[x]$.

② B may now decipher this encipherment e to obtain the information x by using the secret decipherment key Bs. Note that B is the only possessor of Bs, and because this key may never be disclosed or sent, it is impossible for any other party to obtain the information x. The possession of Bs determines the identity of B. The decipherment operation is represented by:

$x = Bs[e]$, or $x = Bs[Bp[x]]$.

③ B may now similarly send some secret information, x', to A, under A's enciphering key, Ap:

$e' = Ap[x']$.

④ A obtains x' by deciphering e':

$x' = As[e']$, or $x' = As[Ap[x']]$.

By this means, A and B have exchanged secret information x and x'. This information may not be obtained by anyone other than A and B, providing that their secret keys are not revealed.

Such an exchange can, as well as transferring secret information between the parties, serve to verify their identities. Specifically, A and B are identified by their possession of the secret deciphering keys, As and Bs respectively. A may determine if B is in possession of the secret deciphering key, Bs, by having returned part of his information x in B's message x'. This indicates to A that communication is taking place with the possessor of Bs. B may similarly test the identity of A.

It is a property of some PKCS that the steps of decipherment and encipherment can be reversed, as in $D=Xp[Xs[D]]$. This allows a piece of information which could only have been originated by X, to be readable by any user (who has

---

[1] For further information, see:

Diffie, W. and Hellman, M.E. New Directions in Cryptography, *IEEE Transactions on Information Theory*, IT-22, No. 6, (November 1976).

possession of Xp). This can therefore be used in the certifying of the source of information, and is the basis for digital signatures. Only PKCS which have this (permutability) property are suitable for use in this authentication framework. One such algorithm is described in Annex D.

Annex D Ä The RSA [1] Public Key Cryptosystem[2]

(This annex does not form an integral part of this Recommendation | International Standard}

D.1 Scope and Field of Application

It is beyond the scope of this annex to discuss RSA fully. However, a brief description is given on the method, which relies on the use of modular exponentiation.

D.2 Definitions

Public Key: the pair of parameters consisting of the Public Exponent and the Arithmetic Modulus;

Note The ASN.1 data element **subjectPublicKey** , defined as **BIT STRING** (see Annex A), should be interpreted in the case of RSA as being of type:

SEQUENCE {INTEGER, INTEGER}

where the first integer is the Arithmetic Modulus and the second is the Public Exponent. The sequence is represented by means of the ASN.1 Basic Encoding Rules.

Secret Key: the pair of parameters consisting of the Secret Exponent and the Arithmetic Modulus.

D.3 Symbols and abbreviations

X,Y	data blocks which are arithmetically less than the modulus
n	the Arithmetic Modulus
e	the Public Exponent
d	the Secret Exponent
p,q	the prime numbers whose product forms the Arithmetic Modulus (n)

Note While the prime numbers are preferably two in number, the use of a Modulus with three- or more prime factors is not precluded.

lcm	least common multiple
mod n	arithmetic modulo n

D.4 Description

This asymmetric algorithm uses the power function for transformation of data blocks such that

$Y = X^e \bmod n$ with $0 \phi X < n$

$X = Y^d \bmod n$ with $0 \phi Y < n$

---

[1] The cryptosystem specified in this annex is widely known as *RSA*, Rivst-Shamir-Adleman.

[2] For further information, see:

General

Rivest, R.L., Shamir, A., and Adleman, L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems, *Communications of the ACM*, 21, 2 (February 1978), 120-126.

Key Generation Reference

Gordon, J Strong RSA Keys, *Electronics Letters*, 20, 5, 514-516.

Decipherment Reference

Quisquater, J.J., and Couvreur, C. Fast Decipherment Algorithm for RSA Public-key Cryptosystems, *Electronics Letters*, 18, 21 (October 14, 1982), 905-907.

which may be satisfied, for example, by

ed mod lcm(p-1,q-1) = 1, or

ed mod (p-1)(q-1) = 1

To effect this process, a data block shall be interpreted as an integer. This is accomplished by considering the entire data block to be an ordered sequence of bits (of length λ). The integer is then formed as the sum of the bits after giving a weight of $2^{\lambda-1}$ to the first bit and dividing by 2 for each subsequent bit (the last bit has a weight of 1).

The data block length should be the largest number of octets containing fewer bits that the modulus. Incomplete blocks should be padded in any way desired. Any number of blocks of additional padding may be added.

