



XML Common Biometric Format

Committee Specification 1.1, June 2003

Document identifier:

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

Location:

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

Edited by:

John Larmouth, Individual Member

Contributors:

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

Abstract:

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

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

Status:

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

Copyright   2002, 2003 The Organization for the Advancement of Structured Information Standards (OASIS)

45 **Table of Contents**

46 1 Introduction 4
47 2 Terminology 5
48 3 Acronyms and Abbreviations 6
49 4 Glossary 7
50 5 X9.84 and BioAPI 1.1 Interoperability 9
51 5.1 BiometricSyntaxSets 9
52 5.1.1 BiometricObjects 10
53 5.1.2 IntegrityObjects 26
54 5.1.3 PrivacyObjects 33
55 5.1.4 PrivacyAndIntegrityObjects 43
56 6 References 45
57 6.1 Normative 45
58 7 XCBF Schema 47
59 7.1 X9-84-Biometrics Module 47
60 7.2 X9-84-CMS Module 51
61 7.3 X9-84-Identifiers Module 54
62 7.4 Encodings to be employed 62
63 7.4.1 Encodings used for calculation of digital signatures and MACs 62
64 7.4.2 Octet Strings with Certificates and Certificate Revocation Lists 62
65 7.4.3 Outer-level encodings 63
66 8 Examples 64
67 8.1 BiometricSyntaxSets (CXER, DER) 64
68 8.2 SignedData 65
69 8.3 EncryptedData (fixedKey) 68
70 Appendix A. Acknowledgments 72
71 Appendix B. Revision History 73
72 Appendix C. Notices 74
73

74 1 Introduction

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

81

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

90

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

96 **2 Terminology**

97 The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*,
98 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
BASIC-XER	Basic XML Encoding Rules for ASN.1
BER	Basic Encoding Rules for ASN.1
BioAPI	Biometric Application Programming Interface
BIR	Biometric Information Record
CBC	Cipher Block Chaining
CBEFF	Common Biometric Exchange File Format
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CXER	Canonical XML Encoding Rules
DER	Distinguished Encoding Rules
DES	Digital Encryption Algorithm
DSA	Digital Signature Algorithm
HMAC	Hashed Message Authentication Code
IBIA	International Biometrics Industry Association
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
SHA	Secure Hash Algorithm
TDES	Triple DES
URL	Uniform Resource Locator
UTC	Universal Coordinated Time
X9	Accredited Standards Committee X9 Financial Services
XCMS	XML Cryptographic Message Syntax
XER	XML Encoding Rules
XML	Extensible Markup Language

4 Glossary

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

Template

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

103

5 X9.84 and BioAPI 1.1 Interoperability

104

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

109

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

5.1 BiometricSyntaxSets

116
117 Type **BiometricSyntaxSets** is a series of one or more values of type **BiometricSyntax**. This
118 type is defined as

119

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

121

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

124

```
125 BiometricSyntax ::= CHOICE {  
126   biometricObjects      BiometricObjects,  
127   integrityObjects     IntegrityObjects,  
128   privacyObjects       PrivacyObjects,  
129   privacyAndIntegrityObjects PrivacyAndIntegrityObjects  
130 }
```

131

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

133

134 biometricObjects	a set of unprotected biometric values
135 integrityObjects	a digitally signed set of biometric values
136 privacyObjects	an encrypted set of biometric values
137 privacyAndIntegrityObjects	a digitally signed and encrypted set of biometric values

138

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

145

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

152

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

157

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

159

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

163

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

166 5.1.1 BiometricObjects

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

170

```
BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject  
  
BiometricObject ::= SEQUENCE {  
  biometricHeader BiometricHeader,  
  biometricData   BiometricData  
}
```

177

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

182

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

184

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

188

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

189

190

191

192

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

194

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

198

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

201

```
202 <biometricObjects>  
203   <BiometricObjects>  
204     <BiometricObject>  
205       <biometricHeader>  
206         ...  
207       </biometricHeader>  
208       <biometricData>  
209         ...  
210       </biometricData>  
211     </BiometricObject>  
212   </BiometricObjects>  
213 </biometricObjects>
```

214

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

217

218 5.1.1.1 BiometricHeader

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

221

```
222 BiometricHeader ::= SEQUENCE {  
223   version      BiometricVersion  DEFAULT hv1,  
224   recordType   RecordType  OPTIONAL,  
225   dataType     DataType  OPTIONAL,  
226   purpose      Purpose  OPTIONAL,  
227   quality      Quality  OPTIONAL,  
228   validityPeriod ValidityPeriod  OPTIONAL,  
229   format       Format  OPTIONAL  
230 }
```

231

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

234

```
235 typedef struct bioapi_bir_header {  
236   uint32 Length;  
237   BioAPI_BIR_VERSION      HeaderVersion;  
238   BioAPI_BIR_DATA_TYPE    Type;  
239   BioAPI_BIR_BIOMETRIC_DATA_FORMAT Format;  
240   BioAPI_QUALITY          Quality;  
241   BioAPI_BIR_PURPOSE      Purpose;  
242   BioAPI_BIR_AUTH_FACTORS FactorsMask;
```

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

244

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

250

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

256

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

259

```
260 <biometricHeader>  
261   <version> 0 </version>  
262   <recordType> <id> 6 </id> </recordType>  
263   <dataType> <processed/> </dataType>  
264   <purpose> <audit/> </purpose>  
265   <quality> 100 </quality>  
266   <validityPeriod>  
267     <notBefore> 1980.10.4 </notBefore>  
268     <notAfter> 2015.10.3.23.59.59 </notAfter>  
269   </validityPeriod>  
270   <format>  
271     <formatOwner>  
272       <oid> 2.23.42.9.10.4.2.0 </oid>  
273     </formatOwner>  
274     <formatType>  
275       <BlindedPrimaryAccountNumber>  
276         A23D552FB4490281C1F6683163D9CCB2  
277       </BlindedPrimaryAccountNumber>  
278     </formatType>  
279   </format>  
280 </biometricHeader>
```

281

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

284 5.1.1.1.1 BiometricVersion

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

287

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

289

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

292

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

297

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

300

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

302

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

304 5.1.1.1.2 RecordType

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

306

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

308

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

311

```
312 BiometricTypes BIOMETRIC ::= {  
313   { BIOMETRIC id : unknown-Type      } |  
314   { BIOMETRIC id : body-Odor         } |  
315   { BIOMETRIC id : dna                } |  
316   { BIOMETRIC id : ear-Shape          } |  
317   { BIOMETRIC id : facial-Features   } |  
318   { BIOMETRIC id : finger-Image      } |  
319   { BIOMETRIC id : finger-Geometry   } |  
320   { BIOMETRIC id : hand-Geometry     } |  
321   { BIOMETRIC id : iris-Features     } |  
322   { BIOMETRIC id : keystroke-Dynamics } |  
323   { BIOMETRIC id : palm               } |  
324   { BIOMETRIC id : retina             } |  
325   { BIOMETRIC id : signature          } |  
326   { BIOMETRIC id : speech-Pattern    } |  
327   { BIOMETRIC id : thermal-Image     } |  
328   { BIOMETRIC id : vein-Pattern      } |  
329   { BIOMETRIC id : thermal-Face-Image } |  
330   { BIOMETRIC id : thermal-Hand-Image } |  
331   { BIOMETRIC id : lip-Movement      } |  
332   { BIOMETRIC id : gait               } |  
333  
334   ... -- expect additional biometric types --  
335 }
```

336

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

340

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

343

```
344 typedef sint8 BioAPI_BIR_AUTH_FACTORS;
```

345

```
346 #define BioAPI_FACTOR_MULTIPLE (0x00000001)
347 #define BioAPI_FACTOR_FACIAL_FEATURES (0x00000002)
348 #define BioAPI_FACTOR_VOICE (0x00000004)
349 #define BioAPI_FACTOR_FINGERPRINT (0x00000008)
350 #define BioAPI_FACTOR_IRIS (0x00000010)
351 #define BioAPI_FACTOR_RETINA (0x00000020)
352 #define BioAPI_FACTOR_HAND_GEOMETRY (0x00000040)
353 #define BioAPI_FACTOR_SIGNATURE_DYNAMICS (0x00000080)
354 #define BioAPI_FACTOR_KEYSTROKE_DYNAMICS (0x00000100)
355 #define BioAPI_FACTOR_LIP_MOVEMENT (0x00000200)
356 #define BioAPI_FACTOR_THERMAL_FACE_IMAGE (0x00000400)
357 #define BioAPI_FACTOR_THERMAL_HAND_IMAGE (0x00000800)
358 #define BioAPI_FACTOR_GAIT (0x00001000)
359 #define BioAPI_FACTOR_PASSWORD (0x80000000)
```

360

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

366

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_KEYSTROKE_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

367

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

372

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

377

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

380

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

382

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

385 5.1.1.1.3 DataType

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

387

```
388 DataType ::= ENUMERATED {
389     raw          (0),
390     intermediate (1),
391     processed    (2)
392 }
```

393

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

396

```
397 typedef uint8 BioAPI_BIR_DATA_TYPE;
```

398

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

402

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

408

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

411

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

412

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

415

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

417

418 This markup specifies processed biometric data using an enumerated value.

419 5.1.1.1.4 Purpose

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

421

```
422 Purpose ::= ENUMERATED {
423     verify          (1),
424     identify        (2),
425     enroll          (3),
426     enrollVerify    (4),
427     enrollIdentify  (5),
428     audit           (6),
429
430     ... -- expect others --
431 }
```

432

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

435

```
436 typedef uint8 BioAPI_BIR_PURPOSE;
```

437

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



```

440 #define BioAPI_PURPOSE_ENROLL (3)
441 #define BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY (4)
442 #define BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY (5)
443 #define BioAPI_PURPOSE_AUDIT (6)

```

444
445

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

446

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

449

```

450 <purpose> <audit/> </purpose>

```

451

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

453 5.1.1.1.5 Quality

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

455

```

456 Quality ::= INTEGER {
457     lowest      ( 0),
458     highest     (100),
459     notSet      (-1),
460     notSupported (-2)
461 }
462 (-2..100,...)

```

463

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

466

```

467 typedef sint8 BioAPI_QUALITY;

```

468

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

472

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

473

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

476

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

478

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

480 5.1.1.1.6 ValidityPeriod

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

483

```
484 ValidityPeriod ::= SEQUENCE {
485     notBefore DateTime OPTIONAL,
486     notAfter  DateTime OPTIONAL
487 }
488 (ALL EXCEPT({ -- none; at least one component is present -- })))
```

489

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

492

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

494

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

499

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

502

```
503 <validityPeriod>
504     <notBefore> 1980.10.4 </notBefore>
505     <notAfter> 2003.10.3.23.59.59 </notAfter>
506 </validityPeriod>
```

507

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

510

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

514 5.1.1.1.7 Format

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

516

```
517 Format ::= SEQUENCE {  
518     formatOwner  BIOMETRIC.&name({Owner}),  
519     formatType   BIOMETRIC.&Type({Owner}){@formatOwner})  OPTIONAL  
520 }
```

521

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

524

```
525 BioAPI bioapi_bir_biometric_data_format  
526 typedef struct bioapi_bir_biometric_data_format {  
527     uint16 FormatOwner;  
528     uint16 FormatID;  
529 } BioAPI_BIR_BIOMETRIC_DATA_FORMAT,  
530 *BioAPI_BIR_BIOMETRIC_DATA_FORMAT_PTR;
```

531

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

535

```
536 BIOMETRIC ::= CLASS {  
537     &name  BIOMETRIC-IDENTIFIER  UNIQUE,  
538     &Type  OPTIONAL  
539 }  
540 WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
```

541

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

546

```
547 BIOMETRIC-IDENTIFIER ::= CHOICE {  
548     oid  OBJECT IDENTIFIER,  -- complete object identifier  
549     id   RELATIVE-OID        -- object identifier fragment  
550 }
```

551

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

554

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

557

```
558 <format>  
559   <formatOwner>  
560     <oid> 2.23.42.9.10.4.2 </oid>  
561   </formatOwner>  
562   <formatType>  
563     <URL> http://asn-1.com/biolojava.htm </URL>  
564   </formatType>  
565 </format>
```

566

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

572

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

578

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

582 **5.1.1.1.7.1 Biometric Format Registration**

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

586

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

590

```
591 Owner BIOMETRIC ::= {  
592   CBEFF-Formats | -- http://www.nist.gov/ --  
593   IBIA-Formats | -- http://www.ibia.org/ --  
594   X9-Formats,    -- http://www.x9.org/ --  
595   ... -- expect additional vendor specific formats --  
596 }  
597
```

598

599 **5.1.1.1.7.2 CBEFF-Formats**

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

602

603

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

604

605

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

606

607

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.

608

609

610

611

612

```
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 --
}
```

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

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

632

633

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

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

634

635

636

637

638

639

640

641

642

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

643

644

645

646

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

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

647

648

649

650

651

652

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

657

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

660

```
661 ProcessingAIDs BIOMETRIC ::= {  
662  
663     ... -- Expect CBEFF assignments in BiometricInformationSets --  
664 }
```

665

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

669

```
670 MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation  
671  
672 MatchingInformation ::= SEQUENCE {  
673     id      BIOMETRIC.&name({MatchingAIDs}),  
674     parms  BIOMETRIC.&Type({MatchingAIDs}{@id})  OPTIONAL  
675 }
```

676

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

681

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

684

```
685 MatchingAIDs BIOMETRIC ::= {  
686  
687     ... -- Expect CBEFF assignments in BiometricInformationSets --  
688 }
```

689

690 **5.1.1.1.7.3 IBIA-Formats**

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

692

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

694

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

699

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

701

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

704

```
705 ibia-SAFLINK RelOID ::= { 1 }
706 ibia-Bioscrypt RelOID ::= { 2 }
707 ibia-Visionics RelOID ::= { 3 }
708 ibia-InfineonTechnologiesAG RelOID ::= { 4 }
709 ibia-IridianTechnologies RelOID ::= { 5 }
710 ibia-Veridicom RelOID ::= { 6 }
711 ibia-CyberSIGN RelOID ::= { 7 }
712 ibia-eCryp RelOID ::= { 8 }
713 ibia-FingerprintCardsAB RelOID ::= { 9 }
714 ibia-SecuGen RelOID ::= { 10 }
715 ibia-PreciseBiometric RelOID ::= { 11 }
716 ibia-Identix RelOID ::= { 12 }
717 ibia-DERMALOG RelOID ::= { 13 }
718 ibia-LOGICO RelOID ::= { 14 }
719 ibia-NIST RelOID ::= { 15 }
720 ibia-A4Vision RelOID ::= { 16 }
721 ibia-NEC RelOID ::= { 17 }
722 ibia-STMicroelectronics RelOID ::= { 18 }
723 ibia-Ultra-Scan RelOID ::= { 19 }
724 ibia-Aurora-Wireless RelOID ::= { 20 }
725 ibia-Thales RelOID ::= { 21 }
726 ibia-IBG RelOID ::= { 22 }
727 ibia-Cogent-Systems RelOID ::= { 23 }
728 ibia-Cross-Match RelOID ::= { 24 }
729 ibia-Recognition-Systems RelOID ::= { 25 }
730 ibia-DIN RelOID ::= { 26 }
731 ibia-INCITS-M1 RelOID ::= { 27 }
```

