Sub-Volume VIII

ATN Security Services

8. ATN SECURITY SERVICES

8.1 INTRODUCTION

8.1.1 This Sub-Volume specifies the ATN security services which are intended to support operational requirements for the secure exchange of ATS information via the ATN and for protection of ATN resources from unauthorized access. The ATN security services will accommodate mobile as well as fixed users of the network.

8.1.2 The ATN security services are used to provide assurance that the originator of a message delivered
via the ATN can be unambiguously authenticated by the receiving entity and that appropriate control is applied when ATN resources are accessed.

Note.— Security services in general support but do not guarantee protection from security violations. In particular, cryptanalytic advances and local implementation issues may affect the overall level of protection.

8.1.3 This sub-volume consists of seven sections introduced as follows:

Section 8.1 contains introductory material to the remainder of the sub-volume.

Section 8.2 presents the general ATN security concept in terms of generic security services to be provided in the ATN.

Section 8.3 contains the ATN security framework. It specifies the ATN information security framework, which includes:

- framework standards,
- the framework for public key infrastructure,
- the framework for provision of security services in ATN systems,
- the framework for provision of security services within the Upper Layer Communication Service,
- the framework for provision of security services within the Context Management application,
- the framework for provision of security services within other ATN applications,
- the framework for provision of security services within SM Managers,
- the framework for provision of security services within ATN Directory Servers,
- the framework for provision of security services for auditing of ATN systems,
- the framework for provision of security services for system management of ATN systems, and
- backward compatibility.

Section 8.3 also identifies the ATN physical security framework.

Section 8.4 contains the ATN Public Key Infrastructure (PKI). It specifies general requirements for certificate policy and certificate practice statements, ATN PKI certificate format, ATN PKI certificate revocation list (CRL) format, and ATN PKI certificate and CRL validation.

Section 8.5 contains the ATN cryptographic infrastructure. It contains terms and notational conventions and specifies the ATN cryptographic setting, the ATN key agreement scheme, the ATN digital signature scheme, the ATN keyed message authentication code scheme, and ATN auxiliary cryptographic functions.

Section 8.6 contains the ATN System Security Object. It specifies a set of abstract services for the generation and verification of security items.

Section 8.7 contains the ASN.1 module for ATN security.

8.2 ATN GENERIC SECURITY SERVICES

8.2.1 Access control services shall be provided to prevent the unauthorized use of ATN resources.

8.2.2 Authentication services shall be provided to ensure the identity of participating entities is as claimed.

Note.— The ATN security strategy employs asymmetric and symmetric cryptographic mechanisms (section 8.5) to provide strong authentication services.

8.2.3 Data integrity services shall be provided to ensure that ATN data is not altered or destroyed in an unauthorized manner.

Note.— Although no requirements exist herein for confidentiality services, i.e., for encryption of ATS application data, the ATN information security services consists of a set of security related functions and a cryptographic setting that may be used by States and organizations in support of applications which require confidentiality but not subject to ICAO standardization.

8.3 ATN SECURITY FRAMEWORK

Note.— The ATN information security framework is based on ISO/IEC 7498-2. ISO 7498-2 defines basic terms and concepts used in specifying ATN security.

8.3.1 ATN Information Security

8.3.1.1 Framework Standards

Note 1.— The ATN information security framework is based on ISO/IEC 10181-1: Information Technology - Security Frameworks in Open Systems - Frameworks Overview.

Note 2.— The ATN information security framework is based on ISO/IEC 10181-2: Information Technology - Security Frameworks in Open Systems - Authentication Framework.

Note 3.— The ATN information security framework is based on ISO/IEC 10181-3: Information Technology - Security Frameworks in Open Systems - Access Control Framework.

Note 4.— The ATN information security framework is based on ISO/IEC 10181-6: Information Technology - Security Frameworks in Open Systems - Integrity Framework.

8.3.1.1.1 The ATN upper layer communications services shall provide secured dialogues for ATN security services based on ISO/IEC 11586-1.

Note.— The ATN upper layer communications services are defined in Sub-Volume IV.

8.3.1.1.2 Secured exchanges associated with the ATSMHS application shall provide secured messaging based on ISO/IEC 10021.

Note.— The ATSMHS application is defined in Sub-Volume III.

8.3.1.1.3 The ATN internet communication service shall provide secured exchanges of routing information among Boundary Intermediate Systems based on provisions for authentication in ISO/IEC 10747.

Note.— The ATN internet communication service is defined in Sub-Volume V.

8.3.1.2 Framework for Public Key Infrastructure

8.3.1.2.1 The ATN information security framework shall provide a Public Key Infrastructure for key management and distribution based on ISO/IEC 9594-8 and ITU-T Recommendation X.509.

Note.— ITU-T Recommendation X.509 and ISO/IEC 9594 contain identical text and are hereafter simply referred to by the term X.509.

8.3.1.2.2 Each State participating in the ATN security scheme shall designate a single Certificate Authority (CA) that issues certificates and certificate revocation lists (CRLs).

Note 1.— The generation and control of public keys requires management, and for this purpose X.509 describes the notion of a 'trusted' authority. Such trusted authorities can be either associated with the user's organization or a third party. A Certificate Authority serves as such a trusted third party for the generation of security certificates that contain the user's public key. (The term "Certificate Authority" is used rather than the X.509 term "Certification Authority" due to the interpretation of "certification" commonly used throughout the aviation community.)

Note 2.— The state's certificate authority could be operated by the State or by a commercial CA on behalf of that State. A given CA may be the designated CA for multiple States.

8.3.1.2.3 State CAs shall have a non-transitive peer relationship among one another, rather than a hierarchical relationship.

Note 1.— The State designated CAs are the highest level of CA, i.e., there is no root CA for the ATN public key infrastructure.

Note 2.— The relationship among such CAs is expected to be established and maintained through bilateral and/or multilateral agreements, which includes, for example, provision for cross-certification..

8.3.1.2.4 The State's designated certificate authority shall issue certificates and CRLs to users within the State's domain, to Aircraft Operating Entities, other CAs within the State's domain, and to the designated CAs of other States.

Note 1.— ATN users within the State's domain include airborne and ground ATN applications and ATN routers, i.e., applications processes within end systems and network entities within intermediate systems.

Note 2.— Aircraft Operating Entities, if present, issue certificates and CRLs to airborne ATN applications and ATN routers. Each Aircraft Operating Entity may in turn designate a CA that issues certificates and CRLs

Note 3.— Other CAs within the State's domain include service providers.

8.3.1.2.5 Each State shall establish a distribution service that provides distribution of certificates and CRLs to ATN ground users within the State's domain.

Note.— Relevant certificate data may cached on a local basis.

8.3.1.2.6 CAs shall issue short-lived certificates for relying entities which do not have access to the distribution service.

Note.— Since aircraft do not have access to a distribution service ground certificates provided to them will be short-lived.

8.3.1.2.7 If a directory service is used for certificate and CRL distribution, the service shall conform to the ATN directory service as specified in Sub-Volume 7.

Note 1.— A generic distribution service is identified rather than a specific mechanism so as not to constrain States or their users and since the service needs of individual States will vary; however, it is generally recommended that the ATN directory service be used for certificate and CRL distribution.

Note 2.— State established distribution services are expected to exchange certificates and CRLs with other State distribution services.

Note 3.— The relationship among State established distribution services is expected to be established and maintained through bilateral and/or multilateral agreement.

8.3.1.2.8 CAs shall generate their own key pairs consisting of a public key and a private key.

8.3.1.2.9 A user, a user's CA, or a trusted third party designated by the user's CA shall generate user key pairs consisting of a public key and a private key.

8.3.1.2.10 The private key shall be safeguarded against unauthorized disclosure or use.

Note.— X.509 states that, "it is vital to the overall security that all private keys remain known only to the user to whom they belong." If a private key is generated by a CA or a designated third party, X.509 requires the third party to release the private key to the user in a physically secure manner and then actively destroy all information relating to the creation of the key pair plus the keys themselves.

8.3.1.2.11 Key pairs and the corresponding certificates for airborne users shall be associated with a given airframe.

Note.— Key pairs and certificates are associated with the airframe, and <u>not</u>, for example with a pilot or a particular flight identifier.

8.3.1.2.12 Key pairs and certificates for airborne users shall be valid for 28 days.

Note.— It is intended that security information not be required to be updated more frequently than normal avionics update cycles.

8.3.1.2.13 A unique key pair and the corresponding certificate for airborne users shall be shared among all ATS applications on a given airframe.

Note.— All ATS applications share the CM key pair and corresponding certificate, i.e., the key pair and certificate assigned to the CM application.

8.3.1.2.14 A unique key pair and the corresponding certificate for airborne users shall be established for each ATN router on a given airframe.

8.3.1.2.15 A key pair and corresponding certificate shall be established for each ATN ground router and each ATN ground application.

Note.— On a local basis all instances of applications of the same type (e.g. CPDLC, ADS, etc,) may share the same key pair. In this context, these applications form a CM Domain for key-usage.

8.3.1.3 Framework for Provision of Security Services within ATN Systems

Note.— If an ATN Intermediate System or End System supports ATN security services, it will support the general requirements of this section.

8.3.1.3.1 Intermediate Systems

8.3.1.3.1.1 ATN Boundary Intermediate Systems (BISs) shall support the ATN Key Agreement Scheme (AKAS), the ATN Digital Signature Scheme (ADSS) and the ATN Keyed Message Authentication Code Scheme (AMACS).

8.3.1.3.1.2 ATN Air-Ground BISs shall support peer entity authentication of Airborne BISs.

Note.— The requirement is for single entity (unilateral) authentication of airborne BISs by Air-Ground BISs. Although no requirements are defined herein for peer entity authentication of Air-Ground BISs by Airborne BISs, it is permitted as a matter of local policy.

8.3.1.3.1.3 ATN Air-Ground BISs and Ground BISs shall support mutual entity authentication with peer Ground or Air-Ground BISs.

8.3.1.3.1.4 ATN Air-Ground BISs and Ground BISs shall support data origin authentication of routing information exchanges.

Note.— Data origin authentication may be provided by an airborne BIS on a local basis provided mutual entity authentication with the peer air-ground BIS has also been performed.

8.3.1.3.1.5 ATN Air-Ground BISs and Ground BISs shall support protection of routing information exchanges from replay and manipulation.

Note.— Protection from replay and manipulation of routing information exchanges may be provided by an airborne BIS on a local basis provided mutual entity authentication with the peer air-ground BIS has also been performed.

8.3.1.3.2 End Systems

Note.— The ATN security model defines security services that may be used by applications residing in ATN end systems. Such end systems would support the security provisions of the upper layer communication services defined in Sub-Volume IV for air-ground applications defined in Sub-Volume II or for end systems supporting the ground-ground applications and associated communication services defined in Sub-Volume III. The ATN security services may also be used for the management of end systems as defined in Sub-Volume VI.

8.3.1.3.2.1 ATN end systems shall support the ATN Key Agreement Scheme (AKAS), the ATN Digital Signature Scheme (ADSS), and the ATN Keyed Message Authentication Code Scheme (AMACS).

8.3.1.3.2.2 ATN end systems shall support mutual entity authentication with peer end systems which support ATN security services.

8.3.1.3.2.3 ATN end systems shall support data origin authentication of application information exchanges.

8.3.1.3.2.4 ATN end systems shall support protection of application information exchanges from replay and manipulation attacks.

8.3.1.4 Framework for Provision of Security Services within the Upper Layer Communications Service (ULCS)

Note.— If the ULCS (see Sub-Volume IV) in an ATN End System supports ATN security services, it will support the requirements of this section.

8.3.1.4.1 The ULCS shall support the request of the dialogue service security types as defined in Table 8.3-1 by the dialogue service user.

Table 8.3-1. Dialogue Service Security Types

Dialogue Security Type	Dialogue Abstract Value	Description
1	No Security	No authentication is provided for application user data exchanges over the dialogue.
2	Secured Dialogue Supporting Key Management	Exchange of key agreement data is provided in dialogue establishment. Authentication is provided for dialogue establishment and for all application

Dialogue Security Type	Dialogue Abstract Value	Description
		user data exchanges by user applications over the dialogue when the dialogue is maintained.
3	Secured Dialogue	Authentication is provided for dialogue establishment and for all application user data exchanges by user applications over the dialogue.
4	Reserved	Reserved for future use.

Note.— Dialogue security type 2 is used in initial dialogue establishment between CM applications, i.e., during CM Logon or Contact. Subsequent dialogues by ATS applications may then invoke dialogue security type 3, in which case the ULCS will use the key agreement data exchanged during the initial dialogue..

Note.— Security type 4 is reserved for future use to support confidentiality along with authentication.

8.3.1.4.2 The initiating ULCS shall have prior knowledge of the private keys of the calling application and the information necessary to validate the certificate path to the responding application using the SSO abstract service as specified in section 8.6.3.6..

8.3.1.4.3 The responding ULCS shall have prior knowledge of the private keys of the responding application and the information necessary to validate the certificate path to the calling application using the SSO abstract service as specified in section 8.6.3.6..

