XML Common Biometric Format

Working Draft 01, Sunday 17 November 2002

Document identifier:

{*Working Draft*}-{*XML Common Biometric Format*}-{*XCBF*}-{*01*} (PDF, Word)

Location:

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

Editor:

Phillip H. Griffin, Griffin Consulting <phil.griffin@asn-1.com>

Contributors:

Tyky Aichelen, IBM
Ed Day, Objective Systems
Dr. Paul Gérôme, AULM
Phillip H. Griffin (chair), Griffin Consulting
John Larmouth, Larmouth T&PDS Ltd
Monica Martin, Drake Certivo
Bancroft Scott, OSS Nokalva
Paul Thorpe, OSS Nokalva
Alessandro Triglia, OSS Nokalva

Abstract:

Biometrics are measurable physical characteristics or personal behavioral traits that can be 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 ANS X9.84:2002 Biometrics Information Management and Security [14]. They conform to the canonical variant of the XML Encoding Rules (XER) for ASN.1 defined in ITU-T Rec. X.693, and will rely on the security and processing requirements specified in X9.96 XML Cryptographic Message Syntax (XCMS) [15] and X9.73 Cryptographic Message Syntax (CMS) [13] standards .

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 measurable physical characteristics or personal behavioral traits that can be 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 ANS X9.84:2002 *Biometrics Information Management and Security*. They conform to the canonical variant of the XML Encoding Rules (XER) 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] 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 the origin, and privacy of biometric data in XML based systems and applications. Mechanisms and techniques are described for the secure transmission, storage, for 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**]**.

3 Acronyms and Abbreviations

Term	Definition
ANSI	American National Standards Institute
ASN.1	Abstract Syntax Notation One
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
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

90

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, physical characteristic or personal behavioral trait used to authenticate the claimed identity of an enrollee.
Biometric Authentication	The one-to-one process of comparing a submitted biometric sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. Contrast with Biometric "Identification".
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 Identification	The one-to-many process of comparing a submitted biometric sample against all of the biometric reference templates on file to determine whether it matches any of the templates and, if so, the identity of the enrollee whose template was matched. The biometric system using the one-to-many approach is seeking to find an identity amongst a database rather than authenticate a claimed identity. Contrast with "Authentication".
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.
Biometric System ???	An automated system capable of capturing a biometric sample from a user, extracting biometric data from that sample, comparing that biometric data with data contained in one or more reference templates, deciding how well the sample and reference templates match, and indicating whether or not an attempted authentication

	or identification has been achieved.
Biometric Verification	The biometric process of identifying or authenticating a user.
Capture	The collection of a biometric sample from a user.
Comparison ???	The process of comparing biometric data from a sample or template against a previously stored reference template.
Enrollee	A person who has a biometric reference template stored in a biometric system.
Enrollment	The process of collecting, preparing, and storing biometric samples from a user to create a biometric reference template representing that person's identity.
Template	Reference data formed from the biometric measurement of an enrollee and used by a biometric system for comparison against subsequently submitted biometric samples.
Registry Authority	A super user that maintains a registry (ISO 11179). In the context of biometrics and unique identification, a registry authority can maintain authentication methods, for example. Information can be gathered by a trusted process, verified and signed by a trusted authority and considered authenticated information. Information can be indexed and access controlled by a registry authority.[1]

93

1 http://www.biometrics.org/REPORTS/cert_stu_pt2.pdf

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 ANS 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>
   <BiometricSyntax>
         ...
   </BiometricSyntax>
</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
}
```

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 and 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 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 and 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 and 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> 2 </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 and 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> 6 </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 and 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 and is 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 **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 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 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 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-A3Vision               RelOID ::= { 16 }
ibia-NEC                    RelOID ::= { 17 }
ibia-STMicroelectronics     RelOID ::= { 18 }
```

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-A3Vision               DATA BirInt16 } |
   { BIOMETRIC id : ibia-NEC                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-STMicroelectronics     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-A3Vision               DATA Any } |
   { BIOMETRIC oid : id-ibia-NEC                    DATA Any } |
   { BIOMETRIC oid : id-ibia-STMicroelectronics     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 the X9 arc, both complete and relative object identifiers can be registered for use by an biometric application vendors. PHG – To be completed.

```
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
}
```

Type **EncodedBiometricObjects** is 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.