732

733 These identifiers are associated with a restricted sixteen bit integer value.

734

```
735 IBIAidFormats BIOMETRIC ::= {
736 { BIOMETRIC id : ibia-SAFLINK DATA BirInt16 } |
737 { BIOMETRIC id : ibia-Bioscrypt DATA BirInt16 } |
738 { BIOMETRIC id : ibia-Visionics DATA BirInt16 } |
739 { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
740 { BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
741 { BIOMETRIC id : ibia-Veridicom DATA BirInt16 } |
742 { BIOMETRIC id : ibia-CyberSIGN DATA BirInt16 } |
743 { BIOMETRIC id : ibia-eCryp DATA BirInt16 } |
744 { BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
745 { BIOMETRIC id : ibia-SecuGen DATA BirInt16 } |
746 { BIOMETRIC id : ibia-PreciseBiometric DATA BirInt16 } |
747 { BIOMETRIC id : ibia-Identix DATA BirInt16 } |
748 { BIOMETRIC id : ibia-DERMALOG DATA BirInt16 } |
749 { BIOMETRIC id : ibia-LOGICO DATA BirInt16 } |
750 { BIOMETRIC id : ibia-NIST DATA BirInt16 } |
751 { BIOMETRIC id : ibia-A4Vision DATA BirInt16 } |
752 { BIOMETRIC id : ibia-NEC DATA BirInt16 } |
753 { BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 } |
754 { BIOMETRIC id : ibia-Ultra-Scan DATA BirInt16 } |
755 { BIOMETRIC id : ibia-Aurora-Wireless DATA BirInt16 } |
756 { BIOMETRIC id : ibia-Thales DATA BirInt16 } |
```

```

757 { BIOMETRIC id : ibia-IBG DATA BirInt16 } |
758 { BIOMETRIC id : ibia-Cogent-Systems DATA BirInt16 } |
759 { BIOMETRIC id : ibia-Cross-Match DATA BirInt16 } |
760 { BIOMETRIC id : ibia-Recognition-Systems DATA BirInt16 } |
761 { BIOMETRIC id : ibia-DIN DATA BirInt16 } |
762 { BIOMETRIC id : ibia-INCITS-M1 DATA BirInt16 },
763
764 ... -- Expect others --
765 }

```

766

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

769

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

773

```

774 IBIAoidFormats BIOMETRIC ::= {
775 { BIOMETRIC oid : id-ibia-SAFLINK DATA Any } |
776 { BIOMETRIC oid : id-ibia-Bioscrypt DATA Any } |
777 { BIOMETRIC oid : id-ibia-Visionics DATA Any } |
778 { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
779 { BIOMETRIC oid : id-ibia-IridianTechnologies DATA Any } |
780 { BIOMETRIC oid : id-ibia-Veridicom DATA Any } |
781 { BIOMETRIC oid : id-ibia-CyberSIGN DATA Any } |
782 { BIOMETRIC oid : id-ibia-eCryp DATA Any } |
783 { BIOMETRIC oid : id-ibia-FingerprintCardsAB DATA Any } |
784 { BIOMETRIC oid : id-ibia-SecuGen DATA Any } |
785 { BIOMETRIC oid : id-ibia-PreciseBiometric DATA Any } |
786 { BIOMETRIC oid : id-ibia-Identix DATA Any } |
787 { BIOMETRIC oid : id-ibia-DERMALOG DATA Any } |
788 { BIOMETRIC oid : id-ibia-LOGICO DATA Any } |
789 { BIOMETRIC oid : id-ibia-NIST DATA Any } |
790 { BIOMETRIC oid : id-ibia-A4Vision DATA Any } |
791 { BIOMETRIC oid : id-ibia-NEC DATA Any } |
792 { BIOMETRIC oid : id-ibia-STMicroelectronics DATA Any } |
793 { BIOMETRIC oid : id-ibia-Ultra-Scan DATA Any } |
794 { BIOMETRIC oid : id-ibia-Aurora-Wireless DATA Any } |
795 { BIOMETRIC oid : id-ibia-Thales DATA Any } |
796 { BIOMETRIC oid : id-ibia-IBG DATA Any } |
797 { BIOMETRIC oid : id-ibia-Cogent-Systems DATA Any } |
798 { BIOMETRIC oid : id-ibia-Cross-Match DATA Any } |
799 { BIOMETRIC oid : id-ibia-Recognition-Systems DATA Any } |
800 { BIOMETRIC oid : id-ibia-DIN DATA Any } |
801 { BIOMETRIC oid : id-ibia-INCITS-M1 DATA Any },
802
803 ... -- Expect additional vendor specific formats --
804 }
805
806 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
807

```

808

809 **5.1.1.1.7.4 X9-Formats**

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

811


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

813

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

820

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

827

828

```
829 X9-Formats BIOMETRIC ::= {  
830   X9oidFormats |  
831   X9idFormats,  
832  
833   ... -- Expect additional X9 vendor specific formats --  
834 }  
835  
836 X9oidFormats BIOMETRIC ::= {  
837   ... -- Expect X9 assigned objects --  
838 }  
839  
840 X9idFormats BIOMETRIC ::= {  
841   ... -- Expect X9 assigned objects of the form { 2 x } --  
842 }
```

843

844 5.1.1.2 BiometricData

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

847

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

849

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

852

```
853 typedef uint8 BioAPI_BIR_BIOMETRIC_DATA;
```

854

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

857

858 5.1.2 IntegrityObjects

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

862

```
863 IntegrityObjects ::= SEQUENCE {  
864     biometricObjects  EncodedBiometricObjects,  
865     integrityBlock    IntegrityBlock  
866 }
```

867

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

872

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

876

```
877 IntegrityBlock ::= CHOICE {  
878     digitalSignature      DigitalSignature,  
879     messageAuthenticationCode  MessageAuthenticationCode,  
880     signedData           SignedData,  
881     authenticatedData    AuthenticatedData  
882 }
```

883

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

885

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

886

887 5.1.2.1 DigitalSignature

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

891

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

897

```
}
```

898

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

902

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

905

906

```
<integrityBlock>
907   <digitalSignature>
908     <algorithmID>
909       <algorithm>1.2.840.10040.4.3</algorithm>
910       <parameters><NullParms/></parameters>
911     </algorithmID>
912     <signature>
913       DE340 ... B0123DF
914     </signature>
915   </digitalSignature>
916 </integrityBlock>
```

917

918

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

925 5.1.2.1.1 Digital Signature Process

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

930

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

933

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

940

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

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

948

949 5.1.2.1.2 Digital Signature Verification

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

957 5.1.2.2 MessageAuthenticationCode

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

962

```
963 MessageAuthenticationCode ::= SEQUENCE {  
964     keyName      OCTET STRING OPTIONAL,  
965     algorithmID  MACAlgorithmIdentifier,  
966     mac          OCTET STRING  
967                 ( CONSTRAINED BY { -- MAC or HMAC on a value of --  
968                                     EncodedBiometricObjects })  
969 }
```

970

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

976

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

982

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

985

```
986 <integrityBlock>  
987   <messageAuthenticationCode>  
988     <keyName> 9FCD...AB45 </keyName>  
989     <algorithmID>  
990       <algorithm>1.3.6.1.5.5.8.1.2</algorithm>  
991     </algorithmID>  
992     <mac>  
993       DEA7B ... 59ABD3
```

994
995
996

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

997

998 This markup uses the message authentication code choice alternative. The hashed MAC with
999 SHA1 algorithm and a shared secret key are used to compute an HMAC on a value of
1000 **EncodedBiometricObjects**. An ellipsis is used as a placeholder for part of the HMAC results.

1001 **5.1.2.2.1 Message Authentication Process**

1002 A sender prepares a value of type **EncodedBiometricObjects**, a named cryptographic key
1003 previously created at random and known by the recipient, and uses these as input to a MAC or
1004 HMAC process. This results in a message authentication code over the specified biometric
1005 information. The biometric information and processing results are sent to a recipient who shares
1006 the secret key used in the message authentication code process.

1007

1008 To verify the message authentication code, the user computes a MAC or HMAC on the biometric
1009 information using the same shared secret key identified by its key name, and compares this result
1010 to the message authentication value received to determine the integrity of the biometric
1011 information.

1012 **5.1.2.3 SignedData**

1013 The **signedData** choice alternative of the **integrityBlock** component of type **IntegrityObjects** is
1014 a value of type **SignedData**. This sequence type is defined as

1015

1016
1017
1018
1019
1020
1021
1022
1023

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

1024

1025 The components of type **SignedData** have the following meanings:

1026

- | | |
|-------------------------|---|
| version | An integer version number of the syntax definition. The version shall be 84 in this standard. |
| digestAlgorithms | The set of message digest algorithms used by the signers. This set contains only one element since there is only one signer of the content in this standard. |
| encapContentInfo | An identifier of the type of content signed and optionally, the signed content. In this standard the signed content is not present. The type of content is always ordinary data as the nesting of cryptographic types is neither required nor supported. |
| certificates | An optional set of one or more X.509 [1] or X9.68 [11] certificates. When present, this set may contain more certificates or less than needed to verify the signature on the signed data. This component shall be encoded as specified in 7.4.2 Octet Strings with Certificates and Certificate |

Revocation Lists.

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

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

1027

1028

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

1029

1030

1031

1032

1033

1034

1035

1036

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

1037

1038 The **SignerIdentifier** type is used to identify the public key certificate associated with the private
1039 key used to create the **signature** component of **SignerInfo**. This type is defined in XCBF as a
1040 choice type having only one alternative:

1041

1042

1043

1044

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

1045

1046 The **certHash** choice alternative of **SignerIdentifier** provides a single, simple mechanism that
1047 allows any type of digital certificate to be identified. The hash is computed over the complete
1048 encoding of the certificate. This allows any type of certificate, regardless of its format or encoding,
1049 to be identified.

1050

1051 When X.509 certificates or attribute certificates are used, the hash will be computed over the
1052 complete DER [6] encoding of the certificate, as X.509 only supports these encoding rules. When
1053 X9.68 domain certificates are used, the hash may be computed over the DER or XER encoding
1054 of the certificate, as either encoding format is supported by that standard. In any case, such
1055 details are hidden from an application if the certificates is properly treated as an opaque series of
1056 octets.

1057

1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068

```
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
}
```

1069

1070 **5.1.2.3.1 Message digest and signature process**

1071 A message digest is used to create the digital signature carried in the **SignerInfo** component of
1072 **SignedData**. The message digest is calculated using the algorithm and parameters specified in
1073 the **digestAlgorithm** component of **SignerInfo**, and the value of the **eContent** component of
1074 **EncapsulatedContentInfo**. The **eContentType** component of **EncapsulatedContentInfo**
1075 identifies the type of value being signed. This is always the object identifier value **id-data** in
1076 XCBF.

1077

1078 When a value of type **SignedData** is represented as XML markup, the starting and ending
1079 **eContent** tags are excluded from the message digest process. Only the "value" portion of the
1080 complete canonical XER encoding of **eContent** is digested. The **<eContent>** and **</eContent>**
1081 tags are excluded from the message digest process, and the digest is calculated starting with the
1082 **<BiometricObjects>** tag and ending with the **</BiometricObjects>** tag.

1083

1084 The result of the message digest process is then digitally signed using the signer's private key
1085 and the signature algorithm and parameters specified in the **signatureAlgorithm** component of
1086 **SignerInfo**. The result of the signature process becomes the value of the **signature** component
1087 of the **SignerInfo** component of **SignedData**.

1088

1089 **5.1.2.3.2 Signature Verification**

1090 To verify the signature in a signed data choice alternative of the **integrityBlock** component of
1091 type **IntegrityObjects**, a message digest is computed over the complete canonical XER
1092 encoding of the value of the **biometricObjects** component of type **IntegrityObjects**. This digest
1093 is computed using the **digestAlgorithm** component of type **SignerInfo**.

1094

1095 The public key of the signer is the signature verification key. This key is used to verify the digital
1096 signature on the signed data created with the signer's private key. The signature is carried in the
1097 **SignerInfo** component of type **SignedData**. The value of the **sid** component of **SignerInfo**
1098 identifies the public key certificate associated with the private key used to create the digital
1099 signature on the signed data.

1100

1101 The message digest value computed by the signature verifier is compared to the value of the
1102 hash obtained from applying the signer's public key to the **signature** component of type
1103 **SignerInfo** to determine if this signature is valid. There is only one signer of the content in this
1104 standard.

1105

1106 Complete trust in the validity of the signature on the signed data by the signer must be
1107 determined by validation of the chain of certificates associated with the signer's public key
1108 certificate. The optional **certificates** and **crls** components of type **SignedData** may be used by
1109 the signer to provide information needed to validate the signer's certificate.

1110 **5.1.2.4 AuthenticatedData**

1111 The **messageAuthenticationCode** choice alternative of the **integrityBlock** component of type
1112 **IntegrityObjects** is a value of type **AuthenticatedData**. This sequence type provides a MAC with
1113 key establishment and is defined as

1114

```
1115 AuthenticatedData ::= SEQUENCE {  
1116     version          CMSVersion,  
1117     recipientInfos   RecipientInfos,  
1118     macAlgorithm     MACAlgorithmIdentifier,  
1119     encapContentInfo EncapsulatedContentInfo,  
1120     mac              MessageAuthenticationCode  
1121 }
```

1122

1123 Type **AuthenticatedData** uses a key transport mechanism to convey a one-time MAC key along
1124 with biometric information. Since a key agreement mechanism is not supported, only data
1125 integrity is provided by use of this type and not origin authentication. There is only a single
1126 recipient of this encrypted MAC key present in the **recipientInfos** component of type
1127 **AuthenticatedData**, and the optional content is not present in the **encapContentInfo**
1128 component.

1129

1130 A message authentication code is calculated on biometric information. This biometric information
1131 is a value of type **EncodedBiometricObjects**, which is a value of type **BiometricObjects** in its
1132 encoded form. The calculation is performed using the MAC algorithm and any associated
1133 algorithm parameters indicated in the **macAlgorithm** component of type **AuthenticatedData**,
1134 the biometric information, and an authentication key that is conveyed in the **recipientInfos**
1135 component. The result of this calculation is a message authentication code, which becomes the
1136 value of the **mac** component.

1137

1138 A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented
1139 in XML markup as

1140

```
1141 <privacyObjects>  
1142   <privacyBlock>  
1143     ...  
1144   </privacyBlock>  
1145 </privacyObjects>
```

1148

1149 This markup illustrates the wrapper for a typical privacy object. The optional privacy object
1150 biometric headers are not present. An ellipsis is used as a placeholder and the details of the
1151 privacy block choice alternative are not shown.

1152

1153 5.1.2.5 Biometric Certificate Extensions

1154 Digital signature data integrity protection and origin authentication can be achieved by including
1155 biometric information in digital certificates. A set of one or more biometric reference templates
1156 can be cryptographically bound to the public key associated with the private key of an entity, by
1157 including a value of type **EncodedBiometricObjects** in a certificate extension. In the XCBF
1158 standard, biometric certificate extension values can be encoded in either BER or BASIC-XER
1159 depending on the outer-level encoding (see 7.4.3 Outer-level encodings).