8.3.1.4.4 The ground ULCS shall support the retrieval of certificates and CRLs from a certificate distribution service using the SSO abstract service as specified in 8.6.3.7 and validation of the certificate path of retrieved certificates using the SSO abstract service as specified in section 8.6.3.6.

8.3.1.4.5 The ULCS shall support validation of certificate paths and CRLS.

8.3.1.4.6 The initiating ULCS shall support generation of digital signatures using the SSO abstract service as specified in section 8.6.3.1.

8.3.1.4.7 The responding ULCS shall support verification of digital signatures using the SSO abstract services as specified in section 8.6.3.2.

8.3.1.4.8 The initiating ULCS shall support generation and verification of tags using the SSO abstract services as specified in sections 8.6.3.3 and 8.6.3.4.

8.3.1.4.9 The responding ULCS shall support generation and verification of tags using the SSO abstract services as specified in section 8.6.3.3 and 8.6.3.4.

8.3.1.4.10 The ULCS shall support derivation of a shared session key using the SSO abstract service as specified in section 8.6.3.5.

8.3.1.5 Framework for Provision of Security Services within the Context Management (CM) application

Note.— If the CM application (see Sub-Volume II) supports the ATN security services, it will support the requirements of this section.

8.3.1.5.1 The airborne CM application shall have the ability to request the use of a secured dialogue with the abstract value "secured dialogue supporting key management" when initiating a CM-logon request.

8.3.1.5.2 The ground CM application shall have the ability to request the use of a secured dialogue with the abstract value "secured dialogue supporting key management" when initiating a CM-update request.

8.3.1.5.3 The CM application shall request a dialogue with the type of security services appropriate to the application in the given operational domain.

8.3.1.5.4 The ground CM application shall support the storage, within the ground system's data base, of the shared key derivation parameter (X) for the aircraft along with the other information contained in the CM logon request.

8.3.1.5.5 The ground CM application shall support the authenticated distribution to ground-based ATS applications of the shared key derivation parameter (X) for the aircraft.

Note 1.— How authentication is accomplished is a local matter.

Note 2.— The ground CM may additionally support the authenticated distribution of the aircraft's key agreement public key, e.g., for applications which do not have access to the certificate distribution service.

8.3.1.5.6 The airborne CM application shall support the distribution to airborne-based ATS applications of the shared key derivation parameter (X), ground application public key agreement keys, and the aircraft's private key agreement key.

8.3.1.6 Framework for Provision of Security Services within Other ATN Applications

8.3.1.6.1 Air-Ground Applications

Note.— If air-ground applications (see Sub-Volume II) support ATN security services, they will support the requirements of this section.

8.3.1.6.1.1 The CPDLC, ADS and FIS applications shall request a dialogue with the type of security services appropriate to the application in the given operational domain.

8.3.1.6.1.2 The CPDLC, ADS, and FIS applications shall support access and retrieval of the key derivation parameter (X) for an aircraft from a ground CM application entity serving the associated administrative domain (e.g., State, organization, etc.).

8.3.1.6.2 Ground-Ground Applications

Note.— If a ground-ground application (see Sub-Volume III) supports ATN security services, it will support the requirements of this section.

8.3.1.6.2.1 Ground-ground applications other than ATSMHS shall authenticate ground-ground message exchanges with the type of security services appropriate to the given operational domain.

Note.— Ground-ground applications perform authentication using a purely asymmetrical solution, i.e., using signatures only.

8.3.1.6.2.2 ATSMHS Application

8.3.1.6.2.2.1 ATSMHS applications shall support the provision of the end-to-end security services of content integrity, origin authentication, and message sequence integrity using MHS Security Class S0 with the ATN Digital Signature Scheme as specified in section 8.5.4

8.3.1.6.2.2.2 ATSMHS applications shall support validation of the originator's certificate path as specified in section 8.4.5.1.

Note.— The originator's certificate path may either be included in the message by the originator or may be retrieved by the recipient using the ATN distribution service.

8.3.1.6.2.2.3 ATSMHS applications shall identify the use of the ATN Digital Scheme using the same identifier used for ATN certificates as specified in section 8.4.3.

8.3.1.7 Framework for Provision of Security Services within ATN System Management (SM) Managers

Note.— The technical provisions for SM applications are in Sub-Volume VI.

8.3.1.7.1 SM managers shall authenticate the exchange of management information between SM managers with the type of security services appropriate to the given operational domain.

8.3.1.7.2 **Recommendation.**— *Authentication mechanisms should be employed for the exchange of management information between ATN systems managers and management agents within ATN systems.*

8.3.1.8 Framework for Provision of Security Services within ATN Directory Servers

Note.— The technical provision for a directory server are in Sub-Volume VII

8.3.1.8.1 ATN directory servers shall support authentication services appropriate to the given operational domain in order to limit access to only authorized ATN users.

8.3.1.8.2 ATN directory servers shall support authentication of information exchanged between directory servers using ATN authentication services appropriate to the given operational domain.

8.3.1.9 Framework for Provision of Security Services for Auditing of ATN Systems

8.3.1.9.1 **Recommendation.**— *ATN systems should be capable of detecting and recording authentication failures and unauthorized access attempts.*

8.3.1.9.2 **Recommendation.**— *ATN systems should be able to generate an audit record of authentication failures and unauthorized access attempts.*

8.3.1.9.3 **Recommendation.**— *ATN systems should provide the Security Audit Trail Function in accordance with ISO/IEC 10164-8.*

8.3.1.10 Framework for Provision of Security Services for Security Management of ATN Systems

Note.— If an ATN intermediate system or end system (with a SM agent) supports ATN security services, it will support the specific ATN security related services described in this section.

8.3.1.10.1 ATN end systems and intermediate systems shall provide access control to their local management information base.

8.3.1.10.2 **Recommendation.**— *ATN systems should provide the Objects and Attributes for Access Control in accordance with ISO/IEC 10164-9.*

8.3.1.11 Backward compatibility

8.3.1.11.1 In order to accommodate the evolution of the ATN, ATN end systems and intermediate systems shall be able to support backward compatibility with prior versions of peer systems that either do not support ATN security services or support prior versions of the ATN security services.

Note.— The later generation versions of the ATN security services will thus need to be able to revert to a mode compatible with prior version(s). This can be accomplished by having the later generation implementation recognize either security provisions do not exist in the peer system or that an earlier version of the security provisions are supported. This is generally enabled through the use of version numbers and/or indicators that security options are not supported.

Note.— Backward compatibility of systems providing security with systems which do not provide security weakens and in some cases eliminates the protection of the system with security.

8.3.2 ATN Physical Security Framework

8.3.2.1 Access to ATN end systems, intermediate system and critical subnetwork components/ subsystems shall be controlled consistent with the provisions of Annex 17 and ICAO Doc 8973.

8.4 ATN PUBLIC KEY INFRASTRUCTURE

Note.— This section specifies the requirements for the ATN Public Key Infrastructure (PKI). It specifies required certificate policy and certificate practice statements, the format and content of certificates and CRLs, and the requirements for validation of these certificates and CRLs.

8.4.1 Certificate Policy

8.4.1.1 CAs shall develop a Certificate Policy that defines the creation, management, and use of public key certificates that they issue.

Note.— X.509 defines a certificate policy as a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.

8.4.1.2 The Certificate Policy shall conform to the Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, IETF PKIX RFC2527.

8.4.2 Certificate Practice Statement

8.4.2.1 CAs shall publish a Certificate Practice Statement that describes the expected use of public key certificates that they issue.

Note.— Practices may include such items as initialization/certification of entities and their key pairs, certificate revocation, key backup and recovery, CA key rollover, cross-certification, etc.

8.4.2.2 The Certificate Practice Statement shall conform to the Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, IETF PKIX RFC2527.

Note.— The Certificate Policy and Certificate Practice Statements of a given State could be used by other States in establishing their trust relationships and operating policies such as cross certification.

8.4.3 ATN PKI Certificate Format

Note.— The general certificate format used for ATN PKI certificates is based on the X.509 certificate format. Constraints are placed on certain X.509 fields when used in the ATN to enable the effective operation of the ATN, and these are identified below.

8.4.3.1 Uncompressed Certificates

8.4.3.1.1 Encoding and Syntax of Uncompressed Certificates

8.4.3.1.1.1 Uncompressed ATN certificates shall be an encoding of the following syntax using Distinguished Encoding Rules (DER) for ASN.1 as specified in ISO/IEC 8825-1 (ITU-T Recommendation X.690.)

8.4.3.1.1.2 The syntax of an uncompressed ATN certificate shall be as defined in ITU-T Recommendation X.509.

8.4.3.1.2 ATN Signatures on Uncompressed Certificates

Note.— The ATN signature fields in the following subsections are specified within the context of the SIGNED parameterized type of the X.509 certificate.

8.4.3.1.2.1 ATN CAs sign certificates using the ATN Signature Primitive (see 8.5.5.2.1) which shall be indicated by assigning to **algorithm** the value **ecdsa-with-SHA1** and assigning to **parameters** the value **NULL**.

8.4.3.1.2.1.1 The OID **ecdsa-with-SHA1** shall be defined as:

ecdsa-with-SHA1 OBJECT IDENTIFIER ::= {1 2 840 10045 4 1 }

Note.— The OID ecdsa-with-SHA1 is based on resolved values defined in the American National Standards Institute standard Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ANSI X9.62) and the Standards for Efficient Cryptography Group (SECG) standard Elliptic Curve Cryptography (SEC 1).

8.4.3.1.2.2 The **ENCRYPTED-HASH** field shall contain an encoding of **ECDSA-Sig-Value** containing the ECDSA signature consisting of two integers **r** and **s** with the following syntax:

```
ECDSA-Sig-Value ::= SEQUENCE {
        r           INTEGER,
        s           INTEGER
}
```

Note.— X.509 certificates represent signatures as bit strings. The entire encoding of the ASN.1 type ***ECDSA-Sig-Value*** *is therefore the value of a bit string.*

8.4.3.1.3 To Be Signed Uncompressed Certificates

8.4.3.1.3.1 The **version** field shall indicate a version 3 certificate, i.e., contain the value 2..

8.4.3.1.3.2 The **signature** field shall repeat the information contained in 8.4.3.1.2.1 about which algorithm the CA is using to sign certificates.

8.4.3.1.3.3 Uncompressed certificates shall represent times using the universal time type **UTCTime**.

8.4.3.1.3.4 **subject** field

Note.— The entity associated with the public key stored in the subject public key field may be identified in the **subject** *field or subject naming information may be present in the* ***subjectAltName*** *extension.*

8.4.3.1.3.4.1 If the subject is a CA or an AMHS entity with a distinguished name (rather than an X.400 name), the **subject** field shall contain the distinguished name of the subject in accordance with the directory schema specified in sub-volume VII.

8.4.3.1.3.4.2 If the subject is not a CA or an AMHS entity with a distinguished name, the **subject** field shall be an empty sequence.

8.4.3.1.3.5 **subjectPublicKeyInfo** field

Note.— The **subjectPublicKeyInfo** *field contains information about the public key of the entity being certified. It contains an OID indicating the public key algorithm type, any parameters associated with the key, and the key itself.*

8.4.3.1.3.5.1 The OID in the value **algorithm** shall be **id-ecPublicKey** defined as:

id-ecPublicKey OBJECT IDENTIFIER ::= {1 2 840 10045 2 1 }

Note.— The OID id-ecPublicKey is based on resolved values defined in ANSI X9.62 and SEC 1.

8.4.3.1.3.5.2 The **parameters** field shall contain the type **Parameters** with the following syntax:

```
Parameters ::= CHOICE {
        ecParameters      NULL,
```

```
        namedCurve      CURVES.&id ({CurveNames}),
        implicitCA      NULL
}
```

*Note.— The syntax for **Parameters** is based on that specified in of ANSI X9.62 and SEC 1 with the type for **ecParameters** changed to **NULL** since only the **namedCurve** choice is used for ATN certificates.*

```
CURVES ::= CLASS {
        &id OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX   { ID &id }

CurveNames  CURVES:: = {
        {ID sect163r2} |
        {ID sect233r1} ,
        ...
}
```

8.4.3.1.3.5.2.1 If ATN user elliptic curve domain parameters are used, the following OID shall be indicated:

sect163r2 OBJECT IDENTIFIER ::= { 1 3 132 0 15 }

Note.— The OID sect163r2 is based on the resolved value defined in the SECG standard Recommended Elliptic Curve Domain Parameters (SEC 2).

8.4.3.1.3.5.2.2 If ATN CA elliptic curve domain parameters are used, the following OID shall be indicated:

sect233r1 OBJECT IDENTIFIER ::= { 1 3 132 0 27}

Note.— The OID sect233r1 is based on the resolved value defined in SEC 2.

8.4.3.1.3.5.3 The **subjectPublicKey** field shall contain an encoding of the subject's public key represented using the type **ECPoint.**

ECPoint ::= OCTET STRING

Note.— (from SEC 1) The elliptic curve public key (a value of type ECPoint that is an OCTET STRING) is mapped to a subjectPublicKey (a value encoded as type BIT STRING) as follows: The most significant bit of the value of the OCTET STRING becomes the most significant bit of the value of the BIT STRING and so on with consecutive bits until the least significant bit of the OCTET STRING becomes the least significant bit of the BIT STRING.

