FIPS PUB JJJ

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

DRAFT 1995 March 13 DRAFT

U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology

STANDARD FOR PUBLIC KEY CRYPTOGRAPHIC ENTITY AUTHENTICATION MECHANISMS

(DRAFT)

CATEGORY: COMPUTER SECURITY
SUBCATEGORY: ACCESS CONTROL

U.S. DEPARTMENT OF COMMERCE, Ronald H. Brown, *Secretary*
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY,

Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official series of publications relating to standards and guidelines adopted and promulgated under the provisions of Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235. These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal Government. The NIST, through its Computer Systems Laboratory, provides leadership, technical guidance, and coordination of Government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

James H. Burrows, *Director*
Computer Systems Laboratory

Abstract

This standard specifies two challenge-response mechanisms by which entities in a computer system may authenticate their identities to one another. These mechanisms are used during session initiation, and at any other time that entity authentication is necessary. Depending on which protocol is implemented, either one or both entities involved may be authenticated. The defined protocols are derived from an international standard for entity authentication based on public key cryptography using digital signatures and random number challenges.

Public key based authentication has an advantage over many other authentication schemes because no secret information has to be shared by the entities involved in the exchange. A user (claimant) attempting to authenticate oneself must use a private key to digitally sign a random number challenge issued by the verifying entity. This random number is a time variant parameter which is unique to the authentication exchange. If the verifier can successfully verify the signed response using the claimant's public key, then the claimant has been successfully authenticated.

Key words: authentication, challenge-response, computer security, cryptographic modules, cryptography, Federal Information Processing Standard (FIPS), telecommunications security.

Federal Information Processing Standards Publication JJJ

DRAFT 1995 March 13 DRAFT

ANNOUNCING THE

DRAFT
STANDARD FOR PUBLIC KEY CRYPTOGRAPHIC
ENTITY AUTHENTICATION MECHANISMS

Federal Information Processing Standards (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

1. Name of Standard. Standard for Public Key Cryptographic Entity Authentication Mechanisms (FIPS PUB JJJ).

2. Category of Standard. Computer Security, **Subcategory** Access Control.

3. Explanation. This standard specifies two challenge-response mechanisms by which entities in a computer system may authenticate their identities to one another. These mechanisms are used during session initiation, and at any other time that entity authentication is necessary. Depending on which protocol is implemented, either one or both entities involved may be authenticated. The defined protocols are derived from an international standard for entity authentication based on public key cryptography using digital signatures and random number challenges.

Public key based authentication has an advantage over many other authentication schemes because no secret information has to be shared by the entities involved in the exchange. A user (claimant) attempting to authenticate oneself must use a private key to digitally sign a random number challenge issued by the verifying entity. This random number is a time variant parameter which is unique to the authentication exchange. If the verifier can successfully verify the signed response using the claimant's public key, then the claimant has been successfully authenticated.

4. Approving Authority. Secretary of Commerce.

5. Maintenance Agency. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory.

6. Cross Index.

a. FIPS PUB 140-1, Security Requirements for Cryptographic Modules.
b. FIPS PUB 171, Key Management Using ANSI X9.17.

c. FIPS PUB 180, Secure Hash Standard.
d. FIPS PUB 186, Digital Signature Standard.
e. FIPS PUB 190, Guideline for the Use of Advanced Authentication Technology Alternatives.
f. ISO/IEC 9798-1:1991, Information technology - Security techniques - Entity authentication mechanisms - Part 1: General model.
g. ISO/IEC 9798-3:1993, Information technology - Security techniques - Entity authentication mechanisms - Part 3: Entity authentication using a public key algorithm.

Other NIST publications may be applicable to the implementation and use of this standard. A list (NIST Publications List 91) of currently available computer security publications, including ordering information, can be obtained from NIST.

7. Applicability. This standard is applicable to all Federal departments and agencies that use public key based authentication systems to protect unclassified information within computer and digital telecommunications systems that are not subject to Section 2315 of Title 10, U.S. Code, or Section 3502(2) of Title 44, U.S. Code. This standard shall be used by all Federal departments and agencies in designing, acquiring and implementing public key based, challenge-response authentication systems at the application layer within computer and digital telecommunications systems. This includes all systems that Federal departments and agencies operate or that are operated for them under contract. In addition, this standard may be used at other layers within computer and digital telecommunications systems.

This standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it is either cost effective or provides interoperability for commercial and private organizations.

8. Applications. Numerous applications can benefit from the incorporation of public key authentication. Networking applications that require remote login will be able to authenticate clients who have not previously registered with the host, since secret material (e.g., a password) does not have to be exchanged beforehand. Also, point-to-point authentication can take place between users who are unknown to one another. The authentication mechanisms in this standard may be used in conjunction with other public key based systems (e.g., a public key infrastructure that uses public key certificates) to enhance the security of a computer system.

9. Specifications. Federal Information Processing Standard (FIPS) JJJ, *Standard for Public Key Cryptographic Entity Authentication Mechanisms* (affixed).

10. Implementations. The authentication mechanisms described in this standard may be implemented in software, firmware, hardware, or any combination thereof.

11. Export Control. Implementations of this standard are subject to Federal Government export controls as specified in Title 15, Code of Federal Regulations, Parts 768 through 799.

Exporters are advised to contact the Department of Commerce, Bureau of Export Administration, for more information.

12. Implementation Schedule. This standard becomes effective (insert six months after approval by the Secretary of Commerce).

13. Qualifications. The authentication technology described in this standard is based upon information provided by sources within the Federal Government and private industry. Authentication systems are designed to protect against adversaries mounting cost-effective attacks on unclassified government or commercial data (e.g., hackers, organized crime, economic competitors). The primary goal in designing an effective security system is to make the cost of any attack greater than the possible payoff.

14. Waivers. Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). The head of such agency may re-delegate such authority only to a senior official designated pursuant to section 3506(b) of Title 44, U.S. Code. Waivers shall be granted only when:

a. Compliance with a standard would adversely affect the accomplishment of the mission of an operator of a Federal computer system, or

b. Cause a major adverse financial impact on the operator which is not offset by Governmentwide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision which explains the basis on which the agency head made the required finding(s). A copy of each such decision, with procurement sensitive classified portions clearly identified, shall be sent to: National Institute of Standards and Technology, ATTN: FIPS Waiver Decisions, Technology Building, Room B-154, Gaithersburg, MD 20899.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Governmental Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any supporting and accompanying documents, with such deletions as the agency is authorized and

decides to make under 5 U.S.C. Section 552(b), shall be part of the procurement documentation and retained by the agency.

15. Where to Obtain Copies. Copies of this publication are available for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. When ordering, refer to Federal Information Processing Standards Publication JJJ (FIPS PUB JJJ), and identify the title. When microfiche is desired, this should be specified. Payment may be made by check, money order, credit card, or deposit account.

