



XML Common Biometric Format

Committee Specification, 20 January 2003

4 **Document identifier:**

5 *{Committee Specification}-{XML Common Biometric Format}-{XCBF}-{01}* ([PDF](#), [Word](#))

6 **Location:**

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

8 **Editor:**

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

10 **Contributors:**

11 Tyky Aichelen, IBM

12 Ed Day, Objective Systems

13 Dr. Paul Gérôme, AULM

14 Phillip H. Griffin (chair), Griffin Consulting

15 John Larmouth, Larmouth T&PDS Ltd

16 Monica Martin, Drake Certivo

17 Bancroft Scott, OSS Nokalva

18 Paul Thorpe, OSS Nokalva

19 Alessandro Triglia, OSS Nokalva

20

21 **Abstract:**

22 Biometrics are automated methods of recognizing a person based on physiological or
23 behavioral characteristics. They are used to recognize the identity of an individual, or to
24 verify a claimed identity. This specification defines a common set of secure XML
25 encodings for the patron formats specified in CBEFF, the Common Biometric Exchange
26 File Format (NISTIR 6529) [\[17\]](#). These XML encodings are based on the ASN.1 schema
27 defined in ANSI X9.84 Biometric Information Management and Security [\[14\]](#). They
28 conform to the canonical variant of the XML Encoding Rules (XER) for ASN.1 defined in
29 ITU-T Rec. X.693, and rely on the security and processing requirements specified in the
30 X9.96 XML Cryptographic Message Syntax (XCMS) [\[15\]](#) and X9.73 Cryptographic
31 Message Syntax (CMS) [\[13\]](#) standards .

32 **Status:**

33 If you are on the xcbf@lists.oasis-open.org list for committee members, send comments
34 there. If you are not on that list, subscribe to the xcbf-comment@lists.oasis-open.org list
35 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.

39 Table of Contents

40	1	Introduction	4
41	2	Terminology	5
42	3	Acronyms and Abbreviations	6
43	4	Glossary	7
44	5	X9.84 and BioAPI 1.1 Interoperability	9
45	5.1	BiometricSyntaxSets	9
46	5.1.1	BiometricObjects	10
47	5.1.2	IntegrityObjects	26
48	5.1.3	PrivacyObjects	34
49	5.1.4	PrivacyAndIntegrityObjects	44
50	6	References	46
51	6.1	Normative	46
52	7	XCBF Schema	48
53	7.1	X9-84-Biometrics Module	48
54	7.2	X9-84-CMS Module	52
55	7.3	X9-84-Identifiers Module	55
56	8	Examples	64
57	8.1	BiometricSyntaxSets (cXER, DER)	64
58	8.2	SignedData	65
59	8.3	EncryptedData (fixedKey)	68
60		Appendix A. Acknowledgments	72
61		Appendix B. Revision History	73
62		Appendix C. Notices	74
63			

64 1 Introduction

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

71

72 These XML encodings are based on the ASN.1 [2] [3] [4] [5] schema defined in ANSI
73 X9.84:2003 *Biometric Information Management and Security*. They conform to the canonical
74 variant of the XML Encoding Rules (XER) for ASN.1 defined in ITU-T Rec. X.693 [7], and rely on
75 the same security and processing requirements specified in X9.96 *XML Cryptographic Message*
76 *Syntax (XCMS)*. Values of the Biometric Information Record (BIR) defined in ANSI/INCITS 358-
77 2002 - *Information technology - BioAPI Specification* [16] that can be represented in the X9.84
78 biometric object format can also be represented using XML markup and secured using the
79 techniques in this standard.

80

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

86

2 Terminology

87

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

89

3 Acronyms and Abbreviations

90

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

91

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.

94 5 X9.84 and BioAPI 1.1 Interoperability

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

99
100 This document describes the process for translating between an X9.84 **BiometricObject** and a
101 BioAPI-1.1 Biometric Information Record (BIR). The X9.84 schema is the same as the schema
102 defined in this standard and provides a common means of representing in XML markup the binary
103 values described in the X9.84 and BioAPI-1.1 standards. Once BIR format values are
104 represented as values of type **BiometricObject** they can be secured using the techniques
105 described in this standard.

106 5.1 BiometricSyntaxSets

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

109

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

111

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

114

```
115    BiometricSyntax ::= CHOICE {  
116        biometricObjects              BiometricObjects,  
117        integrityObjects             IntegrityObjects,  
118        privacyObjects               PrivacyObjects,  
119        privacyAndIntegrityObjects PrivacyAndIntegrityObjects  
120    }
```

121

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

123

124 biometricObjects	a set of unprotected biometric values
125 integrityObjects	a digitally signed set of biometric values
126 privacyObjects	an encrypted set of biometric values
127 privacyAndIntegrityObjects	a digitally signed and encrypted set of biometric values

128

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

135
136 The BioAPI specification defines a single format, a BIR, composed of three fields: a record
137 *Header*, an opaque *BiometricData* field, and an optional *Signature*. Ignoring the *Signature* field,
138 the BIR format corresponds closely to the single unprotected biometric value defined in this
139 standard as the **BiometricSyntax** choice alternative **biometricObjects** when it is constrained to
140 contain a single **BiometricObject**. There is no definition for representing sets of biometric
141 records in BioAPI.

142
143 The other **BiometricSyntax** choice alternatives are not supported in the BioAPI specification.
144 These alternatives are cryptographic messages used to provide integrity, authentication and
145 privacy services. When a BIR value is represented in **biometricObjects** format, XCBF security
146 services can be used to protect BioAPI biometric information.

147
148 A value of type **BiometricSyntaxSets** can be represented in XML markup as
149

```
150 <BiometricSyntaxSets>
151   <BiometricSyntax>
152     ...
153   </BiometricSyntax>
154 </BiometricSyntaxSets>
```

155
156 Here an ellipsis is used as a placeholder for the elements of the choice alternative of type
157 **BiometricSyntax** which are not shown.

5.1.1 BiometricObjects

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

```
162
163 BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject
164
165 BiometricObject ::= SEQUENCE {
166   biometricHeader  BiometricHeader,
167   biometricData    BiometricData
168 }
```

169
170 All of the cryptographic processing in this standard is performed on a value of type
171 **EncodedBiometricObjects**. This is a value of type **BiometricObjects** in its encoded form.

172
173

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

174
175 The ASN.1 definition above, in the context of this standard, implies that a value of type
176 **BiometricObjects** is encoded prior to its inclusion in the complete encoding of the enclosing
177 message. This allows the simultaneous use of two different sets of ASN.1 encoding rules for
178 different parts of the same message, namely Canonical XER (CXER) for the **BiometricObjects**
179 value and basic XER (XER) or Variant XER (VXER) for the enclosing message. The rationale for

180 using two different sets of encoding rules is that cryptographic processing requires that
181 **BiometricObjects** be encoded in a canonical manner, while there is no such requirement for the
182 message as a whole. Therefore other, more-flexible encoding rules (either XER or VXER in this
183 standard) can be used to encode the message as a whole.

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

```
189     typedef struct bioapi_bir {  
190         BioAPI_BIR_HEADER             Header;  
191         BioAPI_BIR_BIOMETRIC_DATA_PTR BiometricData;  
192         BioAPI_DATA_PTR               Signature;  
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 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 **5.1.1.2 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

```
<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> 2 </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_FOR_VERIFICATION_ONLY	4
enrollIdentify	5	BioAPI_PURPOSE_ENROLL_FOR_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       (-..100,...)
```

463

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

465

```
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 one hundred. X9.84 specifies named integer constants for the lowest quality, highest quality, quality not set, and quality not supported. These values are presented in the following table:

470

471

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 as

475

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

476

477

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

```
535
536 BIOMETRIC ::= CLASS {
537     &name BIOMETRIC-IDENTIFIER UNIQUE,
538     &Type OPTIONAL
539 }
540 WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
```

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

```
546
547 BIOMETRIC-IDENTIFIER ::= CHOICE {
548     oid OBJECT IDENTIFIER, -- complete object identifier
549     id RELATIVE-OID        -- object identifier fragment
550 }
```

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

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

```
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 This markup associates the biometric data with a specific vendor product using a complete object
 567 identifier value. The optional **formatType** component is present and contains a value of a user
 568

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  
596   ...  -- expect additional vendor specific formats --  
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
608 **CBEFFidFormats**, to identify a value of type **BiometricInformationSets**. This type biometric
609 serves as a placeholder for possible future standardization, which will identify commonly
610 accepted processing algorithms and matching methods.

611

```
612 CBEFF-Formats BIOMETRIC ::= {
```

```

613     CBEFFoidFormats | -- Complete object identifiers
614     CBEFFidFormats, -- Object identifier fragments
615
616     ... -- Expect additional CBEFF vendor specific formats --
617 }
618
619 CBEFFoidFormats BIOMETRIC ::= {
620     { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },
621
622     ... -- Expect other objects --
623 }
624
625 CBEFFidFormats BIOMETRIC ::= {
626     { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },
627
628     ... -- Expect other objects --
629 }
```

630

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

632

```

633 BiometricInformationSets ::= SEQUENCE SIZE(1..MAX) OF BiometricInformation
634
635 BiometricInformation ::= SEQUENCE {
636     processingAlgorithms ProcessingAlgorithms OPTIONAL,
637     matchingMethods MatchingMethods OPTIONAL
638 }
639 (ALL EXCEPT({ -- none; at least one component is present -- }))
```

641

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

645

```

646 ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation
647
648 ProcessingInformation ::= SEQUENCE {
649     id      BIOMETRIC.&name({ProcessingAIDs}),
650     parms   BIOMETRIC.&Type({ProcessingAIDs}{@id}) OPTIONAL
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 ::= {
```

```
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-A3Vision            RelOID ::= { 16 }
721    ibia-NEC                 RelOID ::= { 17 }
722    ibia-STMicroelectronics RelOID ::= { 18 }
723

```

724
725 These identifiers are associated with a restricted sixteen bit integer value.
726

```

727 IBIAidFormats BIOMETRIC ::= {
728     { BIOMETRIC id : ibia-SAFLINK           DATA BirInt16 } |
729     { BIOMETRIC id : ibia-Bioscrypt         DATA BirInt16 } |
730     { BIOMETRIC id : ibia-Visionics         DATA BirInt16 } |
731     { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
732     { BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
733     { BIOMETRIC id : ibia-Veridicom         DATA BirInt16 } |
734     { BIOMETRIC id : ibia-CyberSIGN         DATA BirInt16 } |
735     { BIOMETRIC id : ibia-eCryp             DATA BirInt16 } |
736     { BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
737     { BIOMETRIC id : ibia-SecuGen           DATA BirInt16 } |
738     { BIOMETRIC id : ibia-PreciseBiometric  DATA BirInt16 } |
739     { BIOMETRIC id : ibia-Identix           DATA BirInt16 } |
740     { BIOMETRIC id : ibia-DERMALOG          DATA BirInt16 } |
741     { BIOMETRIC id : ibia-LOGICO            DATA BirInt16 } |
742     { BIOMETRIC id : ibia-NIST              DATA BirInt16 } |
743     { BIOMETRIC id : ibia-A3Vision          DATA BirInt16 } |
744     { BIOMETRIC id : ibia-NEC               DATA BirInt16 } |
745     { BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 },
746
747     ... -- Expect others --
748 }

```

749
750 Note that additional registry entries are expected and that the associated type is optional in XCBF
751 and need not be present.

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

756

IBIAoidFormats BIOMETRIC ::= { { BIOMETRIC oid : id-ibia-SAFLINK	DATA Any }
---	------------

```

759 { BIOMETRIC oid : id-ibia-Bioscrypt           DATA Any } |
760 { BIOMETRIC oid : id-ibia-Visionics          DATA Any } |
761 { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
762 { BIOMETRIC oid : id-ibia-IridianTechnologies DATA Any } |
763 { BIOMETRIC oid : id-ibia-Veridicom          DATA Any } |
764 { BIOMETRIC oid : id-ibia-CyberSIGN          DATA Any } |
765 { BIOMETRIC oid : id-ibia-eCryp              DATA Any } |
766 { BIOMETRIC oid : id-ibia-FingerprintCardsAB DATA Any } |
767 { BIOMETRIC oid : id-ibia-SecuGen            DATA Any } |
768 { BIOMETRIC oid : id-ibia-PreciseBiometric   DATA Any } |
769 { BIOMETRIC oid : id-ibia-Identix            DATA Any } |
770 { BIOMETRIC oid : id-ibia-DERMALOG          DATA Any } |
771 { BIOMETRIC oid : id-ibia-LOGICO            DATA Any } |
772 { BIOMETRIC oid : id-ibia-NIST               DATA Any } |
773 { BIOMETRIC oid : id-ibia-A3Vision          DATA Any } |
774 { BIOMETRIC oid : id-ibia-NEC               DATA Any } |
775 { BIOMETRIC oid : id-ibia-STMicroelectronics DATA Any },
776
777     ... -- Expect additional vendor specific formats --
778 }
779
780 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained

```

781

782 5.1.1.1.7.4 X9-Formats

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

784

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

786

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

793

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

800

801