1160

1161 XCBF supports the version three certificates and version two attribute certificates defined in the
1162 X.509 standard and the compact domain certificate format defined in X9.68. One extension is
1163 defined for use in each standard, and these are defined as

1164

```
1165 biometricTemplates EXTENSION ::= {  
1166     SYNTAX          EncodedBiometricObjects  -- CXER --  
1167     IDENTIFIED BY  x509-biometricTemplates  
1168 }  
1169  
1170 domainBiometricTemplates PRIVATE-X ::= {  
1171     NAME    oid : x968-biometricTemplates  
1172     TYPE   EncodedBiometricObjects  -- CXER --  
1173 }
```

1174

1175 When biometric information is included in a certificate extension stored in a certificate repository,
1176 the repository becomes a biometric storage subsystem, and the biometric information may need
1177 to be protected by encryption or other means. Measures should be taken to prevent an attacker
1178 from using a certificate repository as a large, searchable public database of biometric reference
1179 templates that could be used to find templates that match a given biometric sample. Finding such
1180 a match would allow an attacker to focus its efforts on that user.

1181 5.1.3 PrivacyObjects

1182 The **privacyObjects** choice alternative of type **BiometricSyntax** is a value of type
1183 **PrivacyObjects**. This type is defined as a sequence of two components, **biometricHeaders** and
1184 **privacyBlock**.

1185

```
1186 PrivacyObjects ::= SEQUENCE {  
1187     biometricHeaders  BiometricHeaders  OPTIONAL,  
1188     privacyBlock      PrivacyBlock  
1189 }
```

1190

1191 The **biometricHeaders** component is a series of one or more values of type **BiometricHeader**.

1192

```
1193 BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
```

1194

1195 This optional **biometricHeaders** component is not protected by encryption and should be present
1196 only when a privacy object is used in a secure environment, or when the information contained in
1197 the **biometricHeaders** component does not compromise security or assist an attacker. In a
1198 secure setting these biometric headers may be used as a convenience, to assist in searches of
1199 biometric information and in database management operations.

1200

1201 The encrypted content in the privacy block contains a series of one or more values of type
1202 **BiometricObject**, including their biometric headers. To be useful, the **biometricHeaders**
1203 component need only provide an indication of the information contained in the encrypted privacy
1204 block. But this component need not contain exactly the same information as the headers in the
1205 encrypted privacy block, and may contain only a single **BiometricHeader** value when present.
1206 The **biometricHeaders** component is not protected in any way.

1207

1208 The **privacyBlock** component of type **PrivacyObjects** offers three choice alternatives, **fixedKey**,
1209 **namedKey** and **establishedKey**.

1210

```
1211 PrivacyBlock ::= CHOICE {  
1212     fixedKey      EncryptedData,  
1213     namedKey     NamedKeyEncryptedData,  
1214     establishedKey EnvelopedData  
1215 }
```

1216

1217 The **fixedKey** and **namedKey** choice alternatives are based on the **EncryptedData** type. The
1218 **establishedKey** alternative is based on type **EnvelopedData**. Each of these alternatives has
1219 different characteristics, and the alternative chosen will depend upon application requirements
1220 and the key management scheme being used.

1221

1222 A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented
1223 in XML markup as

1224

```
1225 <privacyObjects>  
1226     <privacyBlock>  
1227         ...  
1228     </privacyBlock>  
1229 </privacyObjects>
```

1232

1233 This markup illustrates the wrapper for a typical privacy object. The optional privacy object
1234 biometric headers are not present. An ellipsis is used as a placeholder and the details of the
1235 privacy block choice alternative are not shown.

1236 5.1.3.1 Encrypted Content Information

1237 All three of the privacy block choice alternatives contain a value of type **EncryptedContentInfo**
1238 defined as

1239

```
1240 EncryptedContentInfo ::= SEQUENCE {  
1241     contentType      ContentType,  
1242     contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,  
1243     encryptedContent [0] EncryptedContent  
1244 }
```

1245

1246 The **contentType** component identifies the type of encrypted content. In XCBF, the type of
1247 encrypted content is always a value of **EncodedBiometricObjects**, a series of one or more
1248 values of type **BiometricObject** encoded using CXER. The type of encrypted content is identified

1249 as ordinary data by the information object identifier value **id-data**, defined in the PKCS #7
1250 Cryptographic Message Syntax Standard [19]. The encoding of this component is

1251

```
1252 <contentType> 1.2.840.113549.1.7.1 </contentType>
```

1253

1254 The **contentEncryptionAlgorithm** component identifies the content encryption algorithm and
1255 any associated parameters used to encrypt and decrypt the **EncodedBiometricObjects**. This
1256 content encryption algorithm is a value of type **ContentEncryptionAlgorithmIdentifier** defined
1257 as

1258

```
1259 ContentEncryptionAlgorithmIdentifier ::=  
1260     AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
```

1261

1262 The definition of type **ContentEncryptionAlgorithmIdentifier** is based on the parameterized
1263 type **AlgorithmIdentifier** {} and the information object set **ContentEncryptionAlgorithms**,
1264 defined as

1265

```
1266 ContentEncryptionAlgorithms ALGORITHM ::= {  
1267     { OID des-ede3-cbc PARMS IV },  
1268     ... -- Expect other content encryption algorithms --  
1269 }  
1270  
1271 IV ::= OCTET STRING (SIZE(8))
```

1272

1273 **ContentEncryptionAlgorithms** specifies an extensible set of **ALGORITHM** information objects.
1274 The fields of these information objects are used to constrain the valid values of the components
1275 of type **ContentEncryptionAlgorithmIdentifier**. Though only one content encryption algorithm
1276 object is defined explicitly in this set, implementations should expect additional algorithms.

1277

1278 The **ContentEncryptionAlgorithms** information object set contains a single object that identifies
1279 the encryption algorithm described in ANSI X9.52 [10] as Triple DES (TDES) in CBC (cipher
1280 block chaining) mode. Only the two key and three key variants of TDES are supported in XCBF.
1281 The single key variant of TDES is simply the DES algorithm and is generally used only for
1282 backwards compatibility with existing DES based applications and is considered vulnerable to
1283 attack.

1284

1285 The Triple DES algorithm consists of three sequential DES operations, encrypt, decrypt, and
1286 encrypt. For three key TDES a different key is used for each DES operation. For two key TDES
1287 one key is used for both DES encrypt operations, and the second key is used for the DES decrypt
1288 operation.

1289

1290 The **encryptedContent** component contains a value of type **EncodedBiometricObjects**
1291 encrypted using the content encryption algorithm given in the **contentEncryptionAlgorithm**
1292 component. A value of **encryptedContent** is an opaque string of octets treated as having no
1293 discernable structure. This string is a value of type **EncryptedContent** defined as

1294

```
1295 EncryptedContent ::= OCTET STRING
```

1296

1297 A value of the **encryptedContentInfo** component of any of the privacy block choice alternative
1298 types can be represented in XML markup as

1299

```

1300 <encryptedContentInfo>
1301   <contentType>1.2.840.113549.1.7.1</contentType>
1302   <contentEncryptionAlgorithm>
1303     <algorithm>1.2.840.113549.3.7</algorithm>
1304     <parameters>
1305       <IV>7EA13D6E143CB5C9</IV>
1306     </parameters>
1307   </contentEncryptionAlgorithm>
1308   <encryptedContent>
1309     D8F6 ... F766
1310   </encryptedContent>
1311 </encryptedContentInfo>

```

1312

1313 This markup illustrates a typical value. The encrypted content type is identified as ordinary data.
1314 The Triple DES content encryption algorithm is identified along with its associated parameters, an
1315 initialization vector, <IV>. An ellipsis is used as a placeholder for part of the encrypted content.

1316 5.1.3.2 Fixed Key EncryptedData

1317 The **fixedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1318 value of type **EncryptedData**. This type is a sequence of two components, an integer version
1319 number and a value of type **EncryptedContentInfo**. Type **EncryptedData** is defined as

1320

```

1321 EncryptedData ::= SEQUENCE {
1322   version          CMSVersion,
1323   encryptedContentInfo  EncryptedContentInfo
1324 }

```

1325

1326 The **fixedKey** alternative assumes that the recipient of the **EncryptedData** value knows the key
1327 used to encrypt the biometric information, perhaps by prior agreement or as the result of a key
1328 exchange. The **version** component of type **EncryptedData** is always the integer value eighty-
1329 four. The components of type **EncryptedContentInfo** are described in section 4.1.3.1 *Encrypted*
1330 *Content Information*.

1331

1332 A value of the **fixedKey** choice alternative of the **privacyBlock** component of type
1333 **PrivacyObjects** can be represented in XML markup as

1334

```

1335 <fixedKey>
1336   <version>84</version>
1337   <encryptedContentInfo>
1338     <contentType>1.2.840.113549.1.7.1</contentType>
1339     <contentEncryptionAlgorithm>
1340       <algorithm>1.2.840.113549.3.7</algorithm>
1341       <parameters>
1342         <IV>7EA13D6E143CB5C9</IV>
1343       </parameters>
1344     </contentEncryptionAlgorithm>
1345     <encryptedContent>
1346     ...

```

1347
1348
1349

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

1350

1351 This markup uses the fixed key choice alternative of the privacy block, a value of version number
1352 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
1353 identified as ordinary data. The Triple DES content encryption algorithm is identified along with its
1354 associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder and the
1355 encrypted content is not shown.

1356 5.1.3.2.1 Encryption Process

1357 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1358 **BiometricObject** in their encoded form using a content encryption algorithm and a fixed content
1359 encryption key known to the sender and recipient. The content to be encrypted is a value of type
1360 **EncodedBiometricObjects**. This value is always encoded using CXER. The content encryption
1361 algorithm used to encrypt the biometric objects is one of the algorithms specified in the
1362 information object set **ContentEncryptionAlgorithms**.

1363

1364 The **contentType** component of type **EncryptedContentInfo** is set to indicate ordinary data. The
1365 associated **contentEncryptionAlgorithm** value is set to identify the algorithm used to encrypt
1366 the content, and the **encryptedContent** value is set to the results of encrypting the content using
1367 this content encryption algorithm.

1368 5.1.3.2.2 Decryption Process

1369 To decrypt a value of type **EncryptedData**, the content encryption algorithm specified in the
1370 **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1371 associated **encryptedContent** component using a known fixed key to recover a value of type
1372 **EncodedBiometricObjects**. This recovered value will contain one or more values of type
1373 **BiometricObject** encoded using CXER.

1374 5.1.3.3 Named Key EncryptedData

1375 The **namedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1376 value of type **NamedKeyEncryptedData**. This type is sequence with two components, **keyName**
1377 and **encryptedData**. Type **NamedKeyEncryptedData** is defined as

1378

1379
1380
1381
1382

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

1383

1384 The **keyName** component explicitly identifies the key used to encrypt and decrypt the content by
1385 name. The **encryptedData** component is a value of type **EncryptedData**. This type contains two
1386 components, an integer **version** number that is always eighty-four in this standard, and an
1387 **encryptedContentInfo** that is a value of type **EncryptedContentInfo** as described in section
1388 4.1.3.1 *Encrypted Content Information*.

1389

1390 A value of the **namedKey** choice alternative of the **privacyBlock** component of type
1391 **PrivacyObjects** can be represented in XML markup as

1392

1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410

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

1411
1412
1413
1414
1415
1416

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.

1417 **5.1.3.3.1 Encryption Process**

1418 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1419 **BiometricObject** in their encoded form using a content encryption algorithm and a named key
1420 that is known to the recipient of the encrypted biometric information. The content to be encrypted
1421 is a value of type **EncodedBiometricObjects**. This value is always encoded using CXER. The
1422 content encryption algorithm used to encrypt the biometric objects is one of the algorithms
1423 specified in the information object set **ContentEncryptionAlgorithms**.

1424

1425 The **keyName** component of type **NamedKeyEncryptedData** is set to the name of the content
1426 encryption key. The **contentType** component of type **EncryptedContentInfo** is set to indicate
1427 ordinary data. The associated **contentEncryptionAlgorithm** value is set to identify the algorithm
1428 used to encrypt the content, and the **encryptedContent** value is set to the results of encrypting
1429 the content using this content encryption algorithm.

1430 **5.1.3.3.2 Decryption Process**

1431 To decrypt a value of type **NamedKeyEncryptedData**, the content encryption algorithm specified
1432 in the **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1433 associated **encryptedContent** component using the key identified by the **keyName** component
1434 of type **NamedKeyEncryptedData** to recover a value of type **EncodedBiometricObjects**. This
1435 recovered value will contain one or more values of type **BiometricObject** encoded using CXER.

1436 **5.1.3.4 Established Key EnvelopedData**

1437 The **establishedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects**
1438 is a value of type **EnvelopedData**. Using this type, a message sender can encrypt content that
1439 only the intended message recipient can decrypt.

1440

1441 **EnvelopedData** is defined as a sequence of four components, an integer version number,
1442 message sender information, message recipient information, and a value of type

1443 **EncryptedContentInfo** which is described in section 4.1.3.1 *Encrypted Content Information*.
1444 Type **EnvelopedData** is defined as

1445

```
1446 EnvelopedData ::= SEQUENCE {  
1447     version          CMSVersion,  
1448     originatorInfo   [0] OriginatorInfo OPTIONAL,  
1449     recipientInfos   RecipientInfos,  
1450     encryptedContentInfo EncryptedContentInfo  
1451 }
```

1452

1453 The combination of encrypted content and an encrypted content encryption key forms a “digital
1454 envelope”. The **establishedKey** alternative uses a randomly generated content encryption key to
1455 encrypt digital content. The same key is used to decrypt the content. The content encryption key
1456 shall be protected during transport, so the recipient’s public and private key pair is used to
1457 encrypt and decrypt the content encryption key.

1458

1459 The encrypted content is value of type **EncodedBiometricObjects**. This type is a series of one
1460 or more values of type **BiometricObject** in their encoded form. In XCBF these values are
1461 encoded using CXER.

1462

1463 The **version** component of type **EnvelopedData** is the integer value eighty-four. The optional
1464 **originatorInfo** component facilitates distribution of digital certificates and certificate revocation
1465 lists. The **recipientInfos** component contains information needed to recover the encrypted
1466 content encryption key used to encrypt the biometric information. The **encryptedContentInfo**
1467 component is a value of type **EncryptedContentInfo**. This type is described in section 4.1.3.1
1468 *Encrypted Content Information*.

1469

1470 A value of the **establishedKey** choice alternative of the **privacyBlock** component of type
1471 **PrivacyObjects** can be represented in XML markup as

1472

```
1473 <establishedKey>  
1474   <version>84</version>  
1475   <recipientInfos>  
1476     <RecipientInfo>  
1477       <ktri>  
1478         <version>84</version>  
1479         <rid>  
1480           <certHash>  
1481             <ietf>6E143CF31A562FA9492681D27A22013D2AAD435D</ietf>  
1482           </certHash>  
1483         </rid>  
1484         <keyEncryptionAlgorithm>  
1485           <algorithm>  
1486             1.2.840.113549.1.1.1  
1487           </algorithm>  
1488           <parameters><NullParms/></parameters>  
1489         </keyEncryptionAlgorithm>  
1490         <encryptedKey>  
1491           ...  
1492         </encryptedKey>  
1493       </ktri>  
1494     </RecipientInfo>  
1495   </recipientInfos>  
1496   <encryptedContentInfo>
```


1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508

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

1509

1510 This markup uses the established key choice alternative of the privacy block, a value of version
1511 number eighty-four of the cryptographic message type **EnvelopedData**. The optional originator
1512 information is not present. The recipient information uses the key transport choice alternative.
1513 The recipient public and private key pair is indicated by a SHA1 hash of a public key certificate.
1514 The content encryption key is encrypted using the **rsaEncryption** algorithm, which has no
1515 associated parameters indicating that no initialization vector is required. The encrypted content
1516 type is identified as ordinary data. The Triple DES content encryption algorithm is identified along
1517 with its associated parameters, an initialization vector, **<IV>**. An ellipsis is used as a placeholder
1518 and the encrypted content encryption key and the encrypted content are not shown.