Federal Information
Processing Standards Publication JJJ

1995 March 13

DRAFT

Specifications for the

DRAFT STANDARD FOR PUBLIC KEY CRYPTOGRAPHIC ENTITY AUTHENTICATION MECHANISMS

1. INTRODUCTION

This publication specifies two mechanisms for entity authentication that use a public key cryptographic algorithm for generating and verifying digital signatures. One entity can prove its identity to another entity by using a private key to generate a digital signature on a random challenge. The use of cryptography provides for strong authentication, which does not require authenticating entities to share secret information. Federal government computer systems which implement this standard shall generate and verify digital signatures in accordance with a FIPS approved public key digital signature algorithm (e.g., FIPS 186, *Digital Signature Standard*).

International Standard ISO/IEC 9798-3:1993, *Entity authentication using a public key algorithm*, serves as the basis for the authentication mechanisms defined in Section 3 below. That international standard defines five different protocols, concerned with unilateral and mutual entity authentication, that make use of a public key cryptographic algorithm. In particular, digital signatures are used in determining an entity's authenticity. Only one mechanism for each unilateral and mutual authentication was selected from ISO/IEC 9798-3. Certain authentication token fields and mechanism steps are described in greater detail in this standard than in ISO/IEC 9798-3

2. GENERAL

2.1 Scope

The protocols defined in this standard make use of a digital signature algorithm for the generation of authentication tokens. All claimants involved in an authentication exchange *must* have the

ability to generate digital signatures, and only principals that attempt to verify an authentication token are required to have a digital signature verification capability. The authentication protocols defined in this standard are independent of the nature of the authenticating entities (e.g., for mutual authentication, the same protocol is used for human-human, human-process, and process-process authentication). In situations where a human is involved as a principal, a two-step sequence usually takes place, due to the complexity of cryptography. A human user first authenticates oneself to a cryptographic module, which then acts as a claimant and performs the actual signature generation and/or verification on behalf of the human user (See Section 3.1.1).

Public key certificates are not required by this standard, and the utilization of a public key infrastructure lies outside the scope of this standard. Whether or not public key certificates are used in the implementation of this standard, each public/private key pair shall be bound to a particular entity. Such a binding may be performed by a verifier or a third party that is trusted by the verifier. Additionally, trusted third parties *may* be used in conjunction with this standard to distribute delegation keys, login tickets, public keys, or public key certificates. Delegation keys are keys issued by one entity to let another entity act upon the issuer's behalf; login tickets are generated by a third party, and used by one entity to authenticate to another. Neither of these items is used in this standard, but may be included in the optional text fields provided in the exchanged authentication tokens. Trusted third parties may provide any other necessary services, such as a courier service to transport valid public keys.

Since the protocols defined in this standard utilize (pseudo)random integer values for the authentication tokens' time variant parameters, authenticating entities do not have to use synchronized clocks to verify the timeliness of authentication tokens. A date-time value *may* be included in an authentication token; however, it shall *not* replace the (pseudo)random integer value as the authentication token's time variant parameter. Authentication exchanges using date-time values and sequence numbers were not chosen for this standard due to their requirements of maintaining synchronized clocks and sequence number log tables, respectively. Random number challenges are generally easier to use in widely distributed environments where entities don't necessarily know one another prior to authentication.

A naming convention for the entities involved in authentication exchanges described in Section 3 is not defined in this standard. However, the names of participating authentication entities should be established prior to the execution of either of those mechanisms. Each entity being authenticated must have an identifier which can be used to uniquely associate it with a public/private key pair.

Biometric authentication techniques are not included in the authentication exchanges found in Section 3. Information pertaining to biometrics, or a result of performing a biometric live scan, may be included in an authentication token's optional data field, and that data may be used in determining authentication success or failure. However, the lack of that biometric functionality, in an implementation of either protocol in Section 3, will not prevent that implementation from meeting the requirements of this standard.

Upon completion of either of the authentication exchange mechanisms in Section 3, the principal performing the final verification step may send an acknowledgement of verification success or failure to the other principal involved in the exchange. Although the format and use of such an acknowledgement is not within the scope of this standard, in certain implementations it may be necessary to inform the other principal of a(n) (un)successful authentication.

2.2 Threats

The protocols defined in this document address several threats, including masquerade, password compromise, replay attacks, and signing of pre-defined data. Using challenges and digital signatures eliminates the need for passwords, along with the threat of their compromise. Such a compromise would allow an attacker to use the same information to repeatedly authenticate. By using a private key (which never leaves a claimant's sole control) to generate digital signatures for authentication, it becomes computationally infeasible for an attacker to masquerade as another entity. The use of random integer challenges also prevents an intruder from copying an authentication token signed by another user and replaying it successfully at a later time. Of course, the security of replay prevention hinges on the generation of random number challenges which have a low probability of being repeated. Finally, by including a random integer of its own in an authentication token, a claimant can preclude the signing of only data that is pre-defined by the verifier. If a claimant uses a private key for more than just signing authentication tokens, for example, then a verifier could maliciously create a challenge consisting of information which is meaningful in another context. This can be prevented when the claimant signs the challenge and unpredictable, meaningless data - a random integer.

If time stamps are included in tokens, then a verifier has the potential to detect a forced delay. Such a threat occurs when an attacker purposefully disrupts an authentication session by delaying the transmission of an authentication token. One way to detect that attack is for a verifier to compare a token's time stamp with a current time value. If the difference between the two values falls outside of a certain window, then the verifier may determine that a forced delay attack could have occurred, and the authentication exchange may be terminated. Synchronized clocks may be required, depending on the particular application.

Authentication protocols defined in Section 3 do not address other threats, which include denial of service (other than the optional detection of forced delays), session capture, transmission modification, and the use of another entity's compromised private key. No aspect of the authentication tokens or protocols preclude another entity from rerouting or modifying authentication transmissions. Maintaining confidentiality of one's private key is of utmost importance, and failure to do so may result in one entity masquerading as another entity by using the latter's private key for authentication. Procedures for dealing with these problems are outside the scope of this standard.

2.3 Definitions and notation

Definitions

Some of the following definitions are taken from terminology defined in ISO/IEC 9798-1:1991, which describes the general model for the ISO/IEC 9798 series of authentication standards. Other definitions are from FIPS 140-1, *Security Requirements for Cryptographic Modules*, Section 2.1.

Authentication token (viz., *token*): authentication information conveyed during an authentication exchange.

Claimant: an entity which is or represents a principal for the purposes of authentication, together with the functions involved in an authentication exchange on behalf of that entity. A claimant acting on behalf of a principal must include the functions necessary for engaging in an authentication exchange. (e.g., a smart token (claimant) can act on behalf of a human user (principal))

Cryptographic module: the set of hardware, software, firmware, or some combination thereof that implements cryptographic logic or processes, including cryptographic algorithms, and is contained within the cryptographic boundary of the module. (See FIPS 140-1)

Digital signature (viz., *signature*): a nonforgeable transformation of data that allows the proof of the source (with nonrepudiation) and the verification of the integrity of that data.