```
802 X9-Formats BIOMETRIC ::= {
803     X9oidFormats |
804     X9idFormats,
805
806     ... -- Expect additional X9 vendor specific formats --
807 }
808
```

```
809 X9oidFormats BIOMETRIC ::= {  
810     ... -- Expect X9 assigned objects --  
811 }  
812  
813 X9idFormats BIOMETRIC ::= {  
814     ... -- Expect X9 assigned objects of the form { 2 x } --  
815 }
```

816

817 5.1.1.2 BiometricData

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

820

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

822

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

825

```
826 typedef uint8 BioAPI_BIR_BIOMETRIC_DATA;
```

827

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

830

831 5.1.2 IntegrityObjects

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

835

```
836 IntegrityObjects ::= SEQUENCE {  
837     biometricObjects EncodedBiometricObjects,  
838     integrityBlock    IntegrityBlock  
839 }
```

840

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

845

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

849

```
850 IntegrityBlock ::= CHOICE {
```

```

851     digitalSignature      DigitalSignature,
852     messageAuthenticationCode MessageAuthenticationCode,
853     signedData            SignedData,
854     authenticatedData    AuthenticatedData
855 }

```

856

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

858

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

859

860 5.1.2.1 DigitalSignature

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

```

865   DigitalSignature ::= SEQUENCE {
866     algorithmID  SignatureAlgorithmIdentifier,
867     signature     OCTET STRING
868       ( CONSTRAINED BY { -- signature on a value of --
869         EncodedBiometricObjects } )
870   }

```

871

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

875

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

878

```

879 <integrityBlock>
880   <digitalSignature>
881     <algorithmID>
882       <algorithm>1.2.840.10040.4.3</algorithm>
883       <parameters><NoIV/></parameters>
884     </algorithmID>
885     <signature>
886       DE340 ... B0123DF
887     </signature>
888   </digitalSignature>
889 </integrityBlock>

```

890

891

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

898 **5.1.2.1.1 Digital Signature Process**

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

903

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

910

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

915

916 **5.1.2.1.2 Digital Signature Verification**

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

924 **5.1.2.2 MessageAuthenticationCode**

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

929

930 **MessageAuthenticationCode** ::= SEQUENCE {
931 keyName OCTET STRING OPTIONAL,
932 algorithmID MACAlgorithmIdentifier,
933 mac OCTET STRING

```

934             ( CONSTRAINED BY { -- MAC or HMAC on a value of --
935               EncodedBiometricObjects } )
936
937
938 A MessageAuthenticationCode provides a way to verify the integrity of biometric information
939 using a secret authentication key. This secret key is shared between a sender and recipient. An
940 HMAC is a message authentication method based on a cryptographic hash function, a keyed-
941 hash method. The cryptographic strength of an HMAC depends on the strength of the underlying
942 hash function. For this reason, the Secure Hash Algorithm (SHA1) is widely used.
943
944 For both MAC and HMAC, cryptographic keys shall be chosen at random when created, and shall
945 be protected and kept secret, and exchanged securely. The minimum length of the key used with
946 HMAC depends on the choice of underlying hash function. Good security practices demand that
947 keys be refreshed periodically to guard against weaknesses in keys and to minimize exposure
948 from an attack.
949
950 A value of the messageAuthenticationCode choice alternative of the integrityBlock
951 component of type IntegrityObjects can be represented in XML markup as
952
953 <integrityBlock>
954   <messageAuthenticationCode>
955     <keyName> 9FCD...AB45 </keyName>
956     <algorithmID>
957       <algorithm>1.3.6.1.5.5.8.1.2</algorithm>
958     </algorithmID>
959     <mac>
960       DEA7B ... 59ABD3
961     </mac>
962   </messageAuthenticationCode>
963 </integrityBlock>
964
965 This markup uses the message authentication code choice alternative. The hashed MAC with
966 SHA1 algorithm and a shared secret key are used to compute an HMAC on a value of
967 EncodedBiometricObjects. An ellipsis is used as a placeholder for part of the HMAC results.
968
969 

### 5.1.2.2.1 Message Authentication Process


970 A sender prepares a value of type EncodedBiometricObjects, a named cryptographic key
971 previously created at random and known by the recipient, and uses these as input to a MAC or
972 HMAC process. This results in a message authentication code over the specified biometric
973 information. The biometric information and processing results are sent to a recipient who shares
974 the secret key used in the message authentication code process.
975
976 To verify the message authentication code, the user computes a MAC or HMAC on the biometric
977 information using the same shared secret key identified by its key name, and compares this result
978 to the message authentication value received to determine the integrity of the biometric
information.

```

979 **5.1.2.3 SignedData**

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

982

```
983 SignedData ::= SEQUENCE {
984     version          CMSVersion,
985     digestAlgorithms DigestAlgorithmIdentifiers,
986     encapContentInfo EncapsulatedContentInfo,
987     certificates      [0] CertificateSet OPTIONAL,
988     crls             [1] CertificateRevocationLists OPTIONAL,
989     signerInfos       SignerInfos
990 }
```

991

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

993

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.
crls	An optional set of one or more X.509 certificate revocation lists (CRLs). When present, there may be more CRLs or less than needed to determine whether or not the certificate of the signer is valid.
signerInfos	A set of information for each signer of the content. This set contains only one element since there is only one signer of the content in this standard.

994

995

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

996

```
997 SignerInfo ::= SEQUENCE {
998     version          CMSVersion,
999     sid              SignerIdentifier,
1000    digestAlgorithm DigestAlgorithmIdentifier,
1001    signatureAlgorithm SignatureAlgorithmIdentifier,
1002    signature        SignatureValue
1003 }
```

1004

1005 The **SignerIdentifier** type is used to identify the public key certificate associated with the private
1006 key used to create the **signature** component of **SignerInfo**. This type is defined in XCBF as a
1007 choice type having only one alternative:

1008

```
1009 SignerIdentifier ::= CHOICE {  
1010   certHash [1] EXPLICIT Hash  
1011 }
```

1012

1013 The **certHash** choice alternative of **SignerIdentifier** provides a single, simple mechanism that
1014 allows any type of digital certificate to be identified. The hash is computed over the complete
1015 encoding of the certificate. This allows any type of certificate, regardless of its format or encoding,
1016 to be identified.

1017

1018 When X.509 certificates or attribute certificates are used, the hash will be computed over the
1019 complete DER [6] encoding of the certificate, as X.509 only supports these encoding rules. When
1020 X.9.68 domain certificates are used, the hash may be computed over the DER or XER encoding
1021 of the certificate, as either encoding format is supported by that standard. In any case, such
1022 details are hidden from an application if the certificates is properly treated as an opaque series of
1023 octets.

1024

```
1025 Hash ::= CHOICE {  
1026   ietf      CertHash, -- SHA-1 hash of entire certificate  
1027   withAlgID DigestInfo  
1028 }  
1029  
1030 CertHash ::= OCTET STRING (ENCODED BY sha-1)  
1031  
1032 DigestInfo ::= SEQUENCE {  
1033   hashAlgorithm DigestAlgorithmIdentifier,  
1034   digest          OCTET STRING  
1035 }
```

1036

1037 5.1.2.3.1 Message digest and signature process

1038 A message digest is used to create the digital signature carried in the **SignerInfo** component of
1039 **SignedData**. The message digest is calculated using the algorithm and parameters specified in
1040 the **digestAlgorithm** component of **SignerInfo**, and the value of the **eContent** component of
1041 **EncapsulatedContentInfo**. The **eContentType** component of **EncapsulatedContentInfo**
1042 identifies the type of value being signed. This is always the object identifier value **id-data** in
1043 XCBF.

1044

1045 When a value of type **SignedData** is represented as XML markup, the starting and ending
1046 **eContent** tags are excluded from the message digest process. Only the "value" portion of the
1047 complete canonical XER encoding of **eContent** is digested. The **<eContent>** and **</eContent>**
1048 tags are excluded from the message digest process, and the digest is calculated starting with the
1049 **<BiometricObjects>** tag and ending with the **</BiometricObjects>** tag.

1050

1051 The result of the message digest process is then digitally signed using the signer's private key
1052 and the signature algorithm and parameters specified in the **signatureAlgorithm** component of
1053 **SignerInfo**. The result of the signature process becomes the value of the **signature** component
1054 of the **SignerInfo** component of **SignedData**.

1055

1056 **5.1.2.3.2 Signature Verification**

1057 To verify the signature in a signed data choice alternative of the **integrityBlock** component of
1058 type **IntegrityObjects**, a message digest is computed over the complete canonical XER
1059 encoding of the value of the **biometricObjects** component of type **IntegrityObjects**. This digest
1060 is computed using the **digestAlgorithm** component of type **SignerInfo**.

1061

1062 The public key of the signer is the signature verification key. This key is used to verify the digital
1063 signature on the signed data created with the signer's private key. The signature is carried in the
1064 **SignerInfo** component of type **SignedData**. The value of the **sid** component of **SignerInfo**
1065 identifies the public key certificate associated with the private key used to create the digital
1066 signature on the signed data.

1067

1068 The message digest value computed by the signature verifier is compared to the value of the
1069 hash obtained from applying the signer's public key to the **signature** component of type
1070 **SignerInfo** to determine if this signature is valid. There is only one signer of the content in this
1071 standard.

1072

1073 Complete trust in the validity of the signature on the signed data by the signer must be
1074 determined by validation of the chain of certificates associated with the signer's public key
1075 certificate. The optional **certificates** and **crls** components of type **SignedData** may be used by
1076 the signer to provide information needed to validate the signer's certificate.

1077 **5.1.2.4 AuthenticatedData**

1078 The **messageAuthenticationCode** choice alternative of the **integrityBlock** component of type
1079 **IntegrityObjects** is a value of type **AuthenticatedData**. This sequence type provides a MAC with
1080 key establishment and is defined as

1081

```
1082 AuthenticatedData ::= SEQUENCE {
1083   version      CMSVersion,
1084   recipientInfos RecipientInfos,
1085   macAlgorithm  MACAlgorithmIdentifier,
1086   encapContentInfo EncapsulatedContentInfo,
1087   mac          MessageAuthenticationCode
1088 }
```

1089

1090 Type **AuthenticatedData** uses a key transport mechanism to convey a one-time MAC key along
1091 with biometric information. Since a key agreement mechanism is not supported, only data
1092 integrity is provided by use of this type and not origin authentication. There is only a single
1093 recipient of this encrypted MAC key present in the **recipientInfos** component of type
1094 **AuthenticatedData**, and the optional content is not present in the **encapContentInfo**
1095 component.

1096

1097 A message authentication code is calculated on biometric information. This biometric information
1098 is a value of type **EncodedBiometricObjects**, which is a value of type **BiometricObjects** in its
1099 encoded form. The calculation is performed using the MAC algorithm and any associated
1100 algorithm parameters indicated in the **macAlgorithm** component of type **AuthenticatedData**,
1101 the biometric information, and an authentication key that is conveyed in the **recipientInfos**
1102 component. The result of this calculation is a message authentication code, which becomes the
1103 value of the **mac** component.

1104

1105 A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented
1106 in XML markup as

1107

```
1108 <privacyObjects>
1109   <privacyBlock>
1110     ...
1111
1112   </privacyBlock>
1113 </privacyObjects>
```

1115

1116 This markup illustrates the wrapper for a typical privacy object. The optional privacy object
1117 biometric headers are not present. An ellipsis is used as a placeholder and the details of the
1118 privacy block choice alternative are not shown.

1119

1120 5.1.2.5 Biometric Certificate Extensions

1121 Digital signature data integrity protection and origin authenticaion can be achieved by including
1122 biometric information in digital certificates. A set of one or more biometric reference templates
1123 can be cryptographically bound to the public key associated with the private key of an entity, by
1124 including a value of type **EncodedBiometricObjects** in a certificate extension. In the XCBF
1125 standard, biometric certificate extension values can be encoded in either binary or XML markup
1126 formats.

1127

1128 XCBF supports the version three certificates and version two attribute certificates defined in the
1129 X.509 standard and the compact domain certificate format defined in X9.68. On extension is
1130 defined for use in each standard, and these are defined as

1131