Type **IntegrityBlock** is a choice type with four alternatives:

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

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

digitalSignature	simple digital signature using a fixed secret key
messageAuthenticationCode	A simple MAC or HMAC [12]
signedData	a public key signature
authenticatedData	**???????** on a value of type **EncodedBiometricObjects**

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 ::= SIGNATURE { EncodedBiometricObjects }
```

This type is defined in terms of the parameterized type **SIGNATURE {}** whose single parameter **ToBeSigned** is an open type which can be a value of any ASN.1 type. This parameter is to be input to a digital signature process whose results provide the **BIT STRING** value **signature**:

```
SIGNATURE { ToBeSigned } ::= SEQUENCE {
   algorithmID  SignatureAlgorithmIdentifier,
   signature    BIT STRING( CONSTRAINED BY {
                  -- signature on a value of -- ToBeSigned })
}
```

After the reference to this parameterized type is processed, type **DigitalSignature** is defined as:

```
DigitalSignature ::= SEQUENCE {
   algorithmID  SignatureAlgorithmIdentifier,
   signature    BIT STRING
                 ( CONSTRAINED BY { 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><NoIV></parameters>
      </algorithmID>
      <signature>
         ...
      </signature>
```

```
   </digitalSignature>
</integrityBlock>
```

This markup uses the digital signature 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, an initialization vector, `<IV>` is used for signing a value of **EncodedBiometricObjects**. An ellipsis is used as a placeholder and 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 canonical XER encoding of a value of type **BiometricObjects**.

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 canonical XER 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 a bit string, which becomes the value of the **signature** component of **DigitalSignature**.

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 ::= MAC { EncodedBiometricObjects }
```

This type is defined in terms of the parameterized type **MAC {}** whose single parameter **ToBeMACed** is an open type that is to be input to either a message authentication process that results in the **OCTET STRING** value **mac**:

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

After the reference to this parameterized type is processed, type **MessageAuthenticationCode** is defined as:

```
MessageAuthenticationCode ::= SEQUENCE {
   keyName      OCTET STRING  OPTIONAL,
   algorithmID  MACAlgorithmIdentifier,
   mac          OCTET STRING
                   ( CONSTRAINED BY { 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, protected and kept secret, and exchanged securely. The minimum length of the key used with HMAC depends on the choice of underlying hash function. And 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> ... </keyName>
      <algorithmID>
         <algorithm>1.3.6.1.5.5.8.1.2</algorithm>
      </algorithmID>
      <mac>
         ...
      </mac>
   </messageAuthenticationCode>
</integrityBlock>
```

This markup uses the message authentication code choice alternative of the integrity block, a value of type **MessageAuthenticationCode**. Here this type is used to provide a hashed MAC on a value of type **EncodedBiometricObjects**. The HMAC 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 the shared secret key and the HMAC results, and neither of these values is shown.

5.1.2.2.1 Message Authentication Process

A sender prepares a value of type **EncodedBiometricObjects**, chooses a cryptographic key at random, 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 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 or less.

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.

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.

PHG: Remember that the content is not included in the SignedData in XCBF as it already appears in the IntegrityObject.biometricObjects and so should not be duplicated.

```
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.

PHG: Will this be the hash be over the DER [6] or the cXER or DER encoding of X.509 and X9.68 certificates? Seems that if they are included in the SignedData message, then it should be over

the hash of whatever they are encoded in. But if not included how do we know? Application dependent? Need words for this.

```
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**.

5.1.2.4 AuthenticatedData

Type **AuthenticatedData** is defined as

```
AuthenticatedData ::= SEQUENCE {
   version           CMSVersion,
   recipientInfos    RecipientInfos,
   macAlgorithm      MACAlgorithmIdentifier,
```

```
   encapContentInfo  EncapsulatedContentInfo,
   mac               MessageAuthenticationCode
}
```