Entity: any participant in an authentication exchange; such a participant may be human or non-human, and may take the role of a claimant or verifier.

FIPS approved security method: a security method (e.g., cryptographic algorithm, cryptographic key generation algorithm or key distribution technique, authentication technique, or evaluation criteria) that is either a) specified in a FIPS, or b) adopted in a FIPS and specified either in an appendix to the FIPS or in a document referenced by the FIPS.

Initiator: the entity that initiates an authentication exchange.

Principal: an entity whose identity can be authenticated.

Private key: a cryptographic key used with a public key cryptographic algorithm, which is uniquely associated with an entity, and not made public; it is used to generate a digital signature; this key is mathematically linked with a corresponding public key.

Public key: a cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity, and which may be made public; it is used to verify a digital signature; this key is mathematically linked with a corresponding private key.

Public key certificate: a set of data that unambiguously identifies an entity, contains the entity's public key, and is digitally signed by a trusted party.

Responder: the entity that responds to the initiator of the authentication exchange.

Signed data: authentication token data on which a digital signature is generated.

Unsigned data: data included in an authentication token, but which is not digitally signed.

Verifier: an entity which is or represents the entity requiring an authenticated identity. A verifier includes the functions necessary for engaging in authentication exchanges.

Notation

In Section 3, the following notation is used in describing parts of the authentication exchanges. Most of this notation is used in ISO/IEC 9798-3.

A - the distinguishing identifier (name) of entity A.

B - the distinguishing identifier (name) of entity B.

S_X - a private key associated with entity X, used in generating a digital signature.

R_X - a random number issued by entity X.

T_X - a time stamp issued by entity X.

K_X - key exchange information issued by entity X, which is used in the establishment of a cryptographic key.

X || Y - the result of the concatenation of data items X and Y in that order.

CertX - a trusted third party's certificate for entity X, that binds X with a public-private key pair; may also indicate a chain of certificates beginning or ending with CertX.

TextN - data of unspecified format or length that may be included in a token.

TokenXY - a token sent from entity X to entity Y.

$TokenXY_i$ - the i^{th} token sent from entity X to entity Y.

$sS_X(Z)$ - the digital signature of data Z using the private key S_X; Z is referred to as "signed data".

[A] - indicates that item A is an optional element.

[OPTIONAL] - indicates that the accompanying step is optional; it does not have to be executed in order for the implementation to conform with this standard.

3 ENTITY AUTHENTICATION MECHANISMS

3.1 Pre-exchange issues

3.1.1 Cryptographic modules

Certain actions should take place before initiating either of the authentication exchanges described in this standard. Each entity involved in an authentication exchange must have a cryptographic module which can generate (pseudo)random integers. Entities acting as claimants must have a digital signature generation capability, and verifiers must have a digital signature verification capability. All cryptographic modules used to protect unclassified information shall conform, at a minimum, to security level one as defined in FIPS 140-1. However, the cryptographic module shall employ identity-based authentication mechanisms "in order to verify the authorization of the operator to assume the desired roles and to request corresponding services" [FIPS140-1].

If an entity is to be authenticated, its cryptographic module, acting as the claimant, must also have a digital signature generation capability. A public/private key pair compliant with a FIPS approved public key digital signature algorithm (e.g., FIPS 186, *Digital Signature Standard*) must be possessed by each entity which is to be authenticated, and the plaintext private key shall be stored in part of the cryptographic module. In order to protect the key material in the cryptographic module, the module shall employ identity-based operator authentication (See FIPS 140-1, Section 4.3.3).

3.1.2 Challenges

To create (pseudo)random integer challenges for the two authentication exchanges described in this standard, only FIPS-approved (pseudo)random integer generators shall be used. Several may be found in Appendix 3 of FIPS 186, Digital Signature Standard (DSS), and another is supplied in Appendix C of ANSI X9.17, "Financial Institution Key Management (Wholesale)".

3.1.3 Identifiers

Distinguishing identifiers shall be determined for all entities which have the potential for communicating with one another, before the authentication exchange is initiated. A naming convention shall be established such that a verifier can differentiate between all entities which act as claimants, and each claimant uses only one distinguished identifier for each verifier. If certificates are used, the naming convention used in identifying entities to one another during the authentication exchange does not have to be the same as the certificate naming convention.

However, each entity must have some means of correlating a name in a certificate with the identifier used during authentication.

In Section 3, a token identifier may be included with each transmission of an authentication exchange. Such an identifier may be used to indicate the type of token being sent, and the authentication exchange to which it belongs. The format for these token identifiers is outside the scope of this standard.

3.1.4 Certificate usage

If certificates are used in the authentication process, then they must be generated prior to the authentication exchange, and should be maintained so that they are readily accessible to any entity that wishes to authenticate another's identity. Typically, a certificate is generated by a trusted third party and then either distributed or stored where prospective authenticating entities have access to it. When an entity wishes to obtain a certificate for verification purposes, it may be retrieved from a directory service, a local cache, or else it may be passed during an authentication exchange. If certificates are retrieved from a directory service or a local cache, it is most expedient to do this prior to the authentication exchange. During an authentication exchange, the entity generating the token may send its certificate or a chain of certificates with the token. Certificate formats are outside the scope of this standard.

How verification of a claimant's certificate is performed depends on whether or not both the claimant and verifier are members of the same public key infrastructure. If so, the verifier needs to obtain the certificate of the claimant's certificate issuer, as well as every preceding certificate in the infrastructure, up to the point of a common ancestor for which the verifier has a public key. All of those certificates between the claimant and that common ancestor should be verified. In the case where the claimant and verifier are not members of the same public key infrastructure, the verifier should obtain and verify all certificates between the claimant and the root of its infrastructure. The manner in which the certificate path is retrieved in either of the above scenarios is outside the scope of this standard. Failure of any verification in the certificate path shall result in a failure to verify the signature on the claimant's certificate.

If certificates are not used, then somehow the signer's public key and any global variables necessary for signature verification must be exchanged prior to initiating the authentication exchange. Each entity which performs authentication verifications may choose to maintain a public key database. The method by which these tasks are accomplished is outside the scope of this standard.