```
1132 biometricTemplates EXTENSION ::= {
1133   SYNTAX          EncodedBiometricObjects -- DER or cXER --
1134   IDENTIFIED BY  x509-biometricTemplates
1135 }
1136
1137 domainBiometricTemplates PRIVATE-X ::= {
1138   NAME    oid : x968-biometricTemplates
1139   TYPE    EncodedBiometricObjects -- DER or cXER --
1140 }
```

1141

1142 When biometric information is included in a certificate extension stored in a certificate repository,
1143 the repository becomes a biometric storage subsystem, and the biometric information may need
1144 to be protected by encryption or other means. Measures should be taken to prevent an attacker

1145 from using a certificate repository as a large, searchable public database of biometric reference
1146 templates that could be used to find templates that match a given biometric sample. Finding such
1147 a match would allow an attacker to focus its efforts on that user.

1148 5.1.3 PrivacyObjects

1149 The **privacyObjects** choice alternative of type **BiometricSyntax** is a value of type
1150 **PrivacyObjects**. This type is defined as a sequence of two components, **biometricHeaders** and
1151 **privacyBlock**.

1152

```
1153 PrivacyObjects ::= SEQUENCE {  
1154   biometricHeaders  BiometricHeaders  OPTIONAL,  
1155   privacyBlock      PrivacyBlock  
1156 }
```

1157

1158 The **biometricHeaders** component is a series of one or more values of type **BiometricHeader**.

1159

```
1160 BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
```

1161

1162 This optional **biometricHeaders** component is not protected by encryption and should be present
1163 only when a privacy object is used in a secure environment, or when the information contained in
1164 the **biometricHeaders** component does not compromise security or assist an attacker. In a
1165 secure setting these biometric headers may be used as a convenience, to assist in searches of
1166 biometric information and in database management operations.

1167

1168 The encrypted content in the privacy block contains a series of one or more values of type
1169 **BiometricObject**, including their biometric headers. To be useful, the **biometricHeaders**
1170 component need only provide an indication of the information contained in the encrypted privacy
1171 block. But this component need not contain exactly the same information as the headers in the
1172 encrypted privacy block, and may contain only a single **BiometricHeader** value when present.
1173 The **biometricHeaders** component is not protected in any way.

1174

1175 The **privacyBlock** component of type **PrivacyObjects** offers three choice alternatives, **fixedKey**,
1176 **namedKey** and **establishedKey**.

1177

```
1178 PrivacyBlock ::= CHOICE {  
1179   fixedKey        EncryptedData,  
1180   namedKey        NamedKeyEncryptedData,  
1181   establishedKey EnvelopedData  
1182 }
```

1183

1184 The **fixedKey** and **namedKey** choice alternatives are based on the **EncryptedData** type. The
1185 **establishedKey** alternative is based on type **EnvelopedData**. Each of these alternatives has
1186 different characteristics, and the alternative chosen will depend upon application requirements
1187 and the key management scheme being used.

1188

1189 A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented
1190 in XML markup as

1191

```
1192 <privacyObjects>
1193   <privacyBlock>
1194     ...
1195   ...
1196   </privacyBlock>
1197 </privacyObjects>
```

1199

1200 This markup illustrates the wrapper for a typical privacy object. The optional privacy object
1201 biometric headers are not present. An ellipsis is used as a placeholder and the details of the
1202 privacy block choice alternative are not shown.

1203 5.1.3.1 Encrypted Content Information

1204 All three of the privacy block choice alternatives contain a value of type **EncryptedContentInfo**
1205 defined as

1206

```
1207 EncryptedContentInfo ::= SEQUENCE {
1208   contentType             ContentType,
1209   contentEncryptionAlgorithm ContentEncryptAlgorithmIdentifier,
1210   encryptedContent         [0] EncryptedContent
1211 }
```

1212

1213 The **contentType** component identifies the type of encrypted content. In XCBF, the type of
1214 encrypted content is always a value of **EncodedBiometricObjects**, a series of one or more
1215 values of type **BiometricObject** encoded using the XML Encoding Rules. The type of encrypted
1216 content is identified as ordinary data by the information object identifier value **id-data**, defined in
1217 the PKCS #7 Cryptographic Message Syntax Standard [19]. The encoding of this component is

1218

```
1219 <contentType> 1.2.840.113549.1.7.1 </contentType>
```

1220

1221 The **contentEncryptionAlgorithm** component identifies the content encryption algorithm and
1222 any associated parameters used to encrypt and decrypt the **EncodedBiometricObjects**. This
1223 content encryption algorithm is a value of type **ContentEncryptionAlgorithmIdentifier** defined
1224 as

1225

```
1226 ContentEncryptAlgorithmIdentifier ::= 
1227   AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
```

1228

1229 The definition of type **ContentEncryptionAlgorithmIdentifier** is based on the parameterized
1230 type **AlgorithmIdentifier {}** and the information object set **ContentEncryptionAlgorithms**,
1231 defined as

1232

```

1233 ContentEncryptionAlgorithms ALGORITHM ::= {
1234     { OID des-ede3-cbc PARMs IV },
1235     ... -- Expect other content encryption algorithms --
1236 }
1237
1238 IV ::= OCTET STRING (SIZE(8))

```

1239

1240 **ContentEncryptionAlgorithms** specifies an extensible set of **ALGORITHM** information objects.
 1241 The fields of these information objects are used to constrain the valid values of the components
 1242 of type **ContentEncryptionAlgorithmIdentifier**. Though only one content encryption algorithm
 1243 object is defined explicitly in this set, implementations should expect additional algorithms.

1244

1245 The **ContentEncryptionAlgorithms** information object set contains a single object that identifies
 1246 the encryption algorithm described in ANSI X9.52 [10] as Triple DES (TDES) in CBC (cipher
 1247 block chaining) mode. Only the two key and three key variants of TDES are supported in XCBF.
 1248 The single key variant of TDES is simply the DES algorithm and is generally used only for
 1249 backwards compatibility with existing DES based applications and is considered vulnerable to
 1250 attack.

1251

1252 The Triple DES algorithm consists of three sequential DES operations, encrypt, decrypt, and
 1253 encrypt. For three key TDES a different key is used for each DES operation. For two key TDES
 1254 one key is used for both DES encrypt operations, and the second key is used for the DES decrypt
 1255 operation.

1256

1257 The **encryptedContent** component contains a value of type **EncodedBiometricObjects**
 1258 encrypted using the content encryption algorithm given in the **contentEncryptionAlgorithm**
 1259 component. A value of **encryptedContent** is an opaque string of octets treated as having no
 1260 discernable structure. This string is a value of type **EncryptedContent** defined as

1261

```
1262 EncryptedContent ::= OCTET STRING
```

1263

1264 A value of the **encryptedContentInfo** component of any of the privacy block choice alternative
 1265 types can be represented in XML markup as

1266

```

1267 <encryptedContentInfo>
1268     <contentType>1.2.840.113549.1.7.6</contentType>
1269     <contentEncryptionAlgorithm>
1270         <algorithm>1.2.840.113549.3.7</algorithm>
1271         <parameters>
1272             <IV>7EA13D6E143CB5C9</IV>
1273         </parameters>
1274     </contentEncryptionAlgorithm>
1275     <encryptedContent>
1276         D8F6 ...
1277         F766
1278     </encryptedContent>
1279 </encryptedContentInfo>

```

1279

1280 This markup illustrates a typical value. The encrypted content type is identified as ordinary data.
1281 The Triple DES content encryption algorithm is identified along with its associated parameters, an
1282 initialization vector, <IV>. An ellipsis is used as a placeholder for part of the encrypted content.

1283 5.1.3.2 Fixed Key EncryptedData

1284 The **fixedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1285 value of type **EncryptedData**. This type is a sequence of two components, an integer version
1286 number and a value of type **EncryptedContentInfo**. Type **EncryptedData** is defined as

1287

```
1288 EncryptedData ::= SEQUENCE {  
1289   version          CMSVersion,  
1290   encryptedContentInfo  EncryptedContentInfo  
1291 }
```

1292

1293 The **fixedKey** alternative assumes that the recipient of the **EncryptedData** value knows the key
1294 used to encrypt the biometric information, perhaps by prior agreement or as the result of a key
1295 exchange. The **version** component of type **EncryptedData** is always the integer value eighty-
1296 four. The components of type **EncryptedContentInfo** are described in section 4.1.3.1 *Encrypted
1297 Content Information*.

1298

1299 A value of the **fixedKey** choice alternative of the **privacyBlock** component of type
1300 **PrivacyObjects** can be represented in XML markup as

1301

```
1302 <fixedKey>  
1303   <version>84</version>  
1304   <encryptedContentInfo>  
1305     <contentType>1.2.840.113549.1.7.6</contentType>  
1306     <contentEncryptionAlgorithm>  
1307       <algorithm>1.2.840.113549.3.7</algorithm>  
1308       <parameters>  
1309         <IV>7EA13D6E143CB5C9</IV>  
1310       </parameters>  
1311     </contentEncryptionAlgorithm>  
1312     <encryptedContent>  
1313       ...  
1314     </encryptedContent>  
1315   </encryptedContentInfo>  
1316 </fixedKey>
```

1317

1318 This markup uses the fixed key choice alternative of the privacy block, a value of version number
1319 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
1320 identified as ordinary data. The Triple DES content encryption algorithm is identified along with its
1321 associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder and the
1322 encrypted content is not shown.

1323 5.1.3.2.1 Encryption Process

1324 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1325 **BiometricObject** in their encoded form using a content encryption algorithm and a fixed content
1326 encryption key known to the sender and recipient. The content to be encrypted is a value of type

1327 **EncodedBiometricObjects**. This value is always encoded using the XML Encoding Rules. The
1328 content encryption algorithm used to encrypt the biometric objects is one of the algorithms
1329 specified in the information object set **ContentEncryptionAlgorithms**.

1330

1331 The **contentType** component of type **EncryptedContentInfo** is set to indicate ordinary data. The
1332 associated **contentEncryptionAlgorithm** value is set to identify the algorithm used to encrypt
1333 the content, and the **encryptedContent** value is set to the results of encrypting the content using
1334 this content encryption algorithm.

1335 5.1.3.2.2 Decryption Process

1336 To decrypt a value of type **EncryptedData**, the content encryption algorithm specified in the
1337 **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1338 associated **encryptedContent** component using a known fixed key to recover a value of type
1339 **EncodedBiometricObjects**. This recovered value will contain one or more values of type
1340 **BiometricObject** encoded using the XML Encoding Rules.

1341 5.1.3.3 Named Key EncryptedData

1342 The **namedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1343 value of type **NamedKeyEncryptedData**. This type is sequence with two components, **keyName**
1344 and **encryptedData**. Type **NamedKeyEncryptedData** is defined as

1345

```
1346     NamedKeyEncryptedData ::= SEQUENCE {  
1347         keyName           OCTET STRING (SIZE(1..MAX)),  
1348         encryptedData    EncryptedData  
1349     }
```

1350

1351 The **keyName** component explicitly identifies the key used to encrypt and decrypt the content by
1352 name. The **encryptedData** component is a value of type **EncryptedData**. This type contains two
1353 components, an integer **version** number that is always eighty-four in this standard, and an
1354 **encryptedContentInfo** that is a value of type **EncryptedContentInfo** as described in section
1355 4.1.3.1 *Encrypted Content Information*.

1356

1357 A value of the **namedKey** choice alternative of the **privacyBlock** component of type
1358 **PrivacyObjects** can be represented in XML markup as

1359

```
1360     <namedKey>  
1361         <keyName>6AE173BF5A973D1E</keyName>  
1362         <encryptedData>  
1363             <version>84</version>  
1364             <encryptedContentInfo>  
1365                 <contentType>1.2.840.113549.1.7.6</contentType>  
1366                 <contentEncryptionAlgorithm>  
1367                     <algorithm>1.2.840.113549.3.7</algorithm>  
1368                     <parameters>  
1369                         <IV>7EA13D6E143CB5C9</IV>  
1370                     </parameters>  
1371                 </contentEncryptionAlgorithm>  
1372             <encryptedContent>  
1373                 ...  
1374     </encryptedContent>
```

