FIPS PUB 196

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

1997 February 18

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

ENTITY AUTHENTICATION USING PUBLIC KEY CRYPTOGRAPHY

CATEGORY: COMPUTER SECURITY
SUBCATEGORY: ACCESS CONTROL

U.S. DEPARTMENT OF COMMERCE, William M. Daley, *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 5131 of the Information Technology Management Reform Act of 1996, and the Computer Security Act of 1987, Public Law 104-106. 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 Information Technology 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, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

Shukri Wakid, *Director*
Information Technology Laboratory

Abstract

This standard specifies two challenge-response protocols by which entities in a computer system may authenticate their identities to one another. These may be 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, which uses digital signatures and random number challenges.

Authentication based on public key cryptography 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: access control, authentication, challenge-response, computer security, cryptographic modules, cryptography, Federal Information Processing Standard (FIPS), telecommunications security.

Federal Information Processing Standards Publication 196

1997 February 18

ANNOUNCING

ENTITY AUTHENTICATION USING PUBLIC KEY CRYPTOGRAPHY

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 5131 of the Information Technology Management Reform Act of 1996, and the Computer Security Act of 1987, Public Law 104-106.

1. Name of Standard. Entity Authentication Using Public Key Cryptography (FIPS PUB 196).

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

3. Explanation. This standard specifies two challenge-response protocols by which entities in a computer system may authenticate their identities to one another. These protocols may be 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, which uses digital signatures and random number challenges.

Authentication based on public key cryptography 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, Information Technology 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 180-1, Secure Hash Standard.
d. FIPS PUB 186, Digital Signature Standard.
e. FIPS PUB 190, Guideline for the Use of Advanced Authentication Technology Alternatives.
f. ANSI X9.17-1985, Financial Institution Key Management (Wholesale).
g. ISO/IEC 9798-1:1991, Information technology - Security techniques - Entity authentication mechanisms - Part 1: General model.
h. 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 entity authentication based on public key cryptography, when the implementation of such technology is considered cost-effective. 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 protocols 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) 196, *Entity Authentication Using Public Key Cryptography* (affixed).

10. Implementations. The authentication protocols 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 April 6, 1997.

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 (e.g., hackers, organized crime, economic competitors) mounting cost-effective attacks on unclassified government or commercial data. The primary goal in designing an effective security system is to make the cost of any attack greater than the possible payoff.

While specifications in this standard are intended to maintain the security of an authentication protocol, conformance to this standard does not guarantee that a particular implementation is secure. It is the responsibility of the manufacturer to build the implementation of an authentication protocol in a secure manner. This standard will be reviewed every five years in order to assess its adequacy.

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 Government-wide 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, Building 225, Room A231, 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 196 (FIPS PUB 196), 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 196

1997 February 18

Specifications for

ENTITY AUTHENTICATION USING PUBLIC KEY CRYPTOGRAPHY

CONTENTS

1. INTRODUCTION

This publication specifies two protocols 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 PUB 186, *Digital Signature Standard*).

International Standard ISO/IEC 9798-3:1993, *Entity authentication using a public key algorithm* (ISO/IEC 9798-3), serves as the basis for the authentication protocols defined in Section 3 below. That international standard defines five different protocols, addressing both 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 protocol for each unilateral and mutual authentication was selected from ISO/IEC 9798-3. Certain authentication token fields and protocol steps are described in greater detail in this standard than in ISO/IEC 9798-3.

Although ISO/IEC 9798-3 serves as the basis for this standard, this specification is less strict, in that it allows for the arbitrary ordering of token fields. This allowance therefore enables other non-ISO public key authentication protocols to meet the requirements of this standard. Those requirements are specified in the following sections.

Section 2 of this standard presents a general overview of the entity authentication protocols defined herein. It also (1) highlights the criteria that must be met to conform to this standard, (2) briefly describes threats addressed by using the authentication protocols, and (3) lists definitions and notation used in this standard. The authentication protocols, along with issues pertaining to authentication token information, are described in detail in Section 3. Several appendices accompanying this standard describe optional methods for formatting and encoding authentication information. These methods are included for informational purposes only, to help promote the interoperability of various implementations of the authentication protocols defined in Section 3.

2. GENERAL

2.1 Scope

2.1.1 Implementation Criteria

To acceptably implement this standard, an implementation must meet the following criteria:

1) Each entity in an authentication exchange must use a FIPS approved digital signature algorithm to generate and/or verify digital signatures;

2) Each entity must generate (pseudo)random numbers using a FIPS approved (pseudo)random number generator;

3) Each entity acting as a claimant must be bound to a public/private key pair; the private key should remain in the sole control of the claimant who uses that key to sign a random challenge. The key binding requires a unique authentication identifier for each claimant, so that a verifier can distinguish between multiple claimants; and

4) One or both of the authentication protocols in this standard must be implemented. For each protocol, steps and token fields marked as [OPTIONAL] do not need to be implemented, except where indicated otherwise. However, all other steps and token fields must be implemented.

2.1.2 Overview

Entity authentication protocols in this standard make use of a digital signature algorithm for the generation of authentication tokens. The authentication protocols 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).

The authentication of an entity (viz., a claimant to a verifier) depends on two successful actions: (1) the verification of the claimant's binding with its public/private key pair, and (2) the verification of the claimant's digital signature on a random number challenge. A binding of an entity's unique identifier with its key pair is essential to proving the authenticity of that entity's identity. This must be done prior to any authentication exchange. During an authentication exchange, a random number challenge generated by the verifier is associated with the claimant's identifier. The claimant then generates a signature on that challenge, which is freshly generated for that particular exchange. In order to verify a signature, the verifier uses the claimant's identifier to find a public key that is bound to that identifier. If that public key can be used to successfully verify the claimant's signature on the challenge, then the verifier has in fact verified that the claimant is the same entity that is bound to the key pair. This chain of associations, bindings, and signatures is what allows an entity to successfully authenticate itself. The next several paragraphs address some issues of entity-key bindings, random challenges, and identifiers.

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 an authentication implementation, 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. 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 a particular implementation using the optional text fields provided in the exchanged authentication tokens (see Section 3.1.5). Trusted third parties may provide other necessary services, such as a courier service to transport valid public keys.

Since the protocols defined in this standard utilize (pseudo)random number values for the authentication tokens' time variant parameters, authenticating entities do not have to use synchronized clocks to verify the freshness and timeliness of authentication tokens. (Note: Throughout the rest of this document, the term "random" number will imply the use of either a random or pseudorandom number.) Authentication exchanges using date-time stamps 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 do not necessarily know one another prior to authentication.

A naming convention for the entities involved in an authentication exchange is not defined in this standard. However, unique, distinguishing identifiers for participating authentication entities should be established prior to the execution of either of those protocols. Each entity being authenticated must have such 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 in Section 3. Information pertaining to biometrics, or the results of performing a biometric live scan, may be included in an authentication token's optional text 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, will not prevent that implementation from meeting the requirements of this standard.

Upon completion of either of the entity authentication protocols in this standard, the entity performing the final verification step may send an acknowledgment of verification success or failure to the other entity involved in the exchange. Although the format and use of such an acknowledgment 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 exchange.

2.2 Threats

The protocols defined in this document address several threats, including masquerade, password compromise, replay attacks, and the signing of pre-defined data. Using challenges and digital signatures eliminates the need for transmitting passwords for authentication, which reduces the threat of their compromise. Such a compromise would allow an attacker to use the same information to authenticate repeatedly. By using a private key to generate digital signatures for authentication, it becomes computationally infeasible for an attacker to masquerade as another entity. However, implementations may still rely on passwords for an entity to access its private key. In that case, passwords must be kept secure. Although the use of public key cryptography eliminates the need for entities to share a secret value, it is extremely important that each claimant always keeps its private key secure, and under its sole control.

The use of random number challenges also prevents an intruder from copying an authentication token signed by another user and replaying it successfully at a later time. However, a new random number challenge should be generated for each authentication exchange. The security of replay prevention also hinges on the generation of random number challenges which have a low probability of being repeated.

Finally, by including a random number 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 both the challenge and unpredictable, meaningless data - a random number. This is why the responder in the mutual authentication protocol generates a signature over both random numbers, instead of just signing the initiator's random number challenge.

The authentication protocols in Section 3 do not address other threats, which include denial of service, 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 the secrecy 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. Other security measures, which lie outside the scope of this standard, may be needed in order to address such additional threats and vulnerabilities.

2.3 Definitions and notation

2.3.1 Definitions

Some of the following definitions are taken from terminology defined in ISO/IEC 9798-1:1991, *General Model* (ISO/IEC 9798-1) which describes the general model for the ISO/IEC 9798 series of entity authentication standards. Other definitions are from FIPS PUB 140-1, Section 2.1.

