XML Common Biometric Format

Committee Specification 1.1, June 2003

Document identifier:
{*Committee Specification*}-{*XML Common Biometric Format*}-{*XCBF*}-{*1.1*} (PDF, Word)

Location:
http://www.oasis-open.org/committees/xcbf

Edited by:
John Larmouth, Individual Member

Contributors:
Tyky Aichelen (Chair), IBM
Ed Day, Objective Systems
Dr. Paul Gérôme, Individual Member
Phillip H. Griffin, Individual Member
John Larmouth, Individual Member
Monica Martin, Sun Microsystems
Bancroft Scott, OSS Nokalva
Paul Thorpe, OSS Nokalva
Alessandro Triglia, OSS Nokalva
Rick Randall, Booz Allen Hamilton
John Messing, American Bar Association
Clifford Thompson, Individual Member
John Aerts, LA County Information Systems Advisory Body
Michael Nguyen, The Infocomm Development Authority of Singapore

Abstract:
Biometrics are automated methods of recognizing a person based on physiological or behavioral characteristics. They are used to recognize the identity of an individual, or to verify a claimed identity. This specification defines a common set of secure XML encodings for the patron formats specified in CBEFF, the Common Biometric Exchange File Format (NISTIR 6529) **[17]**. These XML encodings are based on the ASN.1 schema defined in ANSI X9.84 Biometric Information Management and Security **[14]**. For security purposes, they make use of the Canonical XML Encoding Rules (CXER) for ASN.1 defined in ITU-T Rec. X.693, and rely on the security and processing requirements specified in the X9.96 XML Cryptographic Message Syntax (XCMS) **[15]** and X9.73 Cryptographic Message Syntax (CMS) **[13]** standards .

NOTE – Other ASN.1 Encoding Rules are also employed, see 7.4 Encodings to be employed.

Status:
If you are on the xcbf@lists.oasis-open.org list for committee members, send comments there. If you are not on that list, subscribe to the xcbf-comment@lists.oasis-open.org list and send comments there. To subscribe, send an email message to xcbf-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of Contents

1 Introduction

Biometrics are automated methods of recognizing a person based on physiological or behavioral characteristics. They are used to recognize the identity of an individual, or to verify a claimed identity. This specification defines a common set of secure XML encodings for the patron formats specified in CBEFF, the Common Biometric Exchange File Format (NISTIR 6529). These CBEFF formats currently include the binary biometric objects and information records in two ANSI standards.

These XML encodings are based on the ASN.1 **[2] [3] [4] [5]** schema defined in ANSI X9.84:2003 *Biometric Information Management and Security*. They use, for security purposes, the Canonical XML Encoding Rules (CXER) for ASN.1 defined in ITU-T Rec. X.693 **[7]**, and rely on the same security and processing requirements specified in X9.96 *XML Cryptographic Message Syntax (XCMS)*. Values of the Biometric Information Record (BIR) defined in ANSI/INCITS 358-2002 - *Information technology - BioAPI Specification* **[16]** that can be represented in the X9.84 biometric object format can also be represented using XML markup and secured using the techniques in this standard.

This standard defines cryptographic messages represented in XML markup for the secure collection, distribution, and processing, of biometric information. These messages provide the means of achieving data integrity, authentication of origin, and privacy of biometric data in XML based systems and applications. Mechanisms and techniques are described for the secure transmission, storage, and integrity and privacy protection of biometric data.

2 Terminology

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in **[18]**.

99 # 3 Acronyms and Abbreviations

100

Term	Definition
ANSI	American National Standards Institute
ASN.1	Abstract Syntax Notation One
BASIC-XER	Basic XML Encoding Rules for ASN.1
BER	Basic Encoding Rules for ASN.1
BioAPI	Biometric Application Programming Interface
BIR	Biometric Information Record
CBC	Cipher Block Chaining
CBEFF	Common Biometric Exchange File Format
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CXER	Canonical XML Encoding Rules
DER	Distinguished Encoding Rules
DES	Digital Encryption Algorithm
DSA	Digital Signature Algorithm
HMAC	Hashed Message Authentication Code
IBIA	International Biometrics Industry Association
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
SHA	Secure Hash Algorithm
TDES	Triple DES
URL	Uniform Resource Locator
UTC	Universal Coordinated Time
X9	Accredited Standards Committee X9 Financial Services
XCMS	XML Cryptographic Message Syntax
XER	XML Encoding Rules
XML	Extensible Markup Language

101

102

4 Glossary

Term	Definition
Attacker	Any individual who is attempting to subvert the operation of the biometric system. The intention may be either to subsequently gain illegal entry to the portal or to deny entry to legitimate users.
Biometric	A measurable biological or behavioral characteristic, which reliably distinguishes one person from another, used to recognize the identity, or verify the claimed identity, of an enrollee.
Biometrics	Biometrics are automated methods of recognizing a person based on a physiological or behavioral characteristic.
Biometric Data	The extracted information taken from the biometric sample and used either to build a reference template or to compare against a previously created reference template.
Biometric Object	A data record taken from a biometric source or a logical piece of biometric information, which may stand for either a template, or one or more samples. The header is a set of associate attributes that belong with the opaque data, and can include additional information about the purpose, quality, etc. This must be in line with the information content in X9.84 BiometricObject type.
Biometric Sample	Captured data that represents a biometric characteristic of a user of a biometric system.
Canonical Form	The complete, unambiguous and unique encoding of an abstract value obtained by the application of encoding rules that allow one and only one way to encode the abstract value.
Capture	The collection of a biometric sample from a user.
Enrollee	A person who has a biometric reference template stored in a biometric system.
Hash	A mathematical function which evenly and randomly distributes values from a large domain into a smaller range.
HMAC	A mechanism for message authentication using a cryptographic hash function and a specific key.
MAC	A cryptographic value resulting from passing a message through the message authentication algorithm using a specific key.
Octet	A sequence of binary digits of length eight that can be represented as two hexadecimal digits, the first hexadecimal digit representing the four most significant bits of the octet, and the second hexadecimal digit representing the four least significant bits.
Octet String	A sequence of octets.
Private Key	A key of an entity's key pair known only to that entity.
Public Key	A key of an entity's key pair known publicly.

Template	Reference data formed from the biometric measurement of an enrollee and used by a biometric system for comparison against subsequently submitted biometric samples.

103

5 X9.84 and BioAPI 1.1 Interoperability

This standard defines a set of cryptographic messages represented in XML markup that can be used for the secure collection, distribution, and processing, of biometric information. All of the cryptographic operations provided in this standard are applied to a set of values of the ASN.1 type **BiometricObject** defined in the ANSI X9.84 standard.

This document describes the process for translating between an X9.84 **BiometricObject** and a BioAPI-1.1 Biometric Information Record (BIR). The X9.84 schema is the same as the schema defined in this standard and provides a common means of representing in XML markup the binary values described in the X9.84 and BioAPI-1.1 standards. Once BIR format values are represented as values of type **BiometricObject** they can be secured using the techniques described in this standard.

5.1 BiometricSyntaxSets

Type **BiometricSyntaxSets** is a series of one or more values of type **BiometricSyntax**. This type is defined as

```
BiometricSyntaxSets ::= SEQUENCE SIZE(1..MAX) OF BiometricSyntax
```

Type **BiometricSyntax** is a choice type with four choice alternatives, **biometricObjects**, **integrityObjects**, **privacyObjects** and **privacyAndIntegrityObjects**.

```
BiometricSyntax ::= CHOICE {
   biometricObjects            BiometricObjects,
   integrityObjects            IntegrityObjects,
   privacyObjects              PrivacyObjects,
   privacyAndIntegrityObjects  PrivacyAndIntegrityObjects
}
```

The choice alternatives of type **BiometricSyntax** have the following meanings:

biometricObjects	a set of unprotected biometric values
integrityObjects	a digitally signed set of biometric values
privacyObjects	an encrypted set of biometric values
privacyAndIntegrityObjects	a digitally signed and encrypted set of biometric values

Type **BiometricSyntaxSets** is a series of one or more choice alternatives. Since each of these alternatives is itself a set of one or more biometric objects, **BiometricSyntaxSets** is a set of sets. Using these choice alternatives useful collections of biometric information can be constructed. The message sender controls the order of the items in each set, so that records can be ordered for any purpose needed. This includes ordering records by likelihood of matching, by vendor format, type of biometric, data quality, or record age.

The BioAPI specification defines a single format, a BIR, composed of three fields: a record *Header*, an opaque *BiometricData* field, and an optional *Signature*. Ignoring the Signature field, the BIR format corresponds closely to the single unprotected biometric value defined in this standard as the **BiometricSyntax** choice alternative **biometricObjects** when it is constrained to contain a single **BiometricObject**. There is no definition for representing sets of biometric records in BioAPI.

The other **BiometricSyntax** choice alternatives are not supported in the BioAPI specification. These alternatives are cryptographic messages used to provide integrity, authentication and privacy services. When a BIR value is represented in **biometricObjects** format, XCBF security services can be used to protect BioAPI biometric information.

A value of type **BiometricSyntaxSets** can be represented in XML markup as

```
<BiometricSyntaxSets>
        ...
</BiometricSyntaxSets>
```

Here an ellipsis is used as a placeholder for the elements of the choice alternative of type **BiometricSyntax** which are not shown.

5.1.1 BiometricObjects

The **biometricObjects** choice alternative of type **BiometricSyntax** is a value of type **BiometricObjects**., a series of one or more values of type **BiometricObject**. These types are defined as

```
BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject

BiometricObject ::= SEQUENCE {
   biometricHeader  BiometricHeader,
   biometricData    BiometricData
}
```

All of the cryptographic processing in this standard is performed on a value of type **EncodedBiometricObjects**. This is a value of type **BiometricObjects** with the cryptographic transformations performed on the CXER encoding, as specified in 5.1.2.1.1 Digital Signature Process.

```
EncodedBiometricObjects ::= BIOMETRIC.&Type( BiometricObjects )
```

Type **BiometricObject** is composed of two components, **biometricHeader** and **biometricData**, which correspond to the BIR *Header* and *BiometricData* fields defined in the BioAPI *bioapi_bir* structure as

```
typedef struct bioapi_bir {
   BioAPI_BIR_HEADER              Header;
   BioAPI_BIR_BIOMETRIC_DATA_PTR  BiometricData;
   BioAPI_DATA_PTR                Signature;
```

```
} BioAPI_BIR, *BioAPI_BIR_PTR ;
```

The *bioapi_bir.Signature* field is optional and opaque. Since this field does not provide any standard formats, no means of identifying cryptographic algorithms and associated parameters, and no facilities for key management, it is simply ignored for the purposes of XCBF.

A value of the **biometricObjects** choice alternative of type **BiometricSyntax** can be represented in XML markup as

```
<biometricObjects>
   <BiometricObjects>
      <BiometricObject>
         <biometricHeader>
            ...
         </biometricHeader>
         <biometricData>
            ...
         </biometricData>
      </BiometricObject>
   </BiometricObjects>
</biometricObjects>
```

Here an ellipsis is used as a placeholder for the biometric header elements and data which are not shown.

5.1.1.1 BiometricHeader

The **biometricHeader** component of type **BiometricObject** is a value of type **BiometricHeader** defined as

```
BiometricHeader ::= SEQUENCE {
   version         BiometricVersion  DEFAULT hv1,
   recordType      RecordType  OPTIONAL,
   dataType        DataType  OPTIONAL,
   purpose         Purpose  OPTIONAL,
   quality         Quality  OPTIONAL,
   validityPeriod  ValidityPeriod  OPTIONAL,
   format          Format  OPTIONAL
}
```

A value of type **BiometricHeader** corresponds closely to the BIR *Header* field in the BioAPI *bioapi_bir* structure, which is defined as

```
typedef struct bioapi_bir_header {
   uint32 Length;
   BioAPI_BIR_VERSION                HeaderVersion;
   BioAPI_BIR_DATA_TYPE              Type;
   BioAPI_BIR_BIOMETRIC_DATA_FORMAT  Format;
   BioAPI_QUALITY                    Quality;
   BioAPI_BIR_PURPOSE                Purpose;
   BioAPI_BIR_AUTH_FACTORS           FactorsMask;
```

```
} BioAPI_BIR_HEADER, *BioAPI_BIR_HEADER_PTR ;
```

The **BiometricHeader** definition describes abstract values that are independent of an implementations choice of programming language, operating system, hardware or transfer representation. This approach provides applications with maximum flexibility and more than one concrete representation of the same abstract values, making it possible to encode these values in compact binary formats or as XML markup.

The **BiometricHeader** definition does not need a prefix with a length component as required by the BIR C programming language format. Some ASN.1 encoding rules will provide length fields and others will not. The **BiometricHeader** definition contains optional fields that need not be included in a record. This can reduce the record size of encoded ASN.1 values when making them more compact than the same values represented in the BioAPI BIR format.

A value of the **biometricHeader** component of type **BiometricObject** can be represented in XML markup as

```
<biometricHeader>
   <version> 0 </version>
   <recordType> <id> 6 </id> </recordType>
   <dataType> <processed/> </dataType>
   <purpose> <audit/> </purpose>
   <quality> 100 </quality>
   <validityPeriod>
      <notBefore> 1980.10.4 </notBefore>
      <notAfter> 2015.10.3.23.59.59 </notAfter>
   </validityPeriod>
   <format>
      <formatOwner>
         <oid> 2.23.42.9.10.4.2.0 </oid>
      </formatOwner>
      <formatType>
         <BlindedPrimaryAccountNumber>
            A23D552FB4490281C1F6683163D9CCB2
         </BlindedPrimaryAccountNumber>
      </formatType>
   </format>
</biometricHeader>
```

This markup specifies a high quality reference template used for audit purposes. A vendor specific payload is carried in the header.

5.1.1.1.1 BiometricVersion

The **version** component of type **BiometricHeader** is a value of type **BiometricVersion** defined as

```
BiometricVersion ::= INTEGER { hv1(0) } (0..MAX)
```

Type **BiometricVersion** specifies the integer version number of the **BiometricHeader** and has no relationship to the BIR *HeaderVersion* field in the BioAPI *bioapi_bir_header* structure.

This definition includes a constraint on the valid values of the **version** component. Values of type **BiometricVersion** are constrained to be integers greater than or equal to zero. The version number shall be zero in this standard. The biometric header version number zero is identified by the constant **hv1**.

A value of the **version** component of type **BiometricHeader** can be represented in XML markup as

```
<version> 0 </version>
```

This markup specifies the zero header version number used in this standard.

5.1.1.1.2 RecordType

The **recordType** component of type **BiometricHeader** is a value of type **RecordType** defined as

```
RecordType ::= BIOMETRIC.&name({BiometricTypes})
```

Valid values of **RecordType** are constrained by the list of objects in the **BiometricTypes** information object set. This set is defined as

```
BiometricTypes BIOMETRIC ::= {
   { BIOMETRIC id : unknown-Type       } |
   { BIOMETRIC id : body-Odor          } |
   { BIOMETRIC id : dna                } |
   { BIOMETRIC id : ear-Shape          } |
   { BIOMETRIC id : facial-Features    } |
   { BIOMETRIC id : finger-Image       } |
   { BIOMETRIC id : finger-Geometry    } |
   { BIOMETRIC id : hand-Geometry      } |
   { BIOMETRIC id : iris-Features      } |
   { BIOMETRIC id : keystroke-Dynamics } |
   { BIOMETRIC id : palm               } |
   { BIOMETRIC id : retina             } |
   { BIOMETRIC id : signature          } |
   { BIOMETRIC id : speech-Pattern     } |
   { BIOMETRIC id : thermal-Image      } |
   { BIOMETRIC id : vein-Pattern       } |
   { BIOMETRIC id : thermal-Face-Image } |
   { BIOMETRIC id : thermal-Hand-Image } |
   { BIOMETRIC id : lip-Movement       } |
   { BIOMETRIC id : gait               },

   ...  -- expect additional biometric types --
}
```

The **BiometricTypes** information object set contains an extension marker ("…") indicating that message recipients should expect additional values of biometric types not currently in the set. This allows the set to change as new biometric technology types are developed and used.

A value of this type corresponds closely to the BIR *FactorsMask* field in the BioAPI *bioapi_bir_header* structure, which is defined as

```
typedef sint8 BioAPI_BIR_AUTH_FACTORS;
```

```
#define BioAPI_FACTOR_MULTIPLE (0x00000001)
#define BioAPI_FACTOR_FACIAL_FEATURES (0x00000002)
#define BioAPI_FACTOR_VOICE (0x00000004)
#define BioAPI_FACTOR_FINGERPRINT (0x00000008)
#define BioAPI_FACTOR_IRIS (0x00000010)
#define BioAPI_FACTOR_RETINA (0x00000020)
#define BioAPI_FACTOR_HAND_GEOMETRY (0x00000040)
#define BioAPI_FACTOR_SIGNATURE_DYNAMICS (0x00000080)
#define BioAPI_FACTOR_KEYSTOKE_DYNAMICS (0x00000100)
#define BioAPI_FACTOR_LIP_MOVEMENT (0x00000200)
#define BioAPI_FACTOR_THERMAL_FACE_IMAGE (0x00000400)
#define BioAPI_FACTOR_THERMAL_HAND_IMAGE (0x00000800)
#define BioAPI_FACTOR_GAIT (0x00001000)
#define BioAPI_FACTOR_PASSWORD (0x80000000)
```

Any other unrecognized value or settings in this BIR field can be represented by an XCBF application by the **unknownType** without changes to the XCBF schema. Values that are defined in XCBF but not supported in the BioAPI specification cannot be represented in a BIR field in a standard way. These include the values defined for **body-Odor**, **dna**, **ear-Shape**, **finger-Geometry**, **palm**, and **thermal-Image**.