```
1375     </encryptedContentInfo>
1376   </encryptedData>
1377 </namedKey>
```

1378

1379 This markup uses the named key choice alternative of the privacy block, a sequence of a key
1380 name and a value of version number eighty-four of the cryptographic message type
1381 **EncryptedData**. The encrypted content type is identified as ordinary data. The Triple DES
1382 content encryption algorithm is identified along with its associated parameters, an initialization
1383 vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not shown.

1384 **5.1.3.3.1 Encryption Process**

1385 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1386 **BiometricObject** in their encoded form using a content encryption algorithm and a named key
1387 that is known to the recipient of the encrypted biometric information. The content to be encrypted
1388 is a value of type **EncodedBiometricObjects**. This value is always encoded using the XML
1389 Encoding Rules. The content encryption algorithm used to encrypt the biometric objects is one of
1390 the algorithms specified in the information object set **ContentEncryptionAlgorithms**.

1391

1392 The **keyName** component of type **NamedKeyEncryptedData** is set to the name of the content
1393 encryption key. The **contentType** component of type **EncryptedContentInfo** is set to indicate
1394 ordinary data. The associated **contentEncryptionAlgorithm** value is set to identify the algorithm
1395 used to encrypt the content, and the **encryptedContent** value is set to the results of encrypting
1396 the content using this content encryption algorithm.

1397 **5.1.3.3.2 Decryption Process**

1398 To decrypt a value of type **NamedKeyEncryptedData**, the content encryption algorithm specified
1399 in the **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1400 associated **encryptedContent** component using the key identified by the **keyName** component
1401 of type **NamedKeyEncryptedData** to recover a value of type **EncodedBiometricObjects**. This
1402 recovered value will contain one or more values of type **BiometricObject** encoded using the XML
1403 Encoding Rules.

1404 **5.1.3.4 Established Key EnvelopedData**

1405 The **establishedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects**
1406 is a value of type **EnvelopedData**. Using this type, a message sender can encrypt content that
1407 only the intended message recipient can decrypt.

1408

1409 **EnvelopedData** is defined as a sequence of four components, an integer version number,
1410 message sender information, message recipient information, and a value of type
1411 **EncryptedContentInfo** which is described in section 4.1.3.1 *Encrypted Content Information*.
1412 Type **EnvelopedData** is defined as

1413

```
1414 EnvelopedData ::= SEQUENCE {
1415   version           CMSVersion,
1416   originatorInfo    [0] OriginatorInfo  OPTIONAL,
1417   recipientInfos    RecipientInfos,
1418   encryptedContentInfo EncryptedContentInfo
1419 }
```

1420

1421 The combination of encrypted content and an encrypted content encryption key forms a “digital
1422 envelope”. The **establishedKey** alternative uses a randomly generated content encryption key to
1423 encrypt digital content. The same key is used to decrypt the content. The content encryption key
1424 shall be protected during transport, so the recipient’s public and private key pair is used to
1425 encrypt and decrypt the content encryption key.

1426

1427 The encrypted content is value of type **EncodedBiometricObjects**. This type is a series of one
1428 or more values of type **BiometricObject** in their encoded form. In XCBF these values are
1429 encoded using the XML Encoding Rules.

1430

1431 The **version** component of type **EnvelopedData** is the integer value eighty-four. The optional
1432 **originatorInfo** component facilitates distribution of digital certificates and certificate revocation
1433 lists. The **recipientInfos** component contains information needed to recover the encrypted
1434 content encryption key used to encrypt the biometric information. The **encryptedContentInfo**
1435 component is a value of type **EncryptedContentInfo**. This type is described in section 4.1.3.1
1436 *Encrypted Content Information*.

1437

1438 A value of the **establishedKey** choice alternative of the **privacyBlock** component of type
1439 **PrivacyObjects** can be represented in XML markup as

1440

```

1441 <establishedKey>
1442   <version>84</version>
1443   <recipientInfos>
1444     <RecipientInfo>
1445       <ktri>
1446         <version>84</version>
1447         <rid>
1448           <certHash>
1449             <ietf>6E143CF31A562FA9492681D27A22013D2AAD435D</ietf>
1450           </certHash>
1451         </rid>
1452         <keyEncryptionAlgorithm>
1453           <algorithm>
1454             1.2.840.113549.1.1.1
1455           </algorithm>
1456           <parameters><NoIV/></parameters>
1457         </keyEncryptionAlgorithm>
1458         <encryptedKey>
1459           ...
1460           </encryptedKey>
1461         </ktri>
1462       </RecipientInfo>
1463     </recipientInfos>
1464     <encryptedContentInfo>
1465       <contentType>1.2.840.113549.1.7.6</contentType>
1466       <contentEncryptionAlgorithm>
1467         <algorithm>1.2.840.113549.3.7</algorithm>
1468         <parameters>
1469           <IV>7EA13D6E143CB5C9</IV>
1470         </parameters>
1471       </contentEncryptionAlgorithm>
1472       <encryptedContent>
1473         ...

```

```
1474     </encryptedContent>
1475   </encryptedContentInfo>
1476 </establishedKey>
```

1477

1478 This markup uses the established key choice alternative of the privacy block, a value of version
1479 number eighty-four of the cryptographic message type **EnvelopedData**. The optional originator
1480 information is not present. The recipient information uses the key transport choice alternative.
1481 The recipient public and private key pair is indicated by a SHA1 hash of a public key certificate.
1482 The content encryption key is encrypted using the **rsaEncryption** algorithm, which has no
1483 associated parameters indicating that no initialization vector is required. The encrypted content
1484 type is identified as ordinary data. The Triple DES content encryption algorithm is identified along
1485 with its associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder
1486 and the encrypted content encryption key and the encrypted content are not shown.

1487

1488 5.1.3.4.1 Certificates and CRLs

1489 Type **OriginatorInfo** is a sequence of two components that may contain sets of digital certificates
1490 and certificate revocation lists (CRLs). This type is defined as

1491

```
1492 OriginatorInfo ::= SEQUENCE {
1493   certs  [0] CertificateSet  OPTIONAL,
1494   crls   [1] CertificateRevocationLists  OPTIONAL
1495 }
1496 (ALL EXCEPT({ -- none; at least one component is present -- }))
```

1497

1498 Any combination of X9.68 domain certificates, X.509 certificates and attribute certificates may be
1499 included in the **CertificateSet** type in any order. There may be more or fewer certificates needed
1500 for any purpose. Certificates are provided as needed to support content key encryption in the key
1501 transport key management technique used in XCBF. Use of the **CertificateSet** type to distribute
1502 certificates is not required. They may be obtained by other means, or an online certificate
1503 validation service may be used instead. Only version one X9.68 domain certificates, version three
1504 X.509 certificates and version two attribute certificates are supported in this standard.

1505

1506 Any number of CRLs may be included in the **CertificateRevocationLists** type in any order.
1507 There may be more or fewer CRLs needed for any purpose. CRLs are provided as needed to
1508 support certificate validation. Use of the **CertificateRevocationLists** type to distribute CRLs is
1509 not required. CRLs may be obtained by other means, or an online certificate validation service
1510 may be used instead. Only version two certificate revocation lists are supported in this standard.

1511

1512 The certificates and certificate revocation lists used in XCBF are signed binary objects, whose
1513 digital signatures have been calculated on values encoded using the Distinguished Encoding
1514 Rules (DER) of ASN.1. In order to verify the signatures on these objects, their original encodings
1515 must be maintained. But these values must also be represented in XML encodings in a useful
1516 textual format. So the values in the **certs** and **crls** components of type **OriginatorInfo** have been
1517 Base64 [20] armored to minimize their size when represented using XML markup while
1518 preserving their original encodings.

1519 **5.1.3.4.2 Recipient Information**

1520 Type **RecipientInfos** is a series of values of type **RecipientInfo**, one value for each recipient of a
1521 digital envelope in **EnvelopedData**. In XCBF there is always a single digital envelope recipient,
1522 and type **RecipientInfos** is constrained to a series of one **RecipientInfo** and defined as

1523

```
1524    RecipientInfos ::= SET SIZE(1) OF RecipientInfo
```

1525

1526 Several key management techniques can be used in **EnvelopedData**. In XCBF, only key
1527 transport is supported. Other techniques such as constructive key management may be
1528 employed by an application, but such use is not defined in this standard. Type **RecipientInfo** is
1529 restricted to a single choice alternative and defined as

1530

```
1531    RecipientInfo ::= CHOICE {  
1532       ktri KeyTransRecipientInfo  
1533     }
```

1534

1535 Key transport information is provided to the recipient of a digital envelope so that the envelope
1536 can be opened and the protected content encryption key recovered. The content encryption key
1537 may then be used to decrypt the content.

1538

1539 The information needed by the recipient to recover the content encryption key is contained in a
1540 value of type **KeyTransRecipientInfo** defined as

1541

```
1542    KeyTransRecipientInfo ::= SEQUENCE {  
1543       version CMSVersion,  
1544       rid RecipientIdentifier,  
1545       keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,  
1546       encryptedKey EncryptedKey  
1547     }
```

1548

1549 This type is a sequence of four components. The integer **version** number is always set to eighty-
1550 four in XCBF. The **rid** component is used to identify the public key used to encrypt the content
1551 encryption key. This public key is bound to a key encryption algorithm in a public key certificate. It
1552 is associated with the recipient private key needed to decrypt the content encryption key used by
1553 the sender to encrypt the content.

1554

1555 A hash of the public key certificate uniquely identifies the recipient certificate.

1556

```
1557    RecipientIdentifier ::= CHOICE {  
1558       certHash [73] EXPLICIT Hash  
1559     }
```

1560

1561 The **keyEncryptionAlgorithm** component identifies the key encryption algorithm and any
1562 associated parameters used to encrypt the content encryption key.

1563

1564 KeyEncryptionAlgorithmIdentifier ::=
1565 AlgorithmIdentifier {{KeyEncryptionAlgorithms}}
1566
1567 KeyEncryptionAlgorithms ALGORITHM ::= {
1568 { OID rsaEncryption PARMs NoIV },
1569
1570 ... -- expect other key encryption algorithms --
1571 }

1572

1573 The encrypted content encryption key is an opaque string, a value of type **EncryptedKey** defined
1574 as

1575

1576 **EncryptedKey** ::= OCTET STRING

1577

1578 **5.1.3.4.3 Digital Envelope Processing**

1579 To create a digital envelope, a content encryption algorithm is selected. The content encryption
1580 algorithm identifier and any associated parameters form the **contentEncryptionAlgorithm** value
1581 of the **encryptedContentInfo** component of type **EnvelopedData**. The recipient uses this value
1582 to recover the encrypted content.

1583

1584 The content encryption key is encrypted using the key encryption algorithm and key transport
1585 public key from the recipient's public key certificate. This certificate must contain a key usage
1586 extension which asserts the **keyEncipherment** bit. The key encryption algorithm identifier and
1587 any associated parameters used to encrypt the content encryption key with the key transport
1588 public key form the **keyEncryptionAlgorithm** component of type **KeyTransRecipientInfo**.

1589

1590 The result of encrypting the content encryption key forms the **encryptedKey** component of type
1591 **KeyTransRecipientInfo**. A hash of the complete DER encoding of the recipient's public key
1592 certificate is used to populate the **rid** component, and the **version** component is set to the integer
1593 eighty-four. This certificate hash mechanism provides a single way to uniquely identify any type of
1594 certificate. This is the only form of certificate identification supported in XCBF.

1595

1596 The content encryption key is used to encrypt a value of type **EncodedBiometricObjects**. This
1597 type is a series of one or more values of type **BiometricObject** in their encoded form. These
1598 values are encoded using the XML Encoding Rules.

1599

1600 To retrieve the encrypted content, the recipient first decrypts the value of the **encryptedKey**
1601 component of type **KeyTransRecipientInfo** to recover the content encryption key using the
1602 private key associated with the public key used to encrypt the content encryption key. This private
1603 key is indicated by the hash of the associated public key certificate in the **rid** component of type
1604 **KeyTransRecipientInfo**. The recovered content encryption key is then used to decrypt the
1605 content to recover a value of type **EncodedBiometricObjects**.

1606 **5.1.4 PrivacyAndIntegrityObjects**

1607 The **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** is a value of type
1608 **PrivacyAndIntegrityObjects**. This type is defined as a sequence of three components, optional
1609 **biometricHeaders**, a **privacyBlock**, and an **integrityBlock**.

1610