8.4.3.1.3.6 The **issuerUniqueIdentifier** field shall be omitted.

8.4.3.1.3.7 The **subjectUniqueIdentifier** field shall be omitted.

Note.— Additional unique identifiers for the issuer and subject of the certificate are placed in the more flexible subject alternative name and issuer alternative name extensions.

8.4.3.1.3.8 The **Extensions** field shall contain the authority key identifier extension, the key usage extension, the subject alternative name extension, and the issuer alternative name extension.

8.4.3.1.3.8.1 Authority key identifier extension

Note.— The authority key identifier extension helps identify which key the issuer used to sign the certificate. This extension is especially useful during events like CA key rollover.

8.4.3.1.3.8.1.1 The authority key identifier shall be the first extension identified by the OID **id-ce-authorityKeyIdentifier**.

8.4.3.1.3.8.1.2 The authority key identifier extension shall marked non-critical in ATN certificates.

8.4.3.1.3.8.1.3 The value of **keyIdentifier** shall be composed of a four bit type field with the value 0100 followed by the least significant 60 bits of the SHA-1 hash of the value of the **subjectPublicKey** of the certificate issuer.

8.4.3.1.3.8.1.4 The **authorityCertissuer** field shall be omitted.

8.4.3.1.3.8.1.5 The **authoritySerialNumber** field shall be omitted.

8.4.3.1.3.8.2 Key usage extension

Note.— The key usage extension indicates what the certified key is going to be used for.

8.4.3.1.3.8.2.1 The key usage extension shall be the second extension identified by the OID **id-ce-keyUsage**.

8.4.3.1.3.8.2.2 The key usage extension shall be marked non-critical in all ATN certificates.

8.4.3.1.3.8.2.3 **KeyUsage** in ATN certificates shall assert the bits for **digitalSignature**, or for **keyAgreement**, or for both **keyCertSign** and **cRLSign**.

8.4.3.1.3.8.3 Subject alternative name extension

Note.— The subject alternative name extension contains an alternative name for the subject of the certificate.

8.4.3.1.3.8.3.1 The subject alternative name extension shall be the third extension identified by the OID **id-ce-subjectAltName**.

8.4.3.1.3.8.3.2 The subject alternative name extension shall be marked non-critical in all ATN certificates.

8.4.3.1.3.8.3.3 If the subject is an ATN ATS end system other than an AMHS end system, the subject alternative name extension shall contain the entity's AP-title.

8.4.3.1.3.8.3.4 If the subject is an AMHS entity, the subject alternative name extension shall contain the AMHS entity's distinguished name or X.400 address.

8.4.3.1.3.8.3.5 If the subject is an intermediate system, the subject alternative name extension shall contain the entity's Network Entity Title (NET) defined as follows:

NET ::= OCTET STRING (SIZE (20))

8.4.3.1.3.8.3.6 When the subject alternative name extension contains an entity's AP-title, this shall be placed in the extension as the value encoded as a **registeredID**.

8.4.3.1.3.8.3.7 When the subject alternative name extension contains an AMHS X.400 distinguished name, this shall be encoded as the value of **directoryName**.

8.4.3.1.3.8.3.8 When the subject alternative name extension contains an AMHS X.400 address, this shall be encoded as the value of **x400Address**.

8.4.3.1.3.8.3.9 When the subject alternative name extension contains an entity's NET, this shall be encoded as the value of **ipAddress**.

8.4.3.1.3.8.4 Issuer alternative name extension

Note.— The issuer alternative name extension contains an alternative name for the issuer of the certificate.

8.4.3.1.3.8.4.1 The issuer alternative name extension shall be the fourth extension identified by the OID **id-ce-issuerAltName**.

8.4.3.1.3.8.4.2 The issuer alternative name extension shall be marked non-critical in all ATN certificates.

8.4.3.1.3.8.4.3 The issuer alternative name extension in ATN certificates shall contain a single alternative name (which will be issuer's AP-title).

8.4.3.1.3.8.4.4 The issuing entity's AP-title shall be placed in the extension as the value encoded as a **registeredID**.

8.4.3.2 Compressed certificates

Note.— Compressed certificates are formed by omitting fields from the full certificate which can be pre-determined by the communicating entities. The information in the omitted fields is pre-stored by the receiving entity which can recover the full certificate using the pre-stored values and the values in the compressed certificate. After recovering the full certificate from the compressed certificate, the receiving entity validates the full certificate.

8.4.3.2.1 ATN certificates transmitted over air/ground subnetworks shall be sent in a compressed form using the following syntax:

```
CompressedUserCertificate ::= SEQUENCE
{
        serialNumber        CertificateSerialNumber,
        validity            ATNValidity,
        subjectPublicKey    BIT STRING,
        subjectAltName      ATNPeerId,
        issuerAltName       ATNPeerId,
        keyUsage            KeyUsage,
        encrypted           BIT STRING,
        ...
}

ATNValidity ::= SEQUENCE
{
        notBefore     ATNSecurityDateTime,
        notAfter ATNSecurityDateTime
}

ATNSecurityDateTime ::= SEQUENCE
{
        date    Date,
        time    Time
}

Date ::= SEQUENCE
{
        year    Year,
        month   Month,
        day     Day
}

Day ::= INTEGER (1..31)
-- unit = Day, Range (1..31), resolution = 1
```

```
Month ::= INTEGER (1..12)
-- unit = Month, Range (1..12), resolution = 1

Time ::= SEQUENCE
{
        hours     Timehours,
        minutes  Timeminutes,
        seconds  Timeseconds
}

Timehours ::= INTEGER (0..23)
-- units = hour, range (0..23), resolution = 1

Timeminutes ::= INTEGER (0..59)
-- units = minutes, range (0..59, resolution = 1

Timeseconds ::= INTEGER (0..59)
-- units = seconds, range (0..59), resolution = 1

Year ::= INTEGER (1996..2095)
-- unit = Year, Range (1996..2095), resolution = 1


ATNPeerId ::= CHOICE
{
        atn-ats-es-id ATN-ats-es-id,                --- required for ATS app entities
        atn-is-idATN-is-id,
        -- required for all intermediate systems
        atn-ca-idATN-ca-id,    -- required for all ATN CAs
        atn-other-id  ATN-other-id,                 -- available for any non-ATS use
        -- AOC can put PSAPs here
        ...
}

-- ATN ATS End Systems are defined by their AP-title as defined in
-- Sub-volume IV.
ATN-ats-es-id ::= CHOICE
{
        rel-air-ap-title     RELATIVE-OID,
                -- relative to { iso(1) identified-organization(3)
                -- icao(27) atn-end-system-air(1) }
        rel-ground-ap-title     RELATIVE-OID
                -- relative to { iso(1) identified-organization(3)
                -- icao(27) atn-end-system-ground(2) }
}

-- ATN Intermediate Systems are identified by their Network Entity
-- Title, a 20-octet address.  The first 3 octets of this address are
-- fixed to decimal 470027.  The RDF is the 8th octet of the NET and
```

```
-- is fixed to the value 0.  The type below takes advantage of these
-- facts and uses only 16 octets rather than the full 20.
ATN-is-id ::= OCTET STRING (SIZE (16))


ATN-ca-id ::= RELATIVE-OID
                -- relative to { iso(1) identified-organization(3) icao(27)
                --          atn-ca(6) }
                -- Note: this is one OID sub-identifier

ATN-other-id ::= OCTET STRING
```

8.4.3.3 ATN Certificate Path

8.4.3.3.1 ATN compressed certificates shall be encoded using the basic unaligned variant of the Packed Encoding Rules (PER) for ASN.1 as specified in ISO/IEC 8825-2 using the following syntax:

```
ATNCertificates ::= SEQUENCE {
        compressedUserCertificate       CompressedUserCertificate,
        certificatePath                 ForwardCertificatePath OPTIONAL
}

ForwardCertificatePath ::= SEQUENCE OF CACertificates

CACertificates ::= SEQUENCE OF CompressedUserCertificate
```

8.4.4 ATN PKI CRL Format

Note.— The general certificate format used for ATN PKI CRLs is based on the PKIX profile of the X.509 CRL format. Additional constraints are placed on the PKIX format when it is used in the ATN to enable the effective operation of the ATN.

8.4.4.1 General Format

8.4.4.1.1 All CRLs in the ATN shall be an encoding of the following syntax using Distinguished Encoding Rules (DER) for ASN.1 as specified in ITU-T Recommendation X.690.

8.4.4.1.2 The syntax of ATN CRLs shall be as defined in ITU-T Recommendation X.509.

8.4.4.1.3 Signatures for ATN CRLs shall be as specified for ATN certificates in 8.4.3.1.

8.4.4.2 ATN To Be Signed Certificate Revocation Lists

8.4.4.2.1 The **version** field shall be present and indicate a version 2 certificate, i.e., contain the value 1.

8.4.4.2.2 The **signature** field shall repeat the information contained in **signatureAlgorithm** about which algorithm the CA is using to sign CRLs.

8.4.4.2.3 All ATN CRLs shall represent times using the universal time type **UTCTime** as specified in Section 8.4.3.2.

8.4.4.2.4 Each component of the **revokedCertificates** sequence shall contain only the certificate serial number of the revoked certificate and the **revocationDate**

8.4.4.2.4.1 The optional field **crlEntryExtensions**, which can be used to convey information such as the reason for revocation, shall be absent in ATN CRLs.

8.4.4.2.5 The optional field **crlExtensions** shall contain a single CRL extension in this field – the issuer alternative name extension.

8.4.4.2.5.1 The issuer alternative name shall contain the AP-title of the issuing CA, encoded as the certificate issuer alternative name extension as specified in 8.4.3.2.8.4.

8.4.5 ATN PKI Certificate and CRL Validation

Note.— This section specifies how ATN entities validate ATN certificates and CRLs.

8.4.5.1 Certificate Validation

Note.— When an ATN entity validates a certificate, the ATN entity performs the following checks:

8.4.5.1.1 The ATN entity shall retrieve an authentic copy of the issuing CA's public key.

Note.— The issuing CA will be either the designated State CA or a CA within the State's domain.

8.4.5.1.2 The ATN entity shall check that the certificate is an X.509 certificate with the authority key identifier, key usage, subject alternative name, and issuer alternative name extensions present.

Note 1.— The certificate should conform to general X.509 and ATN-specific syntax as specified in 8.4.3.

Note 2.— Validation checks that specified extensions are present even though they are marked non-critical.

8.4.5.1.3 The ATN entity shall check that the subject alternative name and issuer alternative name extensions each contain a single name.

8.4.5.1.4 The ATN entity shall check that the certificate is signed using the ATN Signature Primitive by examining the **signatureAlgorithm** and **signature** fields of the certificate.

8.4.5.1.5 The ATN entity shall check that the certificate is a version 3 certificate by examining the **version** field of the certificate.

8.4.5.1.6 The ATN entity shall check that the certificate was issued by the appropriate CA by examining the **issuer** field and the issuer alternative name extension.

8.4.5.1.7 The ATN entity shall check that the certificate has not been revoked if a CRL list is available to the entity, and that it is fresh by examining the **validity** field.

Note.— If a CRL list which is expected to be available to an entity (in particular, a ground entity), cannot be accessed, then the certificate is to be treated as revoked. This is to prevent attacks wherein the CRL list is blocked.

8.4.5.1.8 The ATN entity shall check that the certificate was issued to the appropriate entity by examining the **subject** field and the subject alternative name extension.

8.4.5.1.9 The ATN entity shall check that the certificate contains the appropriate elliptic curve public key algorithm and parameters by examining the **subjectPublicKeyInfo** field.

8.4.5.1.10 The ATN entity shall check that the certificate certifies the appropriate key type by examining the key usage extension.

8.4.5.1.11 The ATN entity shall check that the certificate is valid by verifying the signature contained in the **signatureValue** field using the issuing CA's public key.

8.4.5.2 CRL Validation

Note.— ATN entities check that certificates have not been revoked by performing the following checks:

8.4.5.2.1 The ATN entity shall retrieve the most recent CRL issued by the issuing CA.

8.4.5.2.2 The ATN entity shall check that the CRL is an X.509 CRL with the **nextUpdate** field present, the issuer alternative name CRL extension present

8.4.5.2.3 The ATN entity shall check that a single name is in the issuer alternative name extension.

8.4.5.2.4 The ATN entity shall check that the CRL is signed using the ATN Signature Primitive by examining the **signatureAlgorithm** and **signature** fields of the CRL.

8.4.5.2.5 The ATN entity shall check that the CRL is a version 2 CRL by examining the **version** field of the CRL.

8.4.5.2.6 The ATN entity shall check that the CRL was issued by the appropriate CA by examining the **issuer** field and the issuer alternative name extension.

8.4.5.2.7 The ATN entity shall check that the CRL is fresh by examining the **thisUpdate** field and the **nextUpdate** field.

8.4.5.2.8 The ATN entity shall check that the issuing CA's certificate is not revoked by checking that the certificate's serial number does not appear in the list of revoked certificates' serial numbers in the **revokedCertificates** field.

8.4.5.2.9 The ATN entity shall check that the CRL is valid by verifying the signature in the **signatureValue** field using the issuing CA's public key.