RecordType	Value	BioAPI FactorsMask	Value
unknownType	0	BioAPI_FACTOR_MULTIPLE	0x00000001
body-Odor	1		
dna	2		
ear-Shape	3		
facial-Features	4	BioAPI_FACTOR_FACIAL_FEATURES	0x00000002
finger-Image	5	BioAPI_FACTOR_FINGERPRINT	0x00000008
finger-Geometry	6		
hand-Geometry	7	BioAPI_FACTOR_HAND_GEOMETRY	0x00000040
iris-Features	8	BioAPI_FACTOR_IRIS	0x00000010
keystroke-Dynamics	9	BioAPI_FACTOR_KEYSTOKE_DYNAMICS	0x00000100
palm	10		
retina	11	BioAPI_FACTOR_RETINA	0x00000020
signature	12	BioAPI_FACTOR_SIGNATURE_DYNAMICS	0x00000080
speech-Pattern	13	BioAPI_FACTOR_VOICE	0x00000004
thermal-Image	14		
vein-Pattern	15		

thermal-Face-Image	16	BioAPI_FACTOR_THERMAL_FACE_IMAGE	0x00000400
thermal-Hand-Image	17	BioAPI_FACTOR_THERMAL_HAND_IMAGE	0x00000800
lip-Movement	18	BioAPI_FACTOR_LIP_MOVEMENT	0x00000200
gait	19	BioAPI_FACTOR_GAIT	0x00001000
		BioAPI_FACTOR_PASSWORD	0x80000000

The **recordType** component of type **BiometricHeader** allows the specification of a single type of biometric record. The BioAPI specification uses a bit mask and allows multiple biometric record types to be specified in the opaque biometric data. In BioAPI, the BioAPI_FACTOR_MULTIPLE bit must be set when multiple record types are specified.

BioAPI does not define a standard way to identify how each type in a multiple type BIR value is delineated, leaving these details to the biometric vendor. When these details are known to an XCBF application, multiple biometric record types may be represented as a value of type **BiometricObjects**, a series of biometric objects.

A value of the **recordType** component of type **BiometricHeader** can be represented in XML markup as

```
<recordType> <id> 9 </id> </recordType>
```

This markup specifies a keystroke dynamics record type using the relative object identifier choice alternative value.

5.1.1.1.3 DataType

The **dataType** component of type **BiometricHeader** is a value of type **DataType** defined as

```
DataType ::= ENUMERATED {
   raw          (0),
   intermediate (1),
   processed    (2)
}
```

A value of this type corresponds closely to the BIR *Type* field in the BioAPI *bioapi_bir_header* structure, which is defined as

```
typedef uint8 BioAPI_BIR_DATA_TYPE;
```

```
#define BioAPI_BIR_DATA_TYPE_RAW (0x01)
#define BioAPI_BIR_DATA_TYPE_INTERMEDIATE (0x02)
#define BioAPI_BIR_DATA_TYPE_PROCESSED (0x04)
```

The following two flags are defined in the BIR *Type* field in the BioAPI *bioapi_bir_header* structure. These are related to the *bioapi_bir.Signature* field and are ignored for the purposes of constructing a value of type **BiometricHeader**, though this information may be used by XCBF applications for determining security requirements where the details of the key management techniques allied to the opaque biometric data can be determined.

```
#define BioAPI_BIR_DATA_TYPE_ENCRYPTED (0x10)
#define BioAPI_BIR_DATA_TYPE_SIGNED (0x20)
```

X9.84 DataType	Value	BioAPI Type	Value
raw	0	BioAPI_BIR_DATA_TYPE_RAW	0x01
intermediate	1	BioAPI_BIR_DATA_TYPE_INTERMEDIATE	0x02
processed	2	BioAPI_BIR_DATA_TYPE_PROCESSED	0x04
		BioAPI_BIR_DATA_TYPE_ENCRYPTED	0x10
		BioAPI_BIR_DATA_TYPE_SIGNED	0x20

A value of the **dataType** component of type **BiometricHeader** can be represented in XML markup as

```
<dataType>  <intermediate/> </dataType>
```

This markup specifies processed biometric data using an enumerated value.

5.1.1.1.4 Purpose

The **purpose** component of type **BiometricHeader** is a value of type **Purpose** defined as

```
Purpose ::= ENUMERATED {
   verify          (1),
   identify        (2),
   enroll          (3),
   enrollVerify    (4),
   enrollIdentify  (5),
   audit           (6),

   ...  -- expect others --
}
```

A value of this type corresponds closely to the BIR *Purpose* field in the BioAPI *bioapi_bir_header* structure, which is defined as

```
typedef uint8 BioAPI_BIR_PURPOSE;
```

```
#define BioAPI_PURPOSE_VERIFY                          (1)
#define BioAPI_PURPOSE_IDENTIFY                        (2)
```



```
#define BioAPI_PURPOSE_ENROLL                          (3)
#define BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY    (4)
#define BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY  (5)
#define BioAPI_PURPOSE_AUDIT                           (6)
```

9.84 Purpose	Value	BioAPI Purpose	**Value**
verify	1	BioAPI_PURPOSE_VERIFY	1
identify	2	BioAPI_PURPOSE_IDENTIFY	2
enroll	3	BioAPI_PURPOSE_ENROLL	3
enrollVerify	4	BioAPI_PURPOSE_ENROLL_VERIFICATION_ONLY	4
enrollIdentify	5	BioAPI_PURPOSE_ENROLL_IDENTIFICATION_ONLY	5
audit	6	BioAPI_PURPOSE_AUDIT	6

A value of the **purpose** component of type **BiometricHeader** can be represented in XML markup as

```
<purpose> <audit/> </purpose>
```

This markup specifies that the purpose of the biometric data is auditing.

5.1.1.1.5 Quality

The **quality** component of type **BiometricHeader** is a value of type **Quality** defined as

```
Quality ::= INTEGER {
   lowest       (  0),
   highest      (100),
   notSet       ( -1),
   notSupported ( -2)
}
 (-2..100,...)
```

A value of this type corresponds closely to the BIR *Quality* field in the BioAPI *bioapi_bir_header* structure, which is defined as

```
typedef sint8 BioAPI_QUALITY;
```

XCBF, X9.84 and BioAPI all define biometric quality as an integer in the range of negative two to one hundred. X9.84 specifies named integer constants for the lowest quality, highest quality, quality not set, and quality not supported. These values are presented in the following table:

Value	Value Range	**Meaning of Value**

-2		Not supported by Biometric Service Provider
-1		Not set by Biometric Service Provider
	0 - 25	Unacceptable
	26 - 50	Marginal
	51 - 75	Adequate
	76 - 100	Excellent

A value of the **quality** component of type **BiometricHeader** can be represented in XML markup as

```
<quality> 100 </quality>
```

This markup specifies that the quality of the biometric data is excellent.

5.1.1.1.6 ValidityPeriod

The **validityPeriod** component of type **BiometricHeader** is a value of type **ValidityPeriod** defined as

```
ValidityPeriod ::= SEQUENCE {
   notBefore  DateTime OPTIONAL,
   notAfter   DateTime  OPTIONAL
}
 (ALL EXCEPT({ -- none; at least one component is present -- }))
```

The **notBefore** and **notAfter** components of type **ValidityPeriod** are values of type **DateTime** defined as

```
DateTime ::= RELATIVE-OID  -- { yyyy mm dd hh mm ss z }
```

These date and time values are a variable length series of integers delimited by the full stop character. No more than seven fields are allowed, and each trailing zero valued field can be omitted. Values of type **DateTime** represent a Universal Coordinated Time (UTC) value and the Zulu indicator is represented by the integer zero.

A value of the **validityPeriod** component of type **BiometricHeader** can be represented in XML markup as

```
<validityPeriod>
   <notBefore> 1980.10.4 </notBefore>
   <notAfter> 2003.10.3.23.59.59 </notAfter>
</validityPeriod>
```

This markup specifies that the biometric data is valid on or after October 4, 1980 and is not valid at midnight October 3, 2003 or thereafter.

When the optional **validityPeriod** component is present in a value of type **BiometricHeader**, either of the `<notBefore>` or `<notAfter>` elements of `<validityPeriod>` may be omitted in a valid value of type **ValidityPeriod**, but not both.

5.1.1.1.7 Format

The **format** component of type **BiometricHeader** is a value of type **Format** defined as

```
Format ::= SEQUENCE {
   formatOwner  BIOMETRIC.&name({Owner}),
   formatType   BIOMETRIC.&Type({Owner}{@formatOwner})  OPTIONAL
}
```

A value of this type corresponds closely to the BIR *Format* field in the BioAPI *bioapi_bir_biometric_data_format* structure, which defined as

```
BioAPI bioapi_bir_biometric_data_format
typedef struct bioapi_bir_biometric_data_format {
   uint16 FormatOwner;
   uint16 FormatID;
} BioAPI_BIR_BIOMETRIC_DATA_FORMAT,
*BioAPI_BIR_BIOMETRIC_DATA_FORMAT_PTR;
```

Type **Format** is composed of two components, **formatOwner** and **formatType**, which are defined in terms of the **&name** and **&Type** fields of the **BIOMETRIC** information object class. This class is defined as

```
BIOMETRIC ::= CLASS {
   &name  BIOMETRIC-IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
   WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
```

The type of the **formatOwner** component is defined in terms of the &name field. This field is defined as a value of type **BIOMETRIC-IDENTIFIER**, a choice type with two alternatives, **oid** and **id**. These alternatives allow a vendor specific format to be identified using a complete object identifier or an object identifier fragment:

```
BIOMETRIC-IDENTIFIER ::= CHOICE {
   oid  OBJECT IDENTIFIER,  -- complete object identifier
   id   RELATIVE-OID        -- object identifier fragment
}
```

The type of the optional **formatType** component is an open type, which can carry the value of any type that can be defined using ASN.1.

A value of the **format** component of type **BiometricHeader** can be represented in XML markup as

```
<format>
   <formatOwner>
      <oid> 2.23.42.9.10.4.2 </oid>
   </formatOwner>
   <formatType>
      <URL> http://asn-1.com/biolojava.htm </URL>
   </formatType>
</format>
```

This markup associates the biometric data with a specific vendor product using a complete object identifier value. The optional **formatType** component is present and contains a value of a user defined type named **URL**. Type **URL** is a Uniform Resource Locator, character string type, but given only the <URL> tag and the element contents, it is not possible to determine the actual ASN.1 schema definition of this type.

While it is easy for human readers to see that the content of the **formatType** open type is a hypertext link, application tools are likely to treat this content as an opaque string. A recipient of this information, without access to the complete ASN.1 Schema and an understanding of the intended semantics, may be able to parse this XML markup, but will not be able to understand or act on the information it provides.

Adopters of this standard can obtain an object identifier and register an associated type for use in their systems and applications. These object identifiers are globally unique and can be used to identify the version of vendor hardware and software needed to process a given biometric object.

5.1.1.1.7.1 Biometric Format Registration

There are three registration authorities for vendor specific formats recognized in this standard, NIST, IBIA and X9. Each organization controls a unique arc under which it may assign vendor specific format identifiers and associated information.

These identifiers and associated types are used to constrain the valid values that may be used in the components of type **Format**. This constraint is specified by objects defined in the **Owner** information object set defined as

```
Owner BIOMETRIC ::= {
   CBEFF-Formats  |  -- http://www.nist.gov/ --
   IBIA-Formats   |  -- http://www.ibia.org/ --
   X9-Formats,       -- http://www.x9.org/ --

   ...  -- expect additional vendor specific formats --
}
```

5.1.1.1.7.2 CBEFF-Formats

All CBEFF registered vendor specific format types are identified by the object identifier **id-x984BioInfo** or the object identifier fragment **x984BioInfo** defined as:

```
id-x984BioInfo  OID ::= { cbeff-Owner x984BioInfo(0) }

x984BioInfo  RelOID ::= { x984BioInfo(0) }  -- CBEFF owner
```

These identifier values are used in the information object sets, **CBEFFoidFormats** and **CBEFFidFormats**, to identify a value of type **BiometricInformationSets**. This type biometric serves as a placeholder for possible future standardization, which will identify commonly accepted processing algorithms and matching methods.

```
CBEFF-Formats BIOMETRIC ::= {
   CBEFFoidFormats  |  -- Complete object identifiers
   CBEFFidFormats,     -- Object identifier fragments

   ...  -- Expect additional CBEFF vendor specific formats --
}

CBEFFoidFormats BIOMETRIC ::= {
   { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },

   ...  -- Expect other objects --
}

CBEFFidFormats BIOMETRIC ::= {
   { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },

   ...  -- Expect other objects --
}
```

Type **BiometricInformationSets** is defined as one or more instances of **BiometricInformation**:

```
BiometricInformationSets ::=
      SEQUENCE SIZE(1..MAX) OF BiometricInformation

BiometricInformation ::= SEQUENCE {
   processingAlgorithms  ProcessingAlgorithms  OPTIONAL,
   matchingMethods       MatchingMethods  OPTIONAL
}
 (ALL EXCEPT({ -- none; at least one component is present -- }))
```

Type **ProcessingAlgorithms** specifies one or more biometric processing algorithms that are to be used to process biometric sample data or which have been used to create a biometric reference template. This type is defined as one or more instances of **ProcessingInformation**:

```
ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation

ProcessingInformation ::= SEQUENCE {
   id     BIOMETRIC.&name({ProcessingAIDs}),
   parms  BIOMETRIC.&Type({ProcessingAIDs}{@id})  OPTIONAL
}
```

Type **ProcessingInformation** is composed of two components, **id** and **parms**, which are defined in terms of the fields **&name** and **&Type** of the **BIOMETRIC** information object class. The valid values of these two components are constrained by the objects specified in the information object set **ProcessingAIDs**.

The **ProcessingAIDs** information object set contains no objects, as no biometric processing algorithms have been assigned by NIST under their CBEFF program.

```
ProcessingAIDs BIOMETRIC ::= {

   ...  -- Expect CBEFF assignments in BiometricInformationSets --
}
```

Type **MatchingMethods** specifies one or more biometric matching methods that can be used to associate a biometric sample to a stored reference template. This type is defined as one or more instances of **MatchingInformation**:

```
MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation

MatchingInformation ::= SEQUENCE {
   id     BIOMETRIC.&name({MatchingAIDs}),
   parms  BIOMETRIC.&Type({MatchingAIDs}{@id})  OPTIONAL
}
```

Type **MatchingInformation** is composed of two components, **id** and **parms**, which are defined in terms of the fields **&name** and **&Type** of the **BIOMETRIC** information object class. The valid values of these two components are constrained by the objects specified in the information object set **MatchingAIDs**.

The **MatchingAIDs** information object set contains no objects, as no biometric matching methods have been assigned by NIST under their CBEFF program.

```
MatchingAIDs BIOMETRIC ::= {

   ...  -- Expect CBEFF assignments in BiometricInformationSets --
}
```

5.1.1.1.7.3 IBIA-Formats

All IBIA registered vendor specific format types are identified by the object identifier

```
ibia-Owner OID ::= { format-Owner ibia(1) }
```

This base object identifier is not used in practice in BioAPI based applications, as all of the vendor specific formats registered under this arc are restricted to small, sixteen bit integers for compatibility with the fixed format requirements of the BioAPI specification. These are values of type **BirInt16** defined as

```
BirInt16 ::= INTEGER (0..65535)
```

In XCBF, the BIR format owner is modeled as a relative object identifier restricted to a single node and must comply with the fixed format requirements of the BioAPI specification.

```
ibia-SAFLINK                RelOID ::= {  1 }
ibia-Bioscrypt              RelOID ::= {  2 }
ibia-Visionics              RelOID ::= {  3 }
ibia-InfineonTechnologiesAG RelOID ::= {  4 }
ibia-IridianTechnologies    RelOID ::= {  5 }
ibia-Veridicom              RelOID ::= {  6 }
ibia-CyberSIGN              RelOID ::= {  7 }
ibia-eCryp                  RelOID ::= {  8 }
ibia-FingerprintCardsAB     RelOID ::= {  9 }
ibia-SecuGen                RelOID ::= { 10 }
ibia-PreciseBiometric       RelOID ::= { 11 }
ibia-Identix                RelOID ::= { 12 }
ibia-DERMALOG               RelOID ::= { 13 }
ibia-LOGICO                 RelOID ::= { 14 }
ibia-NIST                   RelOID ::= { 15 }
ibia-A4Vision               RelOID ::= { 16 }
ibia-NEC                    RelOID ::= { 17 }
ibia-STMicroelectronics     RelOID ::= { 18 }
ibia-Ultra-Scan             RelOID ::= { 19 }
ibia-Aurora-Wireless        RelOID ::= { 20 }
ibia-Thales                 RelOID ::= { 21 }
ibia-IBG                    RelOID ::= { 22 }
ibia-Cogent-Systems         RelOID ::= { 23 }
ibia-Cross-Match            RelOID ::= { 24 }
ibia-Recognition-Systems    RelOID ::= { 25 }
ibia-DIN                    RelOID ::= { 26 }
ibia-INCITS-M1              RelOID ::= { 27 }
```

These identifiers are associated with a restriced sixteen bit integer value.

```
IBIAidFormats BIOMETRIC ::= {
   { BIOMETRIC id : ibia-SAFLINK                DATA BirInt16 } |
   { BIOMETRIC id : ibia-Bioscrypt              DATA BirInt16 } |
   { BIOMETRIC id : ibia-Visionics              DATA BirInt16 } |
   { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
   { BIOMETRIC id : ibia-IridianTechnologies    DATA BirInt16 } |
   { BIOMETRIC id : ibia-Veridicom              DATA BirInt16 } |
   { BIOMETRIC id : ibia-CyberSIGN              DATA BirInt16 } |
   { BIOMETRIC id : ibia-eCryp                  DATA BirInt16 } |
   { BIOMETRIC id : ibia-FingerprintCardsAB     DATA BirInt16 } |
   { BIOMETRIC id : ibia-SecuGen                DATA BirInt16 } |
   { BIOMETRIC id : ibia-PreciseBiometric       DATA BirInt16 } |
   { BIOMETRIC id : ibia-Identix                DATA BirInt16 } |
   { BIOMETRIC id : ibia-DERMALOG               DATA BirInt16 } |
   { BIOMETRIC id : ibia-LOGICO                 DATA BirInt16 } |
   { BIOMETRIC id : ibia-NIST                   DATA BirInt16 } |
   { BIOMETRIC id : ibia-A4Vision               DATA BirInt16 } |
   { BIOMETRIC id : ibia-NEC                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-STMicroelectronics     DATA BirInt16 } |
   { BIOMETRIC id : ibia-Ultra-Scan             DATA BirInt16 } |
   { BIOMETRIC id : ibia-Aurora-Wireless        DATA BirInt16 } |
   { BIOMETRIC id : ibia-Thales                 DATA BirInt16 } |
```

```
   { BIOMETRIC id : ibia-IBG                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-Cogent-Systems         DATA BirInt16 } |
   { BIOMETRIC id : ibia-Cross-Match            DATA BirInt16 } |
   { BIOMETRIC id : ibia-Recognition-Systems    DATA BirInt16 } |
   { BIOMETRIC id : ibia-DIN                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-INCITS-M1              DATA BirInt16 },

   ...  -- Expect others --
}
```

Note that additional registry entries are expected and that the associated type is optional in XCBF and need not be present.