```
1611    PrivacyAndIntegrityObjects ::= SEQUENCE {  
1612     biometricHeaders   BiometricHeaders   OPTIONAL,  
1613     privacyBlock       PrivacyBlock,  
1614     integrityBlock    IntegrityBlock  
1615 }
```

1616

1617 The **biometricHeaders** component is optional and is composed of a series of one or more values
1618 of type **BiometricHeader**. The **privacyBlock** component is a value of type **PrivacyBlock**. The
1619 **BiometricHeader** and **PrivacyBlock** types are described in section 4.1.3. *PrivacyObjects*. The
1620 **integrityBlock** component is a value of type **IntegrityBlock**. This type is described in section
1621 4.1.2. *IntegrityObjects*.

1622

1623 The input to all cryptographic process is a value of type **EncodedBiometricObjects**, a series of
1624 one or more values of type **BiometricObject** in their encoded form based on the XML Encoding
1625 Rules. The order of the components in type **PrivacyAndIntegrityObjects** facilitates one pass
1626 processing for both sender and recipient.

1627

1628 For the sender, a value of type **EncodedBiometricObjects** is created and then used as input to
1629 the cryptographic processing of both the **privacyBlock** and **integrityBlock** components. For the
1630 recipient, the **privacyBlock** component is first processed to recover the encrypted content, a
1631 value of type **EncodedBiometricObjects**. This recovered value is then used to validate the
1632 signature in the **integrityBlock** component.

1633

1634 A value of the **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** can be
1635 represented in XML markup as

1636

```
1637    <privacyAndIntegrityObjects>  
1638     <privacyBlock>  
1639       <namedKey>  
1640         <keyName>6AE173BF5A973D1E</keyName>  
1641         <encryptedData>  
1642           <version>84</version>  
1643           <encryptedContentInfo>  
1644             <contentType>1.2.840.113549.1.7.6</contentType>  
1645             <contentEncryptionAlgorithm>  
1646               <algorithm>1.2.840.113549.3.7</algorithm>  
1647               <parameters>  
1648                 <IV>7EA13D6E143CB5C9</IV>  
1649                 </parameters>  
1650               </contentEncryptionAlgorithm>  
1651               <encryptedContent>  
1652                 ...  
1653                 </encryptedContent>  
1654               </encryptedContentInfo>  
1655     </encryptedData>
```

```
1656      </namedKey>
1657    </privacyBlock>
1658    <integrityBlock>
1659      <digitalSignature>
1660        <algorithmID>
1661          <algorithm>1.2.840.10040.4.3</algorithm>
1662          <parameters><NoIV/></parameters>
1663        </algorithmID>
1664        <signature>
1665          ...
1666          </signature>
1667        </digitalSignature>
1668      </integrityBlock>
1669    </privacyAndIntegrityObjects>
```

1670
1671 This markup combines a privacy block and integrity block. The named key choice alternative of
1672 the privacy block and the digital signature choice alternative of the integrity block are used. The
1673 optional biometric headers are not present.

1674
1675 The named key alternative is a sequence containing a key name and a value of version number
1676 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
1677 identified as ordinary data, and is computed on a value of type **EncodedBiometricObjects**. The
1678 Triple DES content encryption algorithm is identified along with its associated parameters, an
1679 initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not
1680 shown.

1681
1682 The digital signature choice alternative is a simple digital signature on a value of type
1683 **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) with Secure Hash Algorithm
1684 (SHA1) and its associated parameters, an initialization vector, <IV> are used for signing a value
1685 of **EncodedBiometricObjects**. An ellipsis is used as a placeholder and the signature is not
1686 shown.

1687

6 References

1688

6.1 Normative

1689

1690 These references or an understanding of the materials within them are required to implement this
1691 XCBF standard. They are intended to include any amendments and technical corrigenda issued
1692 after their publication. Users of this standard are encouraged to seek the latest versions of these
1693 referenced materials before initiating any serious work.

1694

- 1695 1 ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open
1696 Systems Interconnection -- The Directory: Authentication framework.
- 1697 2 ISO/IEC 8824-1:2002 | ITU-T Recommendation X.680 (2002), Information Technology
1698 - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation,
1699 http://www.itu.int/ITU-T/studygroups/com17/languages/X680_0702.pdf.
- 1700 3 ISO/IEC 8824-2:2002 | ITU-T Recommendation X.681 (2002), Information Technology
1701 - Abstract Syntax Notation One (ASN.1): Information Object Specification,
1702 http://www.itu.int/ITU-T/studygroups/com17/languages/X681_0702.pdf.
- 1703 4 ISO/IEC 8824-3:2002| ITU-T Recommendation X.682 (2002), Information Technology -
1704 Abstract Syntax Notation One (ASN.1): Constraint Specification,
1705 http://www.itu.int/ITU-T/studygroups/com17/languages/X682_0702.pdf .
- 1706 5 ISO/IEC 8824-4:2002| ITU-T Recommendation X.683 (2002), Information Technology -
1707 Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications,
1708 http://www.itu.int/ITU-T/studygroups/com17/languages/X683_0702.pdf.
- 1709 6 ISO/IEC 8825-1:2002| ITU-T Recommendation X.690 (2002), Information Technology -
1710 ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical
1711 Encoding Rules (CER) and Distinguished Encoding Rules (DER),
1712 http://www.itu.int/ITU-T/studygroups/com17/languages/X690_0702.pdf.
- 1713 7 ISO/IEC 8825-4:2002 | X.693 ITU-T Recommendation X.693 (2002) |, Information
1714 Technology - ASN.1 Encoding Rules: XML Encoding Rules (XER),
1715 http://www.itu.int/ITU-T/studygroups/com17/languages/X693_0702.pdf.
- 1716 ISO/IEC 8825-4:2002/Amd.1:2003 | ITU-T Rec. X.693 (2002) / Amd.1 (2003),
1717 Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) –
1718 Amendment 1: XER Control Notation (XCN) and Variable XER (VXER)
- 1719 8 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 1:
1720 The Digital Signature Algorithm (DSA), <http://webstore.ansi.org/>.
- 1721 9 ANSI X9.30-1997 Public Key Cryptography for the Financial Services Industry - Part 2:
1722 The Secure Hash Algorithm (SHA-1), <http://webstore.ansi.org/>.
- 1723 10 ANSI X9.52-1998 Triple Data Encryption Algorithm Modes of Operation,
1724 <http://webstore.ansi.org/>.
- 1725 11 ANSI X9.68:2001 Digital Certificates for Mobile/Wireless and High Transaction Volume
1726 Financial Systems: Part 2: Domain Certificate Syntax, <http://webstore.ansi.org/>.
- 1727 12 ANSI X9.71-1999 Keyed Hash Message Authentication Code (HMAC),
1728 <http://webstore.ansi.org/>.
- 1729 13 ANSI X9.73:2002 Cryptographic Message Syntax (CMS), <http://webstore.ansi.org/>.
- 1730 14 ANSI X9.84:2003 Biometric Information Management and Security For The Financial
1731 Services Industry, <http://webstore.ansi.org/>.

- 1732 15 ANSI X9.96:2003 (draft) XML Cryptographic Message Syntax (XCMS).
1733 16 ANSI INCITS 358-2002 - Information technology - BioAPI Specification,
1734 <http://webstore.ansi.org/>.
1735 17 CBEFF Common Biometric Exchange File Format, NISTIR-6529,
1736 <http://oasis-open.org/committees/xcbf/docs/NISTR6529-CBEFF.pdf>, January 3, 2001.
1737 18 S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,
1738 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
1739 19 PKCS #7 – Cryptographic Message Syntax Standard,
1740 <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/>, 1 November 1993.
1741 20 N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME)
1742 Part 1: Format of Internet Message Bodies, <http://www.ietf.org/rfc/rfc2045.txt>,
1743 IETF RFC 2045, November 1996.

1744 7 XCBF Schema

1745 The following ASN.1 modules provide the schema for all of the XML markup defined in this
1746 standard.

1747

1748 7.1 X9-84-Biometrics Module

1749

```
1750 X9-84-Biometrics {
1751     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
1752         x9Standards(9) x9-84(84) module(0) biometrics(1) rev(1) }
1753     DEFINITIONS AUTOMATIC TAGS ::= BEGIN
1754
1755 -- EXPORTS All;
1756
1757 IMPORTS
1758
1759     -- X9.84 Biometrics Information Management and Security IDs --
1760
1761     BiometricTypes, CBEFF-Formats, IBIA-Formats, MatchingAIDs,
1762     ProcessingAIDs, X9-Formats
1763     FROM X9-84-Identifiers {
1764         iso(1) identified-organization(3) tc68(133) country(16)
1765             x9(840) x9Standards(9) x9-84(84) module(0)
1766                 ids(3) rev(1) }
1767
1768     -- X9.84 Biometrics Information Management and Security CMS --
1769
1770     AuthenticatedData, EncryptedData, EnvelopedData,
1771     MACAlgorithmIdentifier, SignatureAlgorithmIdentifier,
1772     SignedData
1773     FROM X9-84-CMS {
1774         iso(1) identified-organization(3) tc68(133) country(16)
1775             x9(840) x9Standards(9) x9-84(84) module(0)
1776                 cms(2) rev(1) } ;
1777
1778
1779 BiometricSyntaxSets ::= SEQUENCE SIZE(1..MAX) OF BiometricSyntax
1780
1781 BiometricSyntax ::= CHOICE {
1782     biometricObjects           BiometricObjects,
1783     integrityObjects          IntegrityObjects,
1784     privacyObjects            PrivacyObjects,
1785     privacyAndIntegrityObjects PrivacyAndIntegrityObjects
1786 }
1787
1788 BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject
1789
1790 BiometricObject ::= SEQUENCE {
1791     biometricHeader  BiometricHeader,
1792     biometricData    BiometricData
1793 }
1794
1795
```

```

1796
1797 -- All of the cryptographic processing in this standard is performed
1798 -- on a value of type EncodedBiometricObjects. This is a sequence of
1799 -- one or more values of type BiometricObject in its encoded form.
1800 --
1801
1802 EncodedBiometricObjects ::= BIOMETRIC.&Type( BiometricObjects )
1803
1804 BiometricHeader ::= SEQUENCE {
1805   version      BiometricVersion DEFAULT hv1,
1806   recordType   RecordType OPTIONAL,
1807   dataType     DataType OPTIONAL,
1808   purpose      Purpose OPTIONAL,
1809   quality      Quality OPTIONAL,
1810   validityPeriod ValidityPeriod OPTIONAL,
1811   format       Format OPTIONAL
1812 }
1813
1814 BiometricVersion ::= INTEGER { hv1(0) } (0..MAX)
1815
1816 RecordType ::= BIOMETRIC.&name({BiometricTypes})
1817
1818 DataType ::= ENUMERATED {
1819   raw          (0),
1820   intermediate (1),
1821   processed    (2)
1822 }
1823
1824 Purpose ::= ENUMERATED {
1825   verify        (1),
1826   identify      (2),
1827   enroll         (3),
1828   enrollVerify   (4),
1829   enrollIdentity (5),
1830   audit         (6),
1831
1832   ... -- Expect other values --
1833 }
1834
1835 Quality ::= INTEGER {
1836   lowest        ( 0),
1837   highest       (100),
1838   notSet        (-1),
1839   notSupported  (-2)
1840 } (-2..100,...)
1841
1842 ValidityPeriod ::= SEQUENCE {
1843   notBefore   DateTime OPTIONAL,
1844   notAfter    DateTime  OPTIONAL
1845 }
1846 (ALL EXCEPT({ -- none; at least one component is present -- })) )
1847
1848 DateTime ::= RELATIVE-OID -- { yyyy mm dd hh mm ss z } --
1849
1850 Format ::= SEQUENCE {
1851   formatOwner  BIOMETRIC.&name({Owner}),
1852   formatType   BIOMETRIC.&Type({Owner}@formatOwner)) OPTIONAL
1853 }
1854
1855 Owner BIOMETRIC ::= {

```