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

Binding: an acknowledgment by a trusted third party that associates an entity's identity with its public key. This may take place through (1) a certification authority's generation of a public key certificate, (2) a security officer's verification of an entity's credentials and placement of the entity's public key and identifier in a secure database, or (3) an analogous method.

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 smartcard (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 PUB 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.

Distinguishing identifier: information which unambiguously distinguishes an entity in the authentication process.

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

FIPS approved security method: a security method (e.g., cryptographic algorithm, cryptographic key generation algorithm or key distribution technique, random number generator, 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 (certificate): a set of data that unambiguously identifies an entity, contains the entity's public key, and is digitally signed by a trusted third party (certification authority).

Public key infrastructure (PKI): an architecture which is used to bind public keys to entities, enable other entities to verify public key bindings, revoke such bindings, and provide other services critical to managing public keys.

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

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

Unsigned data: data included in an authentication token, in addition to a digital signature. An exception to this is $TokenBA_1$ in each authentication exchange, which contains only unsigned data and no signature. Unsigned data gives information to the token recipient which may be used to verify the token's signature and/or generate a response token. This unsigned data *may* be equivalent to the signed data, but not necessarily.

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

2.3.2 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.

$X \| Y$ - the result of the concatenation of data items X and Y, not necessarily 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.

TokenID - token identifier - information accompanying an authentication token that may be used to identify the token and/or protocol type to assist token processing by the recipient.

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".

[Z] - indicates that item Z 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 to this standard.

3. ENTITY AUTHENTICATION PROTOCOLS

3.1 Authentication protocol issues

Certain factors should be considered before initiating either of the authentication protocols described in this standard.

3.1.1 Digital signatures

Each entity involved in an authentication protocol in Section 3 must have the ability to generate and/or verify a digital signature. Entities acting as claimants must have a digital signature generation capability, and verifiers must have a digital signature verification capability. A FIPS approved public key digital signature algorithm (e.g., FIPS PUB 186, *Digital Signature Standard (DSS)*) must be implemented to provide these digital signature functions.

A public/private key pair compliant with the digital signature algorithm must be possessed by each claimant. *It is critical to the security of the authentication exchanges that a private key remain accessible to only one claimant.* The entity authentication implementation shall employ an identity-based operator authentication mechanism "in order to verify the authorization of the operator to assume the desired roles and to request corresponding services" (FIPS PUB 140-1, *Security Requirements for Cryptographic Modules*). This establishes a one-to-one relationship between an entity and the private (signing) key, and it also helps prevent unauthorized entities from authenticating themselves falsely using the key material.

3.1.2 Random numbers

To create random numbers for the two authentication exchanges described in this standard, only FIPS approved random number generators shall be used. At a minimum, these include generators found in:

-Appendix 3 of FIPS PUB 186, and

-Appendix C of ANSI X9.17-1985, Financial Institution Key Management (Wholesale).

In the authentication exchanges described in Sections 3.2 and 3.3, the verifier uses a random number as a challenge to the claimant, and the claimant uses a random number to preclude signing only data determined by the verifier (see Section 2.2 for a discussion of potential threats).

An implementor may choose to use other types of time variant parameters in the authentication exchanges in this standard, *however* they shall be used in conjunction with the random number challenge, which remains the basis for verifying an entity's authenticity. One possible approach is to combine (e.g., exclusive-OR, concatenate, etc.) a non-repeating sequence number with the output of the random number generator, and using the result as the random number challenge.

It is important to note that each entity that acts as a verifier must maintain state, because knowledge of the original random number challenge is essential when the verifier attempts to verify the claimant's response. To maintain state for the duration of an authentication exchange, a verifier must keep a record of both (1) a freshly generated random number challenge and (2) an association between that challenge and the claimant. Linking the claimant to the appropriate random number challenge is *especially* important when the verifier is involved in several simultaneous authentication sessions. How this is done depends on the particular implementation. In some implementations, it may be necessary to link the random number with the claimant using the optional TokenID field which may be sent with a token.

3.1.3 Identifiers

Unique, distinguishing identifiers shall be determined for all entities which have the potential for communicating with one another, before the authentication protocol is initiated. A naming convention shall be established such that a verifier can differentiate between all entities which act as claimants, and each claimant has a unique 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 the entity authentication protocols below, a token identifier (TokenID) may be included with each token transmission. Such an identifier might 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 Public key certificates

If public key 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 entity'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, the certificate may be retrieved from a directory service, a local cache, or an authentication message. 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, but particular formats may be recommended or mandated by other FIPS, NIST Special Publications, or other Federal regulations.

The manner in which a certificate or certificate chain is verified depends on the configuration of the public key infrastructure in use. Generally speaking, however, to verify the binding between the claimant and its public key, a verifier must have a chain of valid, verifiable certificates - from the claimant's certificate to a certificate issued by a trusted third party whose public key is known to the verifier. The manner in which the certificate chain is retrieved is outside the scope of this standard. Failure of any verification in the certificate chain shall result in a failure to verify the signature on the claimant's certificate.

If certificates are not used, then the claimant must be bound to its public/private key pair in some other manner. The claimant's public key and any global variables necessary for signature verification may be exchanged prior to initiating the authentication exchange. A trusted third party may be used by a verifier to obtain a claimant's public key. 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.

3.1.5 Optional fields and steps

Each of the messages exchanged in the authentication protocols include fields that are marked "[OPTIONAL]", as are some steps in the protocols. Neither those fields nor steps have to be implemented in order to conform to this standard, unless indicated otherwise. The authentication of each claimant does not depend on the use of any optional fields or steps. It is left up to the implementor to determine which optional fields and steps will be implemented, and how they will be used.

Each of the tokens exchanged in both the unilateral and mutual protocols contain generic "Text" fields. These fields are optional and have a format which is implementation-specific. The optional text field in each authentication token represents an aggregation of optional fields, which can appear in any order throughout the token.

Each authentication token includes an optional text field(s) containing data that does not have to be signed. Note that data integrity is *not* guaranteed for information which is included in the unsigned portion of a token but which is *not* digitally signed (i.e., unsigned data that is not included in the signed data). It is recommended, but not required, that a signature be generated over *all* information included in a token. The type of data that may be included is determined by the implementor, and is not limited by the specifications in this standard. The number of different types of data in each optional field is not limited, either. Use of the text fields should be carefully implemented, because their use *may* - depending on the implementation - create vulnerabilities in the authentication exchange. Some possible uses of the text fields are listed below:

<u>Identifiers</u> - An entity may choose to include an identifier for itself in the text field of a token. If certificates are not used to distribute a claimant's public key, then the implementation may require that the claimant include information identifying itself in the authentication token.

<u>Time value</u> - Another time variant parameter may be included in a token's text field, in addition to the random challenges that are used to determine the authenticity of entities. *However, this additional value shall **not** replace the random number value as the verifier's challenge to the claimant.* For example, a time value may be included in a token for access control auditing, if tokens are logged by a verifier upon successful completion of an authentication exchange.

Key exchange data - Text fields may include information used to distribute a cryptographic key (or keys). For example, an encrypted session key, or information used in establishing a session key may be included in the text field. How the key distribution is performed is not specified in or required by this standard. When either of the protocols in this standard is used to establish a key, that key shall not be used until each claimant in the exchange has been successfully authenticated.

Biometric data - An implementation, in addition to requiring a verifiable response to a random challenge, may optionally include biometric authentication data to determine an entity's authenticity. This is a physical characteristic of a claimant (or the principal on whose behalf the claimant is acting), which is an additional factor of authentication. See FIPS PUB 190, *Guideline for the Use of Advanced Authentication Technology Alternatives*, for more information.

3.1.6 Authentication token encoding methods

Descriptions of the authentication protocols in Sections 3.2 and 3.3 specify the contents of authentication tokens and steps for token generation and verification. However, the formatting and encoding of these tokens is not specified in this standard, and they are left to the discretion of the implementor. The interoperability of different implementations of the authentication protocols will rely, to some extent, on how tokens and messages are formatted and encoded. This is especially important since token fields are ordered arbitrarily, and the format and content of optional text fields are not specified. To provide some implementation guidance, several informative appendices (A, B, C, and D) are provided in this standard which specify optional examples of formatting and encoding methods. The implementation of any one of those methods is ***not*** required for meeting the specifications of this standard.

3.2 Unilateral authentication protocol

The following unilateral entity authentication protocol is based on Section 5.1.2, "Two pass authentication", of ISO/IEC 9798-3. Certain authentication token fields and protocol 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 exchange.

The unilateral authentication protocol begins with the verifier (B) challenging the claimant (A). The description below allows for situations where the claimant may be the initiator of the exchange, depending on the application of this protocol.