8.4.5.3 Additional Certificate Path Validation

Note.— In addition to the certificate and CRL validation procedures already described in this section, the following checks are necessary when validating a certificate path.

8.4.5.3.1 When validating a certificate path, ATN entities shall check that the certificate path contains at most one certificate issued by a State CA to another State CA.

Note.— Trust among States is not transitive – a certificate issued to State B's CA from State A's CA together with a certificate to State C's CA from State B's CA does not indicate that State A trusts State C.

8.5 ATN CRYPTOGRAPHIC INFRASTRUCTURE

Note.— This section specifies the ATN cryptographic algorithms which are aligned with other relevant standards to provide interoperability and to insure that non-developmental implementations will be available to ATN system developers. The principle standards are: the American National Standards Institute standards, Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography (ANSI X9.63)and Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ANSI X9.62); and, the industry standard, Standards for Efficient Cryptography Group (SECG), Elliptic Curve Cryptography (SEC 1). In addition, the algorithms are aligned with the International Organization for Standardization standards, Cryptographic techniques based on elliptic curves - Part 2: Signatures, (ISO/IEC 15946-2), and Part 3: Key establishment (ISO/IEC 15946-3). The specific elliptic curves and associated parameters are from Standards for Efficient Cryptography Group, Recommended Elliptic Curve Domain Parameters (SEC 2) and the US National Institute of Standards and Technology, Digital Signature Standard (FIPS 186-2).

8.5.1 Terms

Note.— This section defines terms used throughout 8.5.

Base point (G). A selected point of large prime order *n* on an elliptic curve..

Basis . A representation of the elements of the finite field F_{2m}. In this Sub-Volume, only polynomial basis representations are used..

Binary polynomial. A polynomial whose coefficients are in the field F_2. When adding, multiplying, or dividing two binary polynomials, the coefficient arithmetic is performed modulo 2.

Characteristic 2 finite field. A finite field containing 2^m elements, where $m \$ 1$ is an integer. In this Sub-Volume, only characteristic 2 fields containing 2^m elements with *m* prime are used.

Elliptic curve. An elliptic curve over F_{2m}. is a set of points which satisfy a certain equation specified by 2 parameters *a* and *b*, which are elements of the field F_{2m}.

Elliptic curve key pair (Q, d). Given particular elliptic curve domain parameters, an elliptic curve key pair consists of an elliptic curve public key (*Q*) and the corresponding elliptic curve private key (*d*).

Elliptic curve private key (d). Given particular elliptic curve domain parameters, an elliptic curve private key, *d*, is a statistically unique and unpredictable integer in the interval [1, *n*-1], where *n* is the prime order of the base point *G*.

Elliptic curve public key (Q). Given particular elliptic curve domain parameters, and an elliptic curve private key d, the corresponding elliptic curve public key, Q, is the elliptic curve point $Q = dG$, where G is the base point.

Elliptic curve domain parameters. Elliptic curve domain parameters are comprised of a field size 2^m, the reduction polynomial $f(x)$, an indication of the basis used, an optional SEED, two elements a, b in F_{2m}, which define an elliptic curve E over F_{2m}, a point $G=(x_G,y_G)$ of prime order in $E(F_{2m})$ and the order n of G.

Elliptic curve point. If E is an elliptic curve defined over a field F_{2m}, then an elliptic curve point P is either: a pair of field elements (x_p, y_p) (where $x_p, y_p \in F_{2m}$) such that the values $x = x_p$ and $y = y_p$ satisfy the equation defining E, or a special point O called the point at infinity. O is the identity element of the elliptic curve group.

Irreducible binary polynomial. A binary polynomial $f(x)$ is irreducible if it does not factor as a product of two or more binary polynomials, each of degree less than the degree of $f(x)$.

Order of a point. The order of a point P is the smallest positive integer n such that $nP = O$ (the point at infinity).

Point Compression. Point compression allows a point $P = (x_p, y_p)$ to be represented compactly using x_p and a single additional bit $\underline{y}_p$ derived from x_p and y_p.

Reduction polynomial. A reduction polynomial is the irreducible binary polynomial $f(x)$ of degree m that is used to determine a polynomial basis representation of F_{2m}.

Scalar multiplication. If k is a positive integer, then kP denotes the point obtained by adding together k copies of the point P. The process of computing kP from P and k is called scalar multiplication.

8.5.2 Notational Conventions

Note 1.— This section defines notation for objects and operations used throughout 8.5.

$\|\|$	denotes concatenation. Thus $X \|\| Y$ denotes the string which results when the string Y is appended to the string X.
$\lceil x \rceil$	denotes the ceiling of x, i.e., the smallest integer $\geq x$.
a	denotes a coefficient of the defining elliptic curve equation associated with a set elliptic curve domain parameters.
AHASH(Data)	denotes the established ATN hash function which computes the hash value corresponding to *Data*.
AKDF(Z, keydatalen, SharedInfo)	

denotes the ATN key derivation function which derives keying data of length *keydatalen* using the key derivation function based on *AHASH* from the shared secret value *Z* and other shared data *SharedInfo*.

AMACP(MacKey, MacData, mactaglen)
: denotes the ATN tagging primitive which computes a MAC of length *mactaglen* on the data *MacData* using *HMAC* with *AHASH* under the MAC session key *MacKey*

AMACVP(MacKey; MacData, MacTag, mactaglen)
: denotes the ATN MAC verification primitive which verifies *MacTag* of length *mactaglen* on the data *MacData* under the session key *MacKey* using the ATN tagging primitive *AMACP*.

ASP(d_{sig}, SignData)
: denotes the ATN signature generation primitive which computes a signature (*r*,*s*) on the data *SignData* using ECDSA under the elliptic curve signing private key d_{sig}.

ASVDP($d_{s,U}$, $Q_{s,V}$)
: denotes the ATN secret value derivation primitive which derives a "Diffie-Hellman" shared secret value from the private key $d_{s,U}$, and the public key $Q_{s,V}$.

AVP(SignData,(r',s'), Q_{sig})
: denotes the ATN signature verification primitive which verifies the signature (*r'*,*s'*) on the data *SignData* was generated using ECDSA under the elliptic curve signing private key corresponding to the public key Q_{sig}.

b denotes a coefficient of the defining elliptic curve equation associated with a set elliptic curve domain parameters.

CA denotes the identity of a Certificate Authority.

d denotes an elliptic curve private key.

f(x) denotes the reduction polynomial associated with a set of elliptic curve domain parameters.

G denotes an elliptic curve base point associated with a set of elliptic curve domain parameters.

m denotes the extension degree of the binary field associated with a set of elliptic curve domain parameters.

n denotes the order of the base point associated with a set of elliptic curve domain parameters.

O denotes the point at infinity on an elliptic curve. The additive identity of the elliptic curve group.

Q denotes an elliptic curve public key.

Rand denotes a random challenge.

T denotes a sextuple of elliptic curve domain parameters, *(m,f(x),a,b,G,n)*.

Time denotes a time field.

U,V denotes an ATN application or an ATN router. ATN applications are identified by an Application Process Title (AP-Title). ATN routers are identified by a Network Entity Title (NET).

X denotes a shared public value. *X* is used as a key derivation parameter during the calculation of session keys for applications

z or Z denotes a shared secret value. *z* denotes a shared secret field element and *Z* is a shared secret octet string.

Note 2.— Subscript notation is used to indicate the association of a value to a particular entity, or to indicate the use of a value. For example:

$d_{s,U}$ denotes the static elliptic curve key agreement private key owned by the entity *U*.

$d_{sig,U}$ denotes the elliptic curve signing private key owned by the entity *U*.

G_{cert} denotes the elliptic curve base point associated with the CA strength elliptic curve domain parameters T_{cert}.

G_{stan} denotes the elliptic curve base point associated with the standard strength elliptic curve domain parameters T_{stan}.

$Q_{s,U}$ denotes the static elliptic curve key agreement public key owned by the entity *U*. Since the entity *U* uses the standard strength elliptic curve domain parameters T_{stan} for key agreement, $Q_{s,U} = d_{s,U}G_{stan}$.

$Q_{sig,U}$ denotes the elliptic curve signing public key owned by the entity *U*. If the signing entity is a Certificate Authority CA, the CA strength elliptic curve domain parameters, T_{cert}, are used, and $Q_{sig,U} = d_{sig,U}G_{cert}$; otherwise, the standard strength elliptic curve domain parameters, T_{stan}, are used, and $Q_{sig,U} = d_{sig,U}G_{stan}$.

$Rand_V$ denotes a random challenge chosen by the entity *V*.

T_{stan} denotes the standard strength elliptic curve domain parameters used by ATN entities for key agreement and signing.

T_{cert} denotes the CA strength elliptic curve domain parameters used by ATN Certificate Authorities during to sign certificates and CRLs.

$Time_V$ denotes a time field generated by the entity *V*.

8.5.3 ATN Cryptographic Setting

Note.— The ATN cryptographic schemes are based on arithmetic operations on an elliptic curve over a finite field. This section defines the finite fields, elliptic curves, elliptic curve domain parameters, and constraints on the use of random values used by the ATN cryptographic schemes.

8.5.3.1 ATN finite field F_{2m}

8.5.3.1.1 ATN applications, ATN routers, and supporting CAs shall use a characteristic 2 finite field (F_{2m}) containing 2^m elements, where m is a domain parameter, as the underlying finite field for ATN cryptographic algorithms.

8.5.3.1.2 The elements of F_{2m} shall be represented by the set of binary polynomials of degree m-1, with operations performed using the reduction polynomial $f(x)$ of degree m, where $f(x)$ is a domain parameter.

8.5.3.2 ATN elliptic curves

8.5.3.2.1 ATN applications, ATN routers, and supporting CAs shall use elliptic curves, $E(F_{2m})$, defined by the equation $E: y^2+xy=x^3+ax^2+b$ over F_{2m}, where a, *and* b are domain parameters.

8.5.3.3 ATN elliptic curve point representation

8.5.3.3.1 ATN elliptic curve points shall be represented in compact form using point compression.

Note.— Point compression is mandated for bandwidth efficiency and to facilitate use of non-developmental CA products.

8.5.3.4 ATN elliptic curve domain parameters

Note.— Two sets of domain parameters are specified for ATN use. ATN User Elliptic Curve Parameters, T_{stan} , which is sect163r2 from the SECG Recommended Elliptic Curve Domain Parameters (SEC 2), is a set of standard user-strength parameters to be used for key agreement and signing and verification by ATN application processes and network entities. ATN CA Elliptic Curve Parameters, T_{Cert} ,which is sect233r1 from the SECG Recommended Elliptic Curve Domain Parameters (SEC 2), is a set of CA-strength parameters used for certificate and CRL signing by CAs and for ATN application processes and network entities to verify CA-signed certificates and CRLs.

8.5.3.4.1 ATN user elliptic curve domain parameters

Note 1.— The ATN user parameters T_{stan} are parameters over F_{2m} specified by the sextuple $T_{stan} = (m, f(x) ,a, b, G, n)$.

Note 2.— m is given in decimal form, a, b, S, G and n are given in hexadecimal form.

8.5.3.4.1.1 The extension degree *m* of the binary field F_{2m} for T_{stan} shall be:
$m = 163$

8.5.3.4.1.2 The polynomial basis representation F_{2163} for T_{stan} shall be defined by the reduction polynomial:

$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$

8.5.3.4.1.3 The curve $E: y^2 + xy = x^3 + ax^2 + b$ over F_{2m} for T_{stan} shall be defined by the coefficients *a*, *b*:

a = 00 00000000 00000000 00000000 00000000 00000001
b = 02 0A601907 B8C953CA 1481EB10 512F7874 4A3205FD

Note.— E was chosen verifiably at random from the seed:
S = 85E25BFE 5C86226C DB12016F 7553F9D0 E693A268

8.5.3.4.1.4 The base point *G* for T_{stan} shall be:

G =0303 F0EBA162 86A2D57E A0991168 D4994637 E8343E36

Note.— G is expressed as G=03||x where 03 indicates that point compression is being used.

8.5.3.4.1.5 The order *n* of *G* for T_{stan} shall be:
n = 04 00000000 00000000 000292FE 77E70C12 A4234C33

8.5.3.4.2 ATN CA elliptic curve domain parameters

Note 1.— The ATN CA parameters T_{Cert} are parameters over F_{2m} specified by the sextuple T_{Cert} = (m, f(x) ,a, b, G, n).

Note 2.— m is given in decimal form, a, b, S, G and n are given in hexadecimal form.

8.5.3.4.2.1 The extension degree *m* of the binary field F_{2m} for T_{Cert} shall be:
$m = 233$

8.5.3.4.2.2 The polynomial basis representation of F_{2233} for T_{Cert} shall defined by:
$f(x) = x^{233} + x^{74} + 1$

8.5.3.4.2.3 The curve $E: y^2 + xy = x^3 + ax^2 + b$ over F_{2m} for T_{Cert} shall be defined by the coefficients a,b:

a = 0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001
b = 0066 647EDE6C 332C7F8C 0923BB58 213B333B 20E9CE42 81FE115F 7D8F90AD

Note.— E was chosen verifiably at random from the seed:
S = 74D59FF0 7F6B413D 0EA14B34 4B20A2DB 049B50C3

8.5.3.4.2.4 The base point G for T_{Cert} shall be:

G = 0300FA C9DFCBAC 8313BB21 39F1BB75 5FEF65BC 391F8B36 F8F8EB73 71FD558B

Note.— G is expressed as G=03||x where 03 indicates that point compression is being used.