```

1856 CBEFF-Formats | -- http://www.nist.gov --
1857 I比亚-Formats | -- http://www.ibia.org --
1858 X9-Formats,      -- http://www.x9.org   --
1859
1860     ... -- expect additional vendor specific formats --
1861 }
1862
1863 -- Integrity --
1864
1865 IntegrityObjects ::= SEQUENCE {
1866     biometricObjects EncodedBiometricObjects,
1867     integrityBlock   IntegrityBlock
1868 }
1869
1870 IntegrityBlock ::= CHOICE {
1871     digitalSignature           DigitalSignature,
1872     messageAuthenticationCode MessageAuthenticationCode,
1873     signedData                 SignedData,
1874     authenticatedData          AuthenticatedData
1875 }
1876
1877 DigitalSignature ::= ::= SEQUENCE {
1878     algorithmID SignatureAlgorithmIdentifier,
1879     signature      OCTET STRING( CONSTRAINED BY {
1880         -- signature on -- EncodedBiometricObjects })
1881 }
1882
1883 MessageAuthenticationCode ::= SEQUENCE {
1884     keyName      OCTET STRING OPTIONAL,
1885     algorithmID MACAlgorithmIdentifier,
1886     mac          OCTET STRING (CONSTRAINED BY {
1887         -- MAC or HMAC on -- EncodedBiometricObjects })
1888 }
1889
1890 -- Privacy --
1891
1892 PrivacyObjects ::= SEQUENCE {
1893     biometricHeaders BiometricHeaders OPTIONAL,
1894     privacyBlock    PrivacyBlock
1895 }
1896
1897 BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
1898
1899 PrivacyBlock ::= CHOICE {
1900     fixedKey        EncryptedData,
1901     namedKey       NamedKeyEncryptedData,
1902     establishedKey EnvelopedData
1903 }
1904
1905 NamedKeyEncryptedData ::= SEQUENCE {
1906     keyName        OCTET STRING (SIZE(1..MAX)),
1907     encryptedData  EncryptedData
1908 }
1909
1910 -- Privacy and integrity --
1911
1912 PrivacyAndIntegrityObjects ::= SEQUENCE {
1913     biometricHeaders BiometricHeaders OPTIONAL,
1914     privacyBlock    PrivacyBlock,
1915     integrityBlock  IntegrityBlock

```

```

1916 }
1917
1918 -- Authentication Information (AI) --
1919
1920 BiometricInformationSets ::= 
1921     SEQUENCE SIZE(1..MAX) OF BiometricInformation
1922
1923 BiometricInformation ::= SEQUENCE {
1924     processingAlgorithms ProcessingAlgorithms OPTIONAL,
1925     matchingMethods MatchingMethods OPTIONAL
1926 }
1927     (ALL EXCEPT({ -- none; at least one component is present -- }))
1928
1929 -- Biometric processing algorithms --
1930
1931 ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation
1932
1933 ProcessingInformation ::= SEQUENCE {
1934     id BIOMETRIC.&name({ProcessingAIDs}),
1935     parms BIOMETRIC.&Type({ProcessingAIDs}{@id}) OPTIONAL
1936 }
1937
1938 -- Biometric matching methods --
1939
1940 MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation
1941
1942 MatchingInformation ::= SEQUENCE {
1943     id BIOMETRIC.&name({MatchingAIDs}),
1944     parms BIOMETRIC.&Type({MatchingAIDs}{@id}) OPTIONAL
1945 }
1946
1947 BiometricData ::= OCTET STRING(SIZE(1..MAX))
1948
1949 -- Biometrics information object class --
1950
1951 BIOMETRIC ::= CLASS {
1952     &name BIOMETRIC-IDENTIFIER UNIQUE,
1953     &Type OPTIONAL
1954 }
1955     WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
1956
1957 BIOMETRIC-IDENTIFIER ::= CHOICE {
1958     oid OBJECT IDENTIFIER, -- complete object identifier
1959     id RELATIVE-OID -- object identifier fragment
1960 }
1961
1962 -- Biometric certificate extension --
1963
1964 --
1965 -- A biometricTemplates information object can be used to extend the
1966 -- information bound to a public key in an value of types Certificate
1967 -- or AttributeCertificate as defined in The Directory series of
1968 -- standards, to include biometric identity information.
1969 --
1970
1971 biometricTemplates EXTENSION ::= {
1972     SYNTAX          EncodedBiometricObjects -- DER or cXER --
1973     IDENTIFIED BY x509-biometricTemplates
1974 }
1975

```

```

1976 EXTENSION ::= CLASS {
1977     &id          OBJECT IDENTIFIER UNIQUE,
1978     &ExtnType
1979 }
1980     WITH SYNTAX { SYNTAX &ExtnType IDENTIFIED BY &id }
1981
1982 --
1983 -- A domainBiometricTemplates information object can be used to extend
1984 -- the information bound to a public key in an value of ASN.1 type
1985 -- DomainCertificate as defined in the X9.68 Domain Certificate Syntax
1986 -- standard, to include biometric identity information.
1987 --
1988
1989 domainBiometricTemplates PRIVATE-X ::= {
1990     NAME oid : x968-biometricTemplates
1991     TYPE EncodedBiometricObjects -- DER or cXER --
1992 }
1993
1994 PRIVATE-X ::= CLASS {
1995     &name Identifier UNIQUE,
1996     &Type OPTIONAL
1997 }
1998     WITH SYNTAX { NAME &name [TYPE &Type] }
1999
2000 Identifier ::= CHOICE {
2001     oid OBJECT IDENTIFIER, -- complete object identifier
2002     id  RELATIVE-OID      -- object identifier fragment
2003 }
2004
2005 END -- X9-84-Biometrics --

```

2006

7.2 X9-84-CMS Module

2008

```

2009 X9-84-CMS {
2010     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
2011         x9Standards(9) x9-84(84) module(0) cms(2) rev(1) }
2012     DEFINITIONS IMPLICIT TAGS ::= BEGIN
2013
2014     -- EXPORTS All;
2015
2016     IMPORTS
2017
2018         -- ANSI X9.84 Biometric Information Management & Security IDs --
2019
2020         des-ed3-cbc, dsa-with-sha1, ecdsa-with-SHA1, hmac-with-SHA1,
2021         id-data, NoIV, rsaEncryption, SHA-Algorithms, sha1WithRSAEncryption
2022         FROM X9-84-Identifiers {
2023             iso(1) identified-organization(3) tc68(133) country(16)
2024                 x9(840) x9Standards(9) x9-84(84) module(0)
2025                 ids(3) rev(1) };
2026
2027     SignedData ::= SEQUENCE {
2028         version           CMSVersion,
2029         digestAlgorithms DigestAlgorithmIdentifiers,
2030         encapContentInfo EncapsulatedContentInfo,
2031         certificates      [0] CertificateSet OPTIONAL,

```

```

2032      crls          [1] CertificateRevocationLists  OPTIONAL,
2033      signerInfos    SignerInfos
2034  }
2035
2036 CMSVersion ::= INTEGER { v84(84) } (v84,...)
2037
2038 DigestAlgorithmIdentifiers ::= 
2039     SET SIZE(1) OF DigestAlgorithmIdentifier
2040
2041 DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{DigestAlgorithms}}
2042
2043 DigestAlgorithms ALGORITHM ::= {
2044     SHA-Algorithms,
2045
2046     ... -- Expect other digest algorithms --
2047 }
2048
2049 EncapsulatedContentInfo ::= SEQUENCE {
2050     eContentType  ContentType,
2051     eContent      [0] EXPLICIT OCTET STRING  OPTIONAL
2052 }
2053
2054 ContentType ::= CONTENTS.&id({Contents})
2055
2056 CONTENTS ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1998, Annex A
2057
2058 Contents CONTENTS ::= {
2059     { Data IDENTIFIED BY id-data }
2060 }
2061
2062 Data ::= OCTET STRING
2063
2064 CertificateSet ::= OCTET STRING
2065
2066 CertificateRevocationLists ::= OCTET STRING
2067
2068 SignerInfos ::= SET SIZE(1) OF SignerInfo
2069
2070 SignerInfo ::= SEQUENCE {
2071     version        CMSVersion,
2072     sid            SignerIdentifier,
2073     digestAlgorithm DigestAlgorithmIdentifier,
2074     signatureAlgorithm SignatureAlgorithmIdentifier,
2075     signature       SignatureValue
2076 }
2077
2078 SignerIdentifier ::= CHOICE {
2079     certHash   [1] EXPLICIT Hash
2080 }
2081
2082 Hash ::= CHOICE {
2083     ietf        CertHash, -- SHA-1 hash of entire certificate
2084     withAlgID  DigestInfo
2085 }
2086
2087 CertHash ::= OCTET STRING (ENCODED BY sha-1)
2088
2089 DigestInfo ::= SEQUENCE {
2090     hashAlgorithm DigestAlgorithmIdentifier,
2091     digest        OCTET STRING

```

```

2092 }
2093
2094 SignatureAlgorithmIdentifier ::= AlgorithmIdentifier {{SignatureAlgorithms}}
2095
2096
2097 SignatureAlgorithms ALGORITHM ::= {
2098     { OID dsa-with-sha1          PARMS NoIV } |
2099     { OID ecdsa-with-SHA1       PARMS NoIV } |
2100     { OID sha1WithRSAEncryption PARMS NoIV },
2101
2102     ... -- Expect other signature algorithms --
2103 }
2104
2105 SignatureValue ::= OCTET STRING
2106
2107 EncryptedData ::= SEQUENCE {
2108     version           CMSVersion,
2109     encryptedContentInfo EncryptedContentInfo
2110 }
2111
2112 EncryptedContentInfo ::= SEQUENCE {
2113     contentType        ContentType,
2114     contentEncryptionAlgorithm ContentEncryptAlgorithmIdentifier,
2115     encryptedContent      [0] EncryptedContent
2116 }
2117
2118 ContentEncryptAlgorithmIdentifier ::= AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
2119
2120
2121 ContentEncryptionAlgorithms ALGORITHM ::= {
2122     { OID des-ede3-cbc PARMS IV },
2123
2124     ... -- Expect other content encryption algorithms --
2125 }
2126
2127 IV ::= OCTET STRING (SIZE(8))
2128
2129 EncryptedContent ::= OCTET STRING
2130
2131 EnvelopedData ::= SEQUENCE {
2132     version           CMSVersion,
2133     originatorInfo    [0] OriginatorInfo OPTIONAL,
2134     recipientInfos    RecipientInfos,
2135     encryptedContentInfo EncryptedContentInfo
2136 }
2137
2138 OriginatorInfo ::= SEQUENCE {
2139     certs   [0] CertificateSet OPTIONAL,
2140     crls    [1] CertificateRevocationLists OPTIONAL
2141 }
2142 (ALL EXCEPT({ -- none; at least one component is present -- }))
2143
2144 RecipientInfos ::= SET SIZE(1) OF RecipientInfo
2145
2146 RecipientInfo ::= CHOICE {
2147     ktri   KeyTransRecipientInfo
2148 }
2149
2150 KeyTransRecipientInfo ::= SEQUENCE {
2151     version           CMSVersion,

```

```

2152                   RecipientIdentifier,
2153             keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
2154             encryptedKey         EncryptedKey
2155         }
2156
2157     RecipientIdentifier ::= CHOICE {
2158       certHash [73] EXPLICIT Hash
2159     }
2160
2161     KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier {{KeyEncryptionAlgorithms}}
2162
2163     KeyEncryptionAlgorithms ALGORITHM ::= {
2164       { OID rsaEncryption PARMs NoIV },
2165
2166       ... -- expect other key encryption algorithms --
2167     }
2168
2169
2170     EncryptedKey ::= OCTET STRING
2171
2172     AuthenticatedData ::= SEQUENCE {
2173       version CMSVersion,
2174       recipientInfos RecipientInfos,
2175       macAlgorithm MACAlgorithmIdentifier,
2176       encapContentInfo EncapsulatedContentInfo,
2177       mac MessageAuthenticationCode
2178     }
2179
2180     MACAlgorithmIdentifier ::= AlgorithmIdentifier {{MACAlgorithms}}
2181
2182     MACAlgorithms ALGORITHM ::= {
2183       { OID hmac-with-SHA1 },
2184
2185       ... -- expect other MAC or HMAC algorithms --
2186     }
2187
2188     MessageAuthenticationCode ::= OCTET STRING
2189
2190
2191     -- Cryptographic algorithm identification --
2192
2193     ALGORITHM ::= CLASS {
2194       &id   OBJECT IDENTIFIER UNIQUE,
2195       &Type OPTIONAL
2196     }
2197     WITH SYNTAX { OID &id [PARMS &Type] }
2198
2199     AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
2200       algorithm ALGORITHM.&id( {IOSet} ),
2201       parameters ALGORITHM.&Type( {IOSet}{@algorithm} ) OPTIONAL
2202     }
2203
2204
2205     END -- X9-84-CMS --

```

2206

2207 7.3 X9-84-Identifiers Module

2208