When these vendor specific format values are expressed as complete object identifiers as allowed in XCBF messages, they can be associated with any ASN.1 type needed by an implementation.

```
IBIAoidFormats BIOMETRIC ::= {
   { BIOMETRIC oid : id-ibia-SAFLINK                DATA Any } |
   { BIOMETRIC oid : id-ibia-Bioscrypt              DATA Any } |
   { BIOMETRIC oid : id-ibia-Visionics              DATA Any } |
   { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
   { BIOMETRIC oid : id-ibia-IridianTechnologies    DATA Any } |
   { BIOMETRIC oid : id-ibia-Veridicom              DATA Any } |
   { BIOMETRIC oid : id-ibia-CyberSIGN              DATA Any } |
   { BIOMETRIC oid : id-ibia-eCryp                  DATA Any } |
   { BIOMETRIC oid : id-ibia-FingerprintCardsAB     DATA Any } |
   { BIOMETRIC oid : id-ibia-SecuGen                DATA Any } |
   { BIOMETRIC oid : id-ibia-PreciseBiometric       DATA Any } |
   { BIOMETRIC oid : id-ibia-Identix                DATA Any } |
   { BIOMETRIC oid : id-ibia-DERMALOG               DATA Any } |
   { BIOMETRIC oid : id-ibia-LOGICO                 DATA Any } |
   { BIOMETRIC oid : id-ibia-NIST                   DATA Any } |
   { BIOMETRIC oid : id-ibia-A4Vision               DATA Any } |
   { BIOMETRIC oid : id-ibia-NEC                    DATA Any } |
   { BIOMETRIC oid : id-ibia-STMicroelectronics     DATA Any } |
   { BIOMETRIC oid : id-ibia-Ultra-Scan             DATA Any } |
   { BIOMETRIC oid : id-ibia-Aurora-Wireless        DATA Any } |
   { BIOMETRIC oid : id-ibia-Thales                 DATA Any } |
   { BIOMETRIC oid : id-ibia-IBG                    DATA Any } |
   { BIOMETRIC oid : id-ibia-Cogent-Systems         DATA Any } |
   { BIOMETRIC oid : id-ibia-Cross-Match            DATA Any } |
   { BIOMETRIC oid : id-ibia-Recognition-Systems    DATA Any } |
   { BIOMETRIC oid : id-ibia-DIN                    DATA Any } |
   { BIOMETRIC oid : id-ibia-INCITS-M1              DATA Any },


   ...  -- Expect additional vendor specific formats --
}

Any ::= TYPE-IDENTIFIER.&Type  -- Application constrained
```

5.1.1.1.7.4 X9-Formats

All X9 registered vendor specific format types are identified by the object identifier

```
x9-Owner OID ::= { format-Owner x9(2) }
```

Under this X9 arc, both complete and relative object identifier values can be registered for use by biometric application vendors. This base object identifier may be used to form complete object identifiers in practice. Use of this arc can occur at the application level above the BioAPI layer. For applicatons that require compatibility with BioAPI formats, the details of the fields in the BIR can be ignored and the entire BIR can be carried in a **BiometricObject** as the value of the **biometricData** component.

None of the vendor specific formats registered under the **x9-Owner** arc are restricted to the small, sixteen bit integers required for field level compatibility with the fixed format requirements of the BioAPI specification. Any type needed by the application can be registered under this arc. This capability gives biometric vendors complete control over the content that can be bound to the biometric information in a **BiometricObject**. and the flexibility needed to create biometric applications complete control and flexibility.

```
X9-Formats BIOMETRIC ::= {
   X9oidFormats |
   X9idFormats,

   ...  -- Expect additional X9 vendor specific formats --
}

X9oidFormats BIOMETRIC ::= {
   ...  -- Expect X9 assigned objects --
}

X9idFormats BIOMETRIC ::= {
   ...  -- Expect X9 assigned objects of the form { 2 x } --
}
```

5.1.1.2 BiometricData

The **biometricData** component of type **BiometricObject** is a value of type **BiometricData** defined as

```
BiometricData ::= OCTET STRING (SIZE(1..MAX))
```

A value of this type corresponds to the BIR *BiometricData* field in the BioAPI *bioapi_bir structure* and is defined as

```
typedef uint8 BioAPI_BIR_BIOMETRIC_DATA;
```

Both of these data types are opaque strings that for the purpose of transfer have no internal structure. They contain unprotected binary biometric samples aligned in 8-bit words.

5.1.2 IntegrityObjects

The **integrityObjects** choice alternative of type **BiometricSyntax** is a value of type **IntegrityObjects**. Type **IntegrityObjects** is a sequence of two components, **biometricObjects** and **integrityBlock**, and is defined as

```
IntegrityObjects ::= SEQUENCE {
   biometricObjects  EncodedBiometricObjects,
   integrityBlock    IntegrityBlock
}
```

The **biometricObjects** component is a value of type **EncodedBiometricObjects**, a series of one or more values of type **BiometricObject** in their encoded form. This is the form needed for input to digital signing and signature verification processes. Type **BiometricObject** is a sequence composed of two components, a biometric header and biometric data.

The **integrityBlock** component is a value of type **IntegrityBlock**, a choice type with four alternatives, **digitalSignature**, **messageAuthenticationCode**, **signedData** and **authenticatedData**. This type is defined as:

```
IntegrityBlock ::= CHOICE {
   digitalSignature           DigitalSignature,
   messageAuthenticationCode  MessageAuthenticationCode,
   signedData                 SignedData,
   authenticatedData          AuthenticatedData
}
```

The choice alternatives of type **IntegrityBlock** have the following meanings:

DigitalSignature	A simple digital signature using a fixed key pair
messageAuthenticationCode	A simple MAC or HMAC **[12]**
SignedData	A simple digital signature using a fixed key pair with origin authentication information
AuthenticatedData	A simple MAC or HMAC with origin authentication information

5.1.2.1 DigitalSignature

The **digitalSignature** choice alternative of the **integrityBlock** component of type **IntegrityObjects** is a value of type **DigitalSignature**. This type is a sequence of two components, an algorithm identifier and a digital signature. Type **DigitalSignature** is defined as

```
DigitalSignature ::= SEQUENCE {
   algorithmID  SignatureAlgorithmIdentifier,
   signature    OCTET STRING
                 ( CONSTRAINED BY { -- signature on a value of --
                                          EncodedBiometricObjects })
```

```
}
```

Here **EncodedBiometricObjects** is a value of type **BiometricObjects** in its encoded form. Type **BiometricObjects** is a series of one or more values of type **BiometricObject**. It is a value of type **EncodedBiometricObjects** that is digitally signed.

A value of the **digitalSignature** choice alternative of the **integrityBlock** component of type **IntegrityObjects** can be represented in XML markup as

```
<integrityBlock>
   <digitalSignature>
      <algorithmID>
         <algorithm>1.2.840.10040.4.3</algorithm>
         <parameters><NullParms/></parameters>
      </algorithmID>
      <signature>
         DE340 ... B0123DF
      </signature>
   </digitalSignature>
</integrityBlock>
```

This markup uses the **digitalSignature** choice alternative of the integrity block, a value of type **DigitalSignature**. This type provides a simple digital signature on a value of type **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) **[8]** with Secure Hash Algorithm (SHA1) **[9]** and its associated parameters, `<NullParms/>` is used for signing a value of **EncodedBiometricObjects**. An ellipsis is used as a placeholder where part of the signature is not shown.

5.1.2.1.1 Digital Signature Process

A message digest is used to create the digital signature carried in the **signature** component of **DigitalSignature**. The message digest and signature are calculated using the algorithm and parameters specified in the **algorithmID** component of **DigitalSignature**. The digest is performed on the complete CXER encoding of a value of type **BiometricObjects**.

> NOTE – This encoding is always used for the digest, whether the same encoding is used for transfer or not (see 7.4.1: Encodings used for calculation of digital signatures and MACs).

When a value of type **DigitalSignature** is represented as XML markup, the starting and ending **EncodedBiometricObjects** tags are excluded from the message digest process. Only the "value" portion of the complete CXER encoding of **EncodedBiometricObjects** is digested. The `<EncodedBiometricObject>` and `</EncodedBiometricObject>` tags are excluded from the message digest process, and the digest is calculated starting with the `<BiometricObjects>` tag and ending with the `</BiometricObjects>` tag.

The result of the message digest process is then digitally signed using the signer's private key and the signature algorithm and parameters specified in the **algorithmID** component of **DigitalSignature**. The result of the signature process is an octet string, which becomes the value of the **signature** component of **DigitalSignature**.

NOTE – The value of this octet string is encoded according to the encoding rules used for transfer (see 7.4.3 Outer-level encodings). A BASE64 encoding is not employed.

5.1.2.1.2 Digital Signature Verification

To verify the signature in a digital signature choice alternative of the **integrityBlock** component of type **IntegrityObjects**, a message digest is computed over the complete CXER encoding of the value of the **biometricObjects** component of type **IntegrityObjects** using the algorithm and any associated parameters indicated in the **algorithmID** component of **DigitalSignature**. The resulting message digest value is compared to the value of the hash obtained from applying the signature verification key to the **signature** component of type **DigitalSignature** to determine if this signature is valid.

5.1.2.2 MessageAuthenticationCode

The **messageAuthenticationCode** choice alternative of the **integrityBlock** component of type **IntegrityObjects** is a value of type **MessageAuthenticationCode**. This type is a sequence of two components, an algorithm identifier and a message authentication code (or hashed message authentication code). Type **MessageAuthenticationCode** is defined as

```
MessageAuthenticationCode ::= SEQUENCE {
   keyName      OCTET STRING  OPTIONAL,
   algorithmID  MACAlgorithmIdentifier,
   mac          OCTET STRING
                   ( CONSTRAINED BY { -- MAC or HMAC on a value of --
                                          EncodedBiometricObjects })
}
```

A **MessageAuthenticationCode** provides a way to verify the integrity of biometric information using a secret authentication key. This secret key is shared between a sender and recipient. An HMAC is a message authentication method based on a cryptographic hash function, a keyed-hash method. The cryptographic strength of an HMAC depends on the strength of the underlying hash function. For this reason, the Secure Hash Algorithm (SHA1) is widely used.

For both MAC and HMAC, cryptographic keys shall be chosen at random when created, and shall be protected and kept secret, and exchanged securely. The minimum length of the key used with HMAC depends on the choice of underlying hash function. Good security practices demand that keys be refreshed periodically to guard against weaknesses in keys and to minimize exposure from an attack.

A value of the **messageAuthenticationCode** choice alternative of the **integrityBlock** component of type **IntegrityObjects** can be represented in XML markup as

```
<integrityBlock>
   <messageAuthenticationCode>
      <keyName> 9FCD...AB45 </keyName>
      <algorithmID>
         <algorithm>1.3.6.1.5.5.8.1.2</algorithm>
      </algorithmID>
      <mac>
         DEA7B ... 59ABD3
```

```
      </mac>
   </messageAuthenticationCode>
</integrityBlock>
```

This markup uses the message authentication code choice alternative. The hashed MAC with SHA1 algorithm and a shared secret key are used to compute an HMAC on a value of **EncodedBiometricObjects**. An ellipsis is used as a placeholder for part of the HMAC results.

5.1.2.2.1 Message Authentication Process

A sender prepares a value of type **EncodedBiometricObjects**, a named cryptographic key previously created at random and known by the recipient, and uses these as input to a MAC or HMAC process. This results in a message authentication code over the specified biometric information. The biometric information and processing results are sent to a recipient who shares the secret key used in the message authentication code process.

To verify the message authentication code, the user computes a MAC or HMAC on the biometric information using the same shared secret key identified by its key name, and compares this result to the message authentication value received to determine the integrity of the biometric information.

5.1.2.3 SignedData

The **signedData** choice alternative of the **integrityBlock** component of type **IntegrityObjects** is a value of type **SignedData**. This sequence type is defined as

```
SignedData ::= SEQUENCE {
   version           CMSVersion,
   digestAlgorithms  DigestAlgorithmIdentifiers,
   encapContentInfo  EncapsulatedContentInfo,
   certificates      [0] CertificateSet  OPTIONAL,
   crls              [1] CertificateRevocationLists  OPTIONAL,
   signerInfos       SignerInfos
}
```

The components of type **SignedData** have the following meanings:

version An integer version number of the syntax definition. The version shall be 84 in this standard.

digestAlgorithms The set of message digest algorithms used by the signers. This set contains only one element since there is only one signer of the content in this standard.

encapContentInfo An identifier of the type of content signed and optionally, the signed content. In this standard the signed content is not present. The type of content is always ordinary data as the nesting of cryptographic types is neither required nor supported.

certificates An optional set of one or more X.509 [1] or X9.68 [11] certificates. When present, this set may contain more certificates or less than needed to verify the signature on the signed data. This component shall be encoded as specified in 7.4.2 Octet Strings with Certificates and Certificate

Revocation Lists.

crls An optional set of one or more X.509 certificate revocation lists (CRLs). When present, there may be more CRLs or less than needed to determine whether or not the certificate of the signer is valid. This component shall be encoded as specified in 7.4.2 Octet Strings with Certificates and Certificate Revocation Lists.

signerInfos A set of information for each signer of the content. This set contains only one element since there is only one signer of the content in this standard.

```
SignerInfos ::= SET SIZE(1) OF SignerInfo
```

```
SignerInfo ::= SEQUENCE {
   version             CMSVersion,
   sid                 SignerIdentifier,
   digestAlgorithm     DigestAlgorithmIdentifier,
   signatureAlgorithm  SignatureAlgorithmIdentifier,
   signature           SignatureValue
}
```

The **SignerIdentifier** type is used to identify the public key certificate associated with the private key used to create the **signature** component of **SignerInfo**. This type is defined in XCBF as a choice type having only one alternative:

```
SignerIdentifier ::= CHOICE {
   certHash  [1] EXPLICIT Hash
}
```

The **certHash** choice alternative of **SignerIdentifier** provides a single, simple mechanism that allows any type of digital certificate to be identified. The hash is computed over the complete encoding of the certificate. This allows any type of certificate, regardless of its format or encoding, to be identified.

When X.509 certificates or attribute certificates are used, the hash will be computed over the complete DER **[6]** encoding of the certificate, as X.509 only supports these encoding rules. When X9.68 domain certificates are used, the hash may be computed over the DER or XER encoding of the certificate, as either encoding format is supported by that standard. In any case, such details are hidden from an application if the certificates is properly treated as an opaque series of octets.

```
Hash ::= CHOICE {
   ietf       CertHash,  -- SHA-1 hash of entire certificate
   withAlgID  DigestInfo
}

CertHash ::= OCTET STRING (ENCODED BY sha-1)

DigestInfo ::= SEQUENCE {
   hashAlgorithm  DigestAlgorithmIdentifier,
   digest         OCTET STRING
}
```

5.1.2.3.1 Message digest and signature process

A message digest is used to create the digital signature carried in the **SignerInfo** component of **SignedData**. The message digest is calculated using the algorithm and parameters specified in the **digestAlgorithm** component of **SignerInfo**, and the value of the **eContent** component of **EncapsulatedContentInfo**. The **eContentType** component of **EncapsulatedContentInfo** identifies the type of value being signed. This is always the object identifier value **id-data** in XCBF.

When a value of type **SignedData** is represented as XML markup, the starting and ending **eContent** tags are excluded from the message digest process. Only the "value" portion of the complete canonical XER encoding of **eContent** is digested. The **<eContent>** and **</eContent>** tags are excluded from the message digest process, and the digest is calculated starting with the **<BiometricObjects>** tag and ending with the **</BiometricObjects>** tag.

The result of the message digest process is then digitally signed using the signer's private key and the signature algorithm and parameters specified in the **signatureAlgorithm** component of **SignerInfo**. The result of the signature process becomes the value of the **signature** component of the **SignerInfo** component of **SignedData**.

5.1.2.3.2 Signature Verification

To verify the signature in a signed data choice alternative of the **integrityBlock** component of type **IntegrityObjects**, a message digest is computed over the complete canonical XER encoding of the value of the **biometricObjects** component of type **IntegrityObjects**. This digest is computed using the **digestAlgorithm** component of type **SignerInfo**.

The public key of the signer is the signature verification key. This key is used to verify the digital signature on the signed data created with the signer's private key. The signature is carried in the **SignerInfo** component of type **SignedData**. The value of the **sid** component of **SignerInfo** identifies the public key certificate associated with the private key used to create the digital signature on the signed data.

The message digest value computed by the signature verifier is compared to the value of the hash obtained from applying the signer's public key to the **signature** component of type **SignerInfo** to determine if this signature is valid. There is only one signer of the content in this standard.

Complete trust in the validity of the signature on the signed data by the signer must be determined by validation of the chain of certificates associated with the signer's public key certificate. The optional **certificates** and **crls** components of type **SignedData** may be used by the signer to provide information needed to validate the signer's certificate.