It is important to note that the success of an entity's authentication, according to this standard, is not dependent on the information contained in the text fields. As described in Section 2.1, the authentication of an entity depends on two things: (1) the verification of the claimant's binding with its key pair, and (2) the verification of the claimant's digital signature on the random number challenge. How text field information is used is beyond the scope of this standard.

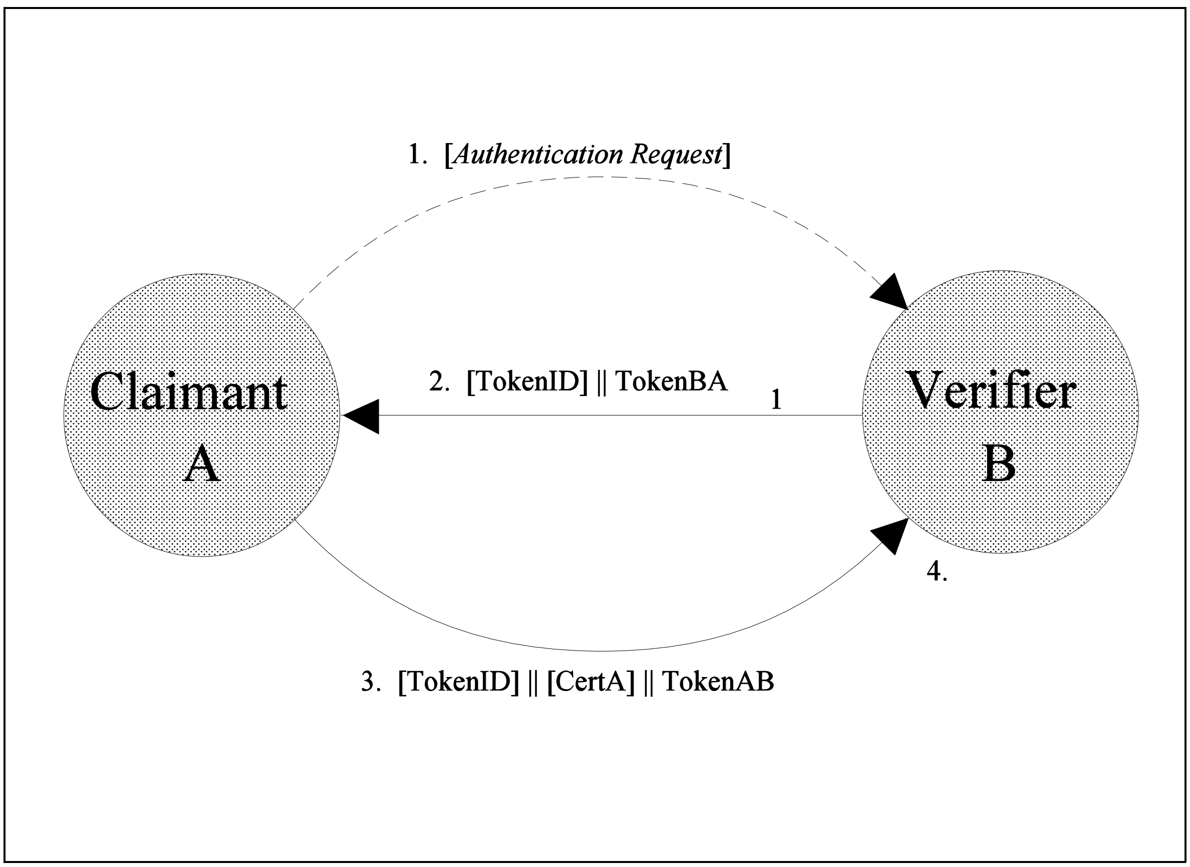

Figure 1 *Unilateral Authentication Protocol*

Unilateral entity authentication occurs as follows:

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, B, determines if it will continue, initiate, or terminate the authentication exchange. If it attempts to authenticate the claimant, the verifier then

   a) generates a random number challenge, which is the value for the R_B field in $TokenBA_1$ below, and retains this value.

   b) [OPTIONAL] generates and/or selects other data which is to be included in the Text1 field of $TokenBA_1$.

   The verifier creates a challenge token of the following form:

   $TokenBA_1$ = $R_B \parallel$ [Text1]

   Entity B sends a message consisting of $TokenBA_1$ and an optional TokenID to the claimant. The message from the verifier to the claimant is of the form:

   [TokenID] $\parallel$ $TokenBA_1$

3) Upon receiving the message including $TokenBA_1$, the claimant, A,

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

   b) [OPTIONAL] retrieves information from the Text1 field, using it in a manner which is outside the scope of this standard.

   c) generates a random number challenge, which is the value for the R_A field in TokenAB below.

   d) [OPTIONAL] selects an identifier for the verifier, and includes that in the B field of TokenAB.

   e) [OPTIONAL] generates and/or selects other data which is to be included in the Text2 and Text3 fields. In TokenAB, Text2 is a subset of the Text3 field, including cases where Text2 is the NULL set, and where Text2 equals Text3.

The claimant creates an authentication token, TokenAB, by concatenating data and generating a digital signature:

TokenAB = $R_A \| [R_B] \| [B] \| [Text3] \| sS_A(R_A \| R_B \| [B] \| [Text2])$

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.

In addition to containing TokenAB, the message may optionally include a token identifier, TokenID, and the claimant's certificate (or chain of certificates), CertA. The message from the claimant to the verifier is of the form:

$$[TokenID] \| [\ CertA\ ] \| TokenAB$$

4) Upon receiving the message including TokenAB, the verifier, B,

   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), depending on which optional information has been included. In the event that certificates are not used for signature verification, then the identifier for entity A should be used to retrieve the claimant's public key in some unspecified manner; continue with step 4)e).

   d) verifies the claimant's certificate or certificate chain, assuming certificates are used (see Section 3.1.4).

   e) verifies the claimant's signature in TokenAB.

   f) [OPTIONAL] retrieves data from the B, Text2, and Text3 fields, using them in a manner which is outside the scope of this standard.

Successful completion of parts b) through e) in step 4) means that the claimant, A, has authenticated itself to the verifier, B. If any of the verifications in parts b) through e) fail, then the authentication exchange is terminated. If certain conditions either are or are not met in the optional parts a) and f), the verifier may choose to terminate the authentication exchange; these conditions are implementation-specific, and outside the scope of this standard.

3.3 Mutual authentication protocol

The following mutual entity authentication protocol is based on Section 5.2.2, "Three pass authentication", of ISO/IEC 9798-3. Certain authentication token fields and protocol steps are specified in greater detail in this section than in ISO/IEC 9798-3. Either entity may choose to terminate the authentication exchange at any time. Figure 2 illustrates this exchange.

The mutual authentication protocol refers to entities A and B as "initiator" and "responder". This differs from terminology used to describe unilateral authentication in Section 3.2, because each entity acts as *both* a claimant and a verifier in the protocol below.

It is important to note that the success of an entity's authentication, according to this standard, is not dependent on the information contained in the text fields. As described in Section 2.1, the authentication of an entity depends on two things: (1) the verification of the claimant's binding with its key pair, and (2) the verification of the claimant's digital signature on the random number challenge. How text field information is used once an entity's authenticity is verified is beyond the scope of this standard.