1519

1520 5.1.3.4.1 Certificates and CRLs

1521 Type **OriginatorInfo** is a sequence of two components that may contain sets of digital certificates
1522 and certificate revocation lists (CRLs). This type is defined as

1523

1524
1525
1526
1527
1528

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

1529

1530 Any combination of X9.68 domain certificates, X.509 certificates and attribute certificates may be
1531 included in the **CertificateSet** type in any order. There may be more or fewer certificates needed
1532 for any purpose. Certificates are provided as needed to support content key encryption in the key
1533 transport key management technique used in XCBF. Use of the **CertificateSet** type to distribute
1534 certificates is not required. They may be obtained by other means, or an online certificate
1535 validation service may be used instead. Only version one X9.68 domain certificates, version three
1536 X.509 certificates and version two attribute certificates are supported in this standard. The value
1537 of the **CertificateSet** octet string shall be the concatenation of the DER encodings of one or more
1538 X.509 certificate and attribute certificate types. The encoding of this octet string is specified in
1539 7.4.2: Octet Strings with Certificates and Certificate Revocation Lists.

1540

1541 Any number of CRLs may be included in the **CertificateRevocationLists** type in any order.
1542 There may be more or fewer CRLs needed for any purpose. CRLs are provided as needed to
1543 support certificate validation. Use of the **CertificateRevocationLists** type to distribute CRLs is
1544 not required. CRLs may be obtained by other means, or an online certificate validation service
1545 may be used instead. Only version two certificate revocation lists are supported in this standard.
1546 The value of the **CertificateRevocationList** octet string shall be the concatenation of the DER
1547 encodings of one or more X.509 CRLs. The encoding of this octet string is specified in 7.4.2:
1548 Octet Strings with Certificates and Certificate Revocation Lists.

1549

1550 The X.509 certificates and certificate revocation lists used in XCBF are signed binary objects,
1551 whose digital signatures have been calculated on values encoded using the Distinguished
1552 Encoding Rules (DER) of ASN.1. In order to verify the signatures on these objects, their original
1553 encodings must be used. This is not affected by any representation (such as a base 64
1554 encoding) used for transfer (see 7.4.2: Octet Strings with Certificates and Certificate Revocation
1555 Lists).

1556 5.1.3.4.2 Recipient Information

1557 Type **RecipientInfos** is a series of values of type **RecipientInfo**, one value for each recipient of a
1558 digital envelope in **EnvelopedData**. In XCBF there is always a single digital envelope recipient,
1559 and type **RecipientInfos** is constrained to a series of one **RecipientInfo** and defined as

1560

```
1561 RecipientInfos ::= SET SIZE(1) OF RecipientInfo
```

1562

1563 Several key management techniques can be used in **EnvelopedData**. In XCBF, only key
1564 transport is supported. Other techniques such as constructive key management may be
1565 employed by an application, but such use is not defined in this standard. Type **RecipientInfo** is
1566 restricted to a single choice alternative and defined as

1567

```
1568 RecipientInfo ::= CHOICE {  
1569     ktri KeyTransRecipientInfo  
1570 }
```

1571

1572 Key transport information is provided to the recipient of a digital envelope so that the envelope
1573 can be opened and the protected content encryption key recovered. The content encryption key
1574 may then be used to decrypt the content.

1575

1576 The information needed by the recipient to recover the content encryption key is contained in a
1577 value of type **KeyTransRecipientInfo** defined as

1578

```
1579 KeyTransRecipientInfo ::= SEQUENCE {  
1580     version          CMSVersion,  
1581     rid              RecipientIdentifier,  
1582     keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,  
1583     encryptedKey     EncryptedKey  
1584 }
```

1585

1586 This type is a sequence of four components. The integer **version** number is always set to eighty-
1587 four in XCBF. The **rid** component is used to identify the public key used to encrypt the content
1588 encryption key. This public key is bound to a key encryption algorithm in a public key certificate. It
1589 is associated with the recipient private key needed to decrypt the content encryption key used by
1590 the sender to encrypt the content.

1591

1592 A hash of the public key certificate uniquely identifies the recipient certificate.

1593

```
1594 RecipientIdentifier ::= CHOICE {  
1595     certHash [73] EXPLICIT Hash
```

1596

```
}
```

1597

1598 The **keyEncryptionAlgorithm** component identifies the key encryption algorithm and any
1599 associated parameters used to encrypt the content encryption key.

1600

1601

```
KeyEncryptionAlgorithmIdentifier ::=  
    AlgorithmIdentifier {{KeyEncryptionAlgorithms}}  
  
KeyEncryptionAlgorithms ALGORITHM ::= {  
    { OID rsaEncryption PARMS NullParms },  
    ... -- expect other key encryption algorithms --  
}
```

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

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

1612

1613

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

1614

1615

5.1.3.4.3 Digital Envelope Processing

1616

1617

1618

1619

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.

1620

1621

1622

1623

1624

1625

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

1626

1627

1628

1629

1630

1631

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

1632

1633

1634

1635

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

1636

1637

1638

1639

1640

1641

1642

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**.

1643 5.1.4 PrivacyAndIntegrityObjects

1644 The **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** is a value of type
1645 **PrivacyAndIntegrityObjects**. This type is defined as a sequence of three components, optional
1646 **biometricHeaders**, a **privacyBlock**, and an **integrityBlock**.

1647

```
1648 PrivacyAndIntegrityObjects ::= SEQUENCE {  
1649     biometricHeaders BiometricHeaders OPTIONAL,  
1650     privacyBlock     PrivacyBlock,  
1651     integrityBlock   IntegrityBlock  
1652 }
```

1653

1654 The **biometricHeaders** component is optional and is composed of a series of one or more values
1655 of type **BiometricHeader**. The **privacyBlock** component is a value of type **PrivacyBlock**. The
1656 **BiometricHeader** and **PrivacyBlock** types are described in section 4.1.3. *PrivacyObjects*. The
1657 **integrityBlock** component is a value of type **IntegrityBlock**. This type is described in section
1658 4.1.2. *IntegrityObjects*.

1659

1660 The input to all cryptographic process is a value of type **EncodedBiometricObjects**, a series of
1661 one or more values of type **BiometricObject** in their encoded form using CXER. The order of the
1662 components in type **PrivacyAndIntegrityObjects** facilitates one pass processing for both sender
1663 and recipient.

1664

1665 For the sender, a value of type **EncodedBiometricObjects** is created and then used as input to
1666 the cryptographic processing of both the **privacyBlock** and **integrityBlock** components. For the
1667 recipient, the **privacyBlock** component is first processed to recover the encrypted content, a
1668 value of type **EncodedBiometricObjects**. This recovered value is then used to validate the
1669 signature in the **integrityBlock** component.

1670

1671 A value of the **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** can be
1672 represented in XML markup as

1673

```
1674 <privacyAndIntegrityObjects>  
1675   <privacyBlock>  
1676     <namedKey>  
1677       <keyName>6AE173BF5A973D1E</keyName>  
1678       <encryptedData>  
1679         <version>84</version>  
1680         <encryptedContentInfo>  
1681           <contentType>1.2.840.113549.1.7.1</contentType>  
1682           <contentEncryptionAlgorithm>  
1683             <algorithm>1.2.840.113549.3.7</algorithm>  
1684             <parameters>  
1685               <IV>7EA13D6E143CB5C9</IV>  
1686             </parameters>  
1687           </contentEncryptionAlgorithm>  
1688           <encryptedContent>  
1689             ...  
1690           </encryptedContent>  
1691         </encryptedContentInfo>  
1692       </encryptedData>  
1693     </namedKey>  
1694   </privacyBlock>  
1695   <integrityBlock>
```

1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706

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

1707

1708 This markup combines a privacy block and integrity block. The named key choice alternative of
1709 the privacy block and the digital signature choice alternative of the integrity block are used. The
1710 optional biometric headers are not present.

1711

1712 The named key alternative is a sequence containing a key name and a value of version number
1713 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
1714 identified as ordinary data, and is computed on a value of type **EncodedBiometricObjects**. The
1715 Triple DES content encryption algorithm is identified along with its associated parameters, an
1716 initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not
1717 shown.

1718

1719 The digital signature choice alternative is a simple digital signature on a value of type
1720 **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) with Secure Hash Algorithm
1721 (SHA1) and its associated parameters, <NullParms/> are used for signing a value of
1722 **EncodedBiometricObjects**. An ellipsis is used as a placeholder and the signature is not shown.

1723

6 References

1724

6.1 Normative

1725

1726 These references or an understanding of the materials within them are required to implement this
1727 XCBF standard. They are intended to include any amendments and technical corrigenda issued
1728 after their publication. Users of this standard are encouraged to seek the latest versions of these
1729 referenced materials before initiating any serious work.

1730

1731 1 ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open
1732 Systems Interconnection -- The Directory: Authentication framework.

1733 2 ISO/IEC 8824-1:2002 | ITU-T Recommendation X.680 (2002), Information Technology
1734 - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation,
1735 http://www.itu.int/ITU-T/studygroups/com17/languages/X680_0702.pdf.

1736 3 ISO/IEC 8824-2:2002 | ITU-T Recommendation X.681 (2002), Information Technology
1737 - Abstract Syntax Notation One (ASN.1): Information Object Specification,
1738 http://www.itu.int/ITU-T/studygroups/com17/languages/X681_0702.pdf.

1739 4 ISO/IEC 8824-3:2002 | ITU-T Recommendation X.682 (2002), Information Technology -
1740 Abstract Syntax Notation One (ASN.1): Constraint Specification,
1741 http://www.itu.int/ITU-T/studygroups/com17/languages/X682_0702.pdf.

1742 5 ISO/IEC 8824-4:2002 | ITU-T Recommendation X.683 (2002), Information Technology -
1743 Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications,
1744 http://www.itu.int/ITU-T/studygroups/com17/languages/X683_0702.pdf.

1745 6 ISO/IEC 8825-1:2002 | ITU-T Recommendation X.690 (2002), Information Technology -
1746 ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical
1747 Encoding Rules (CER) and Distinguished Encoding Rules (DER),
1748 http://www.itu.int/ITU-T/studygroups/com17/languages/X690_0702.pdf.

1749 7 ISO/IEC 8825-4:2002 | X.693 ITU-T Recommendation X.693 (2002) |, Information
1750 Technology - ASN.1 Encoding Rules: XML Encoding Rules (XER),
1751 http://www.itu.int/ITU-T/studygroups/com17/languages/X693_0702.pdf.

1752 8 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 1:
1753 The Digital Signature Algorithm (DSA), <http://webstore.ansi.org/>.

1754 9 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 2:
1755 The Secure Hash Algorithm (SHA-1), <http://webstore.ansi.org/>.

1756 10 ANSI X9.52-1998 Triple Data Encryption Algorithm Modes of Operation,
1757 <http://webstore.ansi.org/>.

1758 11 ANSI X9.68:2001 Digital Certificates for Mobile/Wireless and High Transaction Volume
1759 Financial Systems: Part 2: Domain Certificate Syntax, <http://webstore.ansi.org/>.

1760 12 ANSI X9.71-1999 Keyed Hash Message Authentication Code (HMAC),
1761 <http://webstore.ansi.org/>.

1762 13 ANSI X9.73:2002 Cryptographic Message Syntax (CMS), <http://webstore.ansi.org/>.

1763 14 ANSI X9.84:2003 Biometric Information Management and Security For The Financial
1764 Services Industry, <http://webstore.ansi.org/>.

1765 15 ANSI X9.96:2003 (draft) XML Cryptographic Message Syntax (XCMS).

1766 16 ANSI INCITS 358-2002 - Information technology - BioAPI Specification,
1767 <http://webstore.ansi.org/>.

1768 17 CBEFF Common Biometric Exchange File Format, NISTIR-6529,
1769 <http://oasis-open.org/committees/xcbf/docs/NISTR6529-CBEFF.pdf>, January 3, 2001.

- 1770 18 S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,
1771 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1772 19 PKCS #7 – Cryptographic Message Syntax Standard,
1773 <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/>, 1 November 1993.
- 1774 20 N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME)
1775 Part 1: Format of Internet Message Bodies, <http://www.ietf.org/rfc/rfc2045.txt>,
1776 IETF RFC 2045, November 1996.

1777

7 XCBF Schema

1778 The following ASN.1 modules provide the schema for all of the XML markup defined in this
1779 standard.

1780

1781

7.1 X9-84-Biometrics Module

1782

1783

```
X9-84-Biometrics {
1784   iso(1) identified-organization(3) tc68(133) country(16) x9(840)
1785     x9Standards(9) x9-84(84) module(0) biometrics(1) rev(1) }
1786   DEFINITIONS AUTOMATIC TAGS ::= BEGIN
1787
1788   -- EXPORTS All;
1789
1790   IMPORTS
1791
1792     -- X9.84 Biometrics Information Management and Security IDs --
1793
1794   BiometricTypes, CBEFF-Formats, IBIA-Formats, MatchingAIDs,
1795   ProcessingAIDs, x509-biometricTemplates,
1796   x968-biometricTemplates, X9-Formats
1797     FROM X9-84-Identifiers {
1798       iso(1) identified-organization(3) tc68(133) country(16)
1799       x9(840) x9Standards(9) x9-84(84) module(0)
1800       ids(3) rev(1) }
1801
1802   -- X9.84 Biometrics Information Management and Security CMS --
1803
1804   AuthenticatedData, EncryptedData, EnvelopedData,
1805   MACAlgorithmIdentifier, SignatureAlgorithmIdentifier,
1806   SignedData
1807     FROM X9-84-CMS {
1808       iso(1) identified-organization(3) tc68(133) country(16)
1809       x9(840) x9Standards(9) x9-84(84) module(0)
1810       cms(2) rev(1) } ;
1811
1812
1813
1814   BiometricSyntaxSets ::= SEQUENCE SIZE(1..MAX) OF BiometricSyntax
1815
1816   BiometricSyntax ::= CHOICE {
1817     biometricObjects      BiometricObjects,
1818     integrityObjects      IntegrityObjects,
1819     privacyObjects        PrivacyObjects,
1820     privacyAndIntegrityObjects PrivacyAndIntegrityObjects
1821   }
1822
1823   BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject
1824
1825   BiometricObject ::= SEQUENCE {
1826     biometricHeader BiometricHeader,
1827     biometricData   BiometricData
1828   }
1829
1830   --
1831   -- All of the cryptographic processing in this standard is performed
```

```

1832 -- on a value of type EncodedBiometricObjects. This is a sequence of
1833 -- one or more values of type BiometricObject in its encoded form.
1834 --
1835
1836 EncodedBiometricObjects ::= BIOMETRIC.&Type( BiometricObjects )
1837
1838 BiometricHeader ::= SEQUENCE {
1839     version      BiometricVersion  DEFAULT hv1,
1840     recordType   RecordType  OPTIONAL,
1841     dataType     DataType  OPTIONAL,
1842     purpose      Purpose  OPTIONAL,
1843     quality      Quality  OPTIONAL,
1844     validityPeriod  ValidityPeriod  OPTIONAL,
1845     format       Format  OPTIONAL
1846 }
1847
1848 BiometricVersion ::= INTEGER { hv1(0) } (0..MAX)
1849
1850 RecordType ::= BIOMETRIC.&name({BiometricTypes})
1851
1852 DataType ::= ENUMERATED {
1853     raw          (0),
1854     intermediate (1),
1855     processed    (2)
1856 }
1857
1858 Purpose ::= ENUMERATED {
1859     verify        (1),
1860     identify      (2),
1861     enroll        (3),
1862     enrollVerify  (4),
1863     enrollIdentity (5),
1864     audit         (6),
1865
1866     ... -- Expect other values --
1867 }
1868
1869 Quality ::= INTEGER {
1870     lowest        ( 0),
1871     highest       (100),
1872     notSet        ( -1),
1873     notSupported  ( -2)
1874 } (-2..100,...)
1875
1876 ValidityPeriod ::= SEQUENCE {
1877     notBefore     DateTime  OPTIONAL,
1878     notAfter      DateTime  OPTIONAL
1879 }
1880 (ALL EXCEPT({ -- none; at least one component is present -- }))
1881
1882 DateTime ::= RELATIVE-OID -- { yyyy mm dd hh mm ss z } --
1883
1884 Format ::= SEQUENCE {
1885     formatOwner  BIOMETRIC.&name({Owner}),
1886     formatType   BIOMETRIC.&Type({Owner}{@formatOwner})  OPTIONAL
1887 }
1888
1889 Owner BIOMETRIC ::= {
1890     CBEFF-Formats | -- http://www.nist.gov --
1891     IBIA-Formats  | -- http://www.ibia.org --
1892     X9-Formats,   -- http://www.x9.org --
1893
1894     ... -- expect additional vendor specific formats --