5.1.2.4 AuthenticatedData

The **messageAuthenticationCode** choice alternative of the **integrityBlock** component of type **IntegrityObjects** is a value of type **AuthenticatedData**. This sequence type provides a MAC with key establishment and is defined as

```
AuthenticatedData ::= SEQUENCE {
   version           CMSVersion,
   recipientInfos    RecipientInfos,
   macAlgorithm      MACAlgorithmIdentifier,
   encapContentInfo  EncapsulatedContentInfo,
   mac               MessageAuthenticationCode
}
```

Type **AuthenticatedData** uses a key transport mechanism to convey a one-time MAC key along with biometric information. Since a key agreement mechanism is not supported, only data integrity is provided by use of this type and not origin authentication. There is only a single recipient of this encrypted MAC key present in the **recipientInfos** component of type **AuthenticatedData**, and the optional content is not present in the **encapContentInfo** component.

A message authentication code is calculated on biometric information. This biometric information is a value of type **EncodedBiometricObjects**, which is a value of type **BiometricObjects** in its encoded form. The calculation is performed using the MAC algorithm and any associated algorithm parameters indicated in the **macAlgorithm** component of type **AuthenticatedData**, the biometric information, and an authentication key that is conveyed in the **recipientInfos** component. The result of this calculation is a message authentication code, which becomes the value of the **mac** component.

A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented in XML markup as

```
<privacyObjects>
   <privacyBlock>

      ...

   </privacyBlock>
</privacyObjects>
```

This markup illustrates the wrapper for a typical privacy object. The optional privacy object biometric headers are not present. An ellipsis is used as a placeholder and the details of the privacy block choice alternative are not shown.

5.1.2.5 Biometric Certificate Extensions

Digital signature data integrity protection and origin authentication can be achieved by including biometric information in digital certificates. A set of one or more biometric reference templates can be cryptographically bound to the public key associated with the private key of an entity, by including a value of type **EncodedBiometricObjects** in a certificate extension. In the XCBF standard, biometric certificate extension values can be encoded in either BER or BASIC-XER depending on the outer-level encoding (see 7.4.3 Outer-level encodings).

XCBF supports the version three certificates and version two attribute certificates defined in the X.509 standard and the compact domain certificate format defined in X9.68. One extension is defined for use in each standard, and these are defined as

```
biometricTemplates EXTENSION ::= {
   SYNTAX         EncodedBiometricObjects  -- CXER --
   IDENTIFIED BY  x509-biometricTemplates
}

domainBiometricTemplates PRIVATE-X ::= {
   NAME  oid : x968-biometricTemplates
   TYPE  EncodedBiometricObjects  -- CXER --
}
```

When biometric information is included in a certificate extension stored in a certificate repository, the repository becomes a biometric storage subsystem, and the biometric information may need to be protected by encryption or other means. Measures should be taken to prevent an attacker from using a certificate repository as a large, searchable public database of biometric reference templates that could be used to find templates that match a given biometric sample. Finding such a match would allow an attacker to focus its efforts on that user.

5.1.3 PrivacyObjects

The **privacyObjects** choice alternative of type **BiometricSyntax** is a value of type **PrivacyObjects**. This type is defined as a sequence of two components, **biometricHeaders** and **privacyBlock**.

```
PrivacyObjects ::= SEQUENCE {
   biometricHeaders  BiometricHeaders  OPTIONAL,
   privacyBlock      PrivacyBlock
}
```

The **biometricHeaders** component is a series of one or more values of type **BiometricHeader**.

```
BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
```

This optional **biometricHeaders** component is not protected by encryption and should be present only when a privacy object is used in a secure environment, or when the information contained in the **biometricHeaders** component does not compromise security or assist an attacker. In a secure setting these biometric headers may be used as a convenience, to assist in searches of biometric information and in database management operations.

The encrypted content in the privacy block contains a series of one or more values of type **BiometricObject**, including their biometric headers. To be useful, the **biometricHeaders** component need only provide an indication of the information contained in the encrypted privacy block. But this component need not contain exactly the same information as the headers in the encrypted privacy block, and may contain only a single **BiometricHeader** value when present. The **biometricHeaders** component is not protected in any way.

The **privacyBlock** component of type **PrivacyObjects** offers three choice alternatives, **fixedKey**, **namedKey** and **establishedKey**.

```
PrivacyBlock ::= CHOICE {
   fixedKey        EncryptedData,
   namedKey        NamedKeyEncryptedData,
   establishedKey  EnvelopedData
}
```

The **fixedKey** and **namedKey** choice alternatives are based on the **EncryptedData** type. The **establishedKey** alternative is based on type **EnvelopedData**. Each of these alternatives has different characteristics, and the alternative chosen will depend upon application requirements and the key management scheme being used.

A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented in XML markup as

```
<privacyObjects>
   <privacyBlock>

      ...

   </privacyBlock>
</privacyObjects>
```

This markup illustrates the wrapper for a typical privacy object. The optional privacy object biometric headers are not present. An ellipsis is used as a placeholder and the details of the privacy block choice alternative are not shown.

5.1.3.1 Encrypted Content Information

All three of the privacy block choice alternatives contain a value of type **EncryptedContentInfo** defined as

```
EncryptedContentInfo ::= SEQUENCE {
   contentType                 ContentType,
   contentEncryptionAlgorithm  ContentEncryptAlgorithmIdentifier,
   encryptedContent            [0] EncryptedContent
}
```

The **contentType** component identifies the type of encrypted content. In XCBF, the type of encrypted content is always a value of **EncodedBiometricObjects**, a series of one or more values of type **BiometricObject** encoded using CXER. The type of encrypted content is identified

as ordinary data by the information object identifier value **id-data**, defined in the PKCS #7 Cryptographic Message Syntax Standard [19]. The encoding of this component is

```
<contentType> 1.2.840.113549.1.7.1 </contentType>
```

The **contentEncryptionAlgorithm** component identifies the content encryption algorithm and any associated parameters used to encrypt and decrypt the **EncodedBiometricObjects**. This content encryption algorithm is a value of type **ContentEncryptionAlgorithmIdentifier** defined as

```
ContentEncryptAlgorithmIdentifier ::=
      AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
```

The definition of type **ContentEncryptionAlgorithmIdentifier** is based on the parameterized type **AlgorithmIdentifier {}** and the information object set **ContentEncryptionAlgorithms**, defined as

```
ContentEncryptionAlgorithms ALGORITHM ::= {
   { OID des-ede3-cbc PARMS IV },
   ...  -- Expect other content encryption algorithms --
}

IV ::= OCTET STRING (SIZE(8))
```

ContentEncryptionAlgorithms specifies an extensible set of **ALGORITHM** information objects. The fields of these information objects are used to constrain the valid values of the components of type **ContentEncryptionAlgorithmIdentifier**. Though only one content encryption algorithm object is defined explicitly in this set, implementations should expect additional algorithms.

The **ContentEncryptionAlgorithms** information object set contains a single object that identifies the encryption algorithm described in ANSI X9.52 [10] as Triple DES (TDES) in CBC (cipher block chaining) mode. Only the two key and three key variants of TDES are supported in XCBF. The single key variant of TDES is simply the DES algorithm and is generally used only for backwards compatibility with existing DES based applications and is considered vulnerable to attack.

The Triple DES algorithm consists of three sequential DES operations, encrypt, decrypt, and encrypt. For three key TDES a different key is used for each DES operation. For two key TDES one key is used for both DES encrypt operations, and the second key is used for the DES decrypt operation.

The **encryptedContent** component contains a value of type **EncodedBiometricObjects** encrypted using the content encryption algorithm given in the **contentEncryptionAlgorithm** component. A value of **encryptedContent** is an opaque string of octets treated as having no discernable structure. This string is a value of type **EncryptedContent** defined as

```
EncryptedContent ::= OCTET STRING
```

A value of the **encryptedContentInfo** component of any of the privacy block choice alternative types can be represented in XML markup as

```
<encryptedContentInfo>
   <contentType>1.2.840.113549.1.7.1</contentType>
   <contentEncryptionAlgorithm>
      <algorithm>1.2.840.113549.3.7</algorithm>
      <parameters>
         <IV>7EA13D6E143CB5C9</IV>
      </parameters>
   </contentEncryptionAlgorithm>
   <encryptedContent>
      D8F6 ... F766
   </encryptedContent>
</encryptedContentInfo>
```

This markup illustrates a typical value. The encrypted content type is identified as ordinary data. The Triple DES content encryption algorithm is identified along with its associated parameters, an initialization vector, `<IV>`. An ellipsis is used as a placeholder for part of the encrypted content.

5.1.3.2 Fixed Key EncryptedData

The **fixedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a value of type **EncryptedData**. This type is a sequence of two components, an integer version number and a value of type **EncryptedContentInfo**. Type **EncryptedData** is defined as

```
EncryptedData ::= SEQUENCE {
   version               CMSVersion,
   encryptedContentInfo  EncryptedContentInfo
}
```

The **fixedKey** alternative assumes that the recipient of the **EncryptedData** value knows the key used to encrypt the biometric information, perhaps by prior agreement or as the result of a key exchange. The **version** component of type **EncryptedData** is always the integer value eighty-four. The components of type **EncryptedContentInfo** are described in section 4.1.3.1 *Encrypted Content Information*.

A value of the **fixedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** can be represented in XML markup as

```
<fixedKey>
   <version>84</version>
   <encryptedContentInfo>
      <contentType>1.2.840.113549.1.7.1</contentType>
      <contentEncryptionAlgorithm>
         <algorithm>1.2.840.113549.3.7</algorithm>
         <parameters>
            <IV>7EA13D6E143CB5C9</IV>
         </parameters>
      </contentEncryptionAlgorithm>
      <encryptedContent>
         ...
```

```
      </encryptedContent>
   </encryptedContentInfo>
</fixedKey>
```

This markup uses the fixed key choice alternative of the privacy block, a value of version number eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is identified as ordinary data. The Triple DES content encryption algorithm is identified along with its associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not shown.

5.1.3.2.1 Encryption Process

A value of type **EncryptedData** is created by encrypting a series of one or more values of type **BiometricObject** in their encoded form using a content encryption algorithm and a fixed content encryption key known to the sender and recipient. The content to be encrypted is a value of type **EncodedBiometricObjects**. This value is always encoded using CXER. The content encryption algorithm used to encrypt the biometric objects is one of the algorithms specified in the information object set **ContentEncryptionAlgorithms**.

The **contentType** component of type **EncryptedContentInfo** is set to indicate ordinary data. The associated **contentEncryptionAlgorithm** value is set to identify the algorithm used to encrypt the content, and the **encryptedContent** value is set to the results of encrypting the content using this content encryption algorithm.

5.1.3.2.2 Decryption Process

To decrypt a value of type **EncryptedData**, the content encryption algorithm specified in the **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the associated **encryptedContent** component using a known fixed key to recover a value of type **EncodedBiometricObjects**. This recovered value will contain one or more values of type **BiometricObject** encoded using CXER.

5.1.3.3 Named Key EncryptedData

The **namedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a value of type **NamedKeyEncryptedData**. This type is sequence with two components, **keyName** and **encryptedData**. Type **NamedKeyEncryptedData** is defined as

```
NamedKeyEncryptedData ::= SEQUENCE {
   keyName        OCTET STRING (SIZE(1..MAX)),
   encryptedData  EncryptedData
}
```

The **keyName** component explicitly identifies the key used to encrypt and decrypt the content by name. The **encryptedData** component is a value of type **EncryptedData**. This type contains two components, an integer **version** number that is always eighty-four in this standard, and an **encryptedContentInfo** that is a value of type **EncryptedContentInfo** as described in section 4.1.3.1 *Encrypted Content Information*.

A value of the **namedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** can be represented in XML markup as

```
<namedKey>
   <keyName>6AE173BF5A973D1E</keyName>
   <encryptedData>
      <version>84</version>
      <encryptedContentInfo>
         <contentType>1.2.840.113549.1.7.1</contentType>
         <contentEncryptionAlgorithm>
            <algorithm>1.2.840.113549.3.7</algorithm>
            <parameters>
               <IV>7EA13D6E143CB5C9</IV>
            </parameters>
         </contentEncryptionAlgorithm>
         <encryptedContent>
            ...
         </encryptedContent>
      </encryptedContentInfo>
   </encryptedData>
</namedKey>
```

This markup uses the named key choice alternative of the privacy block, a sequence of a key name and a value of version number eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is identified as ordinary data. The Triple DES content encryption algorithm is identified along with its associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not shown.

5.1.3.3.1 Encryption Process

A value of type **EncryptedData** is created by encrypting a series of one or more values of type **BiometricObject** in their encoded form using a content encryption algorithm and a named key that is known to the recipient of the encrypted biometric information. The content to be encrypted is a value of type **EncodedBiometricObjects**. This value is always encoded using CXER. The content encryption algorithm used to encrypt the biometric objects is one of the algorithms specified in the information object set **ContentEncryptionAlgorithms**.

The **keyName** component of type **NamedKeyEncryptedData** is set to the name of the content encryption key. The **contentType** component of type **EncryptedContentInfo** is set to indicate ordinary data. The associated **contentEncryptionAlgorithm** value is set to identify the algorithm used to encrypt the content, and the **encryptedContent** value is set to the results of encrypting the content using this content encryption algorithm.

5.1.3.3.2 Decryption Process

To decrypt a value of type **NamedKeyEncryptedData**, the content encryption algorithm specified in the **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the associated **encryptedContent** component using the key identified by the **keyName** component of type **NamedKeyEncryptedData** to recover a value of type **EncodedBiometricObjects**. This recovered value will contain one or more values of type **BiometricObject** encoded using CXER.

5.1.3.4 Established Key EnvelopedData

The **establishedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a value of type **EnvelopedData**. Using this type, a message sender can encrypt content that only the intended message recipient can decrypt.

EnvelopedData is defined as a sequence of four components, an integer version number, message sender information, message recipient information, and a value of type

EncryptedContentInfo which is described in section 4.1.3.1 *Encrypted Content Information*. Type **EnvelopedData** is defined as

```
EnvelopedData ::= SEQUENCE {
   version               CMSVersion,
   originatorInfo        [0] OriginatorInfo  OPTIONAL,
   recipientInfos        RecipientInfos,
   encryptedContentInfo  EncryptedContentInfo
}
```

The combination of encrypted content and an encrypted content encryption key forms a “digital envelope”. The **establishedKey** alternative uses a randomly generated content encryption key to encrypt digital content. The same key is used to decrypt the content. The content encryption key shall be protected during transport, so the recipient’s public and private key pair is used to encrypt and decrypt the content encryption key.

The encrypted content is value of type **EncodedBiometricObjects**. This type is a series of one or more values of type **BiometricObject** in their encoded form. In XCBF these values are encoded using CXER.

The **version** component of type **EnvelopedData** is the integer value eighty-four. The optional **originatorInfo** component facilitates distribution of digital certificates and certificate revocation lists. The **recipientInfos** component contains information needed to recover the encrypted content encryption key used to encrypt the biometric information. The **encryptedContentInfo** component is a value of type **EncryptedContentInfo**. This type is described in section 4.1.3.1 *Encrypted Content Information*.

A value of the **establishedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** can be represented in XML markup as

```
<establishedKey>
   <version>84</version>
   <recipientInfos>
      <RecipientInfo>
         <ktri>
            <version>84</version>
            <rid>
               <certHash>
                  <ietf>6E143CF31A562FA9492681D27A22013D2AAD435D</ietf>
               </certHash>
            </rid>
            <keyEncryptionAlgorithm>
               <algorithm>
                  1.2.840.113549.1.1.1
               </algorithm>
               <parameters><NullParms/></parameters>
            </keyEncryptionAlgorithm>
            <encryptedKey>
               ...
            </encryptedKey>
         </ktri>
      </RecipientInfo>
   </recipientInfos>
   <encryptedContentInfo>
```

```
      <contentType>1.2.840.113549.1.7.1</contentType>
      <contentEncryptionAlgorithm>
         <algorithm>1.2.840.113549.3.7</algorithm>
         <parameters>
            <IV>7EA13D6E143CB5C9</IV>
         </parameters>
      </contentEncryptionAlgorithm>
      <encryptedContent>
         ...
      </encryptedContent>
   </encryptedContentInfo>
</establishedKey>
```

This markup uses the established key choice alternative of the privacy block, a value of version number eighty-four of the cryptographic message type **EnvelopedData**. The optional originator information is not present. The recipient information uses the key transport choice alternative. The recipient public and private key pair is indicated by a SHA1 hash of a public key certificate. The content encryption key is encrypted using the **rsaEncryption** algorithm, which has no associated parameters indicating that no initialization vector is required. The encrypted content type is identified as ordinary data. The Triple DES content encryption algorithm is identified along with its associated parameters, an initialization vector, **<IV>**. An ellipsis is used as a placeholder and the encrypted content encryption key and the encrypted content are not shown.

5.1.3.4.1 Certificates and CRLs

Type **OriginatorInfo** is a sequence of two components that may contain sets of digital certificates and certificate revocation lists (CRLs). This type is defined as

```
OriginatorInfo ::= SEQUENCE {
   certs  [0] CertificateSet  OPTIONAL,
   crls   [1] CertificateRevocationLists  OPTIONAL
}
(ALL EXCEPT({ -- none; at least one component is present -- }))
```

Any combination of X9.68 domain certificates, X.509 certificates and attribute certificates may be included in the **CertificateSet** type in any order. There may be more or fewer certificates needed for any purpose. Certificates are provided as needed to support content key encryption in the key transport key management technique used in XCBF. Use of the **CertificateSet** type to distribute certificates is not required. They may be obtained by other means, or an online certificate validation service may be used instead. Only version one X9.68 domain certificates, version three X.509 certificates and version two attribute certificates are supported in this standard. The value of the **CertificateSet** octet string shall be the concatenation of the DER encodings of one or more X.509 certificate and attribute certificate types. The encoding of this octet string is specified in 7.4.2: Octet Strings with Certificates and Certificate Revocation Lists.

