The drawings contained in this Recommendation have been done in AUTOCAD

ANNEX A

(to Recommendation X.509)

Security requirements

This Annex does not form an integral part of this Recommendation.

[Additional material relevant to this topic can be found in OSI 7498 - Information Processing Systems - OSI Reference Model - Part 2, Security Architecture.]

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.

A.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 - 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.

A.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 A.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 will 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 user's identities;

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.

A.3 Security mechanisms

The security mechanisms outlined here perform the security services described in A.2.

a) autbhentication exchange: there are two grades of 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 results 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 explicitely 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 requirement 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.

A.4 Threats protected against by the security services

The table at the end of this Annex indicates the security threats which each security service can protect against. The presence of an asterisk (*) indicates that a certain security service affords protection against a certain threat.

A.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.

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

ANNEX B

(to Recommendation X.509)

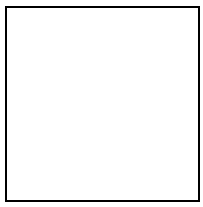
An introduction to public key cryptography

This Annex does not form an integral part of this Recommendation.

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 B-1/X.509.

FIGURE B-1/X.509 - T0704470-88



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/X.509):

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, xw', to A, under A's enciphering key, Ap:

   ew' = Ap[xw'].

4) A obtains xw' by deciphering ew':

   xw' = As[ew'], or xw' = As[Ap[xw']].

By this means, A and B have exchanged secret information x and xw'. 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 xw'. 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 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 C.

For further information, see:

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

ANNEX C

(to Recommendation X.509)

The RSA public key cryptosystem

This Annex does not form an integral part of this Recommendation.

Note - The cryptosystem specified in this Annex, which was invented by R. L. Rivest, A. Shamir and L. Adleman, is widely known as "RSA".

C.1 Scope and field of application

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

C.2 References

For further information, see:

1) General

RIVEST, R. L., SHAMIR, A. and ADLEMAN, L. (February 1978) - A Method for Obtaining Digital Signatures and Public-key Cryptosystems, Communications of the ACM, 21, 2, 120-126.

2) Key Generation Reference

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

3) Decipherment Reference

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

C.3 Definitions

a) 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 G), 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.

b) secret key: the pair of parameters consisting of the Secret Exponent and the Arithmetic Modulus.

C.4 Symbols and abbreviations

X,Ydata blocks which are arithmetically less than the modulus

n the Arithmetic Modulus

e the Public Exponent

d the Secret Exponent

p,qthe 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.

mod n arithmetic modulo n.

C.5 Description

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

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

$X = Y^d \bmod n \quad 0 \leq Y < n$

which may be satisfied, for example, by

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

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

To effect this process, a data block must be interpreted as an integer. This is accomplished by considering the entire data block to be an ordered sequence of bits (of length l, say). The integer is then formed as the sum of the bits after giving a weight of 2^{l-1} to the first bit and dividing the weight 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 than the modulus. Incomplete blocks should be padded in any way desired. Any number of blocks of additional padding may be added.

C.6 Security requirements

C.6.1 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.

C.6.2 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 must be chosen carefully, according to the following rules [e.g. see Reference 2), Section C.2]:

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) must possess a large prime factor;

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

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

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

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

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

After generating the public and secret keys, e.g. "Xp" and "Xs" as defined in 3.3 and 4.1 of this Recommendation 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 [Reference 3)].

It must be ensured that $e > \log_2(n)$ in order to prevent attack by taking the e'th root mod n to disclose the plaintext.

C.7 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.

Note 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 "n", Public Exponent "e" and Secret Exponent "d" is sufficient to determine the factorization of "n". Therefore if the modulus was common, everyone could deduce its factors, thereby finding everyone else's secret exponent.

Note 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.

C.8 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 D

(to Recommendation X.509)

Hash functions

This Annex does not form an integral part of this Recommendation.

D.1 Requirements for hash functions

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

A strong hash function will meet the following requirements:

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

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

D.2 Description of a hash function

The following hash function ("square-mod n") performs the compression of the data on a block by block basis.

Hashing is done in three major steps:

1) The string of data to be hashed is divided into blocks B of equal length. This length is determined by the characteristics of the asymmetric cryptosystem used for signing. With the RSA cryptosystem, this length (in octets) is the largest integer, l, such that, with modulus n, $16\ l < \log_2 n$.

2) For non-invertibility reasons each octet of the block is split in half. Each of the halves is headed ("padded") by binary ones. By this zoning, stiffness or redundancy is introduced that increases the non-invertibility property of the hash function considerably. Each block generated in step 1 is spread to the length of the modulus n.

3) Each block resulting from step 2 is added to the previous block modulo 2, squared, and reduced modulo n, until all m blocks are processed.

   The result is thus the value H_m, where

   $H_0 = 0$

   $H_i = (H_{i-1} + B_i)^2 \bmod n$, for $1 \leq i \leq m$

If the last block of the data to be hashed is incomplete, it is padded with "l"s.

ANNEX E

(to Recommendation X.509)

Threats protected against by the strong authentication method

This Annex does not form an integral part of this Recommendation.

The strong authentication method described in this Recommendation offers protection against the threats as described in Annex A 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 are 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 F

(to Recommendation X.509)

Data confidentiality

This Annex does not form an integral part of this Recommendation.