```



```

1895 }
1896
1897 -- Integrity --
1898
1899 IntegrityObjects ::= SEQUENCE {
1900     biometricObjects  EncodedBiometricObjects,
1901     integrityBlock    IntegrityBlock
1902 }
1903
1904 IntegrityBlock ::= CHOICE {
1905     digitalSignature      DigitalSignature,
1906     messageAuthenticationCode MessageAuthenticationCode,
1907     signedData            SignedData,
1908     authenticatedData    AuthenticatedData
1909 }
1910
1911 DigitalSignature ::= SEQUENCE {
1912     algorithmID  SignatureAlgorithmIdentifier,
1913     signature    OCTET STRING( CONSTRAINED BY {
1914         -- signature on -- EncodedBiometricObjects })
1915 }
1916
1917 MessageAuthenticationCode ::= SEQUENCE {
1918     keyName      OCTET STRING OPTIONAL,
1919     algorithmID  MACAlgorithmIdentifier,
1920     mac          OCTET STRING (CONSTRAINED BY {
1921         -- MAC or HMAC on -- EncodedBiometricObjects })
1922 }
1923
1924 -- Privacy --
1925
1926 PrivacyObjects ::= SEQUENCE {
1927     biometricHeaders  BiometricHeaders OPTIONAL,
1928     privacyBlock      PrivacyBlock
1929 }
1930
1931 BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
1932
1933 PrivacyBlock ::= CHOICE {
1934     fixedKey          EncryptedData,
1935     namedKey          NamedKeyEncryptedData,
1936     establishedKey    EnvelopedData
1937 }
1938
1939 NamedKeyEncryptedData ::= SEQUENCE {
1940     keyName      OCTET STRING (SIZE(1..MAX)),
1941     encryptedData EncryptedData
1942 }
1943
1944 -- Privacy and integrity --
1945
1946 PrivacyAndIntegrityObjects ::= SEQUENCE {
1947     biometricHeaders  BiometricHeaders OPTIONAL,
1948     privacyBlock      PrivacyBlock,
1949     integrityBlock    IntegrityBlock
1950 }
1951
1952 -- Authentication Information (AI) --
1953
1954 BiometricInformationSets ::=
1955     SEQUENCE SIZE(1..MAX) OF BiometricInformation
1956
1957 BiometricInformation ::= SEQUENCE {

```

```

1958     processingAlgorithms  ProcessingAlgorithms  OPTIONAL,
1959     matchingMethods      MatchingMethods    OPTIONAL
1960   }
1961   (ALL EXCEPT({ -- none; at least one component is present -- }))
1962
1963   -- Biometric processing algorithms --
1964
1965   ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation
1966
1967   ProcessingInformation ::= SEQUENCE {
1968     id      BIOMETRIC.&name({ProcessingAIDs}),
1969     parms  BIOMETRIC.&Type({ProcessingAIDs}{@id})  OPTIONAL
1970   }
1971
1972   -- Biometric matching methods --
1973
1974   MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation
1975
1976   MatchingInformation ::= SEQUENCE {
1977     id      BIOMETRIC.&name({MatchingAIDs}),
1978     parms  BIOMETRIC.&Type({MatchingAIDs}{@id})  OPTIONAL
1979   }
1980
1981   BiometricData ::= OCTET STRING(SIZE(1..MAX))
1982
1983   -- Biometrics information object class --
1984
1985   BIOMETRIC ::= CLASS {
1986     &name  BIOMETRIC-IDENTIFIER  UNIQUE,
1987     &Type  OPTIONAL
1988   }
1989     WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
1990
1991   BIOMETRIC-IDENTIFIER ::= CHOICE {
1992     oid  OBJECT IDENTIFIER,  -- complete object identifier
1993     id   RELATIVE-OID        -- object identifier fragment
1994   }
1995
1996   -- Biometric certificate extension --
1997
1998   --
1999   -- A biometricTemplates information object can be used to extend the
2000   -- information bound to a public key in an value of types Certificate
2001   -- or AttributeCertificate as defined in The Directory series of
2002   -- standards, to include biometric identity information.
2003   --
2004
2005   biometricTemplates EXTENSION ::= {
2006     SYNTAX          EncodedBiometricObjects  -- CXER --
2007     IDENTIFIED BY  x509-biometricTemplates
2008   }
2009
2010   EXTENSION ::= CLASS {
2011     &id            OBJECT IDENTIFIER  UNIQUE,
2012     &ExtnType
2013   }
2014     WITH SYNTAX { SYNTAX &ExtnType IDENTIFIED BY &id }
2015
2016   --
2017   -- A domainBiometricTemplates information object can be used to extend
2018   -- the information bound to a public key in an value of ASN.1 type
2019   -- DomainCertificate as defined in the X9.68 Domain Certificate Syntax
2020   -- standard, to include biometric identity information.

```

```

2021 --
2022
2023 domainBiometricTemplates PRIVATE-X ::= {
2024     NAME oid : x968-biometricTemplates
2025     TYPE EncodedBiometricObjects -- CXER --
2026 }
2027
2028 PRIVATE-X ::= CLASS {
2029     &name Identifier UNIQUE,
2030     &Type OPTIONAL
2031 }
2032 WITH SYNTAX { NAME &name [TYPE &Type] }
2033
2034 Identifier ::= CHOICE {
2035     oid OBJECT IDENTIFIER, -- complete object identifier
2036     id RELATIVE-OID -- object identifier fragment
2037 }
2038
2039 END -- X9-84-Biometrics --

```

2040

2041 7.2 X9-84-CMS Module

2042

```

2043 X9-84-CMS {
2044     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
2045     x9Standards(9) x9-84(84) module(0) cms(2) rev(1) }
2046     DEFINITIONS IMPLICIT TAGS ::= BEGIN
2047
2048     -- EXPORTS All;
2049
2050     IMPORTS
2051
2052     -- ANSI X9.84 Biometric Information Management & Security IDs --
2053
2054     des-ede3-cbc, dsa-with-sha1, ecdsa-with-SHA1, hmac-with-SHA1,
2055     id-data, NullParms, OID, rsaEncryption, SHA-Algorithms,
2056     sha1WithRSAEncryption, sha-1
2057     FROM X9-84-Identifiers {
2058         iso(1) identified-organization(3) tc68(133) country(16)
2059         x9(840) x9Standards(9) x9-84(84) module(0)
2060         ids(3) rev(1) };
2061
2062     SignedData ::= SEQUENCE {
2063         version CMSVersion,
2064         digestAlgorithms DigestAlgorithmIdentifiers,
2065         encapContentInfo EncapsulatedContentInfo,
2066         certificates [0] CertificateSet OPTIONAL,
2067         crls [1] CertificateRevocationLists OPTIONAL,
2068         signerInfos SignerInfos
2069     }
2070
2071     CMSVersion ::= INTEGER { v84(84) } (v84,...)
2072
2073     DigestAlgorithmIdentifiers ::=
2074         SET SIZE(1) OF DigestAlgorithmIdentifier
2075
2076     DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{DigestAlgorithms}}
2077
2078     DigestAlgorithms ALGORITHM ::= {
2079         SHA-Algorithms,

```

```

2080
2081     ... -- Expect other digest algorithms --
2082 }
2083
2084 EncapsulatedContentInfo ::= SEQUENCE {
2085     eContentType ContentType,
2086     eContent      [0] EXPLICIT OCTET STRING OPTIONAL
2087 }
2088
2089 ContentType ::= CONTENTS.&id({Contents})
2090
2091 CONTENTS ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1998, Annex A
2092
2093 Contents CONTENTS ::= {
2094     { Data IDENTIFIED BY id-data }
2095 }
2096
2097 Data ::= OCTET STRING
2098
2099 CertificateSet ::= --[XER:BASE64]1-- OCTET STRING
2100
2101 CertificateRevocationLists ::= --[XER:BASE64] -- OCTET STRING
2102
2103 SignerInfos ::= SET SIZE(1) OF SignerInfo
2104
2105 SignerInfo ::= SEQUENCE {
2106     version          CMSVersion,
2107     sid              SignerIdentifier,
2108     digestAlgorithm  DigestAlgorithmIdentifier,
2109     signatureAlgorithm SignatureAlgorithmIdentifier,
2110     signature        SignatureValue
2111 }
2112
2113 SignerIdentifier ::= CHOICE {
2114     certHash [1] EXPLICIT Hash
2115 }
2116
2117 Hash ::= CHOICE {
2118     ietf          CertHash, -- SHA-1 hash of entire certificate
2119     withAlgID    DigestInfo
2120 }
2121
2122 CertHash ::= OCTET STRING (ENCODED BY sha-1)
2123
2124 DigestInfo ::= SEQUENCE {
2125     hashAlgorithm DigestAlgorithmIdentifier,
2126     digest        OCTET STRING
2127 }
2128
2129 SignatureAlgorithmIdentifier ::=
2130     AlgorithmIdentifier {{SignatureAlgorithms}}
2131
2132 SignatureAlgorithms ALGORITHM ::= {
2133     { OID dsa-with-sha1          PARMS NullParms } |
2134     { OID ecdsa-with-SHA1       PARMS NullParms } |
2135     { OID sha1WithRSAEncryption PARMS NullParms },
2136
2137     ... -- Expect other signature algorithms --

```

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

```

2138 }
2139
2140 SignatureValue ::= OCTET STRING
2141
2142 EncryptedData ::= SEQUENCE {
2143     version          CMSVersion,
2144     encryptedContentInfo  EncryptedContentInfo
2145 }
2146
2147 EncryptedContentInfo ::= SEQUENCE {
2148     contentType          ContentType,
2149     contentEncryptionAlgorithm  ContentEncryptAlgorithmIdentifier,
2150     encryptedContent      [0] EncryptedContent
2151 }
2152
2153 ContentEncryptAlgorithmIdentifier ::=
2154     AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
2155
2156 ContentEncryptionAlgorithms ALGORITHM ::= {
2157     { OID des-ede3-cbc PARMS IV },
2158     ... -- Expect other content encryption algorithms --
2159 }
2160
2161 IV ::= OCTET STRING (SIZE(8))
2162
2163 EncryptedContent ::= OCTET STRING
2164
2165 EnvelopedData ::= SEQUENCE {
2166     version          CMSVersion,
2167     originatorInfo    [0] OriginatorInfo  OPTIONAL,
2168     recipientInfos    RecipientInfos,
2169     encryptedContentInfo  EncryptedContentInfo
2170 }
2171
2172 OriginatorInfo ::= SEQUENCE {
2173     certs [0] CertificateSet  OPTIONAL,
2174     crls  [1] CertificateRevocationLists  OPTIONAL
2175 }
2176
2177 (ALL EXCEPT({ -- none; at least one component is present -- }))
2178
2179 RecipientInfos ::= SET SIZE(1) OF RecipientInfo
2180
2181 RecipientInfo ::= CHOICE {
2182     ktri  KeyTransRecipientInfo
2183 }
2184
2185 KeyTransRecipientInfo ::= SEQUENCE {
2186     version          CMSVersion,
2187     rid              RecipientIdentifier,
2188     keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
2189     encryptedKey      EncryptedKey
2190 }
2191
2192 RecipientIdentifier ::= CHOICE {
2193     certHash [73] EXPLICIT Hash
2194 }
2195
2196 KeyEncryptionAlgorithmIdentifier ::=
2197     AlgorithmIdentifier {{KeyEncryptionAlgorithms}}
2198
2199 KeyEncryptionAlgorithms ALGORITHM ::= {
2200     { OID rsaEncryption PARMS NullParms },

```

```

2201
2202     ... -- expect other key encryption algorithms --
2203 }
2204
2205 EncryptedKey ::= OCTET STRING
2206
2207 AuthenticatedData ::= SEQUENCE {
2208     version          CMSVersion,
2209     recipientInfos   RecipientInfos,
2210     macAlgorithm     MACAlgorithmIdentifier,
2211     encapContentInfo EncapsulatedContentInfo,
2212     mac              MessageAuthenticationCode
2213 }
2214
2215 MACAlgorithmIdentifier ::= AlgorithmIdentifier {{MACAlgorithms}}
2216
2217 MACAlgorithms ALGORITHM ::= {
2218     { OID hmac-with-SHA1 },
2219
2220     ... -- expect other MAC or HMAC algorithms --
2221 }
2222
2223 MessageAuthenticationCode ::= OCTET STRING
2224
2225
2226 -- Cryptographic algorithm identification --
2227
2228 ALGORITHM ::= CLASS {
2229     &id   OBJECT IDENTIFIER UNIQUE,
2230     &Type OPTIONAL
2231 }
2232     WITH SYNTAX { OID &id [PARMS &Type] }
2233
2234 AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
2235     algorithm   ALGORITHM.&id( {IOSet} ),
2236     parameters ALGORITHM.&Type( {IOSet}{@algorithm} ) OPTIONAL
2237 }
2238
2239
2240 END -- X9-84-CMS --

```

2241

2242 7.3 X9-84-Identifiers Module

2243

```

2244 X9-84-Identifiers {
2245     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
2246     x9Standards(9) x9-84(84) module(0) ids(3) rev(1) }
2247     DEFINITIONS AUTOMATIC TAGS ::= BEGIN
2248
2249     -- EXPORTS All;
2250
2251     IMPORTS
2252
2253     -- X9.84 Biometrics Information Management and Security --
2254
2255     BIOMETRIC, BiometricInformationSets
2256     FROM X9-84-Biometrics {
2257         iso(1) identified-organization(3) tc68(133) country(16)
2258         x9(840) x9Standards(9) x9-84(84) module(0)
2259         biometrics(1) rev(1) }