Any number of CRLs may be included in the **CertificateRevocationLists** type in any order. There may be more or fewer CRLs needed for any purpose. CRLs are provided as needed to support certificate validation. Use of the **CertificateRevocationLists** type to distribute CRLs is not required. CRLs may be obtained by other means, or an online certificate validation service may be used instead. Only version two certificate revocation lists are supported in this standard. The value of the **CertificateRevocationList** octet string shall be the concatenation of the DER encodings of one or more X.509 CRLs. The encoding of this octet string is specified in 7.4.2: Octet Strings with Certificates and Certificate Revocation Lists.

The X.509 certificates and certificate revocation lists used in XCBF are signed binary objects, whose digital signatures have been calculated on values encoded using the Distinguished Encoding Rules (DER) of ASN.1. In order to verify the signatures on these objects, their original encodings must be used. This is not affected by any representation (such as a base 64 encoding) used for transfer (see 7.4.2: Octet Strings with Certificates and Certificate Revocation Lists).

5.1.3.4.2 Recipient Information

Type **RecipientInfos** is a series of values of type **RecipientInfo**, one value for each recipient of a digital envelope in **EnvelopedData**. In XCBF there is always a single digital envelope recipient, and type **RecipientInfos** is constrained to a series of one **RecipientInfo** and defined as

```
RecipientInfos ::= SET SIZE(1) OF RecipientInfo
```

Several key management techniques can be used in **EnvelopedData**. In XCBF, only key transport is supported. Other techniques such as constructive key management may be employed by an application, but such use in not defined in this standard. Type **RecipientInfo** is restricted to a single choice alternative and defined as

```
RecipientInfo ::= CHOICE {
   ktri  KeyTransRecipientInfo
}
```

Key transport information is provided to the recipient of a digital envelope so that the envelope can be opened and the protected content encryption key recovered. The content encryption key may then be used to decrypt the content.

The information needed by the recipient to recover the content encryption key is contained in a value of type **KeyTransRecipientInfo** defined as

```
KeyTransRecipientInfo ::= SEQUENCE {
   version                CMSVersion,
   rid                    RecipientIdentifier,
   keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
   encryptedKey           EncryptedKey
}
```

This type is a sequence of four components. The integer **version** number is always set to eighty-four in XCBF. The **rid** component is used to identify the public key used to encrypt the content encryption key. This public key is bound to a key encryption algorithm in a public key certificate. It is associated with the recipient private key needed to decrypt the content encryption key used by the sender to encrypt the content.

A hash of the public key certificate uniquely identifies the recipient certificate.

```
RecipientIdentifier ::= CHOICE {
   certHash  [73] EXPLICIT Hash
```

```
}
```

The **keyEncryptionAlgorithm** component identifies the key encryption algorithm and any associated parameters used to encrypt the content encryption key.

```
KeyEncryptionAlgorithmIdentifier ::=
      AlgorithmIdentifier {{KeyEncryptionAlgorithms}}

KeyEncryptionAlgorithms ALGORITHM ::= {
   { OID rsaEncryption PARMS NullParms },

   ...  -- expect other key encryption algorithms --
}
```

The encrypted content encryption key is an opaque string, a value of type **EncryptedKey** defined as

```
EncryptedKey ::= OCTET STRING
```

5.1.3.4.3 Digital Envelope Processing

To create a digital envelope, a content encryption algorithm is selected. The content encryption algorithm identifier and any associated parameters form the **contentEncryptionAlgorithm** value of the **encryptedContentInfo** component of type **EnvelopedData**. The recipient uses this value to recover the encrypted content.

The content encryption key is encrypted using the key encryption algorithm and key transport public key from the recipient's public key certificate. This certificate must contain a key usage extension which asserts the **keyEncipherment** bit. The key encryption algorithm identifier and any associated parameters used to encrypt the content encryption key with the key transport public key form the **keyEncryptionAlgorithm** component of type **KeyTransRecipientInfo**.

The result of encrypting the content encryption key forms the **encryptedKey** component of type **KeyTransRecipientInfo**. A hash of the complete DER encoding of the recipient's public key certificate is used to populate the **rid** component, and the **version** component is set to the integer eighty-four. This certificate hash mechanism provides a single way to uniquely identify any type of certificate. This is the only form of certificate identification supported in XCBF.

The content encryption key is used to encrypt a value of type **EncodedBiometricObjects**. This type is a series of one or more values of type **BiometricObject** in their encoded form. These values are encoded using CXER.

To retrieve the encrypted content, the recipient first decrypts the value of the **encryptedKey** component of type **KeyTransRecipientInfo** to recover the content encryption key using the private key associated with the public key used to encrypt the content encryption key. This private key is indicated by the hash of the associated public key certificate in the **rid** component of type **KeyTransRecipientInfo**. The recovered content encryption key is then used to decrypt the content to recover a value of type **EncodedBiometricObjects**.

5.1.4 PrivacyAndIntegrityObjects

The **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** is a value of type **PrivacyAndIntegrityObjects**. This type is defined as a sequence of three components, optional **biometricHeaders**, a **privacyBlock**, and an **integrityBlock**.

```
PrivacyAndIntegrityObjects ::= SEQUENCE {
   biometricHeaders  BiometricHeaders  OPTIONAL,
   privacyBlock      PrivacyBlock,
   integrityBlock    IntegrityBlock
}
```

The **biometricHeaders** component is optional and is composed of a series of one or more values of type **BiometricHeader**. The **privacyBlock** component is a value of type **PrivacyBlock**. The **BiometricHeader** and **PrivacyBlock** types are described in section 4.1.3. *PrivacyObjects*. The **integrityBlock** component is a value of type **IntegrityBlock**. This type is described in section 4.1.2. *IntegrityObjects*.

The input to all cryptographic process is a value of type **EncodedBiometricObjects**, a series of one or more values of type **BiometricObject** in their encoded form using CXER. The order of the components in type **PrivacyAndIntegrityObjects** facilitates one pass processing for both sender and recipient.

For the sender, a value of type **EncodedBiometricObjects** is created and then used as input to the cryptographic processing of both the **privacyBlock** and **integrityBlock** components. For the recipient, the **privacyBlock** component is first processed to recover the encrypted content, a value of type **EncodedBiometricObjects**. This recovered value is then used to validate the signature in the **integrityBlock** component.

A value of the **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** can be represented in XML markup as

```
<privacyAndIntegrityObjects>
   <privacyBlock>
      <namedKey>
         <keyName>6AE173BF5A973D1E</keyName>
         <encryptedData>
            <version>84</version>
            <encryptedContentInfo>
               <contentType>1.2.840.113549.1.7.1</contentType>
               <contentEncryptionAlgorithm>
                  <algorithm>1.2.840.113549.3.7</algorithm>
                  <parameters>
                     <IV>7EA13D6E143CB5C9</IV>
                  </parameters>
               </contentEncryptionAlgorithm>
               <encryptedContent>
                  ...
               </encryptedContent>
            </encryptedContentInfo>
         </encryptedData>
      </namedKey>
   </privacyBlock>
   <integrityBlock>
```

```
      <digitalSignature>
         <algorithmID>
            <algorithm>1.2.840.10040.4.3</algorithm>
            <parameters><NullParms/></parameters>
         </algorithmID>
         <signature>
            ...
         </signature>
      </digitalSignature>
   </integrityBlock>
</privacyAndIntegrityObjects>
```

This markup combines a privacy block and integrity block. The named key choice alternative of the privacy block and the digital signature choice alternative of the integrity block are used. The optional biometric headers are not present.

The named key alternative is a sequence containing a key name and a value of version number eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is identified as ordinary data, and is computed on a value of type **EncodedBiometricObjects**. The Triple DES content encryption algorithm is identified along with its associated parameters, an initialization vector, `<IV>`. An ellipsis is used as a placeholder and the encrypted content is not shown.

The digital signature choice alternative is a simple digital signature on a value of type **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) with Secure Hash Algorithm (SHA1) and its associated parameters, `<NullParms/>` are used for signing a value of **EncodedBiometricObjects**. An ellipsis is used as a placeholder and the signature is not shown.

6 References

6.1 Normative

These references or an understanding of the materials within them are required to implement this XCBF standard. They are intended to include any amendments and technical corrigenda issued after their publication. Users of this standard are encouraged to seek the latest versions of these referenced materials before initiating any serious work.

1 ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open Systems Interconnection -- The Directory: Authentication framework.

2 ISO/IEC 8824-1:2002 | ITU-T Recommendation X.680 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation, http://www.itu.int/ITU-T/studygroups/com17/languages/X680_0702.pdf.

3 ISO/IEC 8824-2:2002 | ITU-T Recommendation X.681 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Information Object Specification, http://www.itu.int/ITU-T/studygroups/com17/languages/X681_0702.pdf.

4 ISO/IEC 8824-3:2002| ITU-T Recommendation X.682 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Constraint Specification, http://www.itu.int/ITU-T/studygroups/com17/languages/X682_0702.pdf .

5 ISO/IEC 8824-4:2002| ITU-T Recommendation X.683 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications, http://www.itu.int/ITU-T/studygroups/com17/languages/X683_0702.pdf.

6 ISO/IEC 8825-1:2002| ITU-T Recommendation X.690 (2002), Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), http://www.itu.int/ITU-T/studygroups/com17/languages/X690_0702.pdf.

7 ISO/IEC 8825-4:2002 | X.693 ITU-T Recommendation X.693 (2002) |, Information Technology - ASN.1 Encoding Rules: XML Encoding Rules (XER), http://www.itu.int/ITU-T/studygroups/com17/languages/X693_0702.pdf.

8 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA), http://webstore.ansi.org/.

9 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1), http://webstore.ansi.org/.

10 ANSI X9.52-1998 Triple Data Encryption Algorithm Modes of Operation, http://webstore.ansi.org/.

11 ANSI X9.68:2001 Digital Certificates for Mobile/Wireless and High Transaction Volume Financial Systems: Part 2: Domain Certificate Syntax, http://webstore.ansi.org/.

12 ANSI X9.71-1999 Keyed Hash Message Authentication Code (HMAC), http://webstore.ansi.org/.

13 ANSI X9.73:2002 Cryptographic Message Syntax (CMS), http://webstore.ansi.org/.

14 ANSI X9.84:2003 Biometric Information Management and Security For The Financial Services Industry, http://webstore.ansi.org/.

15 ANSI X9.96:2003 (draft) XML Cryptographic Message Syntax (XCMS).

16 ANSI INCITS 358-2002 - Information technology - BioAPI Specification, http://webstore.ansi.org/.

17 CBEFF Common Biometric Exchange File Format, NISTIR-6529, http://oasis-open.org/committees/xcbf/docs/NISTR6529-CBEFF.pdf, January 3, 2001.

18 S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

19 PKCS #7 – Cryptographic Message Syntax Standard, http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/, 1 November 1993.

20 N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part 1: Format of Internet Message Bodies, http://www.ietf.org/rfc/rfc2045.txt, IETF RFC 2045, November 1996.

7 XCBF Schema

The following ASN.1 modules provide the schema for all of the XML markup defined in this standard.

7.1 X9-84-Biometrics Module

```
X9-84-Biometrics {
   iso(1) identified-organization(3) tc68(133) country(16) x9(840)
      x9Standards(9) x9-84(84) module(0) biometrics(1) rev(1) }
   DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS All;

IMPORTS

   -- X9.84 Biometrics Information Management and Security IDs --

   BiometricTypes, CBEFF-Formats, IBIA-Formats, MatchingAIDs,
   ProcessingAIDs, x509-biometricTemplates,
   x968-biometricTemplates, X9-Formats
      FROM X9-84-Identifiers {
         iso(1) identified-organization(3) tc68(133) country(16)
            x9(840) x9Standards(9) x9-84(84) module(0)
               ids(3) rev(1) }

   -- X9.84 Biometrics Information Management and Security CMS --

   AuthenticatedData, EncryptedData, EnvelopedData,
   MACAlgorithmIdentifier, SignatureAlgorithmIdentifier,
   SignedData
      FROM X9-84-CMS {
         iso(1) identified-organization(3) tc68(133) country(16)
            x9(840) x9Standards(9) x9-84(84) module(0)
               cms(2) rev(1) } ;



BiometricSyntaxSets ::= SEQUENCE SIZE(1..MAX) OF BiometricSyntax

BiometricSyntax ::= CHOICE {
   biometricObjects            BiometricObjects,
   integrityObjects            IntegrityObjects,
   privacyObjects              PrivacyObjects,
   privacyAndIntegrityObjects  PrivacyAndIntegrityObjects
}

BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject

BiometricObject ::= SEQUENCE {
   biometricHeader  BiometricHeader,
   biometricData    BiometricData
}

--
-- All of the cryptographic processing in this standard is performed
```

```
-- on a value of type EncodedBiometricObjects. This is a sequence of
-- one or more values of type BiometricObject in its encoded form.
--

EncodedBiometricObjects ::= BIOMETRIC.&Type( BiometricObjects )

BiometricHeader ::= SEQUENCE {
   version         BiometricVersion  DEFAULT hv1,
   recordType      RecordType  OPTIONAL,
   dataType        DataType  OPTIONAL,
   purpose         Purpose  OPTIONAL,
   quality         Quality  OPTIONAL,
   validityPeriod  ValidityPeriod  OPTIONAL,
   format          Format  OPTIONAL
}

BiometricVersion ::= INTEGER { hv1(0) } (0..MAX)

RecordType ::= BIOMETRIC.&name({BiometricTypes})

DataType ::= ENUMERATED {
   raw          (0),
   intermediate (1),
   processed    (2)
}

Purpose ::= ENUMERATED {
   verify          (1),
   identify        (2),
   enroll          (3),
   enrollVerify    (4),
   enrollIdentity  (5),
   audit           (6),

   ...  -- Expect other values --
}

Quality ::= INTEGER {
   lowest       (  0),
   highest      (100),
   notSet       ( -1),
   notSupported ( -2)
} (-2..100,...)

ValidityPeriod ::= SEQUENCE {
   notBefore  DateTime OPTIONAL,
   notAfter   DateTime  OPTIONAL
}
 (ALL EXCEPT({ -- none; at least one component is present --  }))

DateTime ::= RELATIVE-OID  -- { yyyy mm dd hh mm ss z } --

Format ::= SEQUENCE {
   formatOwner  BIOMETRIC.&name({Owner}),
   formatType   BIOMETRIC.&Type({Owner}{@formatOwner})  OPTIONAL
}

Owner BIOMETRIC ::= {
   CBEFF-Formats  |  -- http://www.nist.gov --
   IBIA-Formats   |  -- http://www.ibia.org --
   X9-Formats,       -- http://www.x9.org   --

   ...  -- expect additional vendor specific formats --
```

```
}

-- Integrity --

IntegrityObjects ::= SEQUENCE {
   biometricObjects  EncodedBiometricObjects,
   integrityBlock    IntegrityBlock
}

IntegrityBlock ::= CHOICE {
   digitalSignature           DigitalSignature,
   messageAuthenticationCode  MessageAuthenticationCode,
   signedData                 SignedData,
   authenticatedData          AuthenticatedData
}

DigitalSignature ::= SEQUENCE {
   algorithmID  SignatureAlgorithmIdentifier,
   signature    OCTET STRING( CONSTRAINED BY {
                   -- signature on --  EncodedBiometricObjects })
}

MessageAuthenticationCode ::= SEQUENCE {
   keyName      OCTET STRING  OPTIONAL,
   algorithmID  MACAlgorithmIdentifier,
   mac          OCTET STRING (CONSTRAINED BY {
                   -- MAC or HMAC on -- EncodedBiometricObjects })
}

-- Privacy --

PrivacyObjects ::= SEQUENCE {
   biometricHeaders  BiometricHeaders  OPTIONAL,
   privacyBlock      PrivacyBlock
}

BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader

PrivacyBlock ::= CHOICE {
   fixedKey        EncryptedData,
   namedKey        NamedKeyEncryptedData,
   establishedKey  EnvelopedData
}

NamedKeyEncryptedData ::= SEQUENCE {
   keyName        OCTET STRING (SIZE(1..MAX)),
   encryptedData  EncryptedData
}

-- Privacy and integrity --

PrivacyAndIntegrityObjects ::= SEQUENCE {
   biometricHeaders  BiometricHeaders  OPTIONAL,
   privacyBlock      PrivacyBlock,
   integrityBlock    IntegrityBlock
}

-- Authentication Information (AI) --

BiometricInformationSets ::=
      SEQUENCE SIZE(1..MAX) OF BiometricInformation

BiometricInformation ::= SEQUENCE {
```



```
   processingAlgorithms  ProcessingAlgorithms  OPTIONAL,
   matchingMethods       MatchingMethods  OPTIONAL
}
 (ALL EXCEPT({ -- none; at least one component is present -- }))

-- Biometric processing algorithms --

ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation

ProcessingInformation ::= SEQUENCE {
   id     BIOMETRIC.&name({ProcessingAIDs}),
   parms  BIOMETRIC.&Type({ProcessingAIDs}{@id})  OPTIONAL
}

-- Biometric matching methods --

MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation

MatchingInformation ::= SEQUENCE {
   id     BIOMETRIC.&name({MatchingAIDs}),
   parms  BIOMETRIC.&Type({MatchingAIDs}{@id})  OPTIONAL
}

BiometricData ::= OCTET STRING(SIZE(1..MAX))

-- Biometrics information object class --

BIOMETRIC ::= CLASS {
   &name  BIOMETRIC-IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
   WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }

BIOMETRIC-IDENTIFIER ::= CHOICE {
   oid  OBJECT IDENTIFIER,  -- complete object identifier
   id   RELATIVE-OID        -- object identifier fragment
}

-- Biometric certificate extension --

--
-- A biometricTemplates information object can be used to extend the
-- information bound to a public key in an value of types Certificate
-- or AttributeCertificate as defined in The Directory series of
-- standards, to include biometric identity information.
--

biometricTemplates EXTENSION ::= {
   SYNTAX         EncodedBiometricObjects  -- CXER --
   IDENTIFIED BY  x509-biometricTemplates
}

EXTENSION ::= CLASS {
   &id        OBJECT IDENTIFIER  UNIQUE,
   &ExtnType
}
 WITH SYNTAX { SYNTAX &ExtnType IDENTIFIED BY &id }

--
-- A domainBiometricTemplates information object can be used to extend
-- the information bound to a public key in an value of ASN.1 type
-- DomainCertificate as defined in the X9.68 Domain Certificate Syntax
-- standard, to include biometric identity information.
```

```
--

domainBiometricTemplates PRIVATE-X ::= {
   NAME  oid : x968-biometricTemplates
   TYPE  EncodedBiometricObjects  -- CXER --
}

PRIVATE-X ::= CLASS {
   &name  Identifier  UNIQUE,
   &Type  OPTIONAL
}
 WITH SYNTAX { NAME &name [TYPE &Type] }

Identifier ::= CHOICE {
   oid  OBJECT IDENTIFIER,  -- complete object identifier
   id   RELATIVE-OID        -- object identifier fragment
}

END  -- X9-84-Biometrics --
```