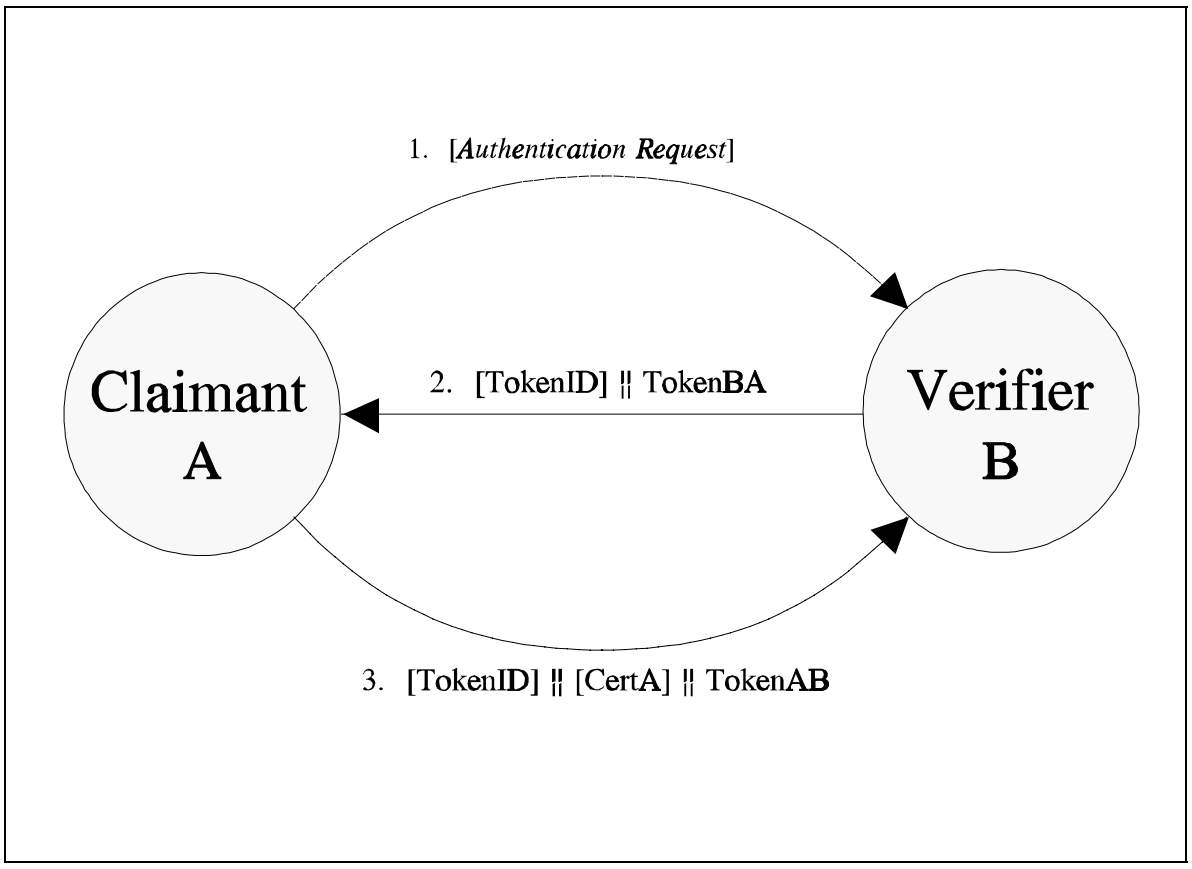

Figure 1 *Unilateral Authentication Exchange*

3.2 Unilateral authentication exchange

The following unilateral authentication exchange mechanism is based on Section 5.1.2, "Two pass authentication", of ISO/IEC 9798-3. Certain authentication token fields and mechanism steps are specified in greater detail in this section than in ISO/IEC 9798-3. The verifier may choose to terminate the authentication exchange at any time. Figure 1 illustrates this authentication exchange. **Note: ISO/IEC 9798-3 "Two pass authentication" begins with the verifier (B) challenging the claimant (A). The description below assumes that the claimant - the entity which is being authenticated - may be the initiator of the exchange, depending on the application of this protocol.**

Throughout the mechanism below, T_X is a time value generated or selected by entity X; an entity has the option to make it equal in value to (or a variation of) a time value transmitted in a previous token in the current exchange. The TextX field may contain undetermined data which may be included by entity X in a token. The existence and content of these fields depend on the specific implementation of this exchange.

1) [OPTIONAL] The claimant, A, selects the verifier, B, to which it will authenticate, and makes an authentication request to B - the format of this request is not defined in this standard.

2) The verifier determines if it will continue the authentication exchange, based on the identity of the claimant. If it chooses to continue, the verifier

   a) generates a (pseudo)random integer, which is the value for the R_B field (as shown in TokenBA below), and retains this value.

   b) [OPTIONAL] selects an identifier for itself which will be the value B in TokenBA.

   c) [OPTIONAL] generates and/or selects other data which is to be included in the A, T_B and Text1 fields.

   The verifier creates an authentication token, TokenBA, of the following form:

   $$R_B \parallel [A] \parallel [B] \parallel [T_B] \parallel [Text1]$$

   B sends a transmission consisting of TokenBA and an optional TokenID (concatenated with TokenBA) to the claimant. The transmission from the verifier to the claimant is of the form:

   $$[TokenID] \parallel TokenBA$$

3) Upon receiving the TokenBA transmission, the claimant, A,

   a) [OPTIONAL] uses TokenID to determine which token is being received.

   b) [OPTIONAL] retrieves information from the A, B, T_B , and Text1 fields, using them in a manner which is outside the scope of this standard.

   c) generates a (pseudo)random integer, which is the value for the R_A field in TokenAB; if this number equals R_B from TokenBA, then it shall be discarded and another value shall be generated. This random value is included so that the claimant does not sign only information generated by the verifier; this is especially important if the claimant's private key is used for more than entity authentication.

   d) selects an identifier for itself which B will be able to use to verify the binding between the claimant and its public key - this will be the value A in TokenAB. If either a claimant identifier was included in the authentication request in step 1), or a certificate is being sent in the transmission with TokenAB, then this field becomes OPTIONAL in TokenAB.

e) [OPTIONAL] generates and/or selects other data which is to be included in the T_A and text fields.

The claimant creates an authentication token, TokenAB, by concatenating data and generating a digital signature. TokenAB consists of the following:

$$R_A \parallel [R_B] \parallel [B] \parallel A \parallel [T_A] \parallel [Text3] \parallel sS_A(R_A \parallel R_B \parallel [B] \parallel [A] \parallel [T_A] \parallel [Text2])$$

In TokenAB, Text2 is a subset of the Text3 field, including the empty set. The signed data that are marked as optional are present only when their corresponding values are present in the unsigned part of TokenAB. Although R_B does not have to be in the unsigned data of TokenAB, it *must* be included in the signed data. The value A in the unsigned data becomes optional according to conditions specified in step 3)d) above.

In addition to containing TokenAB, the transmission may optionally include a token identifier, TokenID, and the claimant's certificate (or chain of certificates), CertA, which are concatenated with the authentication token.