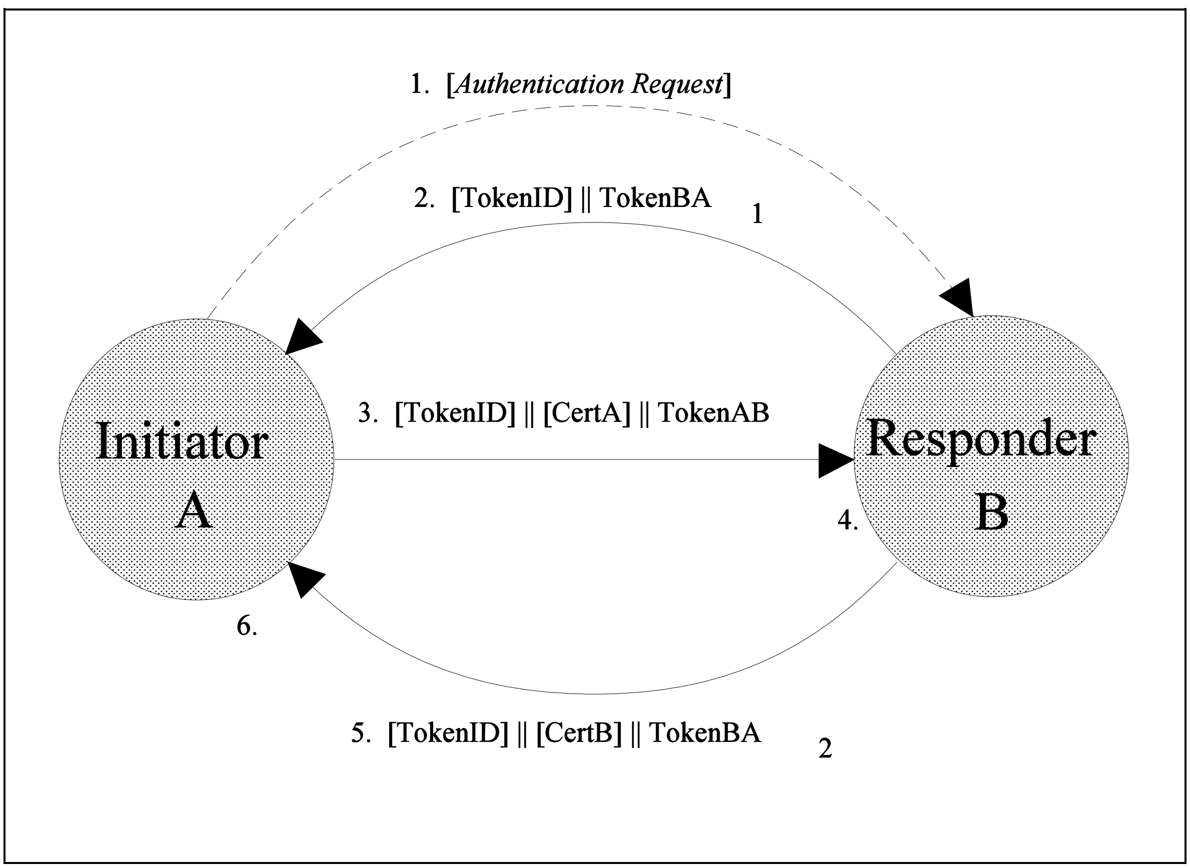

Figure 2 *Mutual Authentication Protocol*

Mutual entity authentication occurs as follows:

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, B, determines if it will continue, initiate, or terminate the authentication exchange. If it attempts to authenticate the initiator, the responder then

   a) generates a random number challenge, which is the value for the R_B field in $TokenBA_1$ below, and retains this value.

   b) [OPTIONAL] generates and/or selects other data which is to be included in the Text1 field of $TokenBA_1$.

   The responder creates a challenge token of the following form:

   $TokenBA_1$ = $R_B \| [Text1]$

   Entity B sends a message consisting of $TokenBA_1$ and an optional TokenID to the initiator. The message from the responder to the initiator is of the form:

   $$[TokenID] \| TokenBA_1$$

3) Upon receiving the message including $TokenBA_1$, the initiator, A,

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

   b) [OPTIONAL] retrieves information from the Text1 field, using it in a manner which is outside the scope of this standard.

   c) generates a random number challenge which is the value for the R_A field in TokenAB below; the value of R_A is retained by the initiator.

   d) [OPTIONAL] selects an identifier for the responder, and includes that in the B field of TokenAB.

   e) [OPTIONAL] generates and/or selects other data which is to be included in the Text2 and Text3 fields. In TokenAB, Text2 is a subset of the Text3 field, including cases where Text2 is the NULL set, and where Text2 equals Text3.

The initiator creates an authentication token, TokenAB, by concatenating data and generating a digital signature:

TokenAB = $R_A \parallel [R_B] \parallel [B] \parallel [Text3] \parallel sS_A(R_A \parallel R_B \parallel [B] \parallel [Text2])$

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 (see Section 2.2).

In addition to containing TokenAB, the message may optionally include a token identifier, TokenID, and the initiator's certificate (or chain of certificates), CertA. The message from the initiator to the responder is of the form:

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

4) Upon receiving the TokenAB transmission, the responder, B,

   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), depending on which optional information has been included. In the event that certificates are not used for signature verification, then the identifier for entity A should be used to retrieve the initiator's public key in some unspecified manner; continue with step 4)e).

   d) verifies the initiator's certificate or certificate chain, assuming certificates are used (see Section 3.1.4).

   e) verifies the initiator's signature in TokenAB.

   f) [OPTIONAL] retrieves data from the B, Text2, and Text3 fields, using them in a manner which is outside the scope of this standard.

Successful completion of parts b) through e) in step 4) means that the initiator, A, has authenticated itself to the responder, B. If any of the verifications in parts b) through e) fail, then the authentication exchange is terminated. If certain conditions either are or are not met in the optional parts a) and f), the responder may choose to terminate the authentication exchange; these conditions are implementation-specific, and outside the scope of this standard.

5) The responder, B,

a) [OPTIONAL] selects an identifier for the initiator, and includes that in the A field of $TokenBA_2$ below.

b) [OPTIONAL] generates and/or selects other data which is to be included in the Text4 and Text5 fields. In $TokenBA_2$, Text4 is a subset of the Text5 field - this includes cases where Text4 is the NULL set, and where Text4 equals Text5.

The responder creates an authentication token, $TokenBA_2$, by concatenating data and generating a digital signature:

$TokenBA_2$ = $[R_B] \parallel [R_A] \parallel [A] \parallel [Text5] \parallel sS_B(R_B \parallel R_A \parallel [A] \parallel [Text4])$

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 in the unsigned data of $TokenBA_2$, they *must* be included in the signed data (see Section 2.2).

In addition to containing $TokenBA_2$, the message may optionally include a token identifier, TokenID, and the responder's certificate (or chain of certificates), CertB. The message from the responder to the initiator is of the form:

$$[TokenID] \parallel [\ CertB\ ] \parallel TokenBA_2$$

6) Upon receiving the message including $TokenBA_2$, 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 $TokenBA_1$, given that R_B is present in $TokenBA_2$. If this value is not present, then the value of R_B retrieved from $TokenBA_1$ 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 $TokenBA_1$, depending on which optional information has been included. In the event that certificates are not used for signature verification, then the identifier for entity B should be used to retrieve the initiator's public key in some unspecified manner; continue with step 6)f).

e) verifies the responder's certificate or certificate chain, assuming certificates are used (see Section 3.1.4).

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

g) [OPTIONAL] retrieves data from the A, Text4, and Text5 fields, using them in a manner which is outside the scope of this standard.

Successful completion of parts b) through f) in step 6) means that the responder, B, has authenticated itself to the initiator, A, and thus the entities have successfully mutually authenticated. If any of the verifications in parts b) through f) fail, then the authentication exchange is terminated. If certain conditions either are or are not met in the optional parts a) and g), the initiator may choose to terminate the authentication exchange - these conditions are implementation-specific, and outside the scope of this standard.

Appendix A

Example Authentication Tokens Using ASN.1 Notation and CER/DER Encoding

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 formatting and encoding authentication tokens and messages from Section 3 of this standard. Abstract Syntax Notation One (ASN.1) is used to format authentication tokens and messages, which are then encoded using Canonical Encoding Rules (CER) or Distinguished Encoding Rules (DER). This appendix also specifies optional representations of other data such as public key certificates. The implementation of a unique set of encoding rules is necessary for interoperability to take place between various authentication domains.

The reader should note that in ASN.1 notation, optional fields are indicated with the word "OPTIONAL", and not with square brackets "[ ]", as used in the body of this standard. In ASN.1 notation, square brackets are used to indicate tagged fields.

A.1 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 ISO and ITU (previously CCITT) protocols as well as many Internet protocols. It is comprised of two parts: the ASN.1 notation [1-4] and the ASN.1 encoding rules [5].

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 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, namely: basic encoding rules, canonical encoding rules, and distinguished encoding rules. Whereas the basic encoding rules give 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, allowing the receiver to unambiguously decode the ASN.1 notation. 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.