7.2 X9-84-CMS Module

```
X9-84-CMS {
   iso(1) identified-organization(3) tc68(133) country(16) x9(840)
      x9Standards(9) x9-84(84) module(0) cms(2) rev(1) }
   DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All;

IMPORTS

   -- ANSI X9.84 Biometric Information Management & Security IDs --

   des-ede3-cbc, dsa-with-sha1, ecdsa-with-SHA1, hmac-with-SHA1,
   id-data, NullParms, OID, rsaEncryption, SHA-Algorithms,
   sha1WithRSAEncryption, sha-1
      FROM X9-84-Identifiers {
         iso(1) identified-organization(3) tc68(133) country(16)
            x9(840) x9Standards(9) x9-84(84) module(0)
               ids(3) rev(1) };

SignedData ::= SEQUENCE {
   version           CMSVersion,
   digestAlgorithms  DigestAlgorithmIdentifiers,
   encapContentInfo  EncapsulatedContentInfo,
   certificates      [0] CertificateSet  OPTIONAL,
   crls              [1] CertificateRevocationLists  OPTIONAL,
   signerInfos       SignerInfos
}

CMSVersion ::= INTEGER { v84(84) } (v84,...)

DigestAlgorithmIdentifiers ::=
      SET SIZE(1) OF DigestAlgorithmIdentifier

DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{DigestAlgorithms}}

DigestAlgorithms ALGORITHM ::= {
   SHA-Algorithms,
```

```
   ... -- Expect other digest algorithms  --
}

EncapsulatedContentInfo ::= SEQUENCE {
   eContentType  ContentType,
   eContent      [0] EXPLICIT OCTET STRING  OPTIONAL
}

ContentType ::= CONTENTS.&id({Contents})

CONTENTS ::= TYPE-IDENTIFIER  -- ISO/IEC 8824-2:1998, Annex A

Contents CONTENTS ::= {
   { Data IDENTIFIED BY id-data }
}

Data ::= OCTET STRING

CertificateSet ::= --[XER:BASE64][1]-- OCTET STRING

CertificateRevocationLists ::= --[XER:BASE64] -- OCTET STRING

SignerInfos ::= SET SIZE(1) OF SignerInfo

SignerInfo ::= SEQUENCE {
   version             CMSVersion,
   sid                 SignerIdentifier,
   digestAlgorithm     DigestAlgorithmIdentifier,
   signatureAlgorithm  SignatureAlgorithmIdentifier,
   signature           SignatureValue
}

SignerIdentifier ::= CHOICE {
   certHash  [1] EXPLICIT Hash
}

Hash ::= CHOICE {
   ietf       CertHash,  -- SHA-1 hash of entire certificate
   withAlgID  DigestInfo
}

CertHash ::= OCTET STRING (ENCODED BY sha-1)

DigestInfo ::= SEQUENCE {
   hashAlgorithm  DigestAlgorithmIdentifier,
   digest         OCTET STRING
}

SignatureAlgorithmIdentifier ::=
      AlgorithmIdentifier {{SignatureAlgorithms}}

SignatureAlgorithms ALGORITHM ::= {
   { OID dsa-with-sha1         PARMS NullParms } |
   { OID ecdsa-with-SHA1       PARMS NullParms  } |
   { OID sha1WithRSAEncryption PARMS NullParms  },

   ...  -- Expect other signature algorithms --
```

[1] The use of [BASE64] in comment anticipates amendments to the ASN.1 standards that will permit such notation without comment marks.

```
}

SignatureValue ::= OCTET STRING

EncryptedData ::= SEQUENCE {
   version              CMSVersion,
   encryptedContentInfo  EncryptedContentInfo
}

EncryptedContentInfo ::= SEQUENCE {
   contentType                 ContentType,
   contentEncryptionAlgorithm  ContentEncryptAlgorithmIdentifier,
   encryptedContent            [0] EncryptedContent
}

ContentEncryptAlgorithmIdentifier ::=
      AlgorithmIdentifier {{ContentEncryptionAlgorithms}}

ContentEncryptionAlgorithms ALGORITHM ::= {
   { OID des-ede3-cbc PARMS IV },

   ...  -- Expect other content encryption algorithms --
}

IV ::= OCTET STRING (SIZE(8))

EncryptedContent ::= OCTET STRING

EnvelopedData ::= SEQUENCE {
   version              CMSVersion,
   originatorInfo       [0] OriginatorInfo  OPTIONAL,
   recipientInfos       RecipientInfos,
   encryptedContentInfo  EncryptedContentInfo
}

OriginatorInfo ::= SEQUENCE {
   certs  [0] CertificateSet  OPTIONAL,
   crls   [1] CertificateRevocationLists  OPTIONAL
}
(ALL EXCEPT({ -- none; at least one component is present -- }))

RecipientInfos ::= SET SIZE(1) OF RecipientInfo

RecipientInfo ::= CHOICE {
   ktri  KeyTransRecipientInfo
}

KeyTransRecipientInfo ::= SEQUENCE {
   version                CMSVersion,
   rid                    RecipientIdentifier,
   keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
   encryptedKey           EncryptedKey
}

RecipientIdentifier ::= CHOICE {
   certHash  [73] EXPLICIT Hash
}

KeyEncryptionAlgorithmIdentifier ::=
      AlgorithmIdentifier {{KeyEncryptionAlgorithms}}

KeyEncryptionAlgorithms ALGORITHM ::= {
   { OID rsaEncryption PARMS NullParms },
```



```
   ...  -- expect other key encryption algorithms --
}

EncryptedKey ::= OCTET STRING

AuthenticatedData ::= SEQUENCE {
   version           CMSVersion,
   recipientInfos    RecipientInfos,
   macAlgorithm      MACAlgorithmIdentifier,
   encapContentInfo  EncapsulatedContentInfo,
   mac               MessageAuthenticationCode
}

MACAlgorithmIdentifier ::= AlgorithmIdentifier {{MACAlgorithms}}

MACAlgorithms ALGORITHM ::= {
   { OID hmac-with-SHA1 },

   ...  -- expect other MAC or HMAC algorithms --
}

MessageAuthenticationCode ::= OCTET STRING


-- Cryptographic algorithm identification --

ALGORITHM ::= CLASS {
   &id    OBJECT IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
   WITH SYNTAX { OID &id [PARMS &Type] }

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


END  -- X9-84-CMS --
```