8.5.3.4.2.5 The order n of G for T_{Cert} shall be:
n=0100 00000000 00000000 00000000 0013E974 E72F8A69 22031D26 03CFE0D7

8.5.3.5 ATN random values

8.5.3.5.1 ATN random or pseudo random values shall be generated in a cryptographically secure manner.

Note 1.— Inadequate random value generation is one of the most common causes of cryptographic failure. Implementations of ATN security must therefore use caution in random and pseudo random number generation.

Note 2.— See ANSI X9.62 for example generation of random and pseudo random values.

8.5.4 ATN Key Agreement Scheme (AKAS)

Note.— ATN applications and ATN routers derive keying data using the key agreement scheme given in this section.

8.5.4.1 AKAS Prerequisites

8.5.4.1.1 Each ATN application or ATN router shall have a copy of the AKAS domain parameters, T_{stan}, specified in section 8.5.3.3.1.

8.5.4.1.2 Each ATN application or ATN router shall be bound to a static key pair associated to the AKAS domain parameters.

8.5.4.1.3 Each ATN application or ATN router shall have a copy of the peer's public key.

Note.— ATN applications will have an authentic copy of the peer application's public key. ATN routers will each have an authentic copy of the peer router's key when mutual authentication is selected, however, in the case of Air/Ground IDRP authentication when single-entity authentication is selected, only the Air/Ground Router has an authentic copy of the Airborne Router's public key.

8.5.4.2 AKAS Operation

Note 1.— An ATN application, U, communicating over an air/ground subnetwork with a peer application, V, agrees on keying data by: 1) deriving a shared secret value (Z) using the primitive ASVDP, 2) deriving a shared key derivation parameter(X) using AKDF, and 3) deriving keying data (KeyData) using AKDF.

Note 2.— An ATN router, U, communicating with a peer ATN router, V, agrees on keying data by: 1) deriving a shared secret value (Z) using the primitive ASVDP and 2) deriving keying data (KeyData) using AKDF.

8.5.4.2.1 Derive Shared Secret Value

8.5.4.2.1.1 ATN applications and routers shall invoke the ATN Secret Value Derivation primitive as specified in 8.5.4.3 to derive a shared secret value *Z* from the private key $d_{s,U}$ and the public key $Q_{s,V}$ under the parameters T_{stan}.

8.5.4.2.2 Derive Shared Key Derivation Parameter

Note.— ATN applications communicating over air/ground subnetworks derive a shared key derivation parameter (X) using the technique given in this section.

8.5.4.2.2.1 An ATN application communicating over an air/ground subnetwork shall form the shared key derivation parameter *X* as output of AHASH (8.5.7.2) applied to the concatenation of a signature *S*, and a random variable *Rand*.

8.5.4.2.3 Derive Keying Data

8.5.4.2.3.1 If the entity *U* is an ATN application communicating over an air-ground subnetwork, it shall form *SharedInfo* as the concatenation of a constant value $01_{16,}$ the shared key derivation parameter *X*, its own AP-Title *U*, and its peer's AP-Title *V*.

8.5.4.2.3.2 If the entity *U* is an ATN router, it shall form *SharedInfo* as the concatenation of its own random variable $Rand_U$ and its peer's random variable $Rand_V$.

8.5.4.2.3.3 The ATN application or ATN router shall invoke the ATN Key Derivation function (AKDF) specified in 8.5.7.1 with *Z*, *SharedInfo,* and *keydatalen = 2* octets to derive a shared key.

8.5.4.3 ATN Secret Value Derivation Primitive (ASVDP)

Note.— ATN applications and ATN routers derive shared secret values for key agreement using the technique given in this section. ASVDP follows the Elliptic Curve Diffie-Hellman primitive (ECDH) as defined in ANSI X9.63 and the SECG Elliptic Curve Cryptography (SEC 1) standard.

8.5.4.3.1 To calculate a shared secret value, the ASVDP shall:

a) accept as input a private key $d_{s,U}$ owned by *U*,

b) accept as input a public key $Q_{s,V}$ owned by *V*,

c) compute the point *P* as the elliptic curve scalar product $d_{s,U}Q_{s,V}$,

d) If the point P is equal to the point at infinity of the curve group, i.e., $P=0$, output 'invalid' and stop.

e) assign the value x_P to z, i.e., set $z=x_P$, where x_P is the *x*-coordinate of *P*,

f) convert $z \in F_{2m}$ to an octet string *Z*, and

g) output Z.

8.5.5 ATN Digital Signature Scheme (ADSS)

Note.— ATN applications, ATN routers, and CAs sign and verify data using the signature scheme given in this section. ADSS follows the Elliptic Curve Digital Signature Algorithm (ECDSA) as defined in ANSI X9.62 and the SECG Elliptic Curve Cryptography (SEC 1) standard.

8.5.5.1 ADSS Prerequisites

8.5.5.1.1 Each ATN application, ATN router, or CA shall have a copy of the appropriate ATN elliptic curve domain parameters, either T_{stan} specified in 8.5.3.3.1 or T_{cert} specified in 8.5.3.3.2.

8.5.5.1.2 Each signing ATN application, ATN router, or CA shall be bound to a signing key pair associated to the appropriate ADSS domain parameters.

8.5.5.1.3 Each verifying ATN application, ATN router, or CA shall have an authentic copy of the signer's public key.

8.5.5.2 ADSS Operation

8.5.5.2.1 ATN Signature Generation Primitive (ASP)

Note.— An ATN application, ATN router, or CA uses the primitive ASP to sign data.

8.5.5.2.1.1 To sign data, the ASP shall:

a) accept as input an octet string *SignData* of data to be signed,

b) accept as input an elliptic curve signing private key $d_{sig,U}$ owned by the signing entity, *U*, which corresponds to the appropriate domain parameters, T_{Cert} or T_{stan},

c) compute the integers *r* and *s* which comprise the signature on *SignData* as follows:

1) compute the hash value $H = AHASH(SignData)$ using the ATN hash function specified in 8.5.7.2,

2) convert *H* to an integer *e*,

3) select a random integer *k* in the interval [1,*n*-1],

4) compute the elliptic curve point $(x_1, y_1) = kG$,

5) convert the field element x_1 to an integer $\underline{x}_1$,

6) set $r = \underline{x}_1 \bmod n$,

7) If $r = 0$, continue at step 3),

8) compute $s = k^{-1}(e + d_{sig,U}r) \bmod n$,

9) If s = 0, continue at step 3), otherwise

10) output the pair of integers *r* and *s*.

8.5.5.2.2 ATN Signature Verification Primitive (AVP)

Note.— An ATN application, ATN router, or CA uses the primitive AVP to verify a purported signature on data.

8.5.5.2.2.1 To verify a signature, the AVP shall:

a) accept as input the bit string *SignData'*,

b) accept as input a pair of integers *r'* and *s'* which are the purported signature of *SignData'*,

c) accept as input an EC signing public key $Q_{sig,U}$ owned by the signing entity, *U*, which corresponds to the appropriate domain parameters, T_{Cert} or T_{stan},

d) verify the purported signature using the verification transformation as follows:

1) compute the hash value *H' = AHASH*(*SignData'*) using the ATN hash function specified in Section 8.5.7.2,

2) convert *H'* to an integer *e'*,

3) If *r'* is not an integer in the interval [1, *n*-1], then reject the signature,

4) If *s'* is not an integer in the interval [1, *n*-1], then reject the signature,

5) compute $c = (s')^{-1} \bmod n$,

6) compute $u_1 = e'c \bmod n$ and $u_2 = r'c \bmod n$,

7) compute the elliptic curve point $(x_1, y_1) = u_1G + u_2Q_{sig,U}$,

8) If $u_1G + u_2Q_{sig,U}$ is the point at infinity, then reject the signature,

9) convert the field element x_1 to an integer $\underline{x}_1$',

10) compute $v = \underline{x}_1' \bmod n$, and

11) If $r' = v$, then the output 'valid' to indicate a valid signature; otherwise, output 'invalid'.

8.5.6 ATN Keyed Message Authentication Code Scheme (AMACS)

Note.— ATN applications and ATN routers perform message authentication using the tagging transformation and the tag checking transformation given in this section. AMACS follows the Hashed Message Authentication Code HMAC scheme as specified in RFC 2104

8.5.6.1 AMACS Prerequisites

8.5.6.1.1 ATN applications shall use the AMACS with 32-bit tags, i.e., HMAC-SHA1-32.

8.5.6.1.2 ATN routers shall use the AMACS with 80-bit tags, i.e., HMAC-SHA1-80.

Note 1.— HMAC is employed using 160-bit keys. 160-bit keys provide assurance that approximately 2^{160} operations are required to achieve universal forgery, i.e., to be able to forge the tag on any message.

Note 2.— For ATN applications, 32-bit tags are employed. 32-bit tags ensure that the chances of simply guessing the tag on a single message are approximately 2^{-32}. The lifetime of the tag is that of a CM session. For ATN routers, 80-bit tags are employed. 80-bit tags ensure that the chances of guessing the tag on a single message are approximately 2^{-80}. Longer tags are used in this environment because the lifetime of the tag in the ground-ground environment will last as long as the IDRP connection is maintained (continuously in the absence of errors).

8.5.6.1.3 ATN applications and ATN routers shall establish a 160-bit session key using AKAS with the peer application or router using AKAS prior to using AMACS.

8.5.6.2 AMACS Operation

8.5.6.2.1 ATN Keyed Message Authentication Code Generation Primitive (AMACP)

Note.— Data will be tagged using the tagging transformation specified as follows:

8.5.6.2.1.1 To compute a tag on data, the AMACP shall:

a) accept as input an octet string *MacData* to be MACed,

b) accept as input *a 20-octet MacKey* to be used as the key,

c) accept as input the integer *mactaglen* which is the size in octets of *MacTag*,

d) calculate the tag $MacTag = MAC_{MacKey}(MacData)$ of length *mactaglen*, where $MAC_{MacKey}(MacData)$ denotes the computation of the tag on *MacData* under *MacKey* using the HMAC tagging transformation specified in RFC 2104 with AHASH as specified in 8.5.7.2., and

e) output the octet string *MacTag* of *mactaglen*.

8.5.6.2.2 ATN Keyed Message Authentication Code Verification Primitive (AMACVP)

Note.— The purported tag on data will be checked using the tag checking transformation specified as follows:

8.5.6.2.2.1 To verify a purported tag on data, the AMACVP shall:

a) accept as input an octet string *MacData*,

b) accept as input the purported tag for *MacData* which is an octet string *MacTag'*,

c) accept as input a 20-octet *MacKey* to be used as the key,

d) accept as input the integer *mactaglen* which is the size in octets of *MacTag*,

e) calculate the tag $MacTag = MAC_{MacKey}(MacData)$, where $MAC_{MacKey}(MacData)$ denotes the computation of the tag on *MacData* under *MacKey* using the HMAC tagging transformation specified in RFC 2104 with AHASH as specified in 8.5.7.2.

f) If *MacTag'*=*MacTag*, output 'valid',

g) If *MacTag'*...*MacTag*, output 'invalid'.

8.5.7 ATN Auxiliary Cryptographic Primitives and Functions

8.5.7.1 ATN Key Derivation Function (AKDF)

Note.— ATN applications and ATN routers derive keying data under the ATN Key Agreement Scheme (8.5.4) using the key derivation function given in this section.

8.5.7.1.1 To calculate keying data, the AKDF shall:

a) accept as input an octet string *Z* which is the shared secret value,

b) accept as input an integer *keydatalen* less than $hashlen \times (2^{32}-1)$ which is the length in octets of the keying data to be generated,

Note.— hashlen is the output of the function AHASH which is 20 octets for SHA-1.

c) accept as input an octet string *SharedInfo* which consists of some data shared by the two entities intended to share the secret value *Z*,

d) compute key data using the sequence of steps in this section,

e) initialize a 32-bit, big-endian bit string *counter* as 00000001_{16} ,

f) For i=1 to $j=\lceil keydatalen/hashlen \rceil$,

1) compute $Hash_i = AHASH(Z \,||\, counter \,||\, SharedInfo)$, where AHASH is as specified in 8.5.7.2,

2) increment *counter*,

3) increment *i*.

g) if *keydatalen/hashlen* is an integer, let $HHash_j$ denote $Hash_j$,

h) if *keydatalen/hashlen* is not an integer, let $HHash_j$ denote the (*keydatalen* - (*hashlen*×*j*)) leftmost bits of $Hash_j$,

i) set $KeyData = Hash_1||Hash_2||\ldots||Hash_{j-1}||HHash_j$, and

j) output the octet string *KeyData* of length *keydatalen*.

8.5.7.2 ATN Hash Function (AHASH)

Note.— ATN applications and ATN routers calculate hash values associated with an octet string using the cryptographic hash function given in this section.