D.5 Security requirements

D.5.1 Key lengths

It is recognized that the acceptable key length is likely to change with time, subject to the cost and availability of hardware, the time taken, advances in techniques and the level of security required. It is recommended that a value for the length of n of 512 bits be adopted initially, but subject to *further study*.

D.5.2 Key generation

The security of RSA relies on the difficulty of factorizing n. There are many algorithms for performing this operation, and in order to thwart the use of any currently known technique, the values p and q shall be chosen carefully, according to the following rules (e.g., see footnote 2, эKey Generation Reference):

a) They should be chosen randomly;

b) They should be large;

c) They should be prime;

d) |p-q should be large;

e) (p+1) shall possess a large prime factor;

f) (q+1) shall possess a large prime factor;

g) (p-1) shall possess a large prime factor, say r;

h) (q-1) shall possess a large prime factor, say s;

i) (r-1) shall possess a large prime factor;

j) (s-1) shall possess a large prime factor.

After generating the public and secret keys, e.g. эXp and эXs as defined in clauses 3 and 4 of this Directory Specification, which consist of d, e and n, the values p and q together with all other data produced such as the product (p-1)(q-1) and the large prime factors should preferably be destroyed. However, keeping p and q locally can improve throughput in decryption by two to four times. The decision to keep p and q is considered to be a local matter (see footnote 2 эDecipherment Reference).

It must be ensured that $e > \log_2(n)$. If not, then the simple operation of taking the integer e th root of a ciphertext block will disclose the plaintext.

D.6 Public exponent

The Public Exponent (e) could be common to the whole environment, in order to minimize the length of that part of the public key that actually has to be distributed, in order to reduce transmission capacity and complexity of transformation (see note 1).

Exponent e should be large enough but such that exponentiation can be performed efficiently with regard to processing time and storage capacity. If a fixed public exponent e is desired, there are notable merits for the use of the Fermat Number F_4 (see note 2).

$$F_4 = 2^{2^4} + 1$$

= 65537 decimal, and

= 1 0000 0000 0000 0001 binary

Notes

1 Although both Modulus n and Exponent e are public, the Modulus should not be the part which is common to a group of users. Knowledge of Modulus эn , Public Exponent эe , and Secret Exponent эd is sufficient to determine the factorization of эn . Therefore, if the modulus were common, everyone could deduce its factors, thereby determining everyone else s secret exponent.

2 The fixed exponent should be large and prime but it should also provide efficient processing. Fermat Number F_4 meets these requirements, e.g. authentication takes only 17 multiplications and is on the average 30 times faster than decipherment.

D.7 Conformance

Whilst this annex specifies an algorithm for the public and secret functions, it does not define the method whereby the calculations are carried out; therefore there may be different products which comply with this annex and are mutually compatible.

Annex E Ä Hash functions

(This annex does not form an integral part of this Recommendation | International Standard)

The square-mod hash function that was described in this annex in the 1988 edition of this Directory Specification is deprecated.

E.1 Requirements for hash functions

To use a hash function as a secure one-way function, it shall not be possible to obtain easily the same hash result from different combinations of the input message.

A strong hash function shall meet the following requirements:

a) The hash function shall be one-way, i.e., given any possible hash result it shall be computationally infeasible to construct an input message which hashes to this result.

b) The hash function shall be collision-free, i.e., it shall be computationally infeasible to construct two distinct input messages which hash to the same result.

Annex F – Threats protected against by the strong authentication method

(This annex does not form an integral part of this Recommendation | International Standard)

The strong authentication method described in this Directory Specification offers protection against the threats as described in Annex B for strong authentication.

In addition, there is a range of potential threats that are specific to the strong authentication method itself. These are:

Compromise of the user's secret key - one of the basic principles of strong authentication is that the user's secret key remain secure. A number of practical methods are available for the user to hold his secret key in a manner that provides adequate security. The consequences of the compromise is limited to subversion of communication involving that user.

Compromise of the CA's secret key - that the secret key of a CA remain secure is also a basic principle of strong authentication. Physical security and "need to know" methods apply. The consequences of the compromise are limited to subversion of communication involving any user certified by that CA.

Misleading CA into producing an invalid certificate - the fact that CAs are off-line affords some protection. The onus is on the CA to check that purported strong credentials are valid before creating a certificate. The consequences of the compromise are limited to subversion of communication involving the user for whom the certificate was created, and anyone impacted by the invalid certificate.