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 **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 is 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 the XML Encoding Rules. The type of encrypted content is identified as ordinary data by the information object identifier value **id-data**. This value is defined as

```
<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 ANS 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>1.2.840.113549.1.7.6</contentType>
   <contentEncryptionAlgorithm>
      <algorithm>1.2.840.113549.3.7</algorithm>
      <parameters>
         <IV>7EA13D6E143CB5C9</IV>
      </parameters>
   </contentEncryptionAlgorithm>
   <encryptedContent>
      ...
   </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 and the encrypted content is not shown.

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>1.2.840.113549.1.7.6</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 the XML Encoding Rules. 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 the XML Encoding Rules.

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>1.2.840.113549.1.7.6</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 the XML

Encoding Rules. 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 the XML Encoding Rules.

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 the XML Encoding Rules.

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>
                  </NoIV>
               </parameters>
            </keyEncryptionAlgorithm>
            <encryptedKey>
               ...
            </encryptedKey>
         </ktri>
      </RecipientInfo>
   </recipientInfos>
   <encryptedContentInfo>
      <contentType>1.2.840.113549.1.7.6</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.

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 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 maintained. But these values must also be represented in XML encodings in a useful textual format. So the values in the **certs** and **crls** components of type **OriginatorInfo** I have been base64 armored to minimize their size when represented using XML markup while preserving their original encodings.

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 NoIV },

   ...  -- 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 public key from the recipient's public key certificate. The key encryption algorithm identifier and any associated parameters used to encrypt the content encryption 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.

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 the XML Encoding Rules.

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 based on the XML Encoding Rules. 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>1.2.840.113549.1.7.6</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><NoIV></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, an initialization vector, <IV> 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

1 ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open Systems Interconnection -- The Directory: Authentication framework", International Organization for Standardization, Geneva, Switzerland, 2000.

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

[2] ISO/IEC 8824-1:2001 | ITU-T Recommendation X.680 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation.

[3] ISO/IEC 8824-2:2001 | ITU-T Recommendation X.681 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Information Object Specification.

[4] ISO/IEC 8824-3:2001 | ITU-T Recommendation X.682 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Constraint Specification.

[5] ISO/IEC 8824-4:2001 | ITU-T Recommendation X.683 (2002), Information Technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications.

[6] ISO/IEC 8825-1:2001 | 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).

[7] ISO/IEC 8825-4:2001 | X.693 ITU-T Recommendation X.693 (2002) |, Information Technology - ASN.1 Encoding Rules: XML Encoding Rules (XER).

[8] ANS X9.30-1997 Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry, Part 1: The Digital Signature Algorithm (DSA).

[9] ANS X9.30-1997 Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry, Part 2: The Secure Hash Algorithm (SHA).

[10] ANS X9.52-1999 Triple Data Encryption Algorithm Modes of Operation.

[11] ANS X9.68:2001 Digital Certificates for Mobile/Wireless and High Transaction Volume Financial Services and Security For The Financial Systems: Part 2: Domain Certificate Syntax.

[12] ANS X9.71-1999 Keyed Hash Message Authentication Code (HMAC).

[13] ANS X9.73:2002 Cryptographic Message Syntax (CMS).

[14] ANS X9.84:2002 Biometrics Information Management and Security For The Financial Services Industry.

[15] ANS X9.96:2002 (draft) XML Cryptographic Message Syntax (XCMS).

[16] ANSI/INCITS 358-2002 - Information technology - BioAPI Specification.

[17] CBEFF Common Biometric Exchange File Format, NISTIR-6529, 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.

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, 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 ::= SIGNATURE { EncodedBiometricObjects }

MessageAuthenticationCode ::= MAC { 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
}

-- Parameterized types --

SIGNATURE { ToBeSigned } ::= SEQUENCE {
   algorithmID  SignatureAlgorithmIdentifier,
   signature    BIT STRING( CONSTRAINED BY {
                   -- signature on a value of --  ToBeSigned })
}

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

-- 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  -- DER or 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 ::= {
   NAME  oid : x968-biometricTemplates
   TYPE  EncodedBiometricObjects  -- DER or cXER --
}
```

```
PRIVATE ::= 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

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

   des-ede3-cbc, dsa-with-sha1, ecdsa-with-SHA1, hmac-with-SHA1,
   id-data, NoIV, rsaEncryption, SHA-Algorithms, sha1WithRSAEncryption
      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 ::= OCTET STRING

CertificateRevocationLists ::= 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 NoIV } |
   { OID ecdsa-with-SHA1       PARMS NoIV } |
   { OID sha1WithRSAEncryption PARMS NoIV },

   ...  -- Expect other signature algorithms --
}

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 NoIV },

   ...  -- 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-A3Vision               { 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 }
-- 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-A3Vision               OID ::= { ibia-Owner 16 }
id-ibia-NEC                    OID ::= { ibia-Owner 17 }
id-ibia-STMicroelectronics     OID ::= { ibia-Owner 18 }


-- 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-A3Vision               DATA Any } |
   { BIOMETRIC oid : id-ibia-NEC                    DATA Any } |
   { BIOMETRIC oid : id-ibia-STMicroelectronics     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-A3Vision               RelOID ::= { 16 }
ibia-NEC                    RelOID ::= { 17 }
ibia-STMicroelectronics     RelOID ::= { 18 }


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-A3Vision               DATA BirInt16 } |
   { BIOMETRIC id : ibia-NEC                    DATA BirInt16 } |
   { BIOMETRIC id : ibia-STMicroelectronics     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 --

id-sha1 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 id-sha1    PARMS NoIV }  |
   { OID id-sha256             }  |
   { OID id-sha384             }  |
   { OID id-sha512             },

   ...  -- Expect others --
}

NoIV ::= 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 }

-- ANS 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 --
```