```

2209 X9-84-Identifiers {
2210     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
2211         x9Standards(9) x9-84(84) module(0) ids(3) rev(1) }
2212 DEFINITIONS AUTOMATIC TAGS ::= BEGIN
2213
2214 -- EXPORTS All;
2215
2216 IMPORTS
2217
2218     -- X9.84 Biometrics Information Management and Security --
2219
2220     BIOMETRIC, BiometricInformationSets
2221         FROM X9-84-Biometrics {
2222             iso(1) identified-organization(3) tc68(133) country(16)
2223                 x9(840) x9Standards(9) x9-84(84) module(0)
2224                     biometrics(1) rev(1) }
2225
2226     -- X9.84 Biometrics Information Management and Security CMS --
2227
2228     ALGORITHM
2229         FROM X9-84-CMS {
2230             iso(1) identified-organization(3) tc68(133) country(16)
2231                 x9(840) x9Standards(9) x9-84(84) module(0)
2232                     cms(2) rev(1) };
2233
2234
2235     OID ::= OBJECT IDENTIFIER -- Alias
2236
2237     RelOID ::= RELATIVE-OID -- Alias
2238
2239     -- x9-84
2240     -- x9-84-Module
2241     -- x9-84-Biometrics
2242     -- x9-84-CMS
2243     -- x9-84-Identifiers
2244     -- biometric
2245     -- id-unknown-Type
2246     -- id-body-Odor
2247     -- id-dna
2248     -- id-ear-Shape
2249     -- id-facial-Features
2250     -- id-finger-Image
2251     -- id-finger-Geometry
2252     -- id-hand-Geometry
2253     -- id-iris-Features
2254     -- id-keystroke-Dynamics
2255     -- id-palm
2256     -- id-retina
2257     -- id-signature
2258     -- id-speech-Pattern
2259     -- id-thermal-Image
2260     -- id-vein-Pattern
2261     -- id-thermal-Face-Image
2262     -- id-thermal-Hand-Image
2263     -- id-lip-Movement
2264     -- id-gait
2265     -- processing-algorithm
2266     -- matching-method
2267     -- format-Owner
2268     -- cbeff-Owner

```

```

2269   -- ibia-Owner           { 1 3 133 16 840 9 84 4 1 1 }
2270   -- id-ibia-SAFLINK      { 1 3 133 16 840 9 84 4 1 1 }
2271   -- id-ibia-Bioscrypt     { 1 3 133 16 840 9 84 4 1 2 }
2272   -- id-ibia-Visionics     { 1 3 133 16 840 9 84 4 1 3 }
2273   -- id-ibia-InfineonTechnologiesAG { 1 3 133 16 840 9 84 4 1 4 }
2274   -- id-ibia-IridianTechnologies { 1 3 133 16 840 9 84 4 1 5 }
2275   -- id-ibia-Veridicom       { 1 3 133 16 840 9 84 4 1 6 }
2276   -- id-ibia-CyberSIGN       { 1 3 133 16 840 9 84 4 1 7 }
2277   -- id-ibia-eCryp          { 1 3 133 16 840 9 84 4 1 8 }
2278   -- id-ibia-FingerprintCardsAB { 1 3 133 16 840 9 84 4 1 9 }
2279   -- id-ibia-SecuGen         { 1 3 133 16 840 9 84 4 1 10 }
2280   -- id-ibia-PreciseBiometric { 1 3 133 16 840 9 84 4 1 11 }
2281   -- id-ibia-Identix         { 1 3 133 16 840 9 84 4 1 12 }
2282   -- id-ibia-DERMALOG        { 1 3 133 16 840 9 84 4 1 13 }
2283   -- id-ibia-LOGICO          { 1 3 133 16 840 9 84 4 1 14 }
2284   -- id-ibia-NIST            { 1 3 133 16 840 9 84 4 1 15 }
2285   -- id-ibia-A3Vision        { 1 3 133 16 840 9 84 4 1 16 }
2286   -- id-ibia-NEC             { 1 3 133 16 840 9 84 4 1 17 }
2287   -- id-ibia-STMicroelectronics { 1 3 133 16 840 9 84 4 1 18 }
2288   -- x9-Owner                { 1 3 133 16 840 9 84 4 2  }
2289   -- certificate-Extensions { 1 3 133 16 840 9 84 5  }
2290   -- x968-biometricTemplates { 1 3 133 16 840 9 84 5 0  }
2291   -- x509-biometricTemplates { 1 3 133 16 840 9 84 5 1  }
2292
2293   -- X9.84 arc; base object identifier --
2294
2295   x9-84 OID ::= {
2296     iso(1) identified-organization(3) tc68(133) country(16)
2297       x9(840) x9Standards(9) x9-84(84)
2298   }
2299
2300   -- X9.84 ASN.1 modules --
2301
2302   x9-84-Module OID ::= { x9-84 modules(0) }
2303
2304   x9-84-Biometrics  OID ::= { x9-84-Module biometrics(1) rev(1) }
2305
2306   x9-84-CMS        OID ::= { x9-84-Module cms(2) rev(1) }
2307
2308   x9-84-Identifiers OID ::= { x9-84-Module ids(3) rev(1) }
2309
2310   -- X9.84 biometric technologies --
2311
2312   biometric OID ::= { x9-84 biometrics(1) }
2313
2314   id-unknown-Type      OID ::= { biometric unknownType(0) }
2315   id-body-Odor          OID ::= { biometric bodyOdor(1) }
2316   id-dna                OID ::= { biometric dna(2) }
2317   id-ear-Shape          OID ::= { biometric ear-Shape(3) }
2318   id-facial-Features    OID ::= { biometric facialFeatures(4) }
2319   id-finger-Image        OID ::= { biometric fingerImage(5) }
2320   id-finger-Geometry     OID ::= { biometric fingerGeometry(6) }
2321   id-hand-Geometry       OID ::= { biometric handGeometry(7) }
2322   id-iris-Features        OID ::= { biometric irisFeatures(8) }
2323   id-keystroke-Dynamics  OID ::= { biometric keystrokeDynamics(9) }
2324   id-palm                 OID ::= { biometric palm(10) }
2325   id-retina               OID ::= { biometric retina(11) }
2326   id-signature             OID ::= { biometric signature(12) }
2327   id-speech-Pattern       OID ::= { biometric speech-Pattern(13) }
2328   id-thermal-Image        OID ::= { biometric thermalImage(14) }

```

```

2329      id-vein-Pattern      OID ::= { biometric veinPattern(15) }
2330      id-thermal-Face-Image OID ::= { biometric thermalFaceImage(16) }
2331      id-thermal-Hand-Image OID ::= { biometric thermalHandImage(17) }
2332      id-lip-Movement       OID ::= { biometric lipMovement(18) }
2333      id-gait                OID ::= { biometric gait(19) }
2334
2335      -- X9.84 biometric technology object identifier fragments --
2336
2337      unknown-Type          RelOID ::= { unknownType(0) }
2338      body-Odor              RelOID ::= { bodyOdor(1) }
2339      dna                     RelOID ::= { dna(2) }
2340      ear-Shape              RelOID ::= { earShape(3) }
2341      facial-Features        RelOID ::= { facialFeatures(4) }
2342      finger-Image           RelOID ::= { fingerImage(5) }
2343      finger-Geometry         RelOID ::= { fingerGeometry(6) }
2344      hand-Geometry           RelOID ::= { handGeometry(7) }
2345      iris-Features           RelOID ::= { irisFeatures(8) }
2346      keystroke-Dynamics     RelOID ::= { keystrokeDynamics(9) }
2347      palm                    RelOID ::= { palm(10) }
2348      retina                  RelOID ::= { retina(11) }
2349      signature               RelOID ::= { signature(12) }
2350      speech-Pattern          RelOID ::= { speechPattern(13) }
2351      thermal-Image           RelOID ::= { thermalImage(14) }
2352      vein-Pattern             RelOID ::= { veinPattern(15) }
2353      thermal-Face-Image      RelOID ::= { thermalFaceImage(16) }
2354      thermal-Hand-Image       RelOID ::= { thermalHandImage(17) }
2355      lip-Movement             RelOID ::= { lipMovement(18) }
2356      gait                     RelOID ::= { gait(19) }
2357
2358      BiometricTypes BIOMETRIC ::= {
2359          { BIOMETRIC id : unknown-Type } |
2360          { BIOMETRIC id : body-Odor } |
2361          { BIOMETRIC id : dna } |
2362          { BIOMETRIC id : ear-Shape } |
2363          { BIOMETRIC id : facial-Features } |
2364          { BIOMETRIC id : finger-Image } |
2365          { BIOMETRIC id : finger-Geometry } |
2366          { BIOMETRIC id : hand-Geometry } |
2367          { BIOMETRIC id : iris-Features } |
2368          { BIOMETRIC id : keystroke-Dynamics } |
2369          { BIOMETRIC id : palm } |
2370          { BIOMETRIC id : retina } |
2371          { BIOMETRIC id : signature } |
2372          { BIOMETRIC id : speech-Pattern } |
2373          { BIOMETRIC id : thermal-Image } |
2374          { BIOMETRIC id : vein-Pattern } |
2375          { BIOMETRIC id : thermal-Face-Image } |
2376          { BIOMETRIC id : thermal-Hand-Image } |
2377          { BIOMETRIC id : lip-Movement } |
2378          { BIOMETRIC id : gait } ,
2379          ...
2380          -- expect additional biometric types --
2381      }
2382
2383
2384      -- X9.84 biometric processing algorithms --
2385
2386      processing-algorithm OID ::= { x9-84 processingAlgorithms(2) }
2387
2388      -- X9.84 biometric matching methods --

```

```

2389 matching-method OID ::= { x9-84 matchingMethods(3) }
2390
2391 -- X9.84 vendor specific formats --
2392
2393 format-Owner OID ::= { x9-84 format-owners(4) }
2394
2395 cbeff-Owner OID ::= { format-Owner cbeff(0) }
2396
2397 ibia-Owner OID ::= { format-Owner ibia(1) }
2398
2399 x9-Owner OID ::= { format-Owner x9(2) }
2400
2401
2402 -- Ibia vendor specific formats registered at http://www.ibia.org
2403
2404 id-ibia-SAFLINK          OID ::= { ibia-Owner 1 }
2405 id-ibia-Bioscrypt         OID ::= { ibia-Owner 2 }
2406 id-ibia-Visionics         OID ::= { ibia-Owner 3 }
2407 id-ibia-InfineonTechnologiesAG OID ::= { ibia-Owner 4 }
2408 id-ibia-IridianTechnologies   OID ::= { ibia-Owner 5 }
2409 id-ibia-Veridicom          OID ::= { ibia-Owner 6 }
2410 id-ibia-CyberSIGN          OID ::= { ibia-Owner 7 }
2411 id-ibia-eCryp              OID ::= { ibia-Owner 8 }
2412 id-ibia-FingerprintCardsAB OID ::= { ibia-Owner 9 }
2413 id-ibia-SecuGen             OID ::= { ibia-Owner 10 }
2414 id-ibia-PreciseBiometric    OID ::= { ibia-Owner 11 }
2415 id-ibia-Identix             OID ::= { ibia-Owner 12 }
2416 id-ibia-DERMALOG            OID ::= { ibia-Owner 13 }
2417 id-ibia-LOGICO              OID ::= { ibia-Owner 14 }
2418 id-ibia-NIST                OID ::= { ibia-Owner 15 }
2419 id-ibia-A3Vision             OID ::= { ibia-Owner 16 }
2420 id-ibia-NEC                  OID ::= { ibia-Owner 17 }
2421 id-ibia-STMicroelectronics  OID ::= { ibia-Owner 18 }
2422
2423
2424 -- When represented as values of type OBJECT IDENTIFIER, these
2425 -- Ibia vendor specific formats may be associated with any ASN.1
2426 -- type.
2427
2428 IBIAboidFormats BIOMETRIC ::= {
2429     { BIOMETRIC oid : id-ibia-SAFLINK           DATA Any } |
2430     { BIOMETRIC oid : id-ibia-Bioscrypt         DATA Any } |
2431     { BIOMETRIC oid : id-ibia-Visionics         DATA Any } |
2432     { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
2433     { BIOMETRIC oid : id-ibia-IridianTechnologies   DATA Any } |
2434     { BIOMETRIC oid : id-ibia-Veridicom          DATA Any } |
2435     { BIOMETRIC oid : id-ibia-CyberSIGN          DATA Any } |
2436     { BIOMETRIC oid : id-ibia-eCryp              DATA Any } |
2437     { BIOMETRIC oid : id-ibia-FingerprintCardsAB DATA Any } |
2438     { BIOMETRIC oid : id-ibia-SecuGen             DATA Any } |
2439     { BIOMETRIC oid : id-ibia-PreciseBiometric    DATA Any } |
2440     { BIOMETRIC oid : id-ibia-Identix             DATA Any } |
2441     { BIOMETRIC oid : id-ibia-DERMALOG            DATA Any } |
2442     { BIOMETRIC oid : id-ibia-LOGICO              DATA Any } |
2443     { BIOMETRIC oid : id-ibia-NIST                DATA Any } |
2444     { BIOMETRIC oid : id-ibia-A3Vision             DATA Any } |
2445     { BIOMETRIC oid : id-ibia-NEC                  DATA Any } |
2446     { BIOMETRIC oid : id-ibia-STMicroelectronics  DATA Any },
2447
2448     ... -- Expect additional vendor specific formats --

```