Collusion between a rogue CA and user - such a collusive attack will defeat the method. This would constitute a betrayal of the trust placed in the CA. The consequences of a rogue CA are limited to subversion of communication involving any user certified by that CA.

Forging of a certificate - the strong authentication method protects against the forging of a certificate by having the CA sign it. The method depends on maintaining the secrecy of the CA's secret key.

Forging of a token - the strong authentication method protects against the forging of a token by having the sender sign it. The method depends on maintaining the secrecy of the sender's secret key.

Replay of a token - the one- and two-way authentication methods protect against the replay of a token by the inclusion of a timestamp in the token. The three-way method does so by checking the random numbers.

Attack on the cryptographic system- the likelihood of effective cryptanalysis of the system, based on advances in computational number theory and leading to the need for a greater key length are reasonably predictable.

Annex G Ä Data confidentiality

(This annex does not form an integral part of this Recommendation | International Standard)

G.1 Introduction

The process of data confidentiality can be initiated after the necessary keys for encipherment have been exchanged. This might be provided by a preceding authentication exchange as described in clause 9 or by some other key exchange process, the latter being outside the scope of this Directory Specification.

Data confidentiality can be provided either by the application of an asymmetric or symmetric enciphering scheme.

G.2 Data confidentiality by asymmetric encipherment

In this case Data Confidentiality is performed by means of an originator enciphering the data to be sent using the intended recipient's public key: the recipient shall then decipher it using its secret key.

G.3 Data confidentiality by symmetric encipherment

In this case Data Confidentiality is achieved by the use of a symmetric enciphering algorithm. Its choice is outside the scope of the authentication framework.

Where an authentication exchange according to clause 9 has been carried out by the two parties involved, then a key for the usage of a symmetric algorithm can be derived. Choosing secret keys depends on the transformation to be used. The parties shall be sure that they are strong keys. This Directory Specification does not specify how this choice is made, although clearly this would need to be agreed by the parties concerned, or specified in other standards.

Annex H Ä Reference definition of algorithm object identifiers

(This annex forms an integral part of this Recommendation | International Standard)

This annex defines object identifiers assigned to authentication and encryption algorithms, in the absence of a formal register. It is intended to make use of such a register as it becomes available. The definitions take the form of the ASN.1 module, э**AlgorithmObjectIdentifiers** .

```
AlgorithmObjectIdentifiers {joint-iso-ccitt ds(5) module(1) algorithmObjectIdentifiers(8) 2}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        algorithm, authenticationFramework
                FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}
        ALGORITHM
                FROM AuthenticationFramework authenticationFramework            ;

-- categories of object identifier --

encryptionAlgorithm     OBJECT IDENTIFIER       ::=     {algorithm 1}
hashAlgorithm           OBJECT IDENTIFIER       ::=     {algorithm 2}
signatureAlgorithm      OBJECT IDENTIFIER       ::=     {algorithm 3}

-- synonyms --

id-ea   OBJECT IDENTIFIER       ::=     encryptionAlgorithm
id-ha   OBJECT IDENTIFIER       ::=     hashAlgorithm
id-sa   OBJECT IDENTIFIER       ::=     signatureAlgorithm

-- algorithms --

rsa     ALGORITHM               ::=     {
        KeySize
        IDENTIFIED BY   id-ea-rsa }

KeySize         ::=     INTEGER

-- object identifier assignments --

id-ea-rsa               OBJECT IDENTIFIER       ::=     {id-ea 1}

-- the following object identifier assignments reserve values assigned to deprecated functions

id-ha-sqMod-n           OBJECT IDENTIFIER       ::=     {id-ha 1}
id-sa-sqMod-nWithRSA    OBJECT IDENTIFIER       ::=     {id-sa 1}

END
```

Annex J Ä Amendments & corrigenda

(This annex does not form an integral part of this Recommendation | International Standard)

This edition of this Directory Specification includes the following amendments:

Amendment 1 for Access Control

This edition of this Directory Specification includes the following technical corrigenda correcting the defects in the following defect reports (some parts of some of the following Technical Corrigenda may have been subsumed by the amendments that formed this edition of this Directory Specification):

Technical Corrigendum 1 (covering Defect Reports 009, 015, 016, 019, 031).