8 Examples

Need some introductory text here

8.1 BiometricSyntaxSets (cXER, DER)

This example illustrates a value of type **BiometricSyntaxSets** encoded in XML markup using the basic XML Encoding Rules (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>
   <BiometricSyntax>
      <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>
   </BiometricSyntax>
</BiometricSyntaxSets>
```

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

```
<BiometricSyntaxSets><BiometricSyntax><biometricObjects><BiometricObjec
t><biometricHeader><version>0</version><recordType><id>4</id></recordTy
pe><dataType>2</dataType><purpose>6</purpose><quality>-1</quality><vali
dityPeriod><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</b
iometricData></BiometricObject></biometricObjects></BiometricSyntax></B
iometricSyntaxSets>
```

The same abstract value encodes in 57 octets using DER:

```
3037A0353033A02BA1038101048201028301 06
8401FFA50F80048F3C0A0481078F530A03173B
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>2</dataType><purpose>6</p
urpose><quality>-1</quality><validityPeriod><notBefore>1998.10.1</notBe
fore><notAfter>2008.10.1</notAfter></validityPeriod><format><formatOwne
r><oid>2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><
biometricData>0102030405060708090A0B0C0D0E0F11</biometricData></Biometr
icObject><BiometricObject><biometricHeader><version>0</version><recordT
ype><oid>4</oid></recordType><dataType>1</dataType><purpose>3</purpose>
<quality>50</quality><validityPeriod><notBefore>1998.10.2</notBefore><n
otAfter>2008.10.2</notAfter></validityPeriod><format><formatOwner><oid>
2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><biometr
icData>0102030405060708090A0B0C0D0E0F11010203040506070809 0A0B0C0D0E0F11
</biometricData></BiometricObject><BiometricObject><biometricHeader><ve
rsion>0</version><recordType><id>4</id></recordType><dataType>0</dataTy
pe><purpose>3</purpose><quality>100</quality><validityPeriod><notBefore
>1998.10.3</notBefore><notAfter>2008.10.3</notAfter></validityPeriod><f
ormat><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner></format></
biometricHeader><biometricData>0102030405060708090A0B0C0D0E0F1101020304
05060708090A0B0C0D0E0F11010203040506070809 0A0B0C0D0E0F11</biometricData
></BiometricObject></BiometricObjects>
```

The complete integrity object represented as an XML document:

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

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

               <BiometricObject>
                  <biometricHeader>
                     <version>0</version>
                     <recordType><id>4</id></recordType>
                     <dataType>2</dataType>
                     <purpose>6</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>1</dataType>
                    <purpose>3</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>0</dataType>
                    <purpose>3</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><NoIV/></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><NoIV/></parameters>
                              </hashAlgorithm>
                              <digest>
                               DA9245BCD6E666749F43C1A1BD070BAF259B70AA
                              </digest>
                           </withAlgID>
                        </certHash>
                     </sid>
                     <digestAlgorithm>
                        <algorithm>1.3.14.3.2.26</algorithm>
                        <parameters><NoIV/></parameters>
                     </digestAlgorithm>
                     <signatureAlgorithm>
                        <algorithm>1.2.840.10040.4.3</algorithm>
                        <parameters><NoIV></parameters>
                     </signatureAlgorithm>
                     <signature>
                        302C02144F9CA4507E2638AE9B632A3698A7AE84858F13
                        3802140BD484312B36B090D2DF8B8A4719353F9A1EFAA5
                     </signature>
                  </SignerInfo>
               </signerInfos>
            </signedData>
         </integrityBlock>

      </integrityObjects>
   </BiometricSyntax>
</BiometricSyntaxSets>
```