8.5.7.2.1 To calculate hash values, the AHASH Function shall:

a) accept as input an octet string *Data* of length less than *maxhashlen* octets,

Note.— maxhashlen is 2^{61}, which is the maximum input length for SHA-1.

b) calculate the hash value *Hash* corresponding to *Data* using SHA-1 as specified in ISO/IEC 10118-3, and

c) output *hashlen* which is the length in octets of Hash.

Note.— hashlen is 20 octets for SHA-1.

d) output the octet string *Hash* of length *hashlen* octets.

8.6 ATN SYSTEM SECURITY OBJECT

8.6.1 Introduction

Note 1.— The System Security Object (SSO) provides a set of abstract services for the generation and verification of security items. It describes all the necessary steps to perform the cryptographic schemes and primitives described in 8.4.5 and 8.5.

Note 2.— The SSO is an architectural component of the Upper Layer Communication Service (ULCS). Its purpose is to isolate the security-related transformations from the protocol used to exchange security data. The SSO is used in the context of a secured association (managed by the S-ASO defined in 4.8) between two ULCS entities for security-related transformations.

Note 3.— A secured association is defined as one or more secured dialogues between a pair of application entities. For most ATN applications, the duration of the secured association is the same as the duration of the secured dialogue used for that application. For Context Management, the duration of the secured association may span across multiple secured dialogues. This property allows for the use of the same secured association data on a new Context Management dialogue when the first dialogue between a pair of Context Management peers is not maintained. The duration is no longer than the duration of a single flight (defined as a takeoff and landing). However, the actual duration is subject to local considerations (e.g., time constraints).

Note 4.— All ATS applications on an airframe share the same key-agreement key pair, which is "owned" by the Context Management application peer. The airframe's Context Management AP-title will be used as the subject in the airframe's public key-agreement key certificate. Therefore, in order to retrieve the public key-agreement key certificate for the airborne peer, the airframe's Context Management AP-title is used.

Note 5.— As with all other abstract interfaces, there is no requirement to implement the SSO in any product. ATN systems will in general be designed in such a way that it is impossible to detect (from external access) whether or not an interface corresponding to the SSO has been implemented. The only requirement is that the implementation be able to perform the set of functions described here.

Note 6.— For the purposes of this specification, the SSO maintains the following information for each pair of ground ULCS peers involved in a secured association:

a) Source Peer: the initiator of the secured association (e.g., the AP-title for ATS applications).

b) Destination Peer: the responder of the secured association (e.g., the AP-title for ATS applications).

c) Secured-Association-Signature: the ATN Appendix sent from the Source Peer to the Destination Peer in the first exchange on a Secured Dialogue Supporting Key Management. The Secured-Association-Signature applies only to communication between an airborne peer and a ground peer.

d) Random Challenge: the random 32-bit unsigned integer generated by the Destination Peer and sent to the Source Peer in reply to the first exchange from the Source Peer to the Destination Peer on a Secured Dialogue Supporting Key Management. The Random Challenge applies only to communication between an airborne peer and a ground peer.

e) Shared Secret Value: the value created using the private key-agreement-key of the local peer and the public key-agreement-key of the remote peer. Due to the properties of the underlying Cryptographic Infrastructure, only the Source Peer and Destination Peer are able to calculate this value. The Shared Secret Value applies only to communication between an airborne peer and a ground peer.

f) Shared Key Derivation Parameter: the value created by the peers that participate in the first exchange on a Secured Dialogue Supporting Key Management. The peers involved in that first exchange are not always the Source and Destination Peers involved in subsequent exchanges. The Source and Destination Peers may acquire the Shared Key Derivation Parameter by local means from one of the peers that actively participated in its calculation. The Shared Key Derivation Parameter may be made public, but its distribution is performed securely. It is used to minimize the number of air-ground exchanges in the calculation of session keys for each supported ATN application. The Shared Key Derivation Parameter applies only to communication between an airborne peer and a ground peer.

g) Session Key: the symmetric key calculated for use between an airborne peer and a ground peer. There is a unique session key for each pair of peers even though all ULCS peers on a given airframe share the same key-agreement key pair and all ULCS peers of a given application type within a Context Management Domain may share the same key-agreement key pair. The Session Key applies only to communication between an airborne peer and a ground peer.

h) Counters: two counters are maintained for each secured association. One counter counts messages sent from the Source Peer to the Destination Peer. The other counter counts messages sent from the Destination Peer to the Source Peer. These counters provide for replay protection. The counters apply only to communication between an airborne peer and a ground peer.

Note 7.— The lifetime of a secured association (including the session key) is implicitly limited by the maximum value of either of the Counters; however unconstrained integers

are used. The encoding of an unconstrained integer is specified. If a counter reaches its maximum value in an implementation, the security-association must be aborted; otherwise, replay protection is lost. A new Shared Key Derivation Parameter must be calculated before the pair of peers may enter into another secured association. With the current ATN applications, it is very unlikely that these Counters will ever reach the point where a secured association must be aborted.

Note 8.— This section is organized as follows. 8.6.2 defines the general processing requirements associated with the set of SSO abstract services. 8.6.3 describes each of the SSO functions.

8.6.2 General Processing Requirements

8.6.2.1 Encoding Rules for SSO Data

8.6.2.1.1 The SSO shall use the basic unaligned variant of the Packed Encoding Rules (PER) for ASN.1, as specified in ISO|IEC 8825-2, when encoding security-related data.

Note 1.— The SSO encodes data structures in the course of constructing security appendices. The SSO also reconstructs these data structures upon receipt of a security appendix. The reconstructed interim data is then encoded and used in the security appendix verification process.

Note 2.— When using the basic unaligned variant of PER in a security context, care must be taken to ensure that the encoding is 'canonical enough' to support security. All SSO abstract data structures are specified so as to produce a canonical encoding using the basic, unaligned variant of PER.

8.6.2.2 Error Processing Requirements

8.6.2.2.1 In the event that the SSO cannot complete any requested function, the SSO shall notify the SSO-user.

Note.— Examples of error events requiring SSO-user notification are incompatible input, required information not available, and failure of an ATN Cryptographic Infrastructure primitive. The means of notification of these errors is a local matter.

8.6.3 SSO Functions

Note 1.— The following external (i.e., visible to SSO users) functions are provided by the SSO:

a) SSO-Sign, which is used to generate an ATN Appendix containing either a Signature-appendix or a MAC-appendix for the passed user data,

b) SSO-SignCheck, which is used to verify an ATN Appendix containing either a Signature-appendix or a MAC-appendix for the passed user data,

c) SSO-ProtectSign, which is used to generate an ATN Appendix containing either a Signature-appendix or a MAC-appendix for the passed user data and to provide the ATN Appendix and user data in one abstract value,

d) SSO-ProtectSignCheck, which is used to verify an ATN Appendix containing either a Signature-appendix or a MAC-appendix that is received along with user data as one abstract value,

e) SSO-CertificateCheck, which is used to verify an X.509 certificate that is reconstructed from an ATN Compressed Certificate,

f) SSO-GetCertificatePath, which is used to retrieve an ATN Compressed Certificate that is constructed from an X.509 certificate, and

g) SSO-Stop, which is used to record security-related information when a security-related error is detected in a secured association.

Note 2.— The following functions are used only by the SSO:

a) SSO-SessionKey, which is used to generate a shared secret key between two ULCS peers,

b) SSO-ASP, which is used to generate an ATN Appendix containing a Signature-appendix,

c) SSO-AVP, which is used to verify an ATN Appendix containing a Signature-appendix,

d) SSO-AMACP, which is used to generate an ATN Appendix containing a MAC-appendix, and

e) SSO-AMACVP, which is used to verify an ATN Appendix containing a MAC-appendix.

Note 3.— Each function has an associated table of parameters. For a given function, the presence of each parameter is described by one of the following values in the parameter tables:

a) ***blank*** *not present;*

b) ***C*** *conditional upon some predicate explained in the text;*

c) ***C(=)*** *conditional upon the value of the parameter to the left being present, and equal to that value;*

d) ***M*** *mandatory;*

e) ***M(=)*** *mandatory, and equal to the value of the parameter to the left;*

f) ***U*** *user option.*

8.6.3.1 SSO-Sign function

Note 1.— The parameters of the SSO-Sign function are specified in Table 8.6-1.

Table 8.6-1

Parameter Name	*In*	*Out*
Appendix Type	*M*	
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
ATN Signature		*C*

Note 2.— The Appendix Type parameter refers to the cryptographic primitive to be applied to UserData and takes an abstract value of either 'Signature-appendix' or 'MAC-appendix'.

Note 3.— The Source Peer parameter refers to the entity that invoked this function and is an abstract value of an AP-title or PSAP .

Note 4.— The Destination Peer parameter refers to the entity that is to receive the output of this function and is an abstract value of an AP-title or PSAP .

Note 5.— The User Data parameter refers to the data that is to be protected .

Note 6.— The ATN Signature parameter is the result of the cryptographic primitive applied to User Data and is an ASN.1 type ATNAppendix. The ATN Signature will be present in the output only if the function succeeds.

8.6.3.1.1 Upon activation of the SSO-Sign function, the SSO shall:

a) build the Source Peer ATNPeerId and the Destination Peer ATNPeerId.

b) build the ATN Signature using the SSO-ASP function with the following parameters when the Appendix Type is Signature-appendix:

1) Source Peer ATNPeerId as the SSO-ASP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-ASP Destination Peer, and

3) User Data as the SSO-ASP User Data.

c) build the ATN Signature using the SSO-AMACP function with the following parameters when the Appendix Type is MAC-appendix:

1) Source Peer ATNPeerId as the SSO-AMACP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AMACP Destination Peer, and

3) User Data as the SSO-AMACP User Data.

8.6.3.2 SSO-SignCheck function

Note 1.— The parameters of the SSO-SignCheck function are specified in Table 8.6-2.

Table 8.6-2

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
Security Item	*M*	
Certificate Path	*C*	
CheckResult		*M*

Note 2.— The Source Peer parameter refers to the entity that generated the signature to be verified and is an abstract value of an AP-title or PSAP .

Note 3.— The Destination Peer parameter refers to the entity that invoked this function and is an abstract value of an AP-title or PSAP.

Note 4.— The User Data parameter refers to the data whose security item is to be verified.

Note 5.— The Security Item parameter is the security item received from the Source Peer and is an ASN.1 type ATNAppendix. It is the ATN Appendix to be verified.

Note 6.— The Certificate Path is the Source Peer's public signature-key compressed certificate path when the Security Item contains a Signature-appendix and is the Source Peer's public key-agreement-key compressed certificate path when the Security Item contains a MAC-appendix. It is included in the input when the SSO-user receives it from the Source Peer and is an ASN.1 type ATNCertificates.

Note 7.— The CheckResult parameter is the result of processing. It takes one of the following abstract values: 'Success' or 'Failure'.

8.6.3.2.1 Upon activation of the SSO-SignCheck function, the SSO shall:

a) build the Source Peer ATNPeerId and the Destination Peer ATNPeerId.

b) if one of the Source Peer and Destination Peer is an airborne Peer and the Security Item contains a Signature-appendix, verify that the Counter for exchanges from the Source Peer to the Destination Peer does not have a value greater than one.

Note.— This check ensures against replay attacks. If one of the peers is an airborne peer, only the first exchange on a secured dialogue supporting key management should be signed. All subsequent exchanges on a secured dialogue supporting key management will use a MAC-appendix. All exchanges between an airborne peer and a ground peer on a secured dialogue will use a MAC-appendix. All exchanges between two ground peers on a secured dialogue will use a Signature-appendix.

c) invoke SSO-CertificateCheck to verify the Certificate Path when the Certificate Path is present in the input.

d) when the Certificate Path is not present in the input:

1) retrieve the Source Peer's public signature-key uncompressed certificate path when the Security Item contains a Signature-appendix, or

2) retrieve the Source Peer's public key-agreement-key uncompressed certificate path when the Security Item contains a MAC-appendix, and

3) verify the Source Peer's uncompressed certificate path according to the procedure in 8.4.5.

e) verify the Security Item using the SSO-AVP function with the following parameters when the Security Item contains a Signature-appendix:

1) Source Peer ATNPeerId as the SSO-AVP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AVP Destination Peer, and

3) User Data as the SSO-AVP User Data, and

4) the Security Item as the SSO-AVP ATN Appendix.

f) verify the Security Item using the SSO-AMACVP function with the following parameters when the Security Item contains a MAC-appendix:

1) Source Peer ATNPeerId as the SSO-AMACVP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AMACVP Destination Peer, and

3) User Data as the SSO-AMACVP User Data, and

4) the Security Item as the SSO-AMACVP ATN Appendix.

g) set the CheckResult parameter as appropriate.

8.6.3.3 SSO-ProtectSign function

Note 1.— The parameters of the SSO-ProtectSign function are specified in Table 8.6-3.

Table 8.6-3

Parameter Name	*In*	*Out*
Appendix Type	*M*	
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
Security Item		*C*

Note 2.— The Appendix Type parameter refers to the cryptographic primitive to be applied to User Data and takes an abstract value of either 'Signature-appendix' or 'MAC-appendix'.

Note 3.— The Source Peer parameter refers to the entity that invoked this function and is an abstract value of an AP-title or PSAP

Note 4.— The Destination Peer parameter refers to the entity that is to receive the output of this function and is an abstract value of an AP-title or PSAP .