7.3 X9-84-Identifiers Module

```
X9-84-Identifiers {
   iso(1) identified-organization(3) tc68(133) country(16) x9(840)
      x9Standards(9) x9-84(84) module(0) ids(3) rev(1) }
   DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS All;

IMPORTS

   -- X9.84 Biometrics Information Management and Security --

   BIOMETRIC, BiometricInformationSets
      FROM X9-84-Biometrics {
         iso(1) identified-organization(3) tc68(133) country(16)
            x9(840) x9Standards(9) x9-84(84) module(0)
               biometrics(1) rev(1) }
```

```
   -- X9.84 Biometrics Information Management and Security CMS --

   ALGORITHM
      FROM X9-84-CMS {
         iso(1) identified-organization(3) tc68(133) country(16)
            x9(840) x9Standards(9) x9-84(84) module(0)
               cms(2) rev(1) };


OID ::= OBJECT IDENTIFIER  -- Alias

RelOID ::= RELATIVE-OID  -- Alias

-- x9-84                          { 1 3 133 16 840 9 84          }
-- x9-84-Module                   { 1 3 133 16 840 9 84 0        }
-- x9-84-Biometrics               { 1 3 133 16 840 9 84 0  1     }
-- x9-84-CMS                      { 1 3 133 16 840 9 84 0  2     }
-- x9-84-Identifiers              { 1 3 133 16 840 9 84 0  3     }
-- biometric                      { 1 3 133 16 840 9 84 1        }
-- id-unknown-Type                { 1 3 133 16 840 9 84 1  0     }
-- id-body-Odor                   { 1 3 133 16 840 9 84 1  1     }
-- id-dna                         { 1 3 133 16 840 9 84 1  2     }
-- id-ear-Shape                   { 1 3 133 16 840 9 84 1  3     }
-- id-facial-Features             { 1 3 133 16 840 9 84 1  4     }
-- id-finger-Image                { 1 3 133 16 840 9 84 1  5     }
-- id-finger-Geometry             { 1 3 133 16 840 9 84 1  6     }
-- id-hand-Geometry               { 1 3 133 16 840 9 84 1  7     }
-- id-iris-Features               { 1 3 133 16 840 9 84 1  8     }
-- id-keystroke-Dynamics          { 1 3 133 16 840 9 84 1  9     }
-- id-palm                        { 1 3 133 16 840 9 84 1 10     }
-- id-retina                      { 1 3 133 16 840 9 84 1 11     }
-- id-signature                   { 1 3 133 16 840 9 84 1 12     }
-- id-speech-Pattern              { 1 3 133 16 840 9 84 1 13     }
-- id-thermal-Image               { 1 3 133 16 840 9 84 1 14     }
-- id-vein-Pattern                { 1 3 133 16 840 9 84 1 15     }
-- id-thermal-Face-Image          { 1 3 133 16 840 9 84 1 16     }
-- id-thermal-Hand-Image          { 1 3 133 16 840 9 84 1 17     }
-- id-lip-Movement                { 1 3 133 16 840 9 84 1 18     }
-- id-gait                        { 1 3 133 16 840 9 84 1 19     }
-- processing-algorithm           { 1 3 133 16 840 9 84 2        }
-- matching-method                { 1 3 133 16 840 9 84 3        }
-- format-Owner                   { 1 3 133 16 840 9 84 4        }
-- cbeff-Owner                    { 1 3 133 16 840 9 84 4  0     }
-- ibia-Owner                     { 1 3 133 16 840 9 84 4  1     }
-- id-ibia-SAFLINK                { 1 3 133 16 840 9 84 4  1  1  }
-- id-ibia-Bioscrypt              { 1 3 133 16 840 9 84 4  1  2  }
-- id-ibia-Visionics              { 1 3 133 16 840 9 84 4  1  3  }
-- id-ibia-InfineonTechnologiesAG { 1 3 133 16 840 9 84 4  1  4  }
-- id-ibia-IridianTechnologies    { 1 3 133 16 840 9 84 4  1  5  }
-- id-ibia-Veridicom              { 1 3 133 16 840 9 84 4  1  6  }
-- id-ibia-CyberSIGN              { 1 3 133 16 840 9 84 4  1  7  }
-- id-ibia-eCryp                  { 1 3 133 16 840 9 84 4  1  8  }
-- id-ibia-FingerprintCardsAB     { 1 3 133 16 840 9 84 4  1  9  }
-- id-ibia-SecuGen                { 1 3 133 16 840 9 84 4  1 10  }
-- id-ibia-PreciseBiometric       { 1 3 133 16 840 9 84 4  1 11  }
-- id-ibia-Identix                { 1 3 133 16 840 9 84 4  1 12  }
-- id-ibia-DERMALOG               { 1 3 133 16 840 9 84 4  1 13  }
-- id-ibia-LOGICO                 { 1 3 133 16 840 9 84 4  1 14  }
-- id-ibia-NIST                   { 1 3 133 16 840 9 84 4  1 15  }
-- id-ibia-A4Vision               { 1 3 133 16 840 9 84 4  1 16  }
-- id-ibia-NEC                    { 1 3 133 16 840 9 84 4  1 17  }
-- id-ibia-STMicroelectronics     { 1 3 133 16 840 9 84 4  1 18  }
```

```
-- id-ibia-Ultra-Scan             { 1 3 133 16 840 9 84 4  1 19 }
-- id-ibia-Aurora-Wireless        { 1 3 133 16 840 9 84 4  1 20 }
-- id-ibia-Thales                 { 1 3 133 16 840 9 84 4  1 21 }
-- id-ibia-IBG                    { 1 3 133 16 840 9 84 4  1 22 }
-- id-ibia-Cogent-Systems         { 1 3 133 16 840 9 84 4  1 23 }
-- id-ibia-Cross-Match            { 1 3 133 16 840 9 84 4  1 24 }
-- id-ibia-Recognition-Systems    { 1 3 133 16 840 9 84 4  1 25 }
-- id-ibia-DIN                    { 1 3 133 16 840 9 84 4  1 26 }
-- id-ibia-INCITS-M1              { 1 3 133 16 840 9 84 4  1 27 }
-- x9-Owner                       { 1 3 133 16 840 9 84 4  2    }
-- certificate-Extensions         { 1 3 133 16 840 9 84 5       }
-- x968-biometricTemplates        { 1 3 133 16 840 9 84 5  0    }
-- x509-biometricTemplates        { 1 3 133 16 840 9 84 5  1    }

-- X9.84 arc; base object identifier --

x9-84 OID ::= {
   iso(1) identified-organization(3) tc68(133) country(16)
      x9(840) x9Standards(9) x9-84(84)
}

-- X9.84 ASN.1 modules --

x9-84-Module OID ::= { x9-84 modules(0) }

x9-84-Biometrics  OID ::= { x9-84-Module biometrics(1) rev(1) }

x9-84-CMS         OID ::= { x9-84-Module cms(2) rev(1) }

x9-84-Identifiers OID ::= { x9-84-Module ids(3) rev(1) }

-- X9.84 biometric technologies --

biometric OID ::= { x9-84 biometrics(1) }

id-unknown-Type       OID ::= { biometric unknownType(0) }
id-body-Odor          OID ::= { biometric bodyOdor(1) }
id-dna                OID ::= { biometric dna(2) }
id-ear-Shape          OID ::= { biometric ear-Shape(3) }
id-facial-Features    OID ::= { biometric facialFeatures(4) }
id-finger-Image       OID ::= { biometric fingerImage(5) }
id-finger-Geometry    OID ::= { biometric fingerGeometry(6) }
id-hand-Geometry      OID ::= { biometric handGeometry(7) }
id-iris-Features      OID ::= { biometric irisFeatures(8) }
id-keystroke-Dynamics OID ::= { biometric keystrokeDynamics(9) }
id-palm               OID ::= { biometric palm(10) }
id-retina             OID ::= { biometric retina(11) }
id-signature          OID ::= { biometric signature(12) }
id-speech-Pattern     OID ::= { biometric speech-Pattern(13) }
id-thermal-Image      OID ::= { biometric thermalImage(14) }
id-vein-Pattern       OID ::= { biometric veinPattern(15) }
id-thermal-Face-Image OID ::= { biometric thermalFaceImage(16) }
id-thermal-Hand-Image OID ::= { biometric thermalHandImage(17) }
id-lip-Movement       OID ::= { biometric lipMovement(18) }
id-gait               OID ::= { biometric gait(19) }

-- X9.84 biometric technology object identifier fragments --

unknown-Type        RelOID ::= { unknownType(0) }
body-Odor           RelOID ::= { bodyOdor(1) }
dna                 RelOID ::= { dna(2) }
ear-Shape           RelOID ::= { earShape(3) }
facial-Features     RelOID ::= { facialFeatures(4) }
```

```
finger-Image        RelOID ::= { fingerImage(5) }
finger-Geometry     RelOID ::= { fingerGeometry(6) }
hand-Geometry       RelOID ::= { handGeometry(7) }
iris-Features       RelOID ::= { irisFeatures(8) }
keystroke-Dynamics  RelOID ::= { keystrokeDynamics(9) }
palm                RelOID ::= { palm(10) }
retina              RelOID ::= { retina(11) }
signature           RelOID ::= { signature(12) }
speech-Pattern      RelOID ::= { speechPattern(13) }
thermal-Image       RelOID ::= { thermalImage(14) }
vein-Pattern        RelOID ::= { veinPattern(15) }
thermal-Face-Image  RelOID ::= { thermalFaceImage(16) }
thermal-Hand-Image  RelOID ::= { thermalHandImage(17) }
lip-Movement        RelOID ::= { lipMovement(18) }
gait                RelOID ::= { gait(19) }

BiometricTypes BIOMETRIC ::= {
   { BIOMETRIC id : unknown-Type       } |
   { BIOMETRIC id : body-Odor          } |
   { BIOMETRIC id : dna                } |
   { BIOMETRIC id : ear-Shape          } |
   { BIOMETRIC id : facial-Features    } |
   { BIOMETRIC id : finger-Image       } |
   { BIOMETRIC id : finger-Geometry    } |
   { BIOMETRIC id : hand-Geometry      } |
   { BIOMETRIC id : iris-Features      } |
   { BIOMETRIC id : keystroke-Dynamics } |
   { BIOMETRIC id : palm               } |
   { BIOMETRIC id : retina             } |
   { BIOMETRIC id : signature          } |
   { BIOMETRIC id : speech-Pattern     } |
   { BIOMETRIC id : thermal-Image      } |
   { BIOMETRIC id : vein-Pattern       } |
   { BIOMETRIC id : thermal-Face-Image } |
   { BIOMETRIC id : thermal-Hand-Image } |
   { BIOMETRIC id : lip-Movement       } |
   { BIOMETRIC id : gait               },

   ...  -- expect additional biometric types --
}


-- X9.84 biometric processing algorithms --

processing-algorithm OID ::= { x9-84 processingAlgorithms(2) }

-- X9.84 biometric matching methods --

matching-method OID ::= { x9-84 matchingMethods(3) }

-- X9.84 vendor specific formats --

format-Owner OID ::= { x9-84 format-owners(4) }

cbeff-Owner OID ::= { format-Owner cbeff(0) }

ibia-Owner  OID ::= { format-Owner ibia(1) }

x9-Owner    OID ::= { format-Owner x9(2) }

-- IBIA vendor specific formats registered at http://www.ibia.org

id-ibia-SAFLINK               OID ::= { ibia-Owner  1 }
```

```
id-ibia-Bioscrypt              OID ::= { ibia-Owner  2 }
id-ibia-Visionics              OID ::= { ibia-Owner  3 }
id-ibia-InfineonTechnologiesAG OID ::= { ibia-Owner  4 }
id-ibia-IridianTechnologies    OID ::= { ibia-Owner  5 }
id-ibia-Veridicom              OID ::= { ibia-Owner  6 }
id-ibia-CyberSIGN              OID ::= { ibia-Owner  7 }
id-ibia-eCryp                  OID ::= { ibia-Owner  8 }
id-ibia-FingerprintCardsAB     OID ::= { ibia-Owner  9 }
id-ibia-SecuGen                OID ::= { ibia-Owner 10 }
id-ibia-PreciseBiometric       OID ::= { ibia-Owner 11 }
id-ibia-Identix                OID ::= { ibia-Owner 12 }
id-ibia-DERMALOG               OID ::= { ibia-Owner 13 }
id-ibia-LOGICO                 OID ::= { ibia-Owner 14 }
id-ibia-NIST                   OID ::= { ibia-Owner 15 }
id-ibia-A4Vision               OID ::= { ibia-Owner 16 }
id-ibia-NEC                    OID ::= { ibia-Owner 17 }
id-ibia-STMicroelectronics     OID ::= { ibia-Owner 18 }
id-ibia-Ultra-Scan             OID ::= { ibia-Owner 19 }
id-ibia-Aurora-Wireless        OID ::= { ibia-Owner 20 }
id-ibia-Thales                 OID ::= { ibia-Owner 21 }
id-ibia-IBG                    OID ::= { ibia-Owner 22 }
id-ibia-Cogent-Systems         OID ::= { ibia-Owner 23 }
id-ibia-Cross-Match            OID ::= { ibia-Owner 24 }
id-ibia-Recognition-Systems    OID ::= { ibia-Owner 25 }
id-ibia-DIN                    OID ::= { ibia-Owner 26 }
id-ibia-INCITS-M1              OID ::= { ibia-Owner 27 }


-- When represented as values of type OBJECT IDENTIFIER, these
-- IBIA vendor specific formats may be associated with any ASN.1
-- type.

IBIAoidFormats BIOMETRIC ::= {
   { BIOMETRIC oid : id-ibia-SAFLINK               DATA Any } |
   { BIOMETRIC oid : id-ibia-Bioscrypt             DATA Any } |
   { BIOMETRIC oid : id-ibia-Visionics             DATA Any } |
   { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
   { BIOMETRIC oid : id-ibia-IridianTechnologies   DATA Any } |
   { BIOMETRIC oid : id-ibia-Veridicom             DATA Any } |
   { BIOMETRIC oid : id-ibia-CyberSIGN             DATA Any } |
   { BIOMETRIC oid : id-ibia-eCryp                 DATA Any } |
   { BIOMETRIC oid : id-ibia-FingerprintCardsAB    DATA Any } |
   { BIOMETRIC oid : id-ibia-SecuGen               DATA Any } |
   { BIOMETRIC oid : id-ibia-PreciseBiometric      DATA Any } |
   { BIOMETRIC oid : id-ibia-Identix               DATA Any } |
   { BIOMETRIC oid : id-ibia-DERMALOG              DATA Any } |
   { BIOMETRIC oid : id-ibia-LOGICO                DATA Any } |
   { BIOMETRIC oid : id-ibia-NIST                  DATA Any } |
   { BIOMETRIC oid : id-ibia-A4Vision              DATA Any } |
   { BIOMETRIC oid : id-ibia-NEC                   DATA Any } |
   { BIOMETRIC oid : id-ibia-STMicroelectronics    DATA Any } |
   { BIOMETRIC oid : id-ibia-Ultra-Scan            DATA Any } |
   { BIOMETRIC oid : id-ibia-Aurora-Wireless       DATA Any } |
   { BIOMETRIC oid : id-ibia-Thales                DATA Any } |
   { BIOMETRIC oid : id-ibia-IBG                   DATA Any } |
   { BIOMETRIC oid : id-ibia-Cogent-Systems        DATA Any } |
   { BIOMETRIC oid : id-ibia-Cross-Match           DATA Any } |
   { BIOMETRIC oid : id-ibia-Recognition-Systems   DATA Any } |
   { BIOMETRIC oid : id-ibia-DIN                   DATA Any } |
   { BIOMETRIC oid : id-ibia-INCITS-M1             DATA Any },

   ...  -- Expect additional vendor specific formats --
}
```

```
Any ::= TYPE-IDENTIFIER.&Type  -- Application constrained

-- Relative object identifier representations of the identical
-- IBIA vendor specific formats defined as OBJECT IDENTIFIER
-- values above are used to identify these formats when they must
-- comply with the fixed format requirements of the BioAPI 1.1
-- specification and are associated with a two byte integer value.

ibia-SAFLINK                RelOID ::= {  1 }
ibia-Bioscrypt              RelOID ::= {  2 }
ibia-Visionics              RelOID ::= {  3 }
ibia-InfineonTechnologiesAG RelOID ::= {  4 }
ibia-IridianTechnologies    RelOID ::= {  5 }
ibia-Veridicom              RelOID ::= {  6 }
ibia-CyberSIGN              RelOID ::= {  7 }
ibia-eCryp                  RelOID ::= {  8 }
ibia-FingerprintCardsAB     RelOID ::= {  9 }
ibia-SecuGen                RelOID ::= { 10 }
ibia-PreciseBiometric       RelOID ::= { 11 }
ibia-Identix                RelOID ::= { 12 }
ibia-DERMALOG               RelOID ::= { 13 }
ibia-LOGICO                 RelOID ::= { 14 }
ibia-NIST                   RelOID ::= { 15 }
ibia-A4Vision               RelOID ::= { 16 }
ibia-NEC                    RelOID ::= { 17 }
ibia-STMicroelectronics     RelOID ::= { 18 }
ibia-Ultra-Scan             RelOID ::= { 19 }
ibia-Aurora-Wireless        RelOID ::= { 20 }
ibia-Thales                 RelOID ::= { 21 }
ibia-IBG                    RelOID ::= { 22 }
ibia-Cogent-Systems         RelOID ::= { 23 }
ibia-Cross-Match            RelOID ::= { 24 }
ibia-Recognition-Systems    RelOID ::= { 25 }
ibia-DIN                    RelOID ::= { 26 }
ibia-INCITS-M1              RelOID ::= { 27 }


IBIAidFormats BIOMETRIC ::= {
   { BIOMETRIC id : ibia-SAFLINK                DATA BirInt16 } |
   { BIOMETRIC id : ibia-Bioscrypt              DATA BirInt16 } |
   { BIOMETRIC id : ibia-Visionics              DATA BirInt16 } |
   { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
   { BIOMETRIC id : ibia-IridianTechnologies    DATA BirInt16 } |
   { BIOMETRIC id : ibia-Veridicom              DATA BirInt16 } |
   { BIOMETRIC id : ibia-CyberSIGN              DATA BirInt16 } |
   { BIOMETRIC id : ibia-eCryp                  DATA BirInt16 } |
   { BIOMETRIC id : ibia-FingerprintCardsAB     DATA BirInt16 } |
   { BIOMETRIC id : ibia-SecuGen                DATA BirInt16 } |
   { BIOMETRIC id : ibia-PreciseBiometric       DATA BirInt16 } |
   { BIOMETRIC id : ibia-Identix                DATA BirInt16 } |
   { BIOMETRIC id : ibia-DERMALOG               DATA BirInt16 } |
   { BIOMETRIC id : ibia-LOGICO                 DATA BirInt16 } |
   { BIOMETRIC id : ibia-NIST                   DATA BirInt16 } |
   { BIOMETRIC id : ibia-A4Vision               DATA BirInt16 } |
   { BIOMETRIC id : ibia-NEC                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-STMicroelectronics     DATA BirInt16 } |
   { BIOMETRIC id : ibia-Ultra-Scan             DATA BirInt16 } |
   { BIOMETRIC id : ibia-Aurora-Wireless        DATA BirInt16 } |
   { BIOMETRIC id : ibia-Thales                 DATA BirInt16 } |
   { BIOMETRIC id : ibia-IBG                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-Cogent-Systems         DATA BirInt16 } |
   { BIOMETRIC id : ibia-Cross-Match            DATA BirInt16 } |
```

```
   { BIOMETRIC id : ibia-Recognition-Systems    DATA BirInt16 } |
   { BIOMETRIC id : ibia-DIN                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-INCITS-M1              DATA BirInt16 },

   ...  -- Expect others --
}

BirInt16 ::= INTEGER (0..65535)


IBIA-Formats BIOMETRIC ::= {
   IBIAoidFormats  |  -- Complete object identifiers
   IBIAidFormats,     -- Object identifier fragments

   ...  -- Expect additional IBIA vendor specific formats --
}

id-x984BioInfo  OID ::= { cbeff-Owner x984BioInfo(0) }


CBEFFoidFormats BIOMETRIC ::= {
   { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },

   ...  -- Expect other objects --
}

x984BioInfo  RelOID ::= { x984BioInfo(0) }  -- CBEFF owner

CBEFFidFormats BIOMETRIC ::= {
   { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },

   ...  -- Expect other objects --
}

CBEFF-Formats BIOMETRIC ::= {
   CBEFFoidFormats  |  -- Complete object identifiers
   CBEFFidFormats,     -- Object identifier fragments

   ...  -- Expect additional CBEFF vendor specific formats --
}

MatchingAIDs BIOMETRIC ::= {

   ...  -- Expect CBEFF assignments in BiometricInformationSets --
}

ProcessingAIDs BIOMETRIC ::= {

   ...  -- Expect CBEFF assignments in BiometricInformationSets --
}

X9-Formats BIOMETRIC ::= {
   X9oidFormats |
   X9idFormats,

   ...  -- Expect additional X9 vendor specific formats --
}

X9oidFormats BIOMETRIC ::= {
   ...  -- Expect X9 assigned objects --
}

X9idFormats BIOMETRIC ::= {
```

```
   ...  -- Expect X9 assigned objects of the form { 2 x } --
}

certificate-Extensions OID ::= { x9-84 certificate-Extensions(5) }

x968-biometricTemplates OID ::= { certificate-Extensions 0 }
x509-biometricTemplates OID ::= { certificate-Extensions 1 }

-- RSA PKCS #7 Content type

id-data OID ::= {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs7(7) data(1)
}

-- Security object identifiers


-- FIPS 180-1 and FIPS 180-2 Secure Hash Algorithm --


sha-1 OID ::= {
   iso(1) identified-organization(3) oiw(14) secsig(3)
      algorithm(2) 26
}

sha2Algorithm OID ::= {
   joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
      csor(3) nistAlgorithm(4) hashAlgs(2)
}

id-sha256 OID ::= { sha2Algorithm sha256(1) }

id-sha384 OID ::= { sha2Algorithm sha384(2) }

id-sha512 OID ::= { sha2Algorithm sha512(3) }

SHA-Algorithms ALGORITHM ::= {
   { OID sha-1      PARMS NullParms }  |
   { OID id-sha256  PARMS NullParms }  |
   { OID id-sha384  PARMS NullParms }  |
   { OID id-sha512  PARMS NullParms},

   ...  -- Expect others --
}

NullParms ::= NULL  -- No initialization vector

-- X9.57 DSA signature generated with SHA-1 hash (DSA X9.30)

dsa-with-sha1 OID ::=  {
   iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }

--  X9.71 HMAC with SHA-1 algorithm

hmac-with-SHA1 OID ::= {
   iso(1)  identified-organization(3) dod(6)
       internet(1) security(5) mechanisms(5) 8 1 2 }

-- RSA PKCS #1 signature generated with SHA-1 hash & encryption scheme

sha1WithRSAEncryption OID  ::=  {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 5 }
```

```
rsaEncryption OBJECT IDENTIFIER ::=  {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 1 }

-- ANSI X9.52 Triple DES Modes of Operation --

des-ede3-cbc OBJECT IDENTIFIER ::= {
   iso(1) member-body(2) us(840) rsadsi(113549)
      encryptionAlgorithm(3) 7
}

-- X9.62 ECDSA signature with SHA-1

ecdsa-with-SHA1 OID ::= {
   iso(1) member-body(2) us(840) ansi-x962(10045) signatures(4) 1 }


END  -- X9-84-Identifiers --
```

7.4 Encodings to be employed

7.4.1 Encodings used for calculation of digital signatures and MACs

All digital signatures and Message Authentication Codes shall be calculated using the binary values obtained by applying a CXER encoding to a value of EncodedBiometricObect (see 5.1.2.1.1 Digital Signature Process).

> NOTE 1 – A CXER encoding is defined by X.693 as an initial character representation, followed by the application of the UTF8 transformation to those characters to produce a set of binary octets.

> NOTE 2 – This is the only use of CXER in this specification, and is independent of any encoding used for transfer.

7.4.2 Octet Strings with Certificates and Certificate Revocation Lists

All values of the **CertificateSet** type (and **CertificateRevocationLists** type) octet strings shall be the concatenation of the octets of the DER encodings of one or more X.509 certificates or attribute certificates (or certificate revocation lists, respectively).

If the outer level encoding is BER (see 7.4.3 below) then the value of these octet string types shall be encoded as specified in BER.

> NOTE – A formal specification of the types forming the contents of these octet strings is not possible, as they are the concatenation of self-delimited encodings of multiple ASN.1 types, not the encoding of a single ASN.1 type. This is for historical reasons.

If the outer level encoding is BASIC-XER then the value of these octet strings shall not be encoded as specified in BASIC-XER (a UTF8 transform of a hexadecimal encoding), but shall be encoded as specified in the following paragraph.

The UTF8 transformation shall be applied to the Content-Transfer-Encoding specified in IETF RFC 2045, 6.8, except that the 76 character limit does not apply, and XML escape sequences for white-space can be present in any position within the encoding.

> NOTE - IETF RFC 2045 mandates the presence of line breaks dividing the encoding into lines of at most 76 characters, but this is not required in XCBF encodings. It also allows "white-space" to be inserted in any position within the

base64 encoding. This is extended in this specification to allow the inclusion of XML escape sequences for white-space.

7.4.3 Outer-level encodings

The outer-level encoding used for XCBF shall by default be the BASIC-XER encoding specified by X.693 (but see 7.4.2 above for the use of base64).

By implementor agreement, the outer-level encoding may be BER.

NOTE 1 – If the outer-level encoding is BER, the CXER encoding specified in 7.4.1 is still employed for cryptographic transformations, but is not used for transfer.

NOTE 2 – If the outer-level encoding is BER, the concatenation of DER encodings specified in 7.4.2 is still employed for encoding the set of X.509 types, but the base64 encoding is not used, as there is no requirement for an initial character encoding.

8 Examples

Three examples are included in this section to assist readers and implementors of this standard, and with the goal of promoting secure, interoperable biometric applications and systems.

8.1 BiometricSyntaxSets (CXER, DER)

This example illustrates a value of type **BiometricSyntaxSets** encoded in XML markup using the basic XML Encoding Rules (BASIC-XER), a canonical variant of the XML Encoding Rules (CXER) and a compact, canonical, binary format using the ASN.1 Distinguished Encoding Rules (DER). The XER, CXER and DER representations use exactly the same abstract values, based on the same XCBF ASN.1 schema.

Two representations are well-formed XML markup. The third representation is a compact, binary DER encoding. Both CXER and DER are suitable for use in cryptographic applications involving digital signatures, since these encoding rules provide one and only one way to encode any given value.

The BiometricSyntaxSets value encoded in basic XER:

```
<BiometricSyntaxSets>
   <biometricObjects>
      <BiometricObject>
         <biometricHeader>
            <version> 0 </version>
            <recordType> <id> 4 </id> </recordType>
            <dataType> <processed/> </dataType>
            <purpose> <audit/> </purpose>
            <quality> -1 </quality>
            <validityPeriod>
               <notBefore> 1980.10.4 </notBefore>
               <notAfter> 2003.10.3.23.59.59 </notAfter>
            </validityPeriod>
            <format>
               <formatOwner>
                  <oid> 2.23.42.9.10.4.2 </oid>
               </formatOwner>
            </format>
         </biometricHeader>
         <biometricData> 0A0B0C0D </biometricData>
      </BiometricObject>
   </biometricObjects>
</BiometricSyntaxSets>
```

The same abstract value encodes in 517 octets using canonical XER:

```
<BiometricSyntaxSets><biometricObjects><BiometricObject><biometricHeade
r><version>0</version><recordType><id>4</id></recordType><dataType><pro
cessed/></dataType><purpose><audit/></purpose><quality>1</quality><vali
dityPeriod><notBefore>1980.10.4</notBefore><notAfter>2003.10.3. 23.59.5
9</notAfter></validityPeriod><format><formatOwner>< oid>2.23.42.9.10.4
.2</oid></formatOwner></format></biometricHeader><biometricData>0A0B0C0
D</biometricData></BiometricObject></biometricObjects></BiometricSyntax
Sets>
```

The same abstract value encodes in 57 octets using DER:

```
3037A0353033A02BA1038101048201028301068401FFA50F80048F3C0A0481078F530A03173B
```

```
3BA60AA0088006672A090A040281040A0B0C0D
```

8.2 SignedData

This example illustrates how to encode one or more biometric samples or templates for cryptographic enhancement to provide authentication of origin and data integrity services for biometric samples or templates.