8.3 EncryptedData (fixedKey)

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

As shown by 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:

```
623131F21CA8B9B6  707A7F3DB68A5401  623131F21CA8B9B6
```

The biometric objects to be encrypted (shown here for ease of reading not in the canonical form used as input to the encryption process):

```
<BiometricObjects>

   <BiometricObject>
      <biometricHeader>
         <version>0</version>
         <recordType><id>4</id></recordType>
         <dataType>2</dataType>
         <purpose>6</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><oid>4</oid></recordType>
         <dataType>1</dataType>
         <purpose>3</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>0</dataType>
         <purpose>3</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:

```
<BiometricSyntaxSets>
   <BiometricSyntax>
      <privacyObjects>
         <privacyBlock>
            <fixedKey>
               <version>84</version>
               <encryptedContentInfo>
                  <contentType>1.2.840.113549.1.7.6</contentType>
                  <contentEncryptionAlgorithm>
                     <algorithm>1.2.840.113549.3.7</algorithm>
                     <parameters>
                        <IV>7EA13D6E143CB5C9</IV>
                     </parameters>
                  </contentEncryptionAlgorithm>
                  <encryptedContent>
                     F9C32CC99FA8CEA043B88D7A4981C32836A0D04444FECA6B
                     0B8CC83622CF4112F04B24710498D82996183196809FA353
                     39BABC6323F0509700F9ACE6CEA6B06033282486BB1E9AD8
                     FEFED445EFB00B103547F3B07E9CC839E636A10C780744B2
                     F0B8536926E38C2FB3F2F55882AFB8E1AA5CEF23B8C67F47
                     70171EB7DEA87772F8E4615C62AD06C71450D8569A3B69B4
                     828DA9D25F7B91E117A122C8025E29EE4947C39C22C2F299
                     EAEDC3AB24F39B307F84005D82502F972F2C5E1834ECF992
                     912CA6DCCF4B4FCBD84F019BD632A72A5513E3E57C697146
                     6CAA0F9BDADA69095876ADA8F4D3FB5C4F211C9915AFA096
                     63C657EDBAD1CDE85D24B2C0B4A97355D72507987D9F83AB
                     24A5E10FD08FBC362BBC54EC7B9E122A8457AFA68D25B82A
                     C897BDA198CB02D4E1A7E2A731908893A7B3286A5B567394
                     823E178F75F4597CA18B0019C99E6DA6D793D5A8284ECDB7
                     AA34C6C85B7C606370248CFB17C63642AA49FFE835C0EE33
                     8EAC25C98E9FC0095DB10F05B405C366F42594389CC5C178
                     9EA2CEC0F388AB2C4DA2006C491751005E55975FD3A6725C
                     7D608E41A688BD6E8DF552DEAAF6ADDE232E03D2B756D3CD
                     8A9C542017C41A2971914D999F4CF630B347F496AA1D5876
                     4CB805DF6ED506456017351ACA4BF7A80B7B6773359C2F03
                     BE0AC907A1951FE2FAC6589321D873BB752104180E7026F4
                     772570896C86E78303024F2EB0D62635E872901849952029
                     E178645E2A2A7E4F0A2255F65692DD66DA3E0CD8382EFA3F
                     3A1DBAC342DBC10D992E01E5A42E7993C29CF6BD71BECA6A
                     BE7CBE2DF2E9D91DA66FA51DF914BDA34A925B26EEF76F6F
                     CD8DF2C9CF9D497C960FD1DD99D3EAF22442F2B73F59E194
                     50B102C848A220764AFCAF9B182C619F505166760BC7D63E
                     09F16BE521C5DB9BFB2784A6BAD69C23899486381A1DEB14
                     31EECAAB4BD02DDAD00F181AF069D48977F2E01D0184A387
                     1F132654651E3965B0076238C200390187FAA213F3A6012F
```