Note 5.— The User Data parameters refers to the data that is to be protected.

Note 6.— The Security Item parameter is the security item to be generated for sending to the Destination Peer and is an ASN.1 type ATNProtectSign. The Security Item will be present in the output only if the function succeeds.

8.6.3.3.1 Upon activation of the SSO-ProtectSign function, the SSO shall:

a) build the Source Peer ATNPeerId and the Destination Peer ATNPeerId.

b) build the ATN Appendix using the SSO-ASP function with the following parameters when the Appendix Type is Signature-appendix:

1) Source Peer ATNPeerId as the SSO-ASP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-ASP Destination Peer, and

3) User Data as the SSO-ASP User Data.

c) build the ATN Appendix using the SSO-AMACP function with the following parameters when the Appendix Type is MAC-appendix:

1) Source Peer ATNPeerId as the SSO-AMACP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AMACP Destination Peer, and

3) User Data as the SSO-AMACP User Data.

d) set the Security Item unprotected user data to User Data.

e) set the Security Item appendix to the ATN Appendix.

8.6.3.4 SSO-ProtectSignCheck function

Note 1.— The parameters of the SSO-ProtectSignCheck function are specified in Table 8.6-4.

Table 8.6-4

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
Security Item	*M*	
CheckResult		*M*

Note 2.— The Source Peer parameter refers to the entity that generated the Security Item to be verified and is an abstract value of an AP-title or PSAP.

Note 3.— The Destination Peer parameter refers to the entity that invoked this function and is an abstract value of an AP-title or PSAP.

Note 4.— The Security Item parameter is the security item to be verified and is an ASN.1 type ATNProtectSign.

Note 5.— The CheckResult parameter is the result of processing. It takes one of the following abstract values: 'Success' or 'Failure'.

8.6.3.4.1 Upon activation of the SSO-ProtectSignCheck function, the SSO shall:

a) build the Source Peer ATNPeerId and the Destination Peer ATNPeerId.

b) if one of the Source Peer and Destination Peer is an airborne Peer, verify that the Security Item contains a MAC-appendix.

Note.— This check ensures against replay attacks. If one of the peers is an airborne peer, only the first exchange on a secured dialogue supporting key management should be signed. All subsequent exchanges on a secured dialogue supporting key management will use a MAC-appendix. All exchanges between an airborne peer and a ground peer on a secured dialogue will use a MAC-appendix. All exchanges between two ground peers on a secured dialogue will use a Signature-appendix.

c) verify the Security Item using the SSO-AVP function when the Security Item contains a Signature-appendix:

1) Source Peer ATNPeerId as the SSO-AVP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AVP Destination Peer, and

3) the unprotected user data of the Security Item as the SSO-AVP User Data, and

4) the appendix of the Security Item as the SSO-AVP ATN Appendix.

d) verify the Security Item using the SSO-AMACVP function when the Security Item contains a MAC-appendix:

1) Source Peer ATNPeerId as the SSO-AMACVP Source Peer, and

2) Destination Peer ATNPeerId as the SSO-AMACVP Destination Peer, and

3) the unprotected user data of the Security Item as the SSO-AMACVP User Data, and

4) the appendix of the Security Item as the SSO-AMACVP ATN Appendix.

e) set the CheckResult parameter as appropriate.

8.6.3.5 SSO-SessionKey function

Note 1.— The parameters of the SSO-SessionKey function are specified in Table 8.6-5.

Table 8.6-5

Parameter Name	*In*	*Out*
Local Peer	*M*	
Remote Peer	*M*	

Note 2.— The Local Peer parameter refers to the entity that invoked this function and is an ASN.1 type ATNPeerId.

Note 3.— The Remote Peer parameter refers to a paired peer to the entity that invoked this function and is an ASN.1 type ATNPeerId.

8.6.3.5.1 Upon activation of the SSO-SessionKey function, the SSO shall:

a) retrieve the Remote Peer's public key-agreement-key uncompressed certificate path,

Note.— The method of retrieving the public key-agreement-key uncompressed certificate path for the Remote Peer is a local matter.

b) validate the Remote Peer's public key-agreement-key uncompressed certificate path according to the procedure of 8.4.5.

c) retrieve the Local Peer's private key-agreement-key.

d) retrieve the Remote Peer's public key-agreement-key from the Remote Peer's public key-agreement-key uncompressed certificate path.

e) compute the Shared Secret Value associated with the Local Peer and Remote Peer by invoking ASVDP with:

1) the Local Peer's private key-agreement-key as the ASVDP input parameter d_U, and

2) the Remote Peer's public key-agreement-key as the ASVDP input parameter Q_V.

8.6.3.5.2 When the output of ASVDP is the Shared Secret Value *Z* for the Local Peer and Remote Peer, the SSO shall retrieve the Shared Key Derivation Parameter for the Local Peer and Remote Peer.

Note.— The method of retrieving the Shared Key Derivation Parameter for the Local Peer and Remote Peer is a local matter.

8.6.3.5.3 When the Shared Key Derivation Parameter for the Local Peer and Remote Peer could not be retrieved, the SSO shall:

a) retrieve the Secured-Association-Signature for the Local Peer and Remote Peer.

b) retrieve the Random Challenge for the Local Peer and Remote Peer when it has been generated previously.

c) generate the Random Challenge for the Local Peer and Remote Peer when it has not been generated previously.

d) retain the Random Challenge for the Local Peer and Remote Peer when it is generated.

e) construct the Shared Key Derivation Parameter for the Local Peer and Remote Peer by invoking AHASH with the concatenation of the Secured-Association-Signature and Random Challenge as the AHASH parameter *Data*.

f) make the Shared Key Derivation Parameter available for use by other entities.

8.6.3.5.4 When the Shared Key Derivation Parameter for the Local Peer and Remote Called Peer is retrieved or successfully calculated, the SSO shall:

a) construct the Session Key Derivation Information as the concatenation of a single octet with value 01_{16}, the Shared Key Derivation Parameter for the Local Peer and the Remote Peer, the Airborne Peer, and the Ground Peer.

Note 1.— Airborne Peer refers to the air end system and Ground Peer refers to the ground end system. These can be determined by the naming conventions used for the Local Peer and Remote Peer.

b) compute the Session Key associated with the Local Peer and Remote Peer by invoking AKDF with:

1) the Shared Secret Value for the Local Peer and Remote Peer as the AKDF input parameter *Z*, and

2) the AKDF input parameter *keydatalen* set to 20 octets, and

3) Session Key Derivation Information as the AKDF input parameter *SharedInfo*.

c) initialize to zero the Counter for messages sent from the Local Peer to the Remote Peer, if not already initialized.

d) initialize to zero the Counter for messages sent from the Remote Peer to the Local Peer, if not already initialized.

Note 2.— Counters are not reset if previously initialized so as to not lose replay protection. The session key is generated for each secured dialogue. In this manner, replay protection spans all secured dialogues in a secured association.

8.6.3.5.5 ***Recommendation.*** *- The session key should be stored securely and its access restricted to its use in computing and verifying appendices only.*

8.6.3.5.6 After generating the Session Key, the SSO shall verify that the generated Session Key has not been revoked via a previous invocation of SSO-Stop (see 8.6.3.8).

8.6.3.6 SSO-CertificateCheck function

Note 1.— The parameters of the SSO-CertificateCheck function are specified in Table 8.6-6.

Table 8.6-6

Parameter Name	*In*	*Out*
Certificate Path	*M*	
CheckResult		*M*

Note 2.— The Certificate Path parameter is the compressed certificate path to be validated and is an ASN.1 type ATNCertificates.

Note 3.— The CheckResult parameter is the result of processing and indicates whether or not the uncompressed certificate path associated with the Certificate Path is valid. It takes one of the following abstract values: 'Success' or 'Failure'.

8.6.3.6.1 Upon activation of the SSO-CertificateCheck function, the SSO shall:

a) reconstruct the uncompressed certificate path from the Certificate Path.

Note.— The method for reconstructing the uncompressed certificate path from a compressed certificate path is a local implementation matter.

b) validate the uncompressed certificate path according to the procedure of 8.4.5.

c) set the CheckResult parameter appropriately.

8.6.3.7 SSO-GetCertificatePath function

Note 1.— The parameters of the SSO-GetCertificatePath function are specified in Table 8.6-7.

Table 8.6-7

Parameter Name	*In*	*Out*
Entity Identification	*M*	
Key Usage	*M*	
Certificate Path		*C*

Note 2.— The Entity Identification parameter refers to the entity whose certificate is to be retrieved and is an abstract value of an AP-title or PSAP. When the entity is an airborne ATS entity, the Context Management AP-title is used to retrieve the uncompressed certificate path.

Note 3.— The Key Usage parameter refers to the type of compressed certificate path that is desired and is an ASN.1 type KeyUsage. Key Usage will have an abstract value of either digitalSignature or keyAgreement.

Note 4.— The Certificate Path parameter is the compressed certificate path for the requested entity and is an ASN.1 type ATNCertificates. It is included in the output when the function succeeds.

8.6.3.7.1 Upon activation of the SSO-GetCertificatePath function, the SSO shall:

a) retrieve the uncompressed certificate path for the public key according to Key Usage for the specified entity from the certificate delivery service.

Note.— The method of retrieval of the uncompressed certificate path is a local matter.

b) validate the uncompressed certificate path according to the procedure of 8.4.5.

c) compress the retrieved uncompressed certificate path into the appropriate compressed certificate path using the abstract syntax specified in 8.4.3.4.

d) set the Certificate Path parameter to the compressed certificate path.

8.6.3.8 SSO-Stop function

Note 1.— The parameters of the SSO-Stop function are specified in Table 8.6-9.

Table 8.6-9

Parameter Name	*In*	*Out*
Calling Peer	*M*	

Called Peer	*M*	

Note 2.— The Calling Peer parameter refers to the entity that invoked this function and is an abstract value of an AP-title or PSAP.

Note 3.— The Called Peer parameter refers to a paired peer to the entity that invoked this function and is an abstract value of an AP-title or PSAP.

8.6.3.8.1 Upon activation of the SSO-Stop function, the SSO shall:

a) delete the following state data associated with the Calling Peer and Called Peer:

1) Shared Secret Value,

2) Shared Key Derivation Parameter,

3) both Counters,

4) Secured-Association-Signature, and

5) Random Challenge.

b) retain the Session Key associated with the Calling Peer and Called Peer as revoked at the time of invocation

Note.— Retention as revoked of the Session Key is used to prevent the replay of previous exchanges between these two peers using this Session Key since a Session Key between the two peers will be recalculated as the same value until a new Secured-Association-Signature is received from the Initiator on a Secured Dialogue Supporting Key Management.

8.6.3.9 SSO-ASP function

Note 1.— The parameters of the SSO-ASP function are specified in Table 8.6-10.

Table 8.6-10

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
ATN Appendix		*C*

Note 2.— The Source Peer parameter refers to the entity that invoked this function and is an ASN.1 type ATNPeerId.

Note 3.— The Destination Peer parameter refers to the entity that is to receive the output of this function and is an ASN.1 type ATNPeerId.

Note 4.— The User Data parameter refers to the data that is to be signed.

Note 5.— The ATN Appendix parameter is the result of the cryptographic primitive applied toUser Data and is an ASN.1 type ATNAppendix. The ATN Appendix will be present in the output only if the function succeeds.

8.6.3.9.1 Upon activation of the SSO-ASP function, the SSO shall:

a) retrieve the Source Peer's private signature-key.

Note.— Retrieval of the Source Peer's private signature-key is a local implementation matter.

b) generate a Time Field using the current system time.

c) construct the To Be Signed Data from the Source Peer, Destination Peer, Time Field generated above, and the User Data.

d) generate a Signature-appendix over the To Be Signed Data by invoking ASP with:

1) To Be Signed Data as the ASP parameter *SignData*, and

2) the Source Peer's private signature-key as the ASP parameter *d*.

8.6.3.9.2 The To Be Signed Data shall conform to the encoding of the abstract syntax SignData using the Encoding Rules for SSO Data.

8.6.3.9.3 The Time Field shall conform to the encoding of the abstract syntax ATNSecurityDateTime using the Encoding Rules for SSO Data.

8.6.3.9.4 The Signature-appendix shall conform to the encoding of the abstract syntax ECDSA-Sig-Value using the Encoding Rules for SSO-Data.

8.6.3.9.5 When the output of ASP is the valid Signature-appendix, the SSO shall:

a) omit ATNAppendix.algorithmId when the default algorithm for ASP is used.

b) set ATNAppendix.algorithmId to the OBJECT IDENTIFIER for the signature algorithm for ASP when the default algorithm for ASP is not used.

c) set ATNAppendix.validity.timeField to the Time Field generated above.

d) set ATNAppendix.value.ecdsa-Signature to the Signature-appendix.

e) retain the ATN Appendix as the Secured-Association-Signature associated with the Source Peer and Destination Peer for use later in generating the Shared Key Derivation Parameter for use with AKDF when either the Source Peer or the Destination Peer is an airborne entity.

Note.— Communication between ground peers is secured using a purely asymmetric solution and does not use AKDF later; therefore the Secured-Association-Signature is not retained when both peers are ground entities.