A.2 ASN.1 Specification of Entity Authentication Tokens and Messages

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

MessageBA$_1$ is used in the unilateral authentication exchange and is sent by the verifier to the claimant. MessageBA$_1$ is also used in the mutual authentication exchange and is sent by the responder to the initiator. MessageBA$_1$ contains an optional token identifier and authentication token BA$_1$.

```
-- tokenId.tokenType = 0x0001 for unilateral authentication
-- tokenId.tokenType = 0x0011 for mutual authentication

MessageBA1 ::= SEQUENCE {
     tokenId               [0]   TokenId OPTIONAL,
     tokenBA1                    TokenBA1
}

TokenBA1 ::= SEQUENCE {
     ranB                        RandomNumber,
     text1                       Text OPTIONAL
}
```

MessageAB is used in the unilateral authentication exchange and is sent by the claimant to the verifier. MessageAB is also used in the mutual authentication exchange and is sent by the initiator to the responder. MessageAB contains an optional token identifier, optional certification information, and authentication token AB.

```
-- tokenId.tokenType = 0x0002 for unilateral authentication
-- tokenId.tokenType = 0x0012 for mutual authentication
```

```
MessageAB ::= SEQUENCE {
     tokenId              [0]   TokenId OPTIONAL,
     certA                [1]   CertData OPTIONAL,
     tokenAB                    TokenAB
}

TokenAB ::= SEQUENCE {
     ranA                       RandomNumber,
     ranB                       RandomNumber OPTIONAL,
     entityB                    EntityName OPTIONAL,
     text3                      Text OPTIONAL,
     signature                  SIGNATURE { SigDataAB }
}

-- if entityB is included in TokenAB, then it shall also be included in SigDataAB

SigDataAB ::= SEQUENCE {
     ranA                       RandomNumber,
     ranB                       RandomNumber,
     entityB                    EntityName OPTIONAL,
     text2                      Text OPTIONAL      -- subset of text3
}
```

MessageBA$_2$ is used in the mutual authentication exchange and is sent by the responder to the initiator. MessageBA$_2$ contains an optional token identifier, optional certification information, and authentication token BA$_2$.

```
-- tokenId.tokenType = 0x0013

MessageBA2 ::= SEQUENCE {
     tokenId              [0]   TokenId OPTIONAL,
     certB                [1]   CertData OPTIONAL,
     tokenBA2                   TokenBA2
}

TokenBA2 ::= SEQUENCE {
     ranB                 [0]   RandomNumber OPTIONAL,
     ranA                 [1]   RandomNumber OPTIONAL,
     entityA                    EntityName OPTIONAL,
     text5                      Text OPTIONAL,
     signature                  SIGNATURE { SigDataBA2 }
}

-- if entityA is included in TokenBA2, then it shall also be included in SigDataBA2

SigDataBA2 ::= SEQUENCE {
     ranB                       RandomNumber,
     ranA                       RandomNumber,
     entityA                    EntityName OPTIONAL,
     text4                      Text OPTIONAL      -- subset of text5
}
```

The token identifier is defined here to include the type of token and the protocol version number. This version is defined here as Version 2, to distinguish it from Version 1, which was defined in an earlier draft of this appendix.

```
TokenId ::= SEQUENCE {
     tokenType                   INTEGER,
     protoVerNo                  INTEGER {v2(2)},
}
```

The certification data includes a certification path and/or a certificate revocation list. The definitions for **CertificationPath** and **CertificateList** are imported from ITU-T Rec. X.509 | ISO/IEC 9594-8, The Directory: Authentication Framework [6], and are given below in Section A.3.

```
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 octet string.

```
RandomNumber ::= OCTET STRING
```

An entity name is defined as the alternative name forms defined in Draft Amendment 1 to ITU-T Rec. X.509 | ISO/IEC 9594-8 on Certificate Extensions [7].

```
EntityName ::= AltNames

AltNames ::= SEQUENCE OF CHOICE {
     otherName            [0]   INSTANCE OF OTHER-NAME,
     rfc822Name           [1]   IA5String,
     dNSName              [2]   IA5String,
     x400Address          [3]   ORAddress,
     directoryName        [4]   Name,
     ediPartyName         [5]   IA5String
}

OTHER-NAME ::= TYPE-IDENTIFIER
```

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
```

A.3 ASN.1 Specification of Public-Key Certification Information

The definitions for **CertificatePath** and **CertificateList** are taken from Annex A of ITU-T Rec. X.509 | ISO/IEC 9594-8, The Directory: Authentication Framework [6]. These definitions describe Version 2 certificates and certificate revocation lists. Version 3 certificate extensions can be found in Draft Amendment 1 to ITU-T Rec. X.509 | ISO/IEC 9594-8 on Certificate Extensions [7].

```
-- types --

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

CertificateList ::= SIGNED { SEQUENCE {
     signature                  AlgorithmIdentifier,
     issuer                     Name,
     lastUpdate                 UTCTime,
     revokedCertificates        SIGNED { SEQUENCE OF SEQUENCE {
              signature               AlgorithmIdentifier,
              issuer                  Name,
              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,
     issuerUniqueIdentifier     [1]   IMPLICIT UniqueIdentifier OPTIONAL,
                                      -- if present, version must be v2
     subjectUniqueIdentifier    [2]   IMPLICIT UniqueIdentifier OPTIONAL
                                      -- if present, version must be v2
}}

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

CertificateSerialNumber ::= INTEGER

AlgorithmIdentifier ::= SEQUENCE {
     algorithm       ALGORITHM.&id({SupportedAlgorithms}),
     parameters      ALGORITHM.&Type({SupportedAlgorithms}{@algorithm})
                           OPTIONAL
}

-- Definition of the following information object set ifs deferred, perhaps to standardized
```

```
-- profiles or to protocol implementation conformance statements.  The set is required to
-- specify a table constraint on the parameters component of AlgorithmIdentifier.
-- SupportedAlgorithms      ALGORITHM ::= { ... | ... }

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

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

-- information object classes --

ALGORITHM ::= TYPE-IDENTIFIER

-- parameterized types --

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

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

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

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

The definition for **UniqueIdentifier** is imported from Annex A of ITU-T X.520 | ISO/IEC 9594-6, The Directory: Selected Attribute Types [8].

```
UniqueIdentifier ::= BIT STRING
```

The definition for **Name** is taken from Annex B of ITU-T Rec. X.501 | ISO/IEC 9594-2, The Directory: The Models [9].

```
-- naming data types --

Name ::= CHOICE {RDNSequence -- one possibility for now --}

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
```

```
RelativeDistinguishedName ::= SET SIZE(1 .. MAX) OF AttributeTypeAndValue

-- attribute data types --

AttributeTypeAndValue ::= SEQUENCE{
     type                AttributeType({SupportedAttributes}),
     value               AttributeValue({SupportedAttributes}{@type})
}

AttributeType ::= ATTRIBUTE.&id

AttributeValue ::= ATTRIBUTE.&Type

-- Definition of the following information object set ifs deferred, perhaps to standardized
-- profiles or to protocol implementation conformance statements.  The set is required to
-- specify a table constraint on the value component of AttributeTypeAndValue.

-- SupportedAttributes     ATTRIBUTE ::= { ... | ... }
```

The reader should consult Annex B of ITU-T Rec. X.501 | ISO/IEC 9594-2 [9] for the **ATTRIBUTE** information object class specification.

Appendix B

Example Authentication Token Formatting and Encoding Based on the Simple Public-Key GSS-API Mechanism (SPKM) Specification

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 method for formatting and encoding authentication tokens in Section 3 of this standard.

The Simple Public-Key GSS-API Mechanism (SPKM) Specification is an Internet Proposed Standard (RFC 2025) [10] that describes several challenge-response mechanisms to enable entity authentication using public-key cryptography. That specification "defines protocols, procedures, and conventions to be employed by peers implementing the Generic Security Service Application Program Interface (as specified in RFCs 1508 [11] and 1509 [12]) when using the Simple Public-Key Mechanism." [10] The SPKM-1 GSS-API mechanism is capable of implementing both authentication exchanges described in this standard, while taking advantage of optional token fields and authentication steps in Section 3. SPKM-1 also provides implementors with a means to establish a key between the authenticating entities and protect the confidentiality of data. The SPKM-2 mechanism in [10] is not addressed in this appendix because it relies on timestamps - not signed random number challenges - to authenticate entities to one another.

By using GSS-API, entities implementing SPKM authentication can perform context establishment, algorithm negotiation and key establishment. For purposes of this appendix, non-FIPS approved security methods referenced in [10] will not be discussed in the following sections.

B.1 ASN.1 Specification of SPKM Tokens and Messages

SPKM describes the various GSS-API tokens that are to be passed between entities to perform unilateral and mutual authentication. This section describes the formats of those tokens in ASN.1 notation, which is explained in Appendix A, Section A.1. What follows is not a complete list of ASN.1 formatted tokens for the SPKM-1 mechanism, and this list does not completely describe those tokens or the steps necessary for performing SPKM-1 authentication using GSS-API. For example, other optional SPKM GSS-API tokens (not equivalent to authentication tokens described in this FIPS) may be implemented to provide for the data integrity and encryption of the SPKM-1 authentication tokens (`SPKM-REQ`, `SPKM-REP-TI`, and `SPKM-REP-IT`). Such GSS-API tokens may also be used for error recovery during an authentication exchange. The implementor should consult the SPKM specification in [10] to obtain complete details on implementing an SPKM-1 mechanism.

In the token and message descriptions below, SPKM terminology differs slightly from terminology defined in Section 2.3 of this FIPS. The SPKM specification refers to an "initiator" and "target". When performing either unilateral or mutual SPKM-1 authentication, the "initiator" is the equivalent of entity B in this FIPS, and the "target" is the equivalent of entity A. The "initiator" in SPKM authentication is actually the "responder" described in the mutual authentication exchange in Section 3.3.