The **SubjectPublicKeyInfo** value from the signer's X.509 Public Key Certificate is shown here in the hexadecimal representation of a DER encoded sequence:

```
308201B73082012C06072A8648CE3804013082011F028181
00FD7F53811D75122952DF4A9C2EECE4E7F611B7523CEF44
00C31E3F80B6512669455D402251FB593D8D58FABFC5F5BA
30F6CB9B556CD7813B801D346FF26660B76B9950A5A49F9F
E8047B1022C24FBBA9D7FEB7C61BF83B57E7C6A8A6150F04
FB83F6D3C51EC3023554135A169132F675F3AE2B61D72AEF
F22203199DD14801C70215009760508F15230BCCB292B982
A2EB840BF0581CF502818100F7E1A085D69B3DDECBBCAB5C
36B857B97994AFBBFA3AEA82F9574C0B3D0782675159578E
BAD4594FE67107108180B449167123E84C281613B7CF0932
8CC8A6E13C167A8B547C8D28E0A3AE1E2BB3A675916EA37F
0BFA213562F1FB627A01243BCCA4F1BEA8519089A883DFE1
5AE59F06928B665E807B552564014C3BFECF492A03818400
028180476ACACB486186A153E25AE0E243FAAF0CD9105CF4
DCF63412F36ABF671F53637E5F9FA7C5ADC78288FDB9FA3C
FAFDEBFDD7A7C3FF2BD63D32F4773413EBD9EAB3CA03BA2D
ED583187763181CB376954FD13F1F8E046D4E3D40652CA8D
4645439A3ADB8D964F98F81E57147BDF4C009885CAD55D13
B38DBAA2F9CBF13DC525F6
```

The biometric objects to be signed with the signer's private key associated with the public key provided above shown in the canonical form used as input to the message digest process used to create the digital signature on the signed content:

```
<BiometricObjects><BiometricObject><biometricHeader><version>0</version
><recordType><id>4</id></recordType><dataType><processed/></dataType><p
urpose><audit/></purpose><quality>-1</quality><validityPeriod><notBefor
e>1998.10.1</notBefore><notAfter>2008.10.1</notAfter></validityPeriod><
format><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner></format><
/biometricHeader><biometricData>0102030405060708090A0B0C0D0E0F11</biome
tricData></BiometricObject><BiometricObject><biometricHeader><version>0
</version><recordType><id>4</id></recordType><dataType><intermediate/><
/dataType><purpose><enroll/></purpose><quality>50</quality><validityPer
iod><notBefore>1998.10.2</notBefore><notAfter>2008.10.2</notAfter></val
idityPeriod><format><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwn
er></format></biometricHeader><biometricData>0102030405060708090A0B0C0D
0E0F1101020304050607080 90A0B0C0D0E0F11</biometricData></BiometricObject
><BiometricObject><biometricHeader><version>0</version><recordType><id>
4</id></recordType><dataType><raw/></dataType><purpose><enroll/></purpo
se><quality>100</quality><validityPeriod><notBefore>1998.10.3</notBefor
e><notAfter>2008.10.3</notAfter></validityPeriod><format><formatOwner><
oid>2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><bio
metricData>0102030405060708090A0B0C0D0E0F1101020304050607080 90A0B0C0D0E
0F1101020304050607080 90A0B0C0D0E0F11</biometricData></BiometricObject><
/BiometricObjects>
```

The complete integrity object represented as an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>

<BiometricSyntaxSets>
   <integrityObjects>
      <biometricObjects>
         <BiometricObjects>

            <BiometricObject>
               <biometricHeader>
                  <version> 0 </version>
                  <recordType> <id> 4 </id> </recordType>
                  <dataType> <processed/> </dataType>
                  <purpose> <audit/> </purpose>
                  <quality> -1 </quality>
                  <validityPeriod>
                     <notBefore> 1998.10.1 </notBefore>
                     <notAfter> 2008.10.1 </notAfter>
                  </validityPeriod>
                  <format>
                     <formatOwner>
                        <oid> 2.23.42.9.10.4.2 </oid>
                     </formatOwner>
                  </format>
               </biometricHeader>
               <biometricData>
                  0102030405060708090A0B0C0D0E0F11
               </biometricData>
            </BiometricObject>

            <BiometricObject>
               <biometricHeader>
                  <version> 0 </version>
                  <recordType> <id> 4 </id> </recordType>
                  <dataType> <intermediate/> </dataType>
                  <purpose> <enroll/> </purpose>
                  <quality> 50 </quality>
                  <validityPeriod>
                     <notBefore> 1998.10.2 </notBefore>
                     <notAfter> 2008.10.2 </notAfter>
                  </validityPeriod>
                  <format>
                     <formatOwner>
                        <oid> 2.23.42.9.10.4.2 </oid>
                     </formatOwner>
                  </format>
               </biometricHeader>
               <biometricData>
                  0102030405060708090A0B0C0D0E0F11
                  0102030405060708090A0B0C0D0E0F11
               </biometricData>
            </BiometricObject>

            <BiometricObject>
               <biometricHeader>
                  <version> 0 </version>
                  <recordType> <id> 4 </id> </recordType>
                  <dataType> <raw/> </dataType>
                  <purpose> <enroll/> </purpose>
                  <quality> 100 </quality>
```

```
                  <validityPeriod>
                     <notBefore> 1998.10.3 </notBefore>
                     <notAfter> 2008.10.3 </notAfter>
                  </validityPeriod>
                  <format>
                     <formatOwner>
                        <oid> 2.23.42.9.10.4.2 </oid>
                     </formatOwner>
                  </format>
               </biometricHeader>
               <biometricData>
                  0102030405060708090A0B0C0D0E0F110102030405060708
                  090A0B0C0D0E0F11010203040506070809 0A0B0C0D0E0F11
               </biometricData>
            </BiometricObject>

      </BiometricObjects>
   </biometricObjects>

   <integrityBlock>
      <signedData>
         <version> 84 </version>
         <digestAlgorithms>
            <DigestAlgorithmIdentifier>
               <algorithm> 1.3.14.3.2.26 </algorithm>
               <parameters> <NullParms/> </parameters>
            </DigestAlgorithmIdentifier>
         </digestAlgorithms>
         <encapContentInfo>
            <eContentType> 1.2.840.113549.1.7.1 </eContentType>
         </encapContentInfo>
         <signerInfos>
            <SignerInfo>
               <version> 84 </version>
               <sid>
                  <certHash>
                     <withAlgID>
                        <hashAlgorithm>
                           <algorithm> 1.3.14.3.2.26 </algorithm>
                           <parameters>
                                 <NullParms/>
                              </parameters>
                        </hashAlgorithm>
                        <digest>
                         DA9245BCD6E666749F43C1A1BD070BAF259B70AA
                        </digest>
                     </withAlgID>
                  </certHash>
               </sid>
               <digestAlgorithm>
                  <algorithm> 1.3.14.3.2.26 </algorithm>
                  <parameters> <NullParms/> </parameters>
               </digestAlgorithm>
               <signatureAlgorithm>
                  <algorithm> 1.2.840.10040.4.3 </algorithm>
                  <parameters> <NullParms/> </parameters>
               </signatureAlgorithm>
               <signature>
                     302C021411BC0D3A74CAD4FA14C263C1B0556C68D7DBF5
                     E60214596C21B62E3715DE81D65F09C21B6CFA3998A5B0
               </signature>
            </SignerInfo>
         </signerInfos>
```

```
         </signedData>
      </integrityBlock>

   </integrityObjects>
</BiometricSyntaxSets>
```

8.3 EncryptedData (fixedKey)

This example illustrates how to encode a series of one or more biometric samples or templates for cryptographic enhancement. A group of three biometric objects is used here, though XCBF allows any number of objects to be included. The optional, cleartext biometric headers are not included in the example message. The group of three biometric objects is encrypted for privacy using a fixed Triple DES key.

As shown in this example, XCBF users can create and exchange arbitrary collections of biometric information to suit the needs of their security applications. This capability provides the application designer complete flexibility. The order of the biometric objects is determined in the application by the sender, allowing them to prioritize or order biometric information for purposes such as aging of data, or grouping records by quality or data type or entity.

Collections of useful biometric information include:

- pairs of iris image templates for an individual; one for each eye
- collections of paired iris image templates for a group of individuals
- collections of finger print image templates, one per digit for an individual
- sets of individual finger print image template collections for a group of persons
- a collection of mixed biometric templates for an individual; say, retina, hand geometry, and DNA
- collections of templates for groups of individuals, such as:
  - all employees at work today, or
  - all of the passengers on Flight 12, or
  - all of the finger print samples collected on Tuesday

The hex encoding of a two key Triple DES key used to encrypt a series of three values of type **BiometricObject** is represented in hexadecimal notation as:

```
D02523B3E561313B  511516297C52A846  D02523B3E561313B
```

The biometric objects to be encrypted are shown here for ease of reading, and are not in the canonical form used as input to the encryption process in this example:

```
<BiometricObjects>

   <BiometricObject>
      <biometricHeader>
         <version> 0 </version>
         <recordType> <id> 4 </id> </recordType>
         <dataType> <processed/> </dataType>
         <purpose> <audit/> </purpose>
```

```
         <quality> -1 </quality>
         <validityPeriod>
            <notBefore> 1998.10.1 </notBefore>
            <notAfter> 2008.10.1 </notAfter>
         </validityPeriod>
         <format>
            <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
         </format>
      </biometricHeader>
      <biometricData>
         0102030405060708090A0B0C0D0E0F11
      </biometricData>
   </BiometricObject>
```

```
   <BiometricObject>
      <biometricHeader>
         <version> 0 </version>
         <recordType> <id> 4 </id> </recordType>
         <dataType> <intermediate/> </dataType>
         <purpose> <enroll/> </purpose>
         <quality> 50 </quality>
         <validityPeriod>
            <notBefore> 1998.10.2 </notBefore>
            <notAfter> 2008.10.2 </notAfter>
         </validityPeriod>
         <format>
            <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
         </format>
      </biometricHeader>
      <biometricData>
         0102030405060708090A0B0C0D0E0F11
         0102030405060708090A0B0C0D0E0F11
      </biometricData>
   </BiometricObject>

   <BiometricObject>
      <biometricHeader>
         <version> 0 </version>
         <recordType> <id> 4 </id> </recordType>
         <dataType> <raw/> </dataType>
         <purpose> <enroll/> </purpose>
         <quality> 100 </quality>
         <validityPeriod>
            <notBefore> 1998.10.3 </notBefore>
            <notAfter> 2008.10.3 </notAfter>
         </validityPeriod>
         <format>
            <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
         </format>
      </biometricHeader>
      <biometricData>
         0102030405060708090A0B0C0D0E0F110102030405060708
         090A0B0C0D0E0F11010203040506070809 0A0B0C0D0E0F11
      </biometricData>
   </BiometricObject>

</BiometricObjects>
```

The complete XCBF privacy message formed after cryptographic processing of the series of three biometric objects:

```
<BiometricSyntaxSets>
   <privacyObjects>
      <privacyBlock>
         <fixedKey>
            <version> 84 </version>
            <encryptedContentInfo>
               <contentType> 1.2.840.113549.1.7.1 </contentType>
               <contentEncryptionAlgorithm>
                  <algorithm> 1.2.840.113549.3.7 </algorithm>
                  <parameters>
                     <IV> 0102030405060708 </IV>
                  </parameters>
               </contentEncryptionAlgorithm>
               <encryptedContent>
                     2C75078EE885D7EE52321E80971A2B8E5C5A646B12CF2F90
                     7BAFA6FA16BDB6FA949898183762A05AB2A488B24D9397B2
                     E63D1885BAEFABD0A24E501EB612CE0118E435C4EEE70230
                     A0D74D766A795C0A23B454A3E6E5DFE60C0F31B3BABCDF23
                     5E1D4B2D7EDAD144BA38A63058ED211368A6887921973C7D
                     BCD1A53CBA2CD36819B488CDEB5008B546F0A7E9EE631D14
                     409A32EA7D784444EF2A8ABD0B54C99A5C0D5070A3A25D36
                     7303C359D4387D12DB2449FB73FC40D850B01555BAF8F86D
                     937A60256E0FDDAF52690CCF8D76770EBE7E359EFF6510A0
                     E39B09E6A8EA0FB844522364E66696965B37382061152D54
                     69332C7D1BF9C2A311D117583AE740DED8849777C3CE56DE
                     EABA687B936693292858CA67CD5D7C2BCD918A736B044173
                     1DDBBB077C348CA6524BEA30FADED63CE76BB47F43232A3C
                     5FA6ABFFE5234C108C6B903DB724B478CD647F52E98A5A28
                     071E881321CA7AAD4F7F1F911C4666F5746A1090FB076251
                     24F4E521BCD78C61C6AB1AC0C142D9CB32E08B1B0C7E8F51
                     07826C883D92963F547F62373CD6CACD791877293E2937EA
                     E93D78FAD09479990BC8488D067C8203A43D9AC132A75712
                     A1A86A4E527FBE941E7E7CAC8CCEADCA72C9853D5FDBFECF
                     972450E13461DD688AB412FAABC912838ED5850A9FF85FC4
                     2D311834670D11FBBB7BE732D7FC44D0326AFD17B92A8353
                     747E3A04065E553901A8C5EA531AC644D204F9209EECD151
                     FF7D623DE5671FC4C07325593D0D442E66C872CC7BB214D5
                     88F01A90C8ECFC19BC9DEA06065C954B673B101EFEFC8FFC
                     E97DBBDFABEE9AE20FFE0927E4F59928CD9E2A0F1D538527
                     2A5C78CE1E6A669B18AAD2498731730FFBDFD72D13EB390E
                     82C402F3033E0201089DFF7758F1C76E1578AE7CF6AC4742
                     631B1743141CD0A29D32F6B7D50C37EC8BC55754FA6D7FF5
                     5B0E1B0E075CF1F7EB0DB0468152BE079703BA71485CA891
                     7EA7CF6ED69CD0AC88AB5E93824B337FDB748542D6A2D1B9
                     96C4634CA52536A25C123D9791173F8668404AE30CA701F0
                     5FF77B976369F7C1F48B834952082BA92E7772C717305E70
                     8E51C89E445C88423E378262315E951A207C30B5E03E805C
                     9C538E9856DBDCF84489BF58757E332BFA01625F76C04A23
                     2271BF4FC985209B51A7CC0E747426007617649ABC75818C
                     A68911E497266E717042EEE6747D9A874F9F174713F8751F
                     2AE5B3D7991299A83F85E78F99C5B01B1FCD8A9EC072F763
                     669854D74CC915611181440305AB483E4A2BE21C516561DC
                     B6E25C5D1DCD5FF84B3740B3BB337BF78585A8395EAB3357
                     132541794115D48BCC37B5B0405201C9CDA503C2641E0283
                     3651CD5E55F442C6E7E8D1EA8D83F9FAE370DA6ADC653651
                     4441C6831E4F1B42A056672EC5152B8BA0D31C6F3603801D
                     C50028562D81055915088D6798E4900481B1C86D35888AC0
                     6E9B61CEBA302BD1A0CB0AC002B640455B14072F9E654E87
                     43AE15F7C46DE69BB2F11408CE22133B9C04D3F097300780
                     DCF653414DA95BF7A1DE0B57E72616BC014108828CE7B027
                     5C391516A9D9AD26A59F9D2D46BE59D93B7DD2525CFAC948
```

```
                        9DDA026F4CF7BADF4552F47B64CC1C65D705697251DC92B9
                        486A53159E9AD9D9C149EDAF78636F6F090763E6C826F9FE
                        EBEE49D8FCCC526BDBB4B2513FB355DC05994DFD9D1B581D
                        DD69C904584E5BDFF9593C9A4AFD9E1FD2FA7056755F9079
                        4F1988F103C7F1EEE9A8BE719B401078E504F56E3B6931C3
                        AC1A4392860367E46F008FFB919B8F1DF0B0D0CE41BF1786
                        9313B14B0E556F94E5573106B7050862B4158BB1C87CEB29
                        B939873D514474D72840B96E70D2661249B856DF2775AE7F
                        1A3378AD9030FD10A1CB5E8EF150808DEC94101449DAADE6
                        4D16EA79E4578FA996AEBFA1E3918A485EFB85364282CCA4
                        073A565C64B8A8704CC95ADA4A9B8796679BD89DAA6A7B0E
                        01E4903080875A3208769CE4B1BDD1B874F5B4E2607FBB72
                        2411F0CAFCF14D7FA870EBE87903D5B583D5C231F95912D5
                </encryptedContent>
            </encryptedContentInfo>
        </fixedKey>
    </privacyBlock>
  </privacyObjects>
</BiometricSyntaxSets>
```

Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this specification:

- Tyky Aichelen (Chair), IBM
- Lieutenant John Aerts, LA County Information Systems Advisory Body
- Karl Best, OASIS
- Taylor Boon, BioNetrix Systems
- Jamie Clark, OASIS
- Robin Cover, Isogen
- Ed Day, Objective Systems
- Dr. Paul Gérôme, Individual Member
- Phillip H. Griffin, Individual Member
- Todd Harbour, FGM
- William Koenig, Bank of America
- John Larmouth, Individual Member
- Monica Martin, Sun Microsystems
- John Messing, American Bar Association
- Michael Nguyen, The Infocomm Development Authority of Singapore
- Rob Philpott, RSA Security
- Rick Randall, Booz Allen Hamilton
- Stefan Ri, Microsoft
- William Riippi, MTG Management Consultants, LLC.
- Darran Rolls, Waveset
- Bancroft Scott, OSS Nokalva
- Clifford Thompson, Individual Member
- Paul Thorpe, OSS Nokalva
- Alessandro Triglia, OSS Nokalva
- Per Vorm, BEA Systems

3213 # Appendix B. Revision History

Rev	Date	By Whom	What
CS-1.0	2003-03-03	Phil Griffin	Committee Specification
CS-1.1	2003-06-09	John Larmouth	Committee Specification

3214

Appendix C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.