8.6.3.10 SSO-AVP function

Note 1.— The parameters of the SSO-AVP function are specified in Table 8.6-11.

Table 8.6-11

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
ATN Appendix	*M*	

Note 2.— The Source Peer parameter refers to the entity that invoked this function and is an ASN.1 type ATNPeerId.

Note 3.— The Destination Peer parameter refers to the entity that is to receive the output of this function and is an ASN.1 type ATNPeerId.

Note 4.— The User Data parameter refers to the data that is to be signed.

Note 5.— The ATN Appendix parameter contains the Signature-appendix that was received from the Source Peer and is to be verified. It is an ASN.1 type ATNAppendix

8.6.3.10.1 Upon activation of the SSO-AVP function, the SSO shall:

a) verify that the Time Field in the ATN Appendix is valid.

Note.— The criteria used to verify the Time Field in the ATN Appendix is a local matter.

b) reconstruct the To Be Signed Data from the Source Peer, Destination Peer, Time Field, and the User Data.

c) retrieve the Signature-appendix from the ATN Appendix.

d) retain the ATN Appendix as the Secured-Association-Signature associated with the Source Peer and Destination Peer for use later in generating the Shared Key Derivation Parameter for use with AKDF when either the Source Peer or the Destination Peer is an airborne entity.

Note.— Communication between ground peers is secured using a purely asymmetric solution and does not use AKDF later.

e) obtain the Source Peer's public signature-key.

f) verify the Signature-appendix by invoking AVP with:

1) To Be Signed Data as the AVP parameter *SignData'*, and

2) Signature-appendix as the AVP parameters *r'* and *s'*, and

3) the Source Peer's public signature-key as the AVP parameter *Q*.

8.6.3.10.2 ***Recommendation.*** *- It is recommended that the criteria for verification of the Time Field include a relationship to the transit delay associated with the Class of Communication requested for the application dialogue.*

8.6.3.10.3 The To Be Signed Data shall conform to the encoding of the abstract syntax SignData using the Encoding Rules for SSO Data.

8.6.3.10.4 The Signature-appendix shall conform to the encoding of the abstract syntax ECDSA-Sig-Value using the Encoding Rules for SSO-Data.

8.6.3.11 SSO-AMACP function

Note 1.— The parameters of the SSO-AMACP function are specified in Table 8.6-12.

Table 8.6-12

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
ATN Appendix		*C*

Note 2.— The Source Peer parameter refers to the entity that invoked this function and is an ASN.1 type ATNPeerId.

Note 3.— The Destination Peer parameter refers to the entity that is to receive the output of this function and is an ASN.1 type ATNPeerId.

Note 4.— The User Data parameter refers to the data that is to be signed.

Note 5.— The ATN Appendix parameter is the result of the cryptographic primitive applied to User Data and is an ASN.1 type ATNAppendix. The ATN Appendix will be present in the output only if the function succeeds.

Note 6.— SSO-AMACP is intended to be used between air and ground peers. Communication between ground peers is secured using a purely asymmetrical solution and this function is not needed; therefore, no distinction is made for airborne users in this function.

8.6.3.11.1 Upon activation of the SSO-AMACP function, the SSO shall:

a) retrieve the Session Key for the Source Peer and Destination Peer when it is available.

b) invoke SSO-SessionKey to calculate the Session Key shared by the two peers when it is not available.

c) retrieve the Counter for the ordered pair of Source Peer and Destination Peer.

d) increment Counter by 1.

e) retrieve the Random Challenge for the Source Peer and Destination Peer when the Counter has a value of one..

f) retrieve the Secured-Association-Signature corresponding to the Source Peer and Destination Peer when the Counter has a value of one.

g) construct the MAC Data for the AMACP from the Source Peer, Destination Peer, the Counter from above, the User Data, the Random Challenge generated above, and the Secured-Association-Signature from above when the Counter has a value of one.

h) construct the MAC Data for the AMACP from the Source Peer, Destination Peer, the Counter from above, and the User Data when the Counter has a value greater than one.

i) generate a MAC over MAC Data by invoking AMACP with:

1) MacData as the AMACP parameter *MacData*,

2) Session Key as the AMACP parameter *MacKey*, and

3) the AMACP parameter *mactaglen* set to 4 octets.

8.6.3.11.2 The MAC Data shall conform to the encoding of the abstract syntax MacData using the Encoding Rules for SSO Data.

8.6.3.11.3 When the output of AMACP is a valid MAC *MacTag*, the SSO shall:

a) omit ATNAppendix.algorithmId when the default algorithm for AMACP is used.

b) set ATNAppendix.algorithmId to the OBJECT IDENTIFIER for the algorithm for AMACP when the default algorithm for AMACP is not used.

c) set ATNAppendix.validity.random to the Random Challenge for the Source Peer and Destination Peer when the Counter has a value of one.

d) omit ATNAppendix.validity.random when the Counter has a value greater than one.

e) set ATNAppendix.value.hmac-Tag to the AMACP returned MAC *MacTag*.

8.6.3.12 SSO-AMACVP function

Note 1.— The parameters of the SSO-AMACVP function are specified in Table 8.6-13.

Table 8.6-13

Parameter Name	*In*	*Out*
Source Peer	*M*	
Destination Peer	*M*	
User Data	*M*	
ATN Appendix	*M*	

Note 2.— The Source Peer parameter refers to the entity that generated the signature to be verified and is an ASN.1 type ATNPeerId.

Note 3.— The Destination Peer parameter refers to the entity that invoked this function and is an ASN.1 type ATNPeerId.

Note 4.— The User Data parameter refers to the data whose security item is to be verified.

Note 5.— The ATN Appendix parameter contains the MAC-appendix that was received from the Source Peer and is to be verified. It is an ASN.1 type ATNAppendix.

Note 6.— SSO-AMACVP is intended to be used between air and ground peers. Ground-ground communication is secured using a purely asymmetrical solution and this function is not needed; therefore, no distinction is made for airborne users in this function.

8.6.3.12.1 Upon activation of the SSO-AMACVP function, the SSO shall:

a) retrieve the Random Challenge from the ATN Appendix and retain it for use later in generating the Shared Key Derivation Parameter for the Source Peer and Destination Peer when the Random Challenge is present in the ATN Appendix.

b) invoke the SSO-SessionKey function to calculate the Session Key shared by the two peers when it is not available.

c) retrieve the Session Key shared by the two peers.

d) retrieve the Counter for the ordered pair of Source Peer and Destination Peer.

e) increment Counter by 1.

f) retrieve the Secured-Association-Signature corresponding to the Source Peer and Destination Peer when the Counter has value one.

g) reconstruct the MAC Data for the AMACVP from the Source Peer, Destination Peer, Counter, the User Data, the Random Challenge , and the Secured-Association-Signature from above when the Counter has value one.

h) reconstruct the MAC Data for the AMACVP from the Source Peer, Destination Peer, Counter, and the User Data when the Counter has value greater than one.

i) retrieve the received MAC from the ATN Appendix.

j) verify that the received message authentication code by invoking AMACVP with:

1) MAC Data as the AMACVP parameter *MacData*, and

2) the received MAC as the AMACVP parameter *MacTag'*,

3) Session Key as the AMACVP parameter *MacKey*, and

4) the AMACVP parameter *mactaglen* set to 4 octets.

8.6.3.12.2 The MAC Data shall conform to the encoding of the abstract syntax MacData using the Encoding Rules for SSO Data.

8.7 ATN SECURITY ASN.1 MODULE

Note.— This chapter contains a description of the information unique to ATN certificates and the information internally used by the SSO.

```
ATN-PKI { iso(1) identified-organization(3) icao(27)
          atn-security-requirements(5) modules(1) atnPKI(3) }

DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS ALL --

IMPORTS

        Certificate, CertificateList, CertificateSerialNumber FROM
        AuthenticationFramework { joint-iso-ccitt ds(5) module(1)
                                  authenticationFramework(7) 3 }

        KeyUsag FROM
        CertificateExtensions { joint-iso-ccitt ds(5) module(1)
                                certificateExtensions(26) 0 }

        secids, securityExchanges FROM
        ATNObjectIdentifiers { iso(1) identified-organization(3)
                                icao(27) atn(0) objectIdentifiers(0) }

        ATNAppendix FROM
        ATNSecurityExchanges securityExchanges

        ;


--
-- Compressed Certificates
--

ATNCertificates ::= SEQUENCE
{
        compressedUserCertificate    CompressedUserCertificate,
        certificatePath                 ForwardCertificatePath OPTIONAL
}

CompressedUserCertificate ::= SEQUENCE
{
        serialNumber          CertificateSerialNumber,
        validity              ATNValidity,
        subjectPublicKeyBIT STRING,
        subjectAltName   ATNPeerId,
```

```
        issuerAltName       ATNPeerId,
        keyUsage            KeyUsage,
        encrypted           BIT STRING,
        ...
}

ForwardCertificatePath ::= SEQUENCE OF CACertificates

CACertificates ::= SEQUENCE OF CompressedUserCertificate


--
-- Entity Identifications
--

ATNPeerId ::= CHOICE
{
        atn-ats-es-id ATN-ats-es-id,
        -- required for ATS app entities
        atn-is-idATN-is-id,
        -- required for all intermediate systems
        atn-ca-idATN-ca-id,    -- required for all ATN CAs
        atn-other-id  ATN-other-id,
        -- available for any non-ATS use
        -- AOC can put PSAPs here
        ...
}

-- ATN ATS End Systems are defined by their AP-title as defined in
-- Sub-volume IV.
ATN-ats-es-id ::= CHOICE
{
        rel-air-ap-title     RELATIVE-OID,
                        -- relative to { iso(1) identified-organization(3)
                        -- icao(27) atn-end-system-air(1) }
        rel-ground-ap-title     RELATIVE-OID
                        -- relative to { iso(1) identified-organization(3)
                        -- icao(27) atn-end-system-ground(2) }
}

-- ATN Intermediate Systems are identified by their Network Entity
-- Title, a 20-octet address.  The first 3 octets of this address are
-- fixed to decimal 470027.  The RDF is the 8th octet of the NET and
-- is fixed to the value 0.  The type below takes advantage of these
-- facts and uses only 16 octets rather than the full 20.
ATN-is-id ::= OCTET STRING (SIZE (16))


ATN-ca-id ::= RELATIVE-OID
                -- relative to { iso(1) identified-organization(3) icao(27)
```

```
                --          atn-ca(6) }
                -- Note: this is one OID sub-identifier

ATN-other-id ::= OCTET STRING

--
-- Supporting Types
--

ATNValidity ::= SEQUENCE
{
        notBefore     ATNSecurityDateTime,
        notAfter ATNSecurityDateTime
}

ATNSecurityDateTime ::= SEQUENCE
{
        date    Date,
        time    Time
}

Date ::= SEQUENCE
{
        year    Year,
        month   Month,
        day     Day
}

Day ::= INTEGER (1..31)
-- unit = Day, Range (1..31), resolution = 1

Month ::= INTEGER (1..12)
-- unit = Month, Range (1..12), resolution = 1

Time ::= SEQUENCE
{
        hours   Timehours,
        minutes Timeminutes,
        seconds Timeseconds
}

Timehours ::= INTEGER (0..23)
-- units = hour, range (0..23), resolution = 1

Timeminutes ::= INTEGER (0..59)
-- units = minutes, range (0..59, resolution = 1

Timeseconds ::= INTEGER (0..59)
-- units = seconds, range (0..59), resolution = 1

Year ::= INTEGER (1996..2095)
```

```
-- unit = Year, Range (1996..2095), resolution = 1

NET ::= OCTET STRING (SIZE (20))

--
-- From ANSI X9.62
--

ECDSA-Sig-Value ::= SEQUENCE
{
        r       INTEGER,
        s       INTEGER
}

Parameters ::= CHOICE {
        ecParameters    NULL,
        namedCurve  CURVES.&id ({CurveNames}),
        implicitCA    NULL
}

ECPoint ::= OCTET STRING

ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { 1 2 840 10045 4 1 }

id-ecPublicKey OBJECT IDENTIFIER ::= { 1 2 840 10045 2 1 }


--
-- From SEC2
--

sect163r2 OBJECT IDENTIFIER ::= { 1 3 132 0 15 }

sect233r1 OBJECT IDENTIFIER ::= { 1 3 132 0 27 }


--
-- Table of valid ATN curves
--

CurveNames CURVES ::= {
        { ID sect163r2 } |
        { ID sect233r1 },
        ...
}

CURVES ::= CLASS {
        &id     OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX { ID &id }
```

```
--
-- SSO Data Types
--

MacData ::= SEQUENCE
{
        sourcePeerId    ATNPeerId,
        destPeerId      ATNPeerId,
        counter         INTEGER (0..MAX),
        userData        OCTET STRING,
        random      INTEGER (0..4294967295) OPTIONAL,
                        -- 32-bit unsigned integer
        atnSignature    ATNAppendix OPTIONAL
}

SignData ::= SEQUENCE
{
        sourcePeerId    ATNPeerId,
        destPeerId      ATNPeerId,
        timeField       ATNSecurityDateTime,
        userData        OCTET STRING
}


END
```