```
                        7D9AECD95292B9069FA1409DFB3C32AF19656D9DDD83F07F
                        D634E9EFAFFF7D7236BC08B4417B4618EF767790E6AE653D
                        27F8C797F9132E497B20E5A13B2BBF4F64187BE84B61C9C5
                        CD25ACCE38E3D22F678C79E4532A0B5C5D66EF557D829CCD
                        466B1607AF462163ED07032474DA7240360A6BF0354795EE
                        01CF95A8776E9D16717CF140A72804C53F9619FE6B96141E
                        C97BDB495FFBDC0194665D2A1196EAEC814307CB2FAAA8C8
                        2476B11EA01B42E2C40C7306710323E288EC1A9166BF9D85
                        2E3976257315ED6BB70F6933017072830DD1CCC399380114
                        8935F5CFB19C84B8FE753AE5FFE1D89A2872C5675C4E8E8A
                        A3A9C4ACBA749E84CBF3969D72C6B84299E7B5F7062C797F
                        9F057370183C468111522ACBC84DC410900395DCB25E81EA
                        66B6D20E0AE5D5B7206EABD451539A7EBB2FBDDFCE2B06EE
                        09E5D5A16904CB2496EDCAE16EFC4B0FAF107B074A06CBE4
                        5747D8CA1FA0F2F2E8BE5BF27D9F248DF87D351F3EC6EAE7
                        36A8A6BE99811373480040117B6A78C5C0EA97A524FCD3BA
                        5F2A4BD3265CE1CA1A94ACA5FE0E6A69FDB6102AD97FA2EE
                        4DBF6F399075B3AD9339304E16D25959506E90AE771D6CF5
                        C1C7F793A77D52121C152F66B173A04A851C596090354504
                        785DBA06B8C49E4DC5E47734DC95451E8D1A58098888F505
                        E6CA12EA4C8DE8C08102C088A66FB42E38F17694166B7841
                        CDAB5E46DA0C9ACDBC92840450AA547491CCD82242E32A02
                        6D1BDB1C6B35B47BCC672E4ED8702DCDC804B20692508747
                        5C939872694192D6088E0353460C875C0801464AA20AFC71
                        8CD61A72FDC730609AAA1A20952F3D8790C489B20B3FF6A3
                        2346B544B18A91FCF485E06F85A58B310746507330CF3376
                        27875EE54C22CD489BFDD89F7020BD59C8AD15D403DE2E78
                        75B5571A5EBF80C78B8990448F5B71144FFEBE4B5726BCD1
                    </encryptedContent>
                </encryptedContentInfo>
            </fixedKey>
        </privacyBlock>
    </privacyObjects>
  </BiometricSyntax>
</BiometricSyntaxSets>
```

Appendix A. Acknowledgments

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

- Tyky Aichelen, IBM
- Karl Best, OASIS
- Taylor Boon, BioNetrix Systems
- Robin Cover, Isogen
- Ed Day, Objective Systems
- Dr. Paul Gérôme, AULM
- Phillip H. Griffin (chair), Griffin Consulting
- Todd Harbour, FGM
- William Koenig, Bank of America
- John Larmouth, Larmouth T&PDS Ltd
- Monica Martin, Drake Certivo
- Rob Philpott, RSA Security
- Stefan Ri, Microsoft
- Darran Rolls, Waveset
- Bancroft Scott, OSS Nokalva
- Clifford Thompson, TGI Solutions
- Paul Thorpe, OSS Nokalva
- Alessandro Triglia, OSS Nokalva
- Per Vorm, BEA

Appendix B. Revision History

Rev	Date	By Whom	What
wd-01	2002-08-31	Phil Griffin	Initial version
wd-01	2002-09-27	Phil Griffin	X9.84 schema revision; glossary
Wd-01	2002-10-24	Phil Griffin	X.509/X9.68 Certificate extensions
Wd-01	2002-11-17	Phil Griffin	Made various signature processing corrections; added text for privacy objects section; added markup examples to header section; text and code for format registry sections; markup for cryptographic types. Added abbreviations section and populated. Updated references and added refs.

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.