```

```

2260
2261      -- X9.84 Biometrics Information Management and Security CMS --
2262
2263      ALGORITHM
2264      FROM X9-84-CMS {
2265          iso(1) identified-organization(3) tc68(133) country(16)
2266          x9(840) x9Standards(9) x9-84(84) module(0)
2267          cms(2) rev(1) };
2268
2269
2270      OID ::= OBJECT IDENTIFIER -- Alias
2271
2272      ReOID ::= RELATIVE-OID -- Alias
2273
2274      -- x9-84 { 1 3 133 16 840 9 84 }
2275      -- x9-84-Module { 1 3 133 16 840 9 84 0 }
2276      -- x9-84-Biometrics { 1 3 133 16 840 9 84 0 1 }
2277      -- x9-84-CMS { 1 3 133 16 840 9 84 0 2 }
2278      -- x9-84-Identifiers { 1 3 133 16 840 9 84 0 3 }
2279      -- biometric { 1 3 133 16 840 9 84 1 }
2280      -- id-unknown-Type { 1 3 133 16 840 9 84 1 0 }
2281      -- id-body-Odor { 1 3 133 16 840 9 84 1 1 }
2282      -- id-dna { 1 3 133 16 840 9 84 1 2 }
2283      -- id-ear-Shape { 1 3 133 16 840 9 84 1 3 }
2284      -- id-facial-Features { 1 3 133 16 840 9 84 1 4 }
2285      -- id-finger-Image { 1 3 133 16 840 9 84 1 5 }
2286      -- id-finger-Geometry { 1 3 133 16 840 9 84 1 6 }
2287      -- id-hand-Geometry { 1 3 133 16 840 9 84 1 7 }
2288      -- id-iris-Features { 1 3 133 16 840 9 84 1 8 }
2289      -- id-keystroke-Dynamics { 1 3 133 16 840 9 84 1 9 }
2290      -- id-palm { 1 3 133 16 840 9 84 1 10 }
2291      -- id-retina { 1 3 133 16 840 9 84 1 11 }
2292      -- id-signature { 1 3 133 16 840 9 84 1 12 }
2293      -- id-speech-Pattern { 1 3 133 16 840 9 84 1 13 }
2294      -- id-thermal-Image { 1 3 133 16 840 9 84 1 14 }
2295      -- id-vein-Pattern { 1 3 133 16 840 9 84 1 15 }
2296      -- id-thermal-Face-Image { 1 3 133 16 840 9 84 1 16 }
2297      -- id-thermal-Hand-Image { 1 3 133 16 840 9 84 1 17 }
2298      -- id-lip-Movement { 1 3 133 16 840 9 84 1 18 }
2299      -- id-gait { 1 3 133 16 840 9 84 1 19 }
2300      -- processing-algorithm { 1 3 133 16 840 9 84 2 }
2301      -- matching-method { 1 3 133 16 840 9 84 3 }
2302      -- format-Owner { 1 3 133 16 840 9 84 4 }
2303      -- cbeff-Owner { 1 3 133 16 840 9 84 4 0 }
2304      -- ibia-Owner { 1 3 133 16 840 9 84 4 1 }
2305      -- id-ibia-SAFLINK { 1 3 133 16 840 9 84 4 1 1 }
2306      -- id-ibia-Bioscrypt { 1 3 133 16 840 9 84 4 1 2 }
2307      -- id-ibia-Visionics { 1 3 133 16 840 9 84 4 1 3 }
2308      -- id-ibia-InfineonTechnologiesAG { 1 3 133 16 840 9 84 4 1 4 }
2309      -- id-ibia-IridianTechnologies { 1 3 133 16 840 9 84 4 1 5 }
2310      -- id-ibia-Veridicom { 1 3 133 16 840 9 84 4 1 6 }
2311      -- id-ibia-CyberSIGN { 1 3 133 16 840 9 84 4 1 7 }
2312      -- id-ibia-eCryp { 1 3 133 16 840 9 84 4 1 8 }
2313      -- id-ibia-FingerprintCardsAB { 1 3 133 16 840 9 84 4 1 9 }
2314      -- id-ibia-SecuGen { 1 3 133 16 840 9 84 4 1 10 }
2315      -- id-ibia-PreciseBiometric { 1 3 133 16 840 9 84 4 1 11 }
2316      -- id-ibia-Identix { 1 3 133 16 840 9 84 4 1 12 }
2317      -- id-ibia-DERMALOG { 1 3 133 16 840 9 84 4 1 13 }
2318      -- id-ibia-LOGICO { 1 3 133 16 840 9 84 4 1 14 }
2319      -- id-ibia-NIST { 1 3 133 16 840 9 84 4 1 15 }
2320      -- id-ibia-A4Vision { 1 3 133 16 840 9 84 4 1 16 }
2321      -- id-ibia-NEC { 1 3 133 16 840 9 84 4 1 17 }
2322      -- id-ibia-STMicroelectronics { 1 3 133 16 840 9 84 4 1 18 }

```

```

2323 -- id-ibia-Ultra-Scan { 1 3 133 16 840 9 84 4 1 19 }
2324 -- id-ibia-Aurora-Wireless { 1 3 133 16 840 9 84 4 1 20 }
2325 -- id-ibia-Thales { 1 3 133 16 840 9 84 4 1 21 }
2326 -- id-ibia-IBG { 1 3 133 16 840 9 84 4 1 22 }
2327 -- id-ibia-Cogent-Systems { 1 3 133 16 840 9 84 4 1 23 }
2328 -- id-ibia-Cross-Match { 1 3 133 16 840 9 84 4 1 24 }
2329 -- id-ibia-Recognition-Systems { 1 3 133 16 840 9 84 4 1 25 }
2330 -- id-ibia-DIN { 1 3 133 16 840 9 84 4 1 26 }
2331 -- id-ibia-INCITS-M1 { 1 3 133 16 840 9 84 4 1 27 }
2332 -- x9-Owner { 1 3 133 16 840 9 84 4 2 }
2333 -- certificate-Extensions { 1 3 133 16 840 9 84 5 }
2334 -- x968-biometricTemplates { 1 3 133 16 840 9 84 5 0 }
2335 -- x509-biometricTemplates { 1 3 133 16 840 9 84 5 1 }
2336
2337 -- X9.84 arc; base object identifier --
2338
2339 x9-84 OID ::= {
2340     iso(1) identified-organization(3) tc68(133) country(16)
2341     x9(840) x9Standards(9) x9-84(84)
2342 }
2343
2344 -- X9.84 ASN.1 modules --
2345
2346 x9-84-Module OID ::= { x9-84 modules(0) }
2347
2348 x9-84-Biometrics OID ::= { x9-84-Module biometrics(1) rev(1) }
2349
2350 x9-84-CMS OID ::= { x9-84-Module cms(2) rev(1) }
2351
2352 x9-84-Identifiers OID ::= { x9-84-Module ids(3) rev(1) }
2353
2354 -- X9.84 biometric technologies --
2355
2356 biometric OID ::= { x9-84 biometrics(1) }
2357
2358 id-unknown-Type OID ::= { biometric unknownType(0) }
2359 id-body-Odor OID ::= { biometric bodyOdor(1) }
2360 id-dna OID ::= { biometric dna(2) }
2361 id-ear-Shape OID ::= { biometric ear-Shape(3) }
2362 id-facial-Features OID ::= { biometric facialFeatures(4) }
2363 id-finger-Image OID ::= { biometric fingerImage(5) }
2364 id-finger-Geometry OID ::= { biometric fingerGeometry(6) }
2365 id-hand-Geometry OID ::= { biometric handGeometry(7) }
2366 id-iris-Features OID ::= { biometric irisFeatures(8) }
2367 id-keystroke-Dynamics OID ::= { biometric keystrokeDynamics(9) }
2368 id-palm OID ::= { biometric palm(10) }
2369 id-retina OID ::= { biometric retina(11) }
2370 id-signature OID ::= { biometric signature(12) }
2371 id-speech-Pattern OID ::= { biometric speech-Pattern(13) }
2372 id-thermal-Image OID ::= { biometric thermalImage(14) }
2373 id-vein-Pattern OID ::= { biometric veinPattern(15) }
2374 id-thermal-Face-Image OID ::= { biometric thermalFaceImage(16) }
2375 id-thermal-Hand-Image OID ::= { biometric thermalHandImage(17) }
2376 id-lip-Movement OID ::= { biometric lipMovement(18) }
2377 id-gait OID ::= { biometric gait(19) }
2378
2379 -- X9.84 biometric technology object identifier fragments --
2380
2381 unknown-Type ReLOID ::= { unknownType(0) }
2382 body-Odor ReLOID ::= { bodyOdor(1) }
2383 dna ReLOID ::= { dna(2) }
2384 ear-Shape ReLOID ::= { earShape(3) }
2385 facial-Features ReLOID ::= { facialFeatures(4) }

```



```

2386 finger-Image RelOID ::= { fingerImage(5) }
2387 finger-Geometry RelOID ::= { fingerGeometry(6) }
2388 hand-Geometry RelOID ::= { handGeometry(7) }
2389 iris-Features RelOID ::= { irisFeatures(8) }
2390 keystroke-Dynamics RelOID ::= { keystrokeDynamics(9) }
2391 palm RelOID ::= { palm(10) }
2392 retina RelOID ::= { retina(11) }
2393 signature RelOID ::= { signature(12) }
2394 speech-Pattern RelOID ::= { speechPattern(13) }
2395 thermal-Image RelOID ::= { thermalImage(14) }
2396 vein-Pattern RelOID ::= { veinPattern(15) }
2397 thermal-Face-Image RelOID ::= { thermalFaceImage(16) }
2398 thermal-Hand-Image RelOID ::= { thermalHandImage(17) }
2399 lip-Movement RelOID ::= { lipMovement(18) }
2400 gait RelOID ::= { gait(19) }
2401
2402 BiometricTypes BIOMETRIC ::= {
2403     { BIOMETRIC id : unknown-Type } |
2404     { BIOMETRIC id : body-Odor } |
2405     { BIOMETRIC id : dna } |
2406     { BIOMETRIC id : ear-Shape } |
2407     { BIOMETRIC id : facial-Features } |
2408     { BIOMETRIC id : finger-Image } |
2409     { BIOMETRIC id : finger-Geometry } |
2410     { BIOMETRIC id : hand-Geometry } |
2411     { BIOMETRIC id : iris-Features } |
2412     { BIOMETRIC id : keystroke-Dynamics } |
2413     { BIOMETRIC id : palm } |
2414     { BIOMETRIC id : retina } |
2415     { BIOMETRIC id : signature } |
2416     { BIOMETRIC id : speech-Pattern } |
2417     { BIOMETRIC id : thermal-Image } |
2418     { BIOMETRIC id : vein-Pattern } |
2419     { BIOMETRIC id : thermal-Face-Image } |
2420     { BIOMETRIC id : thermal-Hand-Image } |
2421     { BIOMETRIC id : lip-Movement } |
2422     { BIOMETRIC id : gait } |,
2423
2424     ... -- expect additional biometric types --
2425 }
2426
2427
2428 -- X9.84 biometric processing algorithms --
2429
2430 processing-algorithm OID ::= { x9-84 processingAlgorithms(2) }
2431
2432 -- X9.84 biometric matching methods --
2433
2434 matching-method OID ::= { x9-84 matchingMethods(3) }
2435
2436 -- X9.84 vendor specific formats --
2437
2438 format-Owner OID ::= { x9-84 format-owners(4) }
2439
2440 cbeff-Owner OID ::= { format-Owner cbeff(0) }
2441
2442 ibia-Owner OID ::= { format-Owner ibia(1) }
2443
2444 x9-Owner OID ::= { format-Owner x9(2) }
2445
2446 -- IBIA vendor specific formats registered at http://www.ibia.org
2447
2448 id-ibia-SAFLINK OID ::= { ibia-Owner 1 }

```

```

2449 id-ibia-Bioscrypt          OID ::= { ibia-Owner  2 }
2450 id-ibia-Visionics         OID ::= { ibia-Owner  3 }
2451 id-ibia-InfineonTechnologiesAG  OID ::= { ibia-Owner  4 }
2452 id-ibia-IridianTechnologies  OID ::= { ibia-Owner  5 }
2453 id-ibia-Veridicom         OID ::= { ibia-Owner  6 }
2454 id-ibia-CyberSIGN        OID ::= { ibia-Owner  7 }
2455 id-ibia-eCryp            OID ::= { ibia-Owner  8 }
2456 id-ibia-FingerprintCardsAB  OID ::= { ibia-Owner  9 }
2457 id-ibia-SecuGen          OID ::= { ibia-Owner 10 }
2458 id-ibia-PreciseBiometric  OID ::= { ibia-Owner 11 }
2459 id-ibia-Identix          OID ::= { ibia-Owner 12 }
2460 id-ibia-DERMALOG        OID ::= { ibia-Owner 13 }
2461 id-ibia-LOGICO          OID ::= { ibia-Owner 14 }
2462 id-ibia-NIST             OID ::= { ibia-Owner 15 }
2463 id-ibia-A4Vision         OID ::= { ibia-Owner 16 }
2464 id-ibia-NEC             OID ::= { ibia-Owner 17 }
2465 id-ibia-STMicroelectronics  OID ::= { ibia-Owner 18 }
2466 id-ibia-Ultra-Scan      OID ::= { ibia-Owner 19 }
2467 id-ibia-Aurora-Wireless  OID ::= { ibia-Owner 20 }
2468 id-ibia-Thales          OID ::= { ibia-Owner 21 }
2469 id-ibia-IBG             OID ::= { ibia-Owner 22 }
2470 id-ibia-Cogent-Systems  OID ::= { ibia-Owner 23 }
2471 id-ibia-Cross-Match     OID ::= { ibia-Owner 24 }
2472 id-ibia-Recognition-Systems  OID ::= { ibia-Owner 25 }
2473 id-ibia-DIN             OID ::= { ibia-Owner 26 }
2474 id-ibia-INCITS-M1      OID ::= { ibia-Owner 27 }
2475
2476
2477 -- When represented as values of type OBJECT IDENTIFIER, these
2478 -- IBIA vendor specific formats may be associated with any ASN.1
2479 -- type.
2480
2481 IBIAoidFormats BIOMETRIC ::= {
2482   { BIOMETRIC oid : id-ibia-SAFLINK          DATA Any } |
2483   { BIOMETRIC oid : id-ibia-Bioscrypt       DATA Any } |
2484   { BIOMETRIC oid : id-ibia-Visionics       DATA Any } |
2485   { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG  DATA Any } |
2486   { BIOMETRIC oid : id-ibia-IridianTechnologies  DATA Any } |
2487   { BIOMETRIC oid : id-ibia-Veridicom       DATA Any } |
2488   { BIOMETRIC oid : id-ibia-CyberSIGN      DATA Any } |
2489   { BIOMETRIC oid : id-ibia-eCryp          DATA Any } |
2490   { BIOMETRIC oid : id-ibia-FingerprintCardsAB  DATA Any } |
2491   { BIOMETRIC oid : id-ibia-SecuGen        DATA Any } |
2492   { BIOMETRIC oid : id-ibia-PreciseBiometric  DATA Any } |
2493   { BIOMETRIC oid : id-ibia-Identix        DATA Any } |
2494   { BIOMETRIC oid : id-ibia-DERMALOG       DATA Any } |
2495   { BIOMETRIC oid : id-ibia-LOGICO        DATA Any } |
2496   { BIOMETRIC oid : id-ibia-NIST          DATA Any } |
2497   { BIOMETRIC oid : id-ibia-A4Vision       DATA Any } |
2498   { BIOMETRIC oid : id-ibia-NEC           DATA Any } |
2499   { BIOMETRIC oid : id-ibia-STMicroelectronics  DATA Any } |
2500   { BIOMETRIC oid : id-ibia-Ultra-Scan    DATA Any } |
2501   { BIOMETRIC oid : id-ibia-Aurora-Wireless  DATA Any } |
2502   { BIOMETRIC oid : id-ibia-Thales        DATA Any } |
2503   { BIOMETRIC oid : id-ibia-IBG          DATA Any } |
2504   { BIOMETRIC oid : id-ibia-Cogent-Systems  DATA Any } |
2505   { BIOMETRIC oid : id-ibia-Cross-Match    DATA Any } |
2506   { BIOMETRIC oid : id-ibia-Recognition-Systems  DATA Any } |
2507   { BIOMETRIC oid : id-ibia-DIN           DATA Any } |
2508   { BIOMETRIC oid : id-ibia-INCITS-M1     DATA Any },
2509
2510   ... -- Expect additional vendor specific formats --
2511 }