The transmission from the claimant to the verifier is of the form:

$$[TokenID] \parallel [\ CertA\ ] \parallel TokenAB$$

4) Upon receiving the TokenAB transmission, the verifier

a) [OPTIONAL] uses TokenID to determine which token is being received,

b) verifies that the value of R_B is the same as the value retained in step 2)a), given that R_B is present in the unsigned part of TokenAB. If this value is *not* present, then the value retained in step 2)a) is used in the signature verification process in step 4)e).

c) verifies that the identifier for the claimant has been obtained in one of three ways: in CertA, TokenAB, or the authentication request in step 1). In the event that certificates are not used in the system, then the value A should be used to retrieve the claimant's public key from a public key database; continue with step 4)e).

d) verifies the claimant's certificate (assuming certificates are used) as described in Section 3.1.4 of this standard,

e) verifies the claimant's signature in TokenAB,

f) [OPTIONAL] verifies that B is correct,

g) [OPTIONAL] retrieves data from the R_A, T_A, and text fields, using them in a manner which is outside the scope of this standard.

If any of the verifications in parts b) through e) fail, then the authentication exchange is terminated at that point.

Successful completion of step 4) means that the claimant, A, has authenticated itself to the verifier, B.

3.3 Mutual authentication exchange

The following mutual authentication exchange mechanism is based on Section 5.2.2, "Three pass authentication", of ISO/IEC 9798-3. Certain authentication token fields and mechanism steps are specified in greater detail in this section than in ISO/IEC 9798-3. The verifier may choose to terminate the authentication exchange at any time. Figure 2 illustrates this authentication exchange. **Note: Different terminology (i.e., initiator (A) and responder (B)) is used to describe this mechanism because each entity acts as *both* a claimant and a verifier during the exchange.**

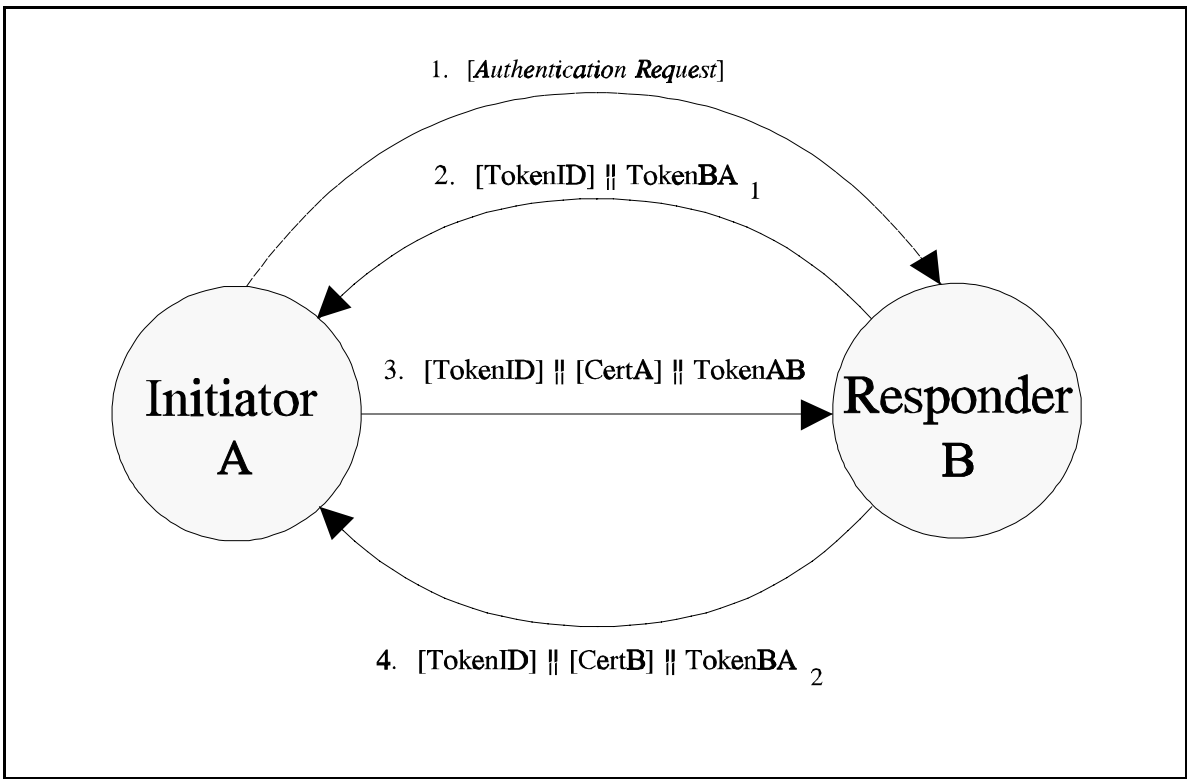

Figure 2 *Mutual Authentication Exchange*

Throughout the mechanism below, T_X is a time value generated or selected by entity X; an entity has the option to make it equal in value to (or a variation of) a time value transmitted in a previous token in the current exchange. The TextX field may contain undetermined data which

may be included by entity X in a token. The existence and content of these fields depend on the specific implementation of this exchange.

1) [OPTIONAL] The initiator, A, selects the responder, B, with which it will mutually authenticate, and makes an authentication request to B - the format of this request is not defined in this standard.

2) The responder determines if it will continue the authentication exchange, based on the identity of the claimant. If it chooses to continue, the responder

   a) generates a (pseudo)random integer, which is the value for the R_B field (as shown in $TokenBA_1$ below), and retains this value.

   b) [OPTIONAL] selects an identifier for itself which A will be able to use to verify the binding between the responder and its public key - this will be the value B in $TokenBA_1$.

   c) [OPTIONAL] generates and/or selects other data which is to be included in the A, T_B and Text1 fields.

   The responder creates an authentication token, $TokenBA_1$, of the following form:

   $$R_B \parallel [A] \parallel [B] \parallel [T_B] \parallel [Text1]$$

   B sends a transmission consisting of $TokenBA_1$ and an optional TokenID (concatenated with $TokenBA_1$) to the initiator. The transmission from the responder to the initiator is of the form:

   $$[TokenID] \parallel TokenBA_1$$

3) Upon receiving the $TokenBA_1$ transmission, the initiator, A,

   a) [OPTIONAL] uses TokenID to determine which token is being received.

   b) [OPTIONAL] retrieves information from the A, B, T_B , and Text1 fields, using them in a manner which is outside the scope of this standard.

   c) generates a (pseudo)random integer which is the value for the R_A field in TokenAB; if this number equals R_B from $TokenBA_1$, then it shall be discarded and another value shall be generated; the value of R_A is retained.

   d) selects an identifier for itself which B will be able to use to verify the binding between the initiator and its public key - this will be the value A in TokenAB. If

either an initiator identifier was included in the authentication request in step 1), or a certificate is being sent in the transmission with TokenAB, then this field becomes OPTIONAL in TokenAB.

e) [OPTIONAL] generates or selects key exchange information which is included in the K_A field in TokenAB. The existence and content of this field depends on the specific implementation of this exchange. **Note: Entity A should not use the corresponding key until the responder has been successfully authenticated (successful completion of step 6)).**

f) [OPTIONAL] generates and/or selects other data which is to be included in the T_A and text fields.