F.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 9 or by some other key exchange process, the latter being outside the scope of this document.

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

F.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 will then decipher it using its secret key.

F.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 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 must be sure that they are strong keys. This Recommendation 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 G

(to Recommendation X.509)

Authentication framework in ASN.1

This Annex is part of the Recommendation.

This Annex includes all of the ASN.1 type, macro and value definitions contained in this Recommendation in the form of the ASN.1 module, "**AuthenticationFramework**".

```
AuthenticationFramework {joint-iso-ccitt ds(5) modules(1)
                         authenticationFramework(7)}

DEFINITIONS ::=
BEGIN

EXPORTS     AlgorithmIdentifier, AuthorityRevocationList, CACertificate, Certificate,
            Certificates, CertificationPath, CertificateRevocationList, UserCertificate,
            CrossCertificatePair, UserPassword, ALGORITHM,
            ENCRYPTED, PROTECTED, SIGNATURE, SIGNED;

IMPORTS
    informationFramework, selectedAttributeTypes, upperBounds
        FROM UsefulDefinitions {joint-iso-ccitt ds(5)modules(1)
                usefulDefinitions(0)}

    Name, ATTRIBUTE,ATTRIBUTE-SYNTAX
        FROM InformationFramework informationFramework

ub-user-passwordFROM Upper Bounds upperBounds;

-- types

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

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

Validity       ::=     SEQUENCE{
                         notBefore              UTCTime
                         notAfter               UTCTime}

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

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

Certificates   ::=     SEQUENCE{
                         certificate            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,
                                lastUpdate              UTCTime,
                                revokedCertificates       SIGNEDSEQUENCE OF SEQUENCE{
                                        signature       AlgorithmIdentifier,
                                        issuer          Name,
                                        userCertificate SerialNumber,
                                        revocationDate  UTCTime}
                                                OPTIONAL}

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

--attribute types

UserCertificate             ::=     ATTRIBUTE
                                    WITH ATTRIBUTE-SYNTAXCertificate

CACertificate               ::=     ATTRIBUTE
                                    WITH ATTRIBUTE-SYNTAXCertificate

CrossCertificatePair        ::=     ATTRIBUTE
                                        WITH ATTRIBUTE-SYNTAXCertificatePair

CertificateRevocationList   ::=     ATTRIBUTE
                                        WITH ATTRIBUTE-SYNTAXCertificateList

AuthorityRevocationList     ::=     ATTRIBUTE
                                        WITH ATTRIBUTE-SYNTAXCertificateList

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

-- macros

ALGORITHM MACRO         ::=
BEGIN
TYPE NOTATION           ::=     "PARAMETER" type
VALUE NOTATION          ::=     value(VALUE OBJECT IDENTIFIER)
END -- of ALGORITHM

ENCRYPTED MACRO         ::=
BEGIN

TYPE NOTATION           ::=     type (ToBeEnciphered)

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

END
```

```
SIGNED MACRO                ::=
BEGIN
TYPE NOTATION               ::=     type (ToBeSigned)
VALUE NOTATION              ::=     value(VALUE

SEQUENCE{
    ToBeSigned,
    AlgorithmIdentifier, -- of the algorithm used to generate the signature
    ENCRYPTED OCTET STRING
    - -where the octet string is the result
    - -of the hashing of the value of
    - -"ToBeSigned"- -}

                            )

END -- of SIGNED

SIGNATURE MACRO             ::=
BEGIN
TYPE NOTATION               ::=     type (OfSignature)
VALUE NOTATION              ::=     value(VALUE

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

END -- of SIGNATURE

PROTECTED MACRO             ::= SIGNATURE

END- -of Authentication Framework Definitions
```

ANNEX H

(to Recommendation X.509)

Reference Definition of algorithm object identifiers

This Annex is not an integral part of the Recommendation.

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, **AlgorithmObjectIdentifiers**.

```
AlgorithmObjectIdentifiers      {joint-iso-ccitt ds(5) modules(1)
                                algorithmObjectIdentifiers(8)}

DEFINITIONS ::=
BEGIN

EXPORTS
    encryptionAlgorithm, hashAlgorithm, signatureAlgorithm,
    rsa,squareMod-n,sqMod-nWithRSA;

IMPORTS
    algorithm,authenticationFramework
        FROM UsefulDefinitions      {joint-iso-ccitt ds(5)modules(1)
                                    usefulDefinitions(0)}
```

```
    ALGORITHM      FROM AuthenticationFramework authenticationFramework;
-- categories of object identifier
encryptionAlgorithm OBJECT IDENTIFIER          ::=      {algorithm 1}
hashAlgorithm OBJECT IDENTIFIER                ::=      {algorithm 2}
signatureAlgorithm OBJECT IDENTIFIER           ::=      {algorithm 3}
-- algorithms
rsa  ALGORITHM
     PARAMETER KeySize
     ::=  {encryptionAlgorithm 1}
KeySize ::=  INTEGER
sqMod-n ALGORITHM
     PARAMETER BlockSize
     ::= {hashAlgorithm 1}
BlockSize ::=  INTEGER
sqMod-nWithRSA ALGORITHM
     PARAMETER KeyAndBlockSize
     ::= {signatureAlgorithm 1}
KeyAndBlockSize ::= INTEGER
END -- of Algorithm Object Identifier Definitions
```