```
SpkmGssTokens      {iso(1) identified-organization(3) dod(6) internet(1)
                    security(5) mechanisms(5) spkm(1) spkmGssTokens(10)}

-- types --

     CertificationData ::= SEQUENCE {
          certificationPath          [0]   CertificationPath OPTIONAL,
          certificateRevocationList  [1]   CertificateList OPTIONAL
     } -- at least one of the above shall be present

     CertificationPath ::= SEQUENCE {
          userKeyId            [0]   OCTET STRING OPTIONAL,
          userCertif           [1]   Certificate OPTIONAL,
          verifKeyId           [2]   OCTET STRING OPTIONAL,
          userVerifCertif      [3]   Certificate OPTIONAL,
          theCACertificates    [4]   SEQUENCE OF CertificatePair OPTIONAL
     } -- Presence of [2] or [3] implies that [0] or [1] must also be
       -- present.  Presence of [4] implies that at least one of [0], [1],
       -- [2], and [3] must also be present.


  -- The requestToken in SPKM-REQ represents TokenBA1 in the FIPS --
  -- authentication exchanges.  It contains B's random challenge to A. --

     SPKM-REQ ::= SEQUENCE {
          requestToken                 REQ-TOKEN,
          certif-data    [0]           CertificationData OPTIONAL,
          auth-data      [1]           AuthorizationData OPTIONAL
     }

     REQ-TOKEN ::= SEQUENCE {
          req-contents                 Req-contents,
          algId                        AlgorithmIdentifier,
          req-integrity                Integrity  -- "token" is Req-contents
     }

     Integrity ::= BIT STRING
          -- See the special note in [10] regarding this type.

     Req-contents ::= SEQUENCE {
          tok-id                       INTEGER (256),  -- shall contain 0100 (hex)
          context-id                   Random-Integer,
          pvno                         BIT STRING,
          timestamp                    UTCTime OPTIONAL,
          randSrc                      Random-Integer,
          targ-name                    Name,
          src-name       [0]           Name OPTIONAL,
          req-data                     Context-Data,
          validity       [1]           Validity OPTIONAL,
          key-estb-set                 Key-Estb-Algs,
          key-estb-req                 BIT STRING OPTIONAL,
          key-src-bind                 OCTET STRING OPTIONAL
     }

     Random-Integer ::= BIT STRING
```

```
   Context-Data ::= SEQUENCE {
        channelId                  ChannelId OPTIONAL,
        seq-number                 INTEGER OPTIONAL,
        options                    Options,
        conf-alg                   Conf-Algs,
        intg-alg                   Intg-Algs,
        owf-alg                    OWF-Algs
   }

   ChannelId ::= OCTET STRING

   Options ::= BIT STRING {
        delegation-state (0),
        mutual-state (1),
        replay-det-state (2),
        sequence-state (3),
        conf-avail (4),
        integ-avail (5),
        target-certif-data-required (6)
   }

   Conf-Algs ::= CHOICE {
        algs           [0]         SEQUENCE OF AlgorithmIdentifier,
        null           [1]         NULL
   }    -- confidentiality algorithms

   Intg-Algs ::= SEQUENCE OF AlgorithmIdentifier     -- integrity algorithms

   OWF-Algs ::= SEQUENCE OF AlgorithmIdentifier

   Key-Estb-Algs ::= SEQUENCE OF AlgorithmIdentifier  -- key establishment
                                                        -- algorithms

   AlgorithmIdentifier ::= SEQUENCE {
        algorithm                  OBJECT IDENTIFIER,
        parameter                  ANY DEFINED BY algorithm OPTIONAL
   }    -- Note that the 1993 AuthenticationFramework module in [6] uses
        -- different syntax for this construct.


-- The responseToken in SPKM-REP-TI represents TokenAB in the FIPS --
-- authentication exchanges.  It contains A's signature on B's challenge, --
-- and a random challenge to B during mutual authentication. --

   SPKM-REP-TI ::= SEQUENCE {
        responseToken              REP-TI-TOKEN,
        certif-data                CertificationData OPTIONAL
          -- present if target-certif-data-required option was
          -- set to TRUE in SPKM-REQ
   }

   REP-TI-TOKEN ::= SEQUENCE {
        rep-ti-contents            Rep-ti-contents,
        algId                      AlgorithmIdentifier,
        rep-ti-integ               Integrity  -- "token" is Rep-ti-contents
   }
```

```
    Rep-ti-contents ::= SEQUENCE {
         tok-id                        INTEGER (512),  -- shall contain 0200 (hex)
         context-id                    Random-Integer,
         pvno            [0]           BIT STRING OPTIONAL,
         timestamp                     UTCTime OPTIONAL,
         randTarg                      Random-Integer,
         src-name        [1]           Name OPTIONAL,
         targ-name                     Name,
         randSrc                       Random-Integer,
         rep-data                      Context-Data,
         validity        [2]           Validity  OPTIONAL,
         key-estb-id                   AlgorithmIdentifier OPTIONAL,
         key-estb-str                  BIT STRING OPTIONAL
    }


  -- The responseToken in SPKM-REP-IT represents TokenBA2 in the FIPS --
  -- mutual authentication exchange.  It contains B's signature on --
  -- A's random challenge. --

    SPKM-REP-IT ::= SEQUENCE {
         responseToken                 REP-IT-TOKEN,
         algId                         AlgorithmIdentifier,
         rep-it-integ                  Integrity  -- "token" is REP-IT-TOKEN
    }

    REP-IT-TOKEN ::= SEQUENCE {
         tok-id                        INTEGER (768),  -- shall contain 0300 (hex)
         context-id                    Random-Integer,
         randSrc                       Random-Integer,
         randTarg                      Random-Integer,
         targ-name                     Name,
         src-name                      Name OPTIONAL,
         key-estb-rep                  BIT STRING OPTIONAL
    }


-- other types --

    -- from [11] --

    MechType ::= OBJECT IDENTIFIER

    InitialContextToken ::= [APPLICATION 0] IMPLICIT SEQUENCE {
         thisMech                      MechType,
         innerContextToken             SPKMInnerContextToken
                                             -- when thisMech is SPKM-1 or SPKM-2
    }

    SPKMInnerContextToken ::= CHOICE {
         req             [0]           SPKM-REQ,
         rep-ti          [1]           SPKM-REP-TI,
         rep-it          [2]           SPKM-REP-IT,
         error           [3]           SPKM-ERROR,
         mic             [4]           SPKM-MIC,
         wrap            [5]           SPKM-WRAP,
         del             [6]           SPKM-DEL
    }
```

```
-- object identifier assignments --

     spkm-1 OBJECT IDENTIFIER ::=
          {iso(1) identified-organization(3) dod(6) internet(1) security(5)
           mechanisms(5) spkm(1) spkm-1(1)}


     -- The following two object identifiers for signature algorithms are derived
     -- from "Stable Implementation Agreements for Open Systems Interconnection
     -- Protocols:  Part 12 - OS Security" [13].  This document was prepared by
     -- the Security Special Interest Group (SECSIG) of the Open Systems
     -- Environment Implementors' Workshop (OIW) hosted by NIST.  The OIW has
     -- registered the following algorithms with ISO.  The DSA and SHA-1 are
     -- examples of a current FIPS approved digital signature and secure hash
     -- algorithm, respectively.  Although these examples are not given in [10],
     -- the SPKM specification states that the examples included in [10], "(and
     -- any other algorithms) may optionally be supported by a given SPKM
     -- implementation" [10].