```

```

2512
2513 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
2514
2515 -- Relative object identifier representations of the identical
2516 -- IBIA vendor specific formats defined as OBJECT IDENTIFIER
2517 -- values above are used to identify these formats when they must
2518 -- comply with the fixed format requirements of the BioAPI 1.1
2519 -- specification and are associated with a two byte integer value.
2520
2521 ibia-SAFLINK RelOID ::= { 1 }
2522 ibia-Bioscrypt RelOID ::= { 2 }
2523 ibia-Visionics RelOID ::= { 3 }
2524 ibia-InfineonTechnologiesAG RelOID ::= { 4 }
2525 ibia-IridianTechnologies RelOID ::= { 5 }
2526 ibia-Veridicom RelOID ::= { 6 }
2527 ibia-CyberSIGN RelOID ::= { 7 }
2528 ibia-eCryp RelOID ::= { 8 }
2529 ibia-FingerprintCardsAB RelOID ::= { 9 }
2530 ibia-SecuGen RelOID ::= { 10 }
2531 ibia-PreciseBiometric RelOID ::= { 11 }
2532 ibia-Identix RelOID ::= { 12 }
2533 ibia-DERMALOG RelOID ::= { 13 }
2534 ibia-LOGICO RelOID ::= { 14 }
2535 ibia-NIST RelOID ::= { 15 }
2536 ibia-A4Vision RelOID ::= { 16 }
2537 ibia-NEC RelOID ::= { 17 }
2538 ibia-STMicroelectronics RelOID ::= { 18 }
2539 ibia-Ultra-Scan RelOID ::= { 19 }
2540 ibia-Aurora-Wireless RelOID ::= { 20 }
2541 ibia-Thales RelOID ::= { 21 }
2542 ibia-IBG RelOID ::= { 22 }
2543 ibia-Cogent-Systems RelOID ::= { 23 }
2544 ibia-Cross-Match RelOID ::= { 24 }
2545 ibia-Recognition-Systems RelOID ::= { 25 }
2546 ibia-DIN RelOID ::= { 26 }
2547 ibia-INCITS-M1 RelOID ::= { 27 }
2548
2549
2550 IBIAidFormats BIOMETRIC ::= {
2551 { BIOMETRIC id : ibia-SAFLINK DATA BirInt16 } |
2552 { BIOMETRIC id : ibia-Bioscrypt DATA BirInt16 } |
2553 { BIOMETRIC id : ibia-Visionics DATA BirInt16 } |
2554 { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
2555 { BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
2556 { BIOMETRIC id : ibia-Veridicom DATA BirInt16 } |
2557 { BIOMETRIC id : ibia-CyberSIGN DATA BirInt16 } |
2558 { BIOMETRIC id : ibia-eCryp DATA BirInt16 } |
2559 { BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
2560 { BIOMETRIC id : ibia-SecuGen DATA BirInt16 } |
2561 { BIOMETRIC id : ibia-PreciseBiometric DATA BirInt16 } |
2562 { BIOMETRIC id : ibia-Identix DATA BirInt16 } |
2563 { BIOMETRIC id : ibia-DERMALOG DATA BirInt16 } |
2564 { BIOMETRIC id : ibia-LOGICO DATA BirInt16 } |
2565 { BIOMETRIC id : ibia-NIST DATA BirInt16 } |
2566 { BIOMETRIC id : ibia-A4Vision DATA BirInt16 } |
2567 { BIOMETRIC id : ibia-NEC DATA BirInt16 } |
2568 { BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 } |
2569 { BIOMETRIC id : ibia-Ultra-Scan DATA BirInt16 } |
2570 { BIOMETRIC id : ibia-Aurora-Wireless DATA BirInt16 } |
2571 { BIOMETRIC id : ibia-Thales DATA BirInt16 } |
2572 { BIOMETRIC id : ibia-IBG DATA BirInt16 } |
2573 { BIOMETRIC id : ibia-Cogent-Systems DATA BirInt16 } |
2574 { BIOMETRIC id : ibia-Cross-Match DATA BirInt16 } |

```

```

2575     { BIOMETRIC id : ibia-Recognition-Systems    DATA BirInt16 } |
2576     { BIOMETRIC id : ibia-DIN                    DATA BirInt16 } |
2577     { BIOMETRIC id : ibia-INCITS-M1              DATA BirInt16 },
2578
2579     ... -- Expect others --
2580 }
2581
2582 BirInt16 ::= INTEGER (0..65535)
2583
2584
2585 IBIA-Formats BIOMETRIC ::= {
2586     IBIAoidFormats | -- Complete object identifiers
2587     IBIAidFormats,  -- Object identifier fragments
2588
2589     ... -- Expect additional IBIA vendor specific formats --
2590 }
2591
2592 id-x984BioInfo  OID ::= { cbeff-Owner x984BioInfo(0) }
2593
2594
2595 CBEFFoidFormats BIOMETRIC ::= {
2596     { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },
2597
2598     ... -- Expect other objects --
2599 }
2600
2601 x984BioInfo  RelOID ::= { x984BioInfo(0) } -- CBEFF owner
2602
2603 CBEFFidFormats BIOMETRIC ::= {
2604     { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },
2605
2606     ... -- Expect other objects --
2607 }
2608
2609 CBEFF-Formats BIOMETRIC ::= {
2610     CBEFFoidFormats | -- Complete object identifiers
2611     CBEFFidFormats,  -- Object identifier fragments
2612
2613     ... -- Expect additional CBEFF vendor specific formats --
2614 }
2615
2616 MatchingAIDs BIOMETRIC ::= {
2617
2618     ... -- Expect CBEFF assignments in BiometricInformationSets --
2619 }
2620
2621 ProcessingAIDs BIOMETRIC ::= {
2622
2623     ... -- Expect CBEFF assignments in BiometricInformationSets --
2624 }
2625
2626 X9-Formats BIOMETRIC ::= {
2627     X9oidFormats |
2628     X9idFormats,
2629
2630     ... -- Expect additional X9 vendor specific formats --
2631 }
2632
2633 X9oidFormats BIOMETRIC ::= {
2634     ... -- Expect X9 assigned objects --
2635 }
2636
2637 X9idFormats BIOMETRIC ::= {

```

```

2638     ... -- Expect X9 assigned objects of the form { 2 x } --
2639     }
2640
2641     certificate-Extensions OID ::= { x9-84 certificate-Extensions(5) }
2642
2643     x968-biometricTemplates OID ::= { certificate-Extensions 0 }
2644     x509-biometricTemplates OID ::= { certificate-Extensions 1 }
2645
2646     -- RSA PKCS #7 Content type
2647
2648     id-data OID ::= {
2649         iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
2650         pkcs7(7) data(1)
2651     }
2652
2653     -- Security object identifiers
2654
2655
2656     -- FIPS 180-1 and FIPS 180-2 Secure Hash Algorithm --
2657
2658
2659     sha-1 OID ::= {
2660         iso(1) identified-organization(3) oiw(14) secsig(3)
2661         algorithm(2) 26
2662     }
2663
2664     sha2Algorithm OID ::= {
2665         joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
2666         csor(3) nistAlgorithm(4) hashAlgs(2)
2667     }
2668
2669     id-sha256 OID ::= { sha2Algorithm sha256(1) }
2670
2671     id-sha384 OID ::= { sha2Algorithm sha384(2) }
2672
2673     id-sha512 OID ::= { sha2Algorithm sha512(3) }
2674
2675     SHA-Algorithms ALGORITHM ::= {
2676         { OID sha-1          PARMS NullParms } |
2677         { OID id-sha256     PARMS NullParms } |
2678         { OID id-sha384     PARMS NullParms } |
2679         { OID id-sha512     PARMS NullParms},
2680
2681     ... -- Expect others --
2682     }
2683
2684     NullParms ::= NULL -- No initialization vector
2685
2686     -- X9.57 DSA signature generated with SHA-1 hash (DSA X9.30)
2687
2688     dsa-with-sha1 OID ::= {
2689         iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }
2690
2691     -- X9.71 HMAC with SHA-1 algorithm
2692
2693     hmac-with-SHA1 OID ::= {
2694         iso(1) identified-organization(3) dod(6)
2695         internet(1) security(5) mechanisms(5) 8 1 2 }
2696
2697     -- RSA PKCS #1 signature generated with SHA-1 hash & encryption scheme
2698
2699     sha1WithRSAEncryption OID ::= {
2700         iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 5 }

```

2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718

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

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

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

-- X9.62 ECDSA signature with SHA-1

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

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

2719 **7.4 Encodings to be employed**

2720 **7.4.1 Encodings used for calculation of digital signatures and MACs**

2721 All digital signatures and Message Authentication Codes shall be calculated using the binary
2722 values obtained by applying a CXER encoding to a value of EncodedBiometricObject (see
2723 5.1.2.1.1 Digital Signature Process).

2724 NOTE 1 – A CXER encoding is defined by X.693 as an initial character
2725 representation, followed by the application of the UTF8 transformation to those
2726 characters to produce a set of binary octets.

2727 NOTE 2 – This is the only use of CXER in this specification, and is independent
2728 of any encoding used for transfer.

2729

2730 **7.4.2 Octet Strings with Certificates and Certificate Revocation Lists**

2731 All values of the **CertificateSet** type (and **CertificateRevocationLists** type) octet strings shall be
2732 the concatenation of the octets of the DER encodings of one or more X.509 certificates or
2733 attribute certificates (or certificate revocation lists, respectively).

2734 If the outer level encoding is BER (see 7.4.3 below) then the value of these octet string types
2735 shall be encoded as specified in BER.

2736 NOTE – A formal specification of the types forming the contents of these octet
2737 strings is not possible, as they are the concatenation of self-delimited encodings
2738 of multiple ASN.1 types, not the encoding of a single ASN.1 type. This is for
2739 historical reasons.

2740 If the outer level encoding is BASIC-XER then the value of these octet strings shall not be
2741 encoded as specified in BASIC-XER (a UTF8 transform of a hexadecimal encoding), but shall be
2742 encoded as specified in the following paragraph.

2743 The UTF8 transformation shall be applied to the Content-Transfer-Encoding specified in IETF
2744 RFC 2045, 6.8, except that the 76 character limit does not apply, and XML escape sequences for
2745 white-space can be present in any position within the encoding.

2746 NOTE - IETF RFC 2045 mandates the presence of line breaks dividing the
2747 encoding into lines of at most 76 characters, but this is not required in XCBF
2748 encodings. It also allows "white-space" to be inserted in any position within the

2749 base64 encoding. This is extended in this specification to allow the inclusion of
2750 XML escape sequences for white-space.
2751

2752 **7.4.3 Outer-level encodings**

2753 The outer-level encoding used for XCBF shall by default be the BASIC-XER encoding specified
2754 by X.693 (but see 7.4.2 above for the use of base64).

2755 By implementor agreement, the outer-level encoding may be BER.

2756 NOTE 1 – If the outer-level encoding is BER, the CXER encoding specified in
2757 7.4.1 is still employed for cryptographic transformations, but is not used for
2758 transfer.

2759 NOTE 2 – If the outer-level encoding is BER, the concatenation of DER
2760 encodings specified in 7.4.2 is still employed for encoding the set of X.509 types,
2761 but the base64 encoding is not used, as there is no requirement for an initial
2762 character encoding.

2763

2764

8 Examples

2765
2766

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

2767

8.1 BiometricSyntaxSets (CXER, DER)

2768
2769
2770
2771
2772

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

2773
2774
2775
2776

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.

2777

The BiometricSyntaxSets value encoded in basic XER:

2778

2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801

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

2802

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

2803
2804
2805
2806
2807
2808
2809
2810

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

2811

2812

The same abstract value encodes in 57 octets using DER:

2813
2814

```
3037A03530333A02BA103810104820102830106
8401FFA50F80048F3C0A0481078F530A03173B
```


2815 3BA60AA0088006672A090A040281040A0B0C0D

2816

2817 8.2 SignedData

2818 This example illustrates how to encode one or more biometric samples or templates for
2819 cryptographic enhancement to provide authentication of origin and data integrity services for
2820 biometric samples or templates.