The initiator creates an authentication token, TokenAB, by concatenating data and generating a digital signature. TokenAB consists of the following:

$$R_A \parallel [R_B] \parallel [B] \parallel A \parallel [T_A] \parallel [K_A] \parallel [Text3] \parallel sS_A(R_A \parallel R_B \parallel [B] \parallel [A] \parallel [T_A] \parallel [K_A] \parallel [Text2])$$

In TokenAB, Text2 is a subset of the Text3 field, including the empty set. The signed data that are marked as optional are present only when their corresponding values are present in the unsigned part of TokenAB. Although R_B does not have to be included in the unsigned data of TokenAB, it *must* be included in the signed data. The value A in the unsigned data becomes optional according to conditions in step 3)d) above.

In addition to containing TokenAB, the transmission may optionally include a token identifier, TokenID, and the initiator's certificate (or chain of certificates), CertA, which are concatenated with the authentication token.

The transmission from the initiator to the responder is of the form:

$$[TokenID] \parallel [\ CertA\ ] \parallel TokenAB$$

4) Upon receiving the TokenAB transmission, the responder

a) [OPTIONAL] uses TokenID to determine which token is being received.

b) verifies that the value of R_B is the same as the value retained in step 2)a), given that R_B is present in the unsigned part of TokenAB. If this value is *not* present, then the value retained in step 2)a) is used in the signature verification process in step 4)e).

c) verifies that the identifier for the initiator has been obtained in one of three ways: in CertA, TokenAB, or the authentication request in step 1). In the event that certificates are not used in the system, then the value A should be used to retrieve the initiator's public key from a public key database; continue with step 4)e).

d) verifies the initiator's certificate (assuming certificates are used) as described in Section 3.1.4 of this standard.

e) verifies the initiator's signature in TokenAB.

f) [OPTIONAL] verifies that B is correct.

g) [OPTIONAL] makes use of the key exchange information in the K_A field. How it is used lies outside the scope of this standard.

h) [OPTIONAL] retrieves data from the T_A and text fields, using them in a manner which is outside the scope of this standard.

If any of the verifications in parts b) through e) fail, then the authentication exchange is terminated at that point.

Successful completion of step 4) means that the initiator, A, has authenticated itself to the responder, B.

5) The responder

a) selects an identifier for itself which A will be able to use to verify the binding between the responder and its public key - this will be the value B in $TokenBA_2$. If either a responder identifier, B, was included in $TokenBA_1$, or a certificate is being sent in the transmission with $TokenBA_2$, then this field becomes OPTIONAL in $TokenBA_2$.

b) [OPTIONAL] generates or selects key exchange information which is included in the K_B field in $TokenBA_2$. The existence and content of this field depends on the specific implementation of this exchange.

c) [OPTIONAL] generates and/or selects other data which is to be included in the T_B and text fields of $TokenBA_2$.

The responder creates an authentication token, $TokenBA_2$, by concatenating data and generating a digital signature. $TokenBA_2$ consists of the following:

$$[R_B] \parallel [R_A] \parallel [A] \parallel B \parallel [T_B] \parallel [K_B] \parallel [Text5] \parallel sS_B(R_B \parallel R_A \parallel [A] \parallel [B] \parallel [T_B] \parallel [K_B] \parallel [Text4])$$

In $TokenBA_2$, Text4 is a subset of the Text5 field, including the empty set. The signed data that are marked as optional are present only when their corresponding values are present in the unsigned part of $TokenBA_2$. Although R_A and R_B do not have to be included in the unsigned data of $TokenBA_2$, they *must* be included in the signed data. (Note that the order of R_A and R_B is reversed from TokenAB - this is to conform with ISO/IEC 9798-3, where the token sender's random challenge is the first token field.) The value B in the unsigned data becomes optional according to conditions specified in step 5)a) above.

In addition to containing $TokenBA_2$, the transmission may optionally include a token identifier, TokenID, and the responder's certificate (or chain of certificates), CertB, which are concatenated with the authentication token.

The transmission from the responder to the initiator is of the form:

$$[\text{TokenID}] \parallel [\ \text{CertB}\ ] \parallel \text{TokenBA}_2$$

6) Upon receiving the $TokenBA_2$ transmission, the initiator, A,

   a) [OPTIONAL] uses TokenID to determine which token is being received.

   b) verifies that the value of R_A is the same as the value retained in step 3)c), given that R_A is present in $TokenBA_2$. If this value is *not* present, then the value retained in step 3)c) is used in the signature verification process in step 6)f).

   c) verifies that the value of R_B is the same value as the R_B field retrieved from $TokenBA_1$ in step 3), given that R_B is present in $TokenBA_2$. If this value is not present, then the value of R_B from step 3) is used in the signature verification process in step 6)f).

   d) verifies that the identifier for the responder has been obtained in one of three ways: in CertB, $TokenBA_2$, or $TokenBA_1$. In the event that certificates are not used in the system, then the value B should be used to retrieve the initiator's public key from a public key database; continue with step 6)f).

   e) verifies the responder's certificate (assuming certificates are used) as described in Section 3.1.4 of this standard.

   f) verifies the responder's signature on $TokenBA_2$.

   g) [OPTIONAL] verifies that A is correct.

   h) [OPTIONAL] makes use of the key exchange information in the K_B field. How it is used lies outside the scope of this standard.

i) [OPTIONAL] retrieves data from the T_B and text fields, using them in a manner which is outside the scope of this standard.

If any of the verifications in parts b) through f) fail, then the authentication exchange is terminated at that point.

Successful completion of step 6) means that the responder has authenticated itself to the initiator, and thus the entities have successfully mutually authenticated.

Appendix A
(Draft, May 3, 1995)

Example ASN.1 Definition and Encoding of Authentication Exchange Tokens

This appendix is provided for informational purposes only, and is not part of this standard. The purpose of this appendix is to provide implementors of this standard with an optional set of rules for defining and encoding the authentication tokens specified in this standard, as well as other data such as public key certificates. Implementation of a unique set of encoding rules is necessary for interoperability to take place between various authentication domains.

Abstract Syntax Notation One (ASN.1)

This example definition is specified using the Abstract Syntax Notation One (ASN.1). ASN.1 is an international standard used by all ISO and ITU (previously CCITT) protocols as well as many Internet protocols. It is comprised of two parts: the ASN.1 notation [1-5] and the ASN.1 encoding rules [6,7].

The *ASN.1 notation* is an abstract specification language used to describe the local representation of data. It allows the description of the complex data types carried in protocol messages, without concern for the underlying binary representation.