     dsaWithSHA1 OBJECT IDENTIFIER ::= {
          iso(1) identified-organization(3) oiw(14) secsig(3)
          algorithm(2) 27
     }

     dsaCommonWithSHA1 OBJECT IDENTIFIER ::= {
          iso(1) identified-organization(3) oiw(14) secsig(3)
          algorithm(2) 28      -- uses common parameters p, q, and g, which
                               -- are distributed externally
     }
```

B.2 ASN.1 Specification of Public-Key Certification Information

The SPKM-1 mechanism uses many certificate-related ASN.1 types that are derived from [6] and [9], which are either the same as or similar to the definitions listed in Section A.3 of Appendix A in this FIPS. Rather than repeat them here and note the differences, the implementor should consult Appendix B of the SPKM specification [10] for more complete details.

Appendix C

Example Authentication Token Formatting Based on ANSI X9.26-1990

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 method for formatting authentication tokens from Section 3 of this FIPS. Formatting specifications are based on message formats defined in ANSI X9.26-1990, *Financial Institution Sign-On Authentication for Wholesale Financial Transactions* [14]. Note that these formatted tokens may additionally have to be encoded before transmission, depending on the nature of a particular implementation. One possible encoding process is described in Appendix D of this standard.

C.1 Background

When authentication is conducted within a security services protocol, the protocol definition generally specifies where the authentication information is located and how it is represented. The individual token fields may be represented as elements in a data structure, or they may be encoded into a transmission unit using, for example, the ASN.1/DER formatting and encoding described in Appendix A. However, many implementations of the FIPS authentication protocols may not be capable of handling (or require the overhead of) ASN.1/DER formatting and encoding. Therefore, this appendix provides an ASCII-based alternative. In this appendix, the token fields are formatted based on notation and message formats in ANSI X9.26 [14]. That standard addresses several types of sign-on authentication that use secret-key, not public-key cryptography. However, with slight modifications to field tags, function definitions, and message formats, an ANSI X9.26-"like" implementation can perform either of the FIPS authentication protocols.

C.2 Token and Message Formats Based on ANSI X9.26

As indicated above, this appendix will allow an implementor to generate authentication tokens based on message formats described in ANSI X9.26. The abbreviations, notation, and basic functions described below were drawn from [14], with additions or slight modifications, which are marked with an asterisk ("*").

C.2.1 Abbreviations

The message formats described in Table I below assume the use of certain abbreviations, some of which are based on abbreviations in [14].

Table I: *Formatting abbreviations based on ANSI X9.26*

ABBREV.	MEANING	REMARKS
*CRA	Entity A's Public Key Certificate	A public key certificate binding entity A to its public/private key pair.
*CRB	Entity B's Public Key Certificate	A public key certificate binding entity B to its public/private key pair.
CSM	Cryptographic Service Message	A message involved in creating a service (such as authentication) using cryptography.
DATA	Data	The input of the Crypto Function (e.g., the result of the Combine Function).
*GSA	GSF generated by entity A	Output of the GSF generated with entity A's KPRI.
*GSB	GSF generated by entity B	Output of the GSF generated with entity B's KPRI.
GSF	General Security Function	* Function used to authenticate ORG to RCV.
INPUT	Input	The input of the Select Function (e.g., the result of the Crypto Function).
*KPRI	Private Key	Key used to generate a digital signature.
MCL	Message Class	The tag for the field that defines the type of CSM.
ORG	Originator	The entity sending the CSM.
RCV	Recipient	The entity receiving the CSM.
*SMA	Entity A's SOM	Sign-on Message with entity A's digital signature.
*SMB	Entity B's SOM	Sign-on Message with entity B's digital signature.
SOM	Sign-on Message	An MCL used in the sign-on authentication CSM.
TTM	TVP Transmission Message	An MCL used in the sign-on authentication CSM.
*TVA	Initiator's TVP	A TVP generated by the initiator, A.
*TVB	Responder's TVP	A TVP generated by the responder, B.
TVP	Time Variant Parameter	A random or pseudorandom number generated with a FIPS approved random number generator.

C.2.2 Notation

The following notation shall be used when formatting authentication tokens as described in Section C.2.3 of this appendix:

1) The character set for Cryptographic Service Messages shall be the following characters: digits (0-9), letters (A-Z), comma (,), period (.), space (b̲), solidus (/), hyphen (-), asterisk (*), and open and close parentheses ((&)). The character (b̲) shall only be used in a message to separate fields. The character (.) shall only be used in a field to separate subfields (if required).

2) The presence of a Cryptographic Service Message is denoted by the field tag, "CSM".

3) The contents of each message shall begin with an open parenthesis "(" and end with a close parenthesis ")".

4) Field tags shall be separated from field contents by a solidus "/".

5) Fields shall be separated by a space (b̲).

*6) For illustration, plaintext fields are represented by "ppp"; fields containing output of the GSF are represented by "fff".

7) It is the responsibility of the implementor to ensure that no delimiters (e.g., "b̲" and ".") appear in the user defined fields (e.g., ORG, RCV).

*8) ORG and RCV refer to the two principals involved in the authentication exchange. For each token, ORG and RCV are the sender and recipient, respectively, of a CSM.

*9) For field tags indicating a certificate (e.g., CRA, CRB), if no certificate is present, the field contents shall consist only of a space, (b̲).

C.2.3 Basic Functions

*1) The Combine Function:

The Combine Function combines multiple input values. For purposes of this appendix, the Combine Function is defined to equal the concatenation of the input values, in the order that they are listed within this function. A solidus '/' shall be used to separate the concatenated fields. In Section C.2.4 below, the Combine Function is of the following form:

```
Combine( RCV,ORG,TVB,TVA )
```

For example, if RCV=`alice`, ORG=`bob`, TVB=`abcdef01`, TVA=`23456789`, then the result of the combine function would equal:

```
alice/bob/abcdef01/23456789
```

*2) The Crypto Function:

The Crypto Function cryptographically transforms the input DATA (e.g., the result of the Combine Function), using the CSM sender's (ORG) private key, KPRI. For purposes of this appendix, the Crypto Function shall use a FIPS approved digital signature algorithm (e.g., the Digital Signature Algorithm specified in FIPS PUB 186) to generate a digital signature. In Section C.2.4 below, the Crypto Function is of the following form:

```
Crypto( KPRI,DATA )
```

where DATA equals the result of the Combine Function in (1) above.

*3) The Select Function:

The Select Function selects and returns all or part of the data from INPUT (e.g., the result of the Crypto Function), facilitating the compression of messages. For purposes of this appendix, the Select Function will select <u>all</u> *n* bits of INPUT. In Section C.2.4 below, the Select Function is of the following form:

```
Select ( INPUT )
```

where INPUT equals the result of the Crypto Function in (2) above

*4) The General Security Function (GSF), is used to authenticate one entity to another. In Section C.2.4 below, the GSF is formed by composing the previously described functions as follows:

```
GSF( PRI,RCV,ORG,TVB,TVA ) =

Select( Crypto( KPRI,Combine( RCV,ORG,TVB,TVA ) ) )
```

C.2.4 Message Formats

In each of the messages described below, the message class tag (MCL) and the field value which follows are considered to be a TokenID. In `MessageAB` and `MessageBA2`, each included certificate should have a predefined format, which is left to the implementor to define. The message formats for entity authentication tokens, based on modified ANSI X9.26 specifications, are as follows:

TokenBA$_1$: In both the unilateral and mutual exchanges, the first message is a random challenge sent from B to A. The message (`MessageBA1`) that includes TokenBA$_1$ ("RCV/...TVB/ppp") is:

```
MessageBA1 =
CSM(MCL/TTMbRCV/pppbORG/pppbTVB/pppb)
```

TTM TVP Transmission Message indicator (message class)
RCV Identity of message recipient, entity A
ORG Identity of message sender, entity B
TVB Random number, in hexadecimal format, generated by entity B, the message sender

----

TokenAB: In both the unilateral and mutual exchanges, the second token is a response to B's random challenge. Also included in the token is a random number generated by A. The message (`MessageAB`) that includes TokenAB ("RCV/...GSA/fff") is:

```
MessageAB =
CSM(MCL/SMAbRCV/pppbORG/pppbTVB/pppbTVA/pppbGSA/fffbCRA/pppb)
```

SMA Entity A's Sign-on Message indicator (message class)
RCV Identity of message recipient, entity B
ORG Identity of message sender, entity A
TVB Random number, in hexadecimal format, generated by entity B, the message recipient
TVA Random number, in hexadecimal format, generated by entity A, the message sender
GSA The output of the GSF generated by entity A.
CRA Entity A's public key certificate (optional).

TokenBA$_2$: In the mutual authentication exchange, a third message is sent, which includes B's response to A's random challenge. The message (`MessageBA2`) that includes TokenBA$_2$ ("RCV/...GSB/fff") is:

```
MessageBA2 =
CSM(MCL/SMBbRCV/pppbORG/pppbTVB/pppbTVA/pppbGSB/fffbCRB/pppb)
```

SMB Entity B's Sign-on Message indicator (message class)
RCV Identity of message recipient, entity A
ORG Identity of message sender, entity B

TVB	Random number, in hexadecimal format, generated by entity B, the message sender
TVA	Random number, in hexadecimal format, generated by entity A, the message recipient
GSB	The output of the GSF generated by entity B.
CRB	Entity B's public key certificate (optional).

Appendix D

Example Entity Authentication Exchange Using Base64 Encoding

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 method for encoding authentication tokens. The method described below for encoding formatted tokens is Base64 Content-Transfer-Encoding, defined in Internet RFC 1521, MIME (Multipurpose Internet Mail Extensions) Part One [15]. By converting data to ASCII characters, this encoding method allows authentication tokens to be exchanged over unstructured, non-binary channels. Note that this encoding method is independent of the format of the authentication tokens. For example, ASN.1/DER-encoded tokens from Appendices A and B might also be base64-encoded, depending on the transmission channel's ability to handle binary data. Also, there exist other, similar encoding schemes that may provide more suitable alternatives for a particular implementation. Section D.5 will present an example of using base64 encoding with authentication tokens described in Appendix C, even though they are already formatted using only ASCII characters.

D.1 Background

When authentication is conducted within a security services protocol, the protocol definition generally specifies where the authentication information is located and how it is represented. Once the authentication information is formatted, it may then have to be encoded in a binary form. If the channel being used to transmit the authentication information is incapable of handling binary data, one possible solution is to further encode the data using base64 encoding as defined in [15].

Regardless of the encoding method used (prior to performing base64 encoding), a formatted authentication message will be referred to in this appendix as a "transfer string", which is simply a sequence of octets with known length. The method for encoding the transfer string is independent of the method used to format the transfer string; however, this appendix describes the use of Base64 Content-Transfer-Encoding [15].

Protocols which explicitly support security services are able to carry authentication transfer strings as-is. However, channels which have no protocol support for security services may also require strong authentication. In such cases the authentication information is carried "in-band" as part of the normal user data. This presents two potential problems: the authentication data does not necessarily appear at a fixed location, and the channel may not be able to pass arbitrary binary information. The format specified in this appendix addresses both problems.

- Tokens begin with a fixed label (token identifier); receivers can scan an incoming data stream for labels to locate potential authentication tokens.

- The token data is converted to a text representation designed to pass through most channels without difficulty.

D.2 Base64 Content-Transfer-Encoding

Assuming that a transfer string has been generated for a an authentication token, the encoding of that token must take place before including it in an "in-band" transmission. The transfer string may be converted to a base64 string according to the procedure defined in Internet RFC 1521. The following description is adapted from [15], Section 5.2:

> The Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. The encoding and decoding algorithms are simple, but the encoded data are consistently only about 33 percent larger than the unencoded data. . . . A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, “=”, is used to signify a special processing function.). . . .
>
> The encoding process represents 24-bit groups of input bits as output strings of [four] encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating [three] 8-bit input groups. These 24 bits are then treated as [four] concatenated 6-bit groups, each of which is translated into a single digit in the base64 alphabet. When encoding a bit stream via the base64 encoding, the bit stream must be presumed to be ordered with the most-significant-bit first. That is, the first bit in the stream will be the high-order bit in the first byte, and the eighth bit will be the low-order bit in the first byte, and so on.

***Table II**: The Base64 Alphabet*

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	I	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

> Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string. These characters, identified in [Table II], are selected so as to be universally representable. . . .
>
> The output stream (encoded bytes) must be represented in lines of no more than 76 characters each. All line breaks or other characters not found in [Table II] must be ignored by decoding software. In base64 data, characters other than those in [Table II], line breaks, and other white space probably indicate a transmission error, about which a warning message or even a message rejection might be appropriate under some circumstances.
>
> Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding [unit] is always completed at the end of a body. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the '=' character. Since all base64 input is an integral number of octets, only the following cases can arise: (1) the final [unit] of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be an integral multiple of [four] characters with no "=" padding, (2) the final [unit] of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or (3) the final [unit] of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character. [15]

D.3 Format of In-Band Authentication Messages

Once a token's transfer string has been base64-encoded, it can then be formatted into a message of the form:

Label":"base64-string":"

The label identifies the type of token being sent, and may be considered to be the TokenID (or part of the TokenID) specified in the FIPS authentication protocols. Labels have been given the "FIPSEA" prefix, which stands for "FIPS Entity Authentication":

FIPSEA_BA1	-Initial token from B to A in both Unilateral and Mutual exchanges.
FIPSEA_AB	-Response token from A to B in both Unilateral and Mutual exchanges.
FIPSEA_BA2	-Response token from B to A in the Mutual exchange.

The base64-string is delimited by colon (:) characters to allow its extent to be determined unambiguously. The initial colon must immediately follow the Label with no intervening whitespace or other characters. The base64-string may include line breaks or other whitespace, as specified in Section D.2.

D.4 Error detection

This formatting method does not explicitly include any error control mechanisms, although it does allow some sanity checks to be applied to received data. Because they are used only at the end of the data, the occurrence of any '=' or ':' characters may be taken as evidence that the end of the data has been reached without truncation in transit. The occurrence of any non-whitespace characters between the '=' characters (if any) and the trailing ':' indicates an error in transit. Any errors in the label/initial colon and any non-base64 character before the final colon should cause the token to be treated as corrupt.

Explicit error control coding (to enable more extensive error detection or correction) can be applied as part of the encoding method, or as a separate step after encoding but before formatting.

D.5 Example

For purposes of example, the authentication messages from Appendix C will be base64-encoded. In many cases, those messages will not need to be encoded, since they are likely to be in ASCII-only format. However, this example will be used to demonstrate both the ANSI X9.26-based formatting and the base64 encoding. The digital signatures in MessageAB and MessageBA2 will be generated using the Digital Signature Algorithm specified in FIPS PUB 186, and no certificates will be included. The base64 encoding is generated beginning with the characters "CSM" and ending with the close parenthesis ")" for each message.

Mutual entity authentication:

Entity A's ID= `alice`
Entity A's TVP= `d6f47bc433299436`
Entity A's private key (hex)= `374b8a09ae40ced4d9510256bf7f3262679e3b3f`

Entity B's ID= `bob`
Entity B's TVP= `e69dfb21ffd051a3`
Entity B's private key (hex): `390382c6aa978875ec7f3160bc9d675e430da255`

```
MessageBA1 =

CSM(MCL/SMA RCV/alice ORG/bob TVB/e69dfb21ffd051a3)
```

Once this message is labeled and base64-encoded, the result is:

FIPSEA_BA1:Q1NNKE1DTC9TTUEgUkNWL2FsaWNlIE9SRy9ib2IgVFZCL2U2OWRmYjIxZmZkMD
UxYTMp:

```
MessageAB =

CSM(MCL/SMA RCV/bob ORG/alice TVB/e69dfb21ffd051a3 TVA/d6f47bc433299436
GSA/b172f97910da3e5dde070c2af334eaa349063695938969d02882a8a7736015e081a38
cf8fd33844c CRA/ )
```

Once this message is labeled and base64-encoded, the result is:


```
FIPSEA_AB:Q1NNKE1DTC9TTUEgUkNWL2JvYiBPUkcvYWxpY2UgVFZCL2U2OWRmYjIxZmZkMDU
xYTMgVFZBL2Q2ZjQ3YmM0MzMyOTk0MzYgR1NBL2IxNzJmOTc5MTBkYTNlNWRkZTA3MGMyYWYz
MzRlYWEzNDkwNjM2OTU5Mzg5NjlkMDI4ODJhOGE3NzM2MDE1ZTA4MWEzOGNmOGZkMzM4NDRjI
ENSQS8gKQ==:
```

```
MessageBA2 =

CSM(MCL/SMB RCV/alice ORG/bob TVB/e69dfb21ffd051a3 TVA/d6f47bc433299436
GSB/c26ac822a93d5c349962d1a78a229d27abfea415b934b2604e6facce2c233afc59be6
be6a9e262f5 CRB/ )
```

Once this message is labeled and base64-encoded, the result is:


```
FIPSEA_BA2:Q1NNKE1DTC9TTUIgUkNWL2FsaWNlIE9SRy9ib2IgVFZCL2U2OWRmYjIxZmZkMD
UxYTMgVFZBL2Q2ZjQ3YmM0MzMyOTk0MzYgR1NCL2MyNmFjODIyYTkzZDVjMzQ5OTYyZDFhNzh
hMjI5ZDI3YWJmZWE0MTViOTM0YjI2MDRlNmZhY2NlMmMyMzNhZmM1OWJlNmJlNmE5ZTI2MmY1
IENSQi8gKQ==:
```

Appendix E

Selected References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 95-94-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions, Editor’s DRAFT, August 1, 1995.

[8] ITU-T X.520 | ISO 9594-6, Information technology - Open Systems Interconnection - The Directory: Selected Attribute Types, Editor's DRAFT, February 14, 1993.

[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] Adams, C., The Simple Public-Key GSS-API Mechanism (SPKM), Internet Proposed Standard RFC 2025, October 1996.

[11] Linn, J., Internet RFC 1508 - Generic Security Service Application Program Interface, September 1993.

[12] Wray, J., Internet RFC 1509 - Generic Security Service API: C-bindings, September 1993.

[13] Mirhakkak, Dr. Mohammad, ed., Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 12 - OS Security, June 1995.

[14] ANSI X9.26, Financial Institution Sign-On Authentication for Wholesale Financial Transactions, Approved February 28, 1990.

[15] Borenstein, N. and N. Freed, Internet RFC 1521 - MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, September 1993.