2821

2822 The **SubjectPublicKeyInfo** value from the signer's X.509 Public Key Certificate is shown here in
2823 the hexadecimal representation of a DER encoded sequence:

2824

```
2825 308201B73082012C06072A8648CE3804013082011F028181
2826 00FD7F53811D75122952DF4A9C2EECE4E7F611B7523CEF44
2827 00C31E3F80B6512669455D402251FB593D8D58FABFC5F5BA
2828 30F6CB9B556CD7813B801D346FF26660B76B9950A5A49F9F
2829 E8047B1022C24FBBA9D7FEB7C61BF83B57E7C6A8A6150F04
2830 FB83F6D3C51EC3023554135A169132F675F3AE2B61D72AEF
2831 F22203199DD14801C70215009760508F15230BCCB292B982
2832 A2EB840BF0581CF502818100F7E1A085D69B3DDECBBCAB5C
2833 36B857B97994AFBBFA3AEA82F9574C0B3D0782675159578E
2834 BAD4594FE67107108180B449167123E84C281613B7CF0932
2835 8CC8A6E13C167A8B547C8D28E0A3AE1E2BB3A675916EA37F
2836 0BFA213562F1FB627A01243BCCA4F1BEA8519089A883DFE1
2837 5AE59F06928B665E807B552564014C3BFECF492A03818400
2838 028180476ACACB486186A153E25AE0E243FAAF0CD9105CF4
2839 DCF63412F36ABF671F53637E5F9FA7C5ADC78288FDB9FA3C
2840 FAFDEBFDD7A7C3FF2BD63D32F4773413EBD9EAB3CA03BA2D
2841 ED583187763181CB376954FD13F1F8E046D4E3D40652CA8D
2842 4645439A3ADB8D964F98F81E57147BDF4C009885CAD55D13
2843 B38DBAA2F9CBF13DC525F6
```

2844

2845 The biometric objects to be signed with the signer's private key associated with the public key
2846 provided above shown in the canonical form used as input to the message digest process used to
2847 create the digital signature on the signed content:

2848

```
2849 <BiometricObjects><BiometricObject><biometricHeader><version>0</version>
2850 <recordType><id>4</id></recordType><dataType><processed/></dataType><p
2851 urpose><audit/></purpose><quality>-1</quality><validityPeriod><notBefore>
2852 <1998.10.1</notBefore><notAfter>2008.10.1</notAfter></validityPeriod><
2853 format><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner></format><
2854 /biometricHeader><biometricData>0102030405060708090A0B0C0D0E0F11</biome
2855 tricData></BiometricObject><BiometricObject><biometricHeader><version>0
2856 </version><recordType><id>4</id></recordType><dataType><intermediate/><
2857 /dataType><purpose><enroll/></purpose><quality>50</quality><validityPer
2858 iod><notBefore>1998.10.2</notBefore><notAfter>2008.10.2</notAfter></val
2859 idityPeriod><format><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwn
2860 er></format></biometricHeader><biometricData>0102030405060708090A0B0C0D
2861 0E0F110102030405060708090A0B0C0D0E0F11</biometricData></BiometricObject
2862 ><BiometricObject><biometricHeader><version>0</version><recordType><id>
2863 4</id></recordType><dataType><raw/></dataType><purpose><enroll/></purpo
2864 se><quality>100</quality><validityPeriod><notBefore>1998.10.3</notBefore
2865 e><notAfter>2008.10.3</notAfter></validityPeriod><format><formatOwner><
2866 oid>2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><bio
2867 metricData>0102030405060708090A0B0C0D0E0F110102030405060708090A0B0C0D0E
2868 0F110102030405060708090A0B0C0D0E0F11</biometricData></BiometricObject><
2869 /BiometricObjects>
```

2870

2871 The complete integrity object represented as an XML document:

2872

2873

2874

2875

2876

2877

2878

2879

2880

2881

2882

2883

2884

2885

2886

2887

2888

2889

2890

2891

2892

2893

2894

2895

2896

2897

2898

2899

2900

2901

2902

2903

2904

2905

2906

2907

2908

2909

2910

2911

2912

2913

2914

2915

2916

2917

2918

2919

2920

2921

2922

2923

2924

2925

2926

2927

2928

2929

2930

2931

```
<?xml version="1.0" encoding="UTF-8"?>
<BiometricSyntaxSets>
  <integrityObjects>
    <biometricObjects>
      <BiometricObjects>
        <BiometricObject>
          <biometricHeader>
            <version> 0 </version>
            <recordType> <id> 4 </id> </recordType>
            <dataType> <processed/> </dataType>
            <purpose> <audit/> </purpose>
            <quality> -1 </quality>
            <validityPeriod>
              <notBefore> 1998.10.1 </notBefore>
              <notAfter> 2008.10.1 </notAfter>
            </validityPeriod>
            <format>
              <formatOwner>
                <oid> 2.23.42.9.10.4.2 </oid>
              </formatOwner>
            </format>
          </biometricHeader>
          <biometricData>
            0102030405060708090A0B0C0D0E0F11
          </biometricData>
        </BiometricObject>
        <BiometricObject>
          <biometricHeader>
            <version> 0 </version>
            <recordType> <id> 4 </id> </recordType>
            <dataType> <intermediate/> </dataType>
            <purpose> <enroll/> </purpose>
            <quality> 50 </quality>
            <validityPeriod>
              <notBefore> 1998.10.2 </notBefore>
              <notAfter> 2008.10.2 </notAfter>
            </validityPeriod>
            <format>
              <formatOwner>
                <oid> 2.23.42.9.10.4.2 </oid>
              </formatOwner>
            </format>
          </biometricHeader>
          <biometricData>
            0102030405060708090A0B0C0D0E0F11
            0102030405060708090A0B0C0D0E0F11
          </biometricData>
        </BiometricObject>
        <BiometricObject>
          <biometricHeader>
            <version> 0 </version>
            <recordType> <id> 4 </id> </recordType>
            <dataType> <raw/> </dataType>
            <purpose> <enroll/> </purpose>
            <quality> 100 </quality>
```

2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994

```
<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
  090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11
</biometricData>
</BiometricObject>

</BiometricObjects>
</biometricObjects>

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

2995
2996
2997
2998
2999
3000
3001

```
        </signedData>  
      </integrityBlock>  
  
    </integrityObjects>  
  </BiometricSyntaxSets>
```

3002 **8.3 EncryptedData (fixedKey)**

3003 This example illustrates how to encode a series of one or more biometric samples or templates
3004 for cryptographic enhancement. A group of three biometric objects is used here, though XCBF
3005 allows any number of objects to be included. The optional, cleartext biometric headers are not
3006 included in the example message. The group of three biometric objects is encrypted for privacy
3007 using a fixed Triple DES key.

3008

3009 As shown in this example, XCBF users can create and exchange arbitrary collections of biometric
3010 information to suit the needs of their security applications. This capability provides the application
3011 designer complete flexibility. The order of the biometric objects is determined in the application by
3012 the sender, allowing them to prioritize or order biometric information for purposes such as aging
3013 of data, or grouping records by quality or data type or entity.

3014

3015 Collections of useful biometric information include:

- 3016 • pairs of iris image templates for an individual; one for each eye
- 3017 • collections of paired iris image templates for a group of individuals
- 3018 • collections of finger print image templates, one per digit for an individual
- 3019 • sets of individual finger print image template collections for a group of persons
- 3020 • a collection of mixed biometric templates for an individual; say, retina, hand geometry, and
3021 DNA
- 3022 • collections of templates for groups of individuals, such as:
 - 3023 – all employees at work today, or
 - 3024 – all of the passengers on Flight 12, or
 - 3025 – all of the finger print samples collected on Tuesday

3026

3027 The hex encoding of a two key Triple DES key used to encrypt a series of three values of type
3028 **BiometricObject** is represented in hexadecimal notation as:

3029

3030
3031

```
D02523B3E561313B 511516297C52A846 D02523B3E561313B
```

3032 The biometric objects to be encrypted are shown here for ease of reading, and are not in the
3033 canonical form used as input to the encryption process in this example:

3034

3035
3036
3037
3038
3039
3040
3041
3042

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

3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055

3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100

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

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

```
<BiometricObject>  
  <biometricHeader>  
    <version> 0 </version>  
    <recordType> <id> 4 </id> </recordType>  
    <dataType> <raw/> </dataType>  
    <purpose> <enroll/> </purpose>  
    <quality> 100 </quality>  
    <validityPeriod>  
      <notBefore> 1998.10.3 </notBefore>  
      <notAfter> 2008.10.3 </notAfter>  
    </validityPeriod>  
    <format>  
      <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>  
    </format>  
  </biometricHeader>  
  <biometricData>  
    0102030405060708090A0B0C0D0E0F110102030405060708  
    090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11  
  </biometricData>  
</BiometricObject>  
</BiometricObjects>
```

3101 The complete XCBF privacy message formed after cryptographic processing of the series of
3102 three biometric objects:

3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164

```
<BiometricSyntaxSets>  
  <privacyObjects>  
    <privacyBlock>  
      <fixedKey>  
        <version> 84 </version>  
        <encryptedContentInfo>  
          <contentType> 1.2.840.113549.1.7.1 </contentType>  
          <contentEncryptionAlgorithm>  
            <algorithm> 1.2.840.113549.3.7 </algorithm>  
            <parameters>  
              <IV> 0102030405060708 </IV>  
            </parameters>  
          </contentEncryptionAlgorithm>  
        <encryptedContent>  
          2C75078EE885D7EE52321E80971A2B8E5C5A646B12CF2F90  
          7BAFA6FA16BDB6FA949898183762A05AB2A488B24D9397B2  
          E63D1885BAEFABD0A24E501EB612CE0118E435C4EEE70230  
          A0D74D766A795C0A23B454A3E6E5DFE60C0F31B3BABCDF23  
          5E1D4B2D7EDAD144BA38A63058ED211368A6887921973C7D  
          BCD1A53CBA2CD36819B488CDEB5008B546F0A7E9EE631D14  
          409A32EA7D784444EF2A8ABD0B54C99A5C0D5070A3A25D36  
          7303C359D4387D12DB2449FB73FC40D850B01555BAF8F86D  
          937A60256E0FDDAF52690CCF8D76770EBE7E359EFF6510A0  
          E39B09E6A8EA0FB844522364E66696965B37382061152D54  
          69332C7D1BF9C2A311D117583AE740DED8849777C3CE56DE  
          EABA687B936693292858CA67CD5D7C2BCD918A736B044173  
          1DDBBB077C348CA6524BEA30FADED63CE76BB47F43232A3C  
          5FA6ABFFFE5234C108C6B903DB724B478CD647F52E98A5A28  
          071E881321CA7AAD4F7F1F911C4666F5746A1090FB076251  
          24F4E521BCD78C61C6AB1AC0C142D9CB32E08B1B0C7E8F51  
          07826C883D92963F547F62373CD6CACD791877293E2937EA  
          E93D78FAD09479990BC8488D067C8203A43D9AC132A75712  
          A1A86A4E527FBE941E7E7CAC8CCEADCA72C9853D5FDBFECF  
          972450E13461DD688AB412FAABC912838ED5850A9FF85FC4  
          2D311834670D11FBBB7BE732D7FC44D0326AFD17B92A8353  
          747E3A04065E553901A8C5EA531AC644D204F9209EED151  
          FF7D623DE5671FC4C07325593D0D442E66C872CC7BB214D5  
          88F01A90C8ECFC19BC9DEA06065C954B673B101EF8FC8FFC  
          E97DBBDFABEE9AE20FFE0927E4F59928CD9E2A0F1D538527  
          2A5C78CE1E6A669B18AAD2498731730FFBDFD72D13EB390E  
          82C402F3033E0201089DF7758F1C76E1578AE7CF6AC4742  
          631B1743141CD0A29D32F6B7D50C37EC8BC55754FA6D7FF5  
          5B0E1B0E075CF1F7EB0DB0468152BE079703BA71485CA891  
          7EA7CF6ED69CD0AC88AB5E93824B337FDB748542D6A2D1B9  
          96C4634CA52536A25C123D9791173F8668404AE30CA701F0  
          5FF77B976369F7C1F48B834952082BA92E7772C717305E70  
          8E51C89E445C88423E378262315E951A207C30B5E03E805C  
          9C538E9856DBDCF84489BF58757E332BFA01625F76C04A23  
          2271BF4FC985209B51A7CC0E747426007617649ABC75818C  
          A68911E497266E717042EEE6747D9A874F9F174713F8751F  
          2AE5B3D7991299A83F85E78F99C5B01B1FCD8A9EC072F763  
          669854D74CC915611181440305AB483E4A2BE21C516561DC  
          B6E25C5D1DCD5FF84B3740B3BB337BF78585A8395EAB3357  
          132541794115D48BCC37B5B0405201C9CDA503C2641E0283  
          3651CD5E55F442C6E7E8D1EA8D83F9FAE370DA6ADC653651  
          4441C6831E4F1B42A056672EC5152B8BA0D31C6F3603801D  
          C50028562D81055915088D6798E4900481B1C86D35888AC0  
          6E9B61CEBA302BD1A0CB0AC002B640455B14072F9E654E87  
          43AE15F7C46DE69BB2F11408CE22133B9C04D3F097300780  
          DCF653414DA95BF7A1DE0B57E72616BC014108828CE7B027  
          5C391516A9D9AD26A59F9D2D46BE59D93B7DD2525CFAC948
```

3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183

```
9DDA026F4CF7BADF4552F47B64CC1C65D705697251DC92B9
486A53159E9AD9D9C149EDAF78636F6F090763E6C826F9FE
EBEE49D8FCCC526BDBB4B2513FB355DC05994DFD9D1B581D
DD69C904584E5BDF9593C9A4AFD9E1FD2FA7056755F9079
4F1988F103C7F1EEE9A8BE719B401078E504F56E3B6931C3
AC1A4392860367E46F008FFB919B8F1DF0B0D0CE41BF1786
9313B14B0E556F94E5573106B7050862B4158BB1C87CEB29
B939873D514474D72840B96E70D2661249B856DF2775AE7F
1A3378AD9030FD10A1CB5E8EF150808DEC94101449DAADE6
4D16EA79E4578FA996AEBFA1E3918A485EFB85364282CCA4
073A565C64B8A8704CC95ADA4A9B8796679BD89DAA6A7B0E
01E4903080875A3208769CE4B1BDD1B874F5B4E2607FBB72
2411F0CAFCE14D7FA870EBE87903D5B583D5C231F95912D5
    </encryptedContent>
  </encryptedContentInfo>
</fixedKey>
</privacyBlock>
</privacyObjects>
</BiometricSyntaxSets>
```

3184 **Appendix A. Acknowledgments**

3185 The following individuals were members of the committee during the development of this
3186 specification:

3187

- 3188 • Tyky Aichelen (Chair), IBM
- 3189 • Lieutenant John Aerts, LA County Information Systems Advisory Body
- 3190 • Karl Best, OASIS
- 3191 • Taylor Boon, BioNetrix Systems
- 3192 • Jamie Clark, OASIS
- 3193 • Robin Cover, Isogen
- 3194 • Ed Day, Objective Systems
- 3195 • Dr. Paul Gérôme, Individual Member
- 3196 • Phillip H. Griffin, Individual Member
- 3197 • Todd Harbour, FGM
- 3198 • William Koenig, Bank of America
- 3199 • John Larmouth, Individual Member
- 3200 • Monica Martin, Sun Microsystems
- 3201 • John Messing, American Bar Association
- 3202 • Michael Nguyen, The Infocomm Development Authority of Singapore
- 3203 • Rob Philpott, RSA Security
- 3204 • Rick Randall, Booz Allen Hamilton
- 3205 • Stefan Ri, Microsoft
- 3206 • William Riippi, MTG Management Consultants, LLC.
- 3207 • Darran Rolls, Waveset
- 3208 • Bancroft Scott, OSS Nokalva
- 3209 • Clifford Thompson, Individual Member
- 3210 • Paul Thorpe, OSS Nokalva
- 3211 • Alessandro Triglia, OSS Nokalva
- 3212 • Per Vorm, BEA Systems

3213

Appendix B. Revision History

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

3214

Appendix C. Notices

3216 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
3217 that might be claimed to pertain to the implementation or use of the technology described in this
3218 document or the extent to which any license under such rights might or might not be available;
3219 neither does it represent that it has made any effort to identify any such rights. Information on
3220 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
3221 website. Copies of claims of rights made available for publication and any assurances of licenses
3222 to be made available, or the result of an attempt made to obtain a general license or permission
3223 for the use of such proprietary rights by implementers or users of this specification, can be
3224 obtained from the OASIS Executive Director.

3225 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
3226 applications, or other proprietary rights, which may cover technology that may be required to
3227 implement this specification. Please address the information to the OASIS Executive Director.

3228 Copyright © 2003 OASIS Open, Inc. All Rights Reserved.

3229 This document and translations of it may be copied and furnished to others, and derivative works
3230 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
3231 published and distributed, in whole or in part, without restriction of any kind, provided that the
3232 above copyright notice and this paragraph are included on all such copies and derivative works.
3233 However, this document itself does not be modified in any way, such as by removing the
3234 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
3235 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
3236 Property Rights document must be followed, or as required to translate it into languages other
3237 than English.

3238 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
3239 successors or assigns.

3240 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3241 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
3242 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
3243 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
3244 PARTICULAR PURPOSE.