The *ASN.1 encoding rules* go hand-in-hand with the ASN.1 notation. Whereas the ASN.1 notation provides an abstract specification of the local representation of data, the ASN.1 encoding rules define the external representation of the data. In order to correctly interpret the binary-pattern representation of a data value, it is necessary to know (usually from the context), the type of the value being represented, as well as the encoding mechanism used being employed to convert the value from its local representation to its external representation. The ASN.1 encoding rules define "tags" that are added to a data value to indicate the data type and length representation. The encoding rules also define the actual bit representation of the data value.

There may be more than one set of encoding rules that can be applied to data values that are defined using the ASN.1 notation. The ASN.1 encoding rules standard defines three sets of encoding rules, called basic encoding rules, canonical encoding rules, and distinguished encoding rules. Whereas the basic encoding rules gives the sender of an encoding various choices as to how data values may be encoded, the canonical and distinguished encoding rules select just one encoding from those allowed by the basic encoding rules, eliminating all of the sender's options. The use of canonical or distinguished encodings is essential with protocol messages containing digitally signed data values, so that the receiver can verify the digital signature over the same exact data representation used by the sender to generate the signature.

The canonical and distinguished encoding rules differ from each other in the set of restrictions they place on the basic encoding rules. The distinguished encoding rules are more suitable if the

encoded value is small enough to fit into the available memory and there is a need to rapidly skip over some nested values. The canonical encoding rules are more suitable if there is a need to encode values that are so large that they cannot readily fit into the available memory or it is necessary to encode and transmit part of a value before the entire value is available.

ASN.1 Specification of FIPS PUB JJJ Protocol Messages

Each protocol message is potentially comprised of three components, an optional token identifier, optional public-key certification data, and an authentication token.

MessageBA is used in the unilateral authentication exchange and is sent by the verifier to the claimant. It contains an optional token identifier and authentication token BA.

```
MessageBA ::= SEQUENCE {
     tokenId         [0]          TokenId OPTIONAL,  -- type 0x0001
     tokenBA                      TokenBA
}

TokenBA ::= SEQUENCE {
     ranB                         RandomNumber,
     entityA         [0]          EntityName OPTIONAL,
     entityB         [1]          EntityName OPTIONAL,
     timestampB                   TimeStamp OPTIONAL,
     text1                        Text OPTIONAL
}
```

MessageABU is used in the unilateral authentication exchange and is sent by the claimant to the verifier. It contains an optional token identifier, optional certification information, and authentication token ABU.

```
MessageABU ::= SEQUENCE {
     tokenId         [0]          TokenId OPTIONAL,  -- type 0x0002
     certA           [1]          CertData OPTIONAL,
     tokenABU                     TokenABU
}

TokenABU ::= SEQUENCE {
     ranA                         RandomNumber,
     ranB                         RandomNumber OPTIONAL,
     entityB         [0]          EntityName OPTIONAL,
     entityA                      EntityName,
     timestampA                   TimeStamp OPTIONAL,
     text3                        Text OPTIONAL,
     signature                    SigDataABU
}

-- if optional components are inclued in TokenABU,
-- then they shall also be included in SigDataABU

SigDataABU ::= SIGNATURE SEQUENCE {
     ranA                         RandomNumber,
     ranB                         RandomNumber,
     entityB         [0]          EntityName OPTIONAL,
     entityA         [1]          EntityName OPTIONAL,
     timestampA                   TimeStamp OPTIONAL,
```

```
    text2                       Text OPTIONAL  -- subset of text3
}
```

MessageBA1 is used in the mutual authentication exchange and is sent by the responder to the initiator. It contains the same authentication token used in the unilateral authentication exchange.

```
MessageBA1 ::= SEQUENCE {
    tokenId         [0]         TokenId OPTIONAL,  -- type 0x0011
    tokenBA1                    TokenBA
}
```

MessageABM is used in the mutual authentication exchange and is sent by the initiator to the responder. It contains an optional token identifier, optional certification information, and authentication token ABM.

```
MessageABM ::= SEQUENCE {
    tokenId         [0]         TokenId OPTIONAL,  -- type 0x0012
    certA           [1]         CertData OPTIONAL,
    tokenABM                    TokenABM
}

TokenABM ::= SEQUENCE {
    ranA                        RandomNumber,
    ranB                        RandomNumber OPTIONAL,
    entityB         [0]         EntityName OPTIONAL,
    entityA                     EntityName,
    timestampA                  TimeStamp OPTIONAL,
    keyInfoA        [1]         KeyInfo OPTIONAL,
    text3           [2]         Text OPTIONAL,
    signature                   SigDataABM
}

-- if optional components are inclued in TokenABM,
-- then they shall also be included in SigDataABM

SigDataABM ::= SIGNATURE SEQUENCE {
    ranA                        RandomNumber,
    ranB                        RandomNumber,
    entityB         [0]         EntityName OPTIONAL,
    entityA         [1]         EntityName OPTIONAL,
    timestampA                  TimeStamp OPTIONAL,
    keyInfoA        [2]         KeyInfo OPTIONAL,
    text2           [3]         Text OPTIONAL  -- subset of text3
}
```

MessageBA2 is used in the mutual authentication exchange and is sent by the responder to the initiator. It contains an optional token identifier, optional certification information, and authentication token BA2.

```
MessageBA2 ::= SEQUENCE {
    tokenId         [0]         TokenId OPTIONAL,  -- type 0x0013
    certB           [1]         CertData OPTIONAL,
    tokenBA2                    TokenBA
}
```

```
TokenBA2 ::= SEQUENCE {
     ranB            [0]         RandomNumber OPTIONAL,
     ranA            [1]         RandomNumber OPTIONAL,
     entityA         [2]         EntityName OPTIONAL,
     entityB                     EntityName,
     timestampB                  TimeStamp OPTIONAL,
     keyInfoB        [3]         KeyInfo OPTIONAL,
     text5           [4]         Text OPTIONAL,
     signature                   SigDataBA2
}

-- if optional components are inclued in TokenBA2,
-- then they shall also be included in SigDataBA2

SigDataBA2 ::= SIGNATURE SEQUENCE {
     ranB                        RandomNumber,
     ranA                        RandomNumber,
     entityA         [0]         EntityName OPTIONAL,
     entityB         [1]         EntityName OPTIONAL,
     timestampB                  TimeStamp OPTIONAL,
     keyInfoB        [2]         KeyInfo OPTIONAL,
     text4           [3]         Text OPTIONAL  -- subset of text5
}
```

The token identifier is defined here to include the type of token and the protocol version number.

```
TokenId ::= SEQUENCE {
     tokenType                   INTEGER,
     protoVerNo                  INTEGER {v1(1)},
}
```

The certification data includes a certification path and/or a certificate revocation list. The definitions for CertificationPath and CertificateList are imported from X.509, and are given below.

```
CertData ::= SEQUENCE {
     certPath        [0]         CertificationPath OPTIONAL,
     certRevList     [1]         CertificateList OPTIONAL
}  -- at least one of the components above shall be present
```

A random number is simply defined as an integer.

```
RandomNumber ::= INTEGER
```

An entity name is defined as a choice between either a distinguished name (as defined in X.501) or an Internet email name, in the standard `user@host` format.

```
EntityName ::= CHOICE {
     Name,
     EmailName
}

EmailName ::= PrintableString  -- in the form user@host
```

A timestamp is defined to be a coordinated universal time, consisting of values representing a calendar date, a time to the precision of either one minute or one second, and optionally a local time differential from coordinated universal time. The canonical and distinguished encoding rules do not specify a single format from among the different alternative formats defined for UTCTime. Therefore, it is recommended that UTCTime values always specify a time to the precision of one second, even though that precision is optional. Similarly, it is recommended that the time be expressed directly as coordinated universal time (i.e., in the format which terminates with the character 'Z'), rather than in the optional format which provides a local time differential from coordinated universal time.

```
TimeStamp ::= UTCTime  -- represented as YYYYMMDDhhmmssZ
```

Key exchange information is undetermined and may be included in a token at the discretion of a specific implementation of this standard. Therefore, it is defined here to simply be a bit string. The key exchange information used in a specific implementation shall be defined and encoded using ASN.1 or some other data representation. In either case, the resulting data shall be unambiguously encoded (e.g., using ASN.1 distinguished or canonical encoding rules) and shall be represented here as a bit string.

```
KeyInfo ::= BIT STRING  -- contains encoded key exchange info
```

Text contains undetermined data which may be included at the discretion of a specific implementation of this standard. Therefore, it is defined here to simply be a bit string. The text data used in a specific implementation shall be defined and encoded using ASN.1 or some other data representation. In either case, the resulting data shall be unambiguously encoded (e.g., using ASN.1 distinguished or canonical encoding rules) and shall be represented here as a bit string.

```
Text ::= BIT STRING  -- contains encoded text data
```

ASN.1 Specification of Public-Key Certification Information

The following definitions are taken from X.501, The Directory: The Models [8,9].

```
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY
AttributeValueAssertion ::= SEQUENCE {AttributeType, AttributeValue}

RelativeDistinguishedName ::= SET OF AttributeValueAssertion
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
DistinguishedName ::= RDNSequence

Name ::= CHOICE {       --only one for now
     DistinguishedName
}
```

The following certification information definitions are taken from X.509, The Directory - Authentication Framework (with proposed 1992 extensions presumed) [10,11].

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

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

Certificate ::= SIGNED SEQUENCE {
     version        [0]         Version DEFAULT v1
     serialNumber               CertificateSerialNumber,
     signature                  AlgorithmIdentifier,
     issuer                     Name,
     validity                   Validity,
     subject                    Name,
     subjectPublicKeyInfo       SubjectPublicKeyInfo,
     issuerUID      [1]         IMPLICIT UID OPTIONAL, -- used in v2 only
     subjectUID     [2]         IMPLICIT UID OPTIONAL, -- used in v2 only
}

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

CertificateSerialNumber ::= INTEGER

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

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

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

UID ::= BIT STRING

ENCRYPTED MACRO ::=
BEGIN
TYPE NOTATION   ::= type(ToBeEnciphered)
VALUE NOTATION  ::= value(VALUE BIT STRING)
```

```
END     -- of ENCRYPTED

SIGNED MACRO    ::=
BEGIN
TYPE NOTATION   ::= type (ToBeSigned)
VALUE NOTATION  ::= value (VALUE
SEQUENCE{
     ToBeSigned,
     AlgorithmIdentifier, -- algorithm used to generate signature
     ENCRYPTED OCTET STRING     -- where 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
     ENCRYPTED OCTET STRING     -- the signature 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
```

References:

[1] ISO 8824, Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), December 15, 1987.

[2] ISO/IEC JTC 1/SC 21 N9148, Final Draft, ITU-T Rec. X.680 | ISO/IEC 8824-1, Information Technology - Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation, November 15, 1994.

[3] ISO/IEC JTC 1/SC 21 N9149, Final Draft, ITU-T Rec. X.681 | ISO/IEC 8824-2, Information Technology - Abstract Syntax Notation One (ASN.1) - Information Object Specification, November 15, 1994.

[4] ISO/IEC JTC 1/SC 21 N9151, Final Draft, ITU-T Rec. X.682 | ISO/IEC 8824-3, Information Technology - Abstract Syntax Notation One (ASN.1) - Constraint Specification, November 15, 1994.

[5] ISO/IEC JTC 1/SC 21 N9152, Final Draft, ITU-T Rec. X.683 | ISO/IEC 8824-4, Information Technology - Abstract Syntax Notation One (ASN.1) - Parameterization of ASN.1 Specifications, November 15, 1994.

[6] ISO 8825, Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), November 15, 1987.

[7] ISO/IEC JTC 1/SC 21 N9153, Final Draft, ITU-T Rec. X.690 | ISO/IEC 8825-1, Information Technology - Abstract Syntax Notation One (ASN.1) - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), November 15, 1994.

[8] CCITT Recommendation X.501, The Directory: The Models, November 1988.

[9] ITU-T Rec. X.501 | ISO/IEC 9594-2, Information Technology - Open Systems Interconnection - The Directory: The Models, Editor's DRAFT, February 14, 1993.

[10] CCITT Recommendation X.509, The Directory: Authentication Framework, November 1988.

[11] ITU-T Rec. X.509 | ISO/IEC 9594-8, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, Editor's DRAFT, February 14, 1993.

Appendix B

Selected References

[ANSIX9.17] ANSI X9.17-1985, Financial Institution Key Management (Wholesale), American Banker's Association, Approved April 4, 1985, Reaffirmed 1991.

[FIPS140-1] FIPS PUB 140-1, Security Requirements for Cryptographic Modules, US DOC/NIST, January 11, 1994.

[FIPS180] FIPS PUB 180, Secure Hash Standard (SHS), US DOC/NIST, May 11, 1993.

[FIPS186] FIPS PUB 186, Digital Signature Standard (DSS), US DOC/NIST, May 19, 1994.

[FIPS190] FIPS PUB 190, Guideline for the Use of Advanced Authentication Technology Alternatives.

[ISO9798-1] ISO/IEC 9798-1:1991, "Information technology - Security techniques - Entity authentication mechanisms - Part 1: General model", ISO/IEC, 1991.

[ISO9798-3] ISO/IEC 9798-3:1993, "Information technology - Security techniques - Entity authentication mechanisms - Part 3: Entity authentication using a public key algorithm", ISO/IEC, 1993.