```

2449 }
2500
2501 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
2502
2503
2504
2505
2506
2507
2508
}
Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
-- Relative object identifier representations of the identical
-- IBI A vendor specific formats defined as OBJECT IDENTIFIER
-- values above are used to identify these formats when they must
-- comply with the fixed format requirements of the BioAPI 1.1
-- specification and are associated with a two byte integer value.
ibia-SAFLINK RelOID ::= { 1 }
ibia-Bioscrypt RelOID ::= { 2 }
ibia-Visionics RelOID ::= { 3 }
ibia-InfineonTechnologiesAG RelOID ::= { 4 }
ibia-IridianTechnologies RelOID ::= { 5 }
ibia-Veridicom RelOID ::= { 6 }
ibia-CyberSIGN RelOID ::= { 7 }
ibia-eCryp RelOID ::= { 8 }
ibia-FingerprintCardsAB RelOID ::= { 9 }
ibia-SecuGen RelOID ::= { 10 }
ibia-PreciseBiometric RelOID ::= { 11 }
ibia-Identix RelOID ::= { 12 }
ibia-DERMALOG RelOID ::= { 13 }
ibia-LOGICO RelOID ::= { 14 }
ibia-NIST RelOID ::= { 15 }
ibia-A3Vision RelOID ::= { 16 }
ibia-NEC RelOID ::= { 17 }
ibia-STMicroelectronics RelOID ::= { 18 }

IBIAidFormats BIOMETRIC ::= {
{ BIOMETRIC id : ibia-SAFLINK DATA BirInt16 } |
{ BIOMETRIC id : ibia-Bioscrypt DATA BirInt16 } |
{ BIOMETRIC id : ibia-Visionics DATA BirInt16 } |
{ BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
{ BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
{ BIOMETRIC id : ibia-Veridicom DATA BirInt16 } |
{ BIOMETRIC id : ibia-CyberSIGN DATA BirInt16 } |
{ BIOMETRIC id : ibia-eCryp DATA BirInt16 } |
{ BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
{ BIOMETRIC id : ibia-SecuGen DATA BirInt16 } |
{ BIOMETRIC id : ibia-PreciseBiometric DATA BirInt16 } |
{ BIOMETRIC id : ibia-Identix DATA BirInt16 } |
{ BIOMETRIC id : ibia-DERMALOG DATA BirInt16 } |
{ BIOMETRIC id : ibia-LOGICO DATA BirInt16 } |
{ BIOMETRIC id : ibia-NIST DATA BirInt16 } |
{ BIOMETRIC id : ibia-A3Vision DATA BirInt16 } |
{ BIOMETRIC id : ibia-NEC DATA BirInt16 } |
{ BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 },
...
    -- Expect others --
}

BirInt16 ::= INTEGER (0..65535)

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

```

```

2509     ... -- Expect additional IRIA vendor specific formats --
2510 }
2511
2512 id-x984BioInfo OID ::= { cbeff-Owner x984BioInfo(0) }
2513
2514
2515 CBEFFoidFormats BIOMETRIC ::= {
2516     { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },
2517
2518     ... -- Expect other objects --
2519 }
2520
2521 x984BioInfo RelOID ::= { x984BioInfo(0) } -- CBEFF owner
2522
2523 CBEFFidFormats BIOMETRIC ::= {
2524     { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },
2525
2526     ... -- Expect other objects --
2527 }
2528
2529 CBEFF-Formats BIOMETRIC ::= {
2530     CBEFFoidFormats | -- Complete object identifiers
2531     CBEFFidFormats, -- Object identifier fragments
2532
2533     ... -- Expect additional CBEFF vendor specific formats --
2534 }
2535
2536 MatchingAIDs BIOMETRIC ::= {
2537
2538     ... -- Expect CBEFF assignments in BiometricInformationSets --
2539 }
2540
2541 ProcessingAIDs BIOMETRIC ::= {
2542
2543     ... -- Expect CBEFF assignments in BiometricInformationSets --
2544 }
2545
2546 X9-Formats BIOMETRIC ::= {
2547     X9oidFormats |
2548     X9idFormats,
2549
2550     ... -- Expect additional X9 vendor specific formats --
2551 }
2552
2553 X9oidFormats BIOMETRIC ::= {
2554     ... -- Expect X9 assigned objects --
2555 }
2556
2557 X9idFormats BIOMETRIC ::= {
2558     ... -- Expect X9 assigned objects of the form { 2 x } --
2559 }
2560
2561 certificate-Extensions OID ::= { x9-84 certificate-Extensions(5) }
2562
2563 x968-biometricTemplates OID ::= { certificate-Extensions 0 }
2564 x509-biometricTemplates OID ::= { certificate-Extensions 1 }
2565
2566 -- RSA PKCS #7 Content type
2567
2568 id-data OID ::= {

```

```

2569     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
2570         pkcs7(7) data(1)
2571     }
2572
2573 -- Security object identifiers
2574
2575
2576 -- FIPS 180-1 and FIPS 180-2 Secure Hash Algorithm --
2577
2578 id-sha1 OID ::= {
2579     iso(1) identified-organization(3) oiw(14) secsig(3)
2580         algorithm(2) 26
2581     }
2582
2583 sha2Algorithm OID ::= {
2584     joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
2585         csor(3) nistAlgorithm(4) hashAlgs(2)
2586     }
2587
2588 id-sha256 OID ::= { sha2Algorithm sha256(1) }
2589
2590 id-sha384 OID ::= { sha2Algorithm sha384(2) }
2591
2592 id-sha512 OID ::= { sha2Algorithm sha512(3) }
2593
2594 SHA-Algorithms ALGORITHM ::= {
2595     { OID id-sha1      PARMS NoIV }  |
2596     { OID id-sha256    }  |
2597     { OID id-sha384    }  |
2598     { OID id-sha512    }, ,
2599
2600     ... -- Expect others --
2601 }
2602
2603 NoIV ::= NULL -- No initialization vector
2604
2605 -- X9.57 DSA signature generated with SHA-1 hash (DSA X9.30)
2606
2607 dsa-with-sha1 OID ::= {
2608     iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }
2609
2610 -- X9.71 HMAC with SHA-1 algorithm
2611
2612 hmac-with-SHA1 OID ::= {
2613     iso(1) identified-organization(3) dod(6)
2614         internet(1) security(5) mechanisms(5) 8 1 2 }
2615
2616 -- RSA PKCS #1 signature generated with SHA-1 hash & encryption scheme
2617
2618 sha1WithRSAEncryption OID ::= {
2619     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 5 }
2620
2621 rsaEncryption OBJECT IDENTIFIER ::= {
2622     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 1 }
2623
2624 -- ANSI X9.52 Triple DES Modes of Operation --
2625
2626 des-edede3-cbc OBJECT IDENTIFIER ::= {
2627     iso(1) member-body(2) us(840) rsadsi(113549)
2628         encryptionAlgorithm(3) 7

```

```
2629 }
2630 -- X9.62 ECDSA signature with SHA-1
2631
2632 ecdsa-with-SHA1 OID ::= {
2633     iso(1) member-body(2) us(840) ansi-x962(10045) signatures(4) 1 }
2634
2635
2636
2637 END -- X9-84-Identifiers --
2638
```

2639 **8 Examples**

2640 Three examples are included in this section to assist readers and implementors of this standard,
2641 and with the goal of promoting secure, interoperable biometric applications and systems.

2642 **8.1 BiometricSyntaxSets (cXER, DER)**

2643 This example illustrates a value of type **BiometricSyntaxSets** encoded in XML markup using the
2644 basic XML Encoding Rules (XER), a canonical variant of the XML Encoding Rules (cXER) and a
2645 compact, canonical binary format using the ASN.1 Distinguished Encoding Rules (DER). The
2646 XER, cXER and DER representations use exactly the same abstract values, based on the same
2647 XCBF ASN.1 schema.

2648 Two representations are well-formed XML markup. The third representation is a compact, binary
2649 DER encoding. Both cXER and DER are suitable for use in cryptographic applications involving
2650 digital signatures, since these encoding rules provide one and only one way to encode any given
2651 value.

2652 The BiometricSyntaxSets value encoded in basic XER:

2653

```
2654 <BiometricSyntaxSets>
2655   <BiometricSyntax>
2656     <biometricObjects>
2657       <BiometricObject>
2658         <biometricHeader>
2659           <version> 0 </version>
2660           <recordType> <id> 4 </id> </recordType>
2661           <dataType> <processed/> </dataType>
2662           <purpose> <audit/> </purpose>
2663           <quality> -1 </quality>
2664           <validityPeriod>
2665             <notBefore> 1980.10.4 </notBefore>
2666             <notAfter> 2003.10.3.23.59.59 </notAfter>
2667           </validityPeriod>
2668           <format>
2669             <formatOwner>
2670               <oid> 2.23.42.9.10.4.2 </oid>
2671             </formatOwner>
2672           </format>
2673         </biometricHeader>
2674         <biometricData> 0A0B0C0D </biometricData>
2675       </BiometricObject>
2676     </biometricObjects>
2677   </BiometricSyntax>
2678 </BiometricSyntaxSets>
```

2679 The same abstract value encodes in 517 octets using canonical XER:

```
2680 <BiometricSyntaxSets><BiometricSyntax><biometricObjects><BiometricObjec
2681 t><biometricHeader><version>0</version><recordType><id>4</id></recordTy
2682 pe><dataType>2</dataType><purpose>6</purpose><quality>-1</quality><val
2683 idityPeriod><notBefore>1980.10.4</notBefore><notAfter>2003.10.3.23.59.59
2684 </notAfter></validityPeriod><format><formatOwner><oid>2.23.42.9.10.4.2<
2685 /oid></formatOwner><format></biometricHeader><biometricData>0A0B0C0D</b
2686 iometricData></BiometricObject></biometricObjects></BiometricSyntax></B
2687 iometricSyntaxSets>
```

2688 The same abstract value encodes in 57 octets using DER:

```
3037A0353033A02BA103810104820102830106  
8401FFA50F80048F3C0A0481078F530A03173B  
3BA60AA0088006672A090A040281040A0B0C0D
```

2692

8.2 SignedData

2694 This example illustrates how to encode one or more biometric samples or templates for
2695 cryptographic enhancement to provide authentication of origin and data integrity services for
2696 biometric samples or templates.

2697

2698 The **SubjectPublicKeyInfo** value from the signer's X.509 Public Key Certificate is shown here in
2699 the hexadecimal representation of a DER encoded sequence:

2700

```
308201B73082012C06072A8648CE3804013082011F028181  
00FD7F53811D75122952DF4A9C2EECE4E7F611B7523CEF44  
00C31E3F80B6512669455D402251FB593D8D58FABFC5F5BA  
30F6CB9B556CD7813B801D346FF26660B76B9950A5A49F9F  
E8047B1022C24FBBA9D7FEB7C61BF83B57E7C6A8A6150F04  
FB83F6D3C51EC3023554135A169132F675F3AE2B61D72AEF  
F22203199DD14801C70215009760508F15230BCCB292B982  
A2EB840BF0581CF502818100F7E1A085D69B3DDECBCBAC5C  
36B857B97994AFBBFA3AEA82F9574C0B3D0782675159578E  
BAD4594FE67107108180B449167123E84C281613B7CF0932  
8CC8A6E13C167A8B547C8D28E0A3AE1E2BB3A675916EA37F  
0BFA213562F1FB627A01243BCCA4F1BEA8519089A883DFE1  
5AE59F06928B665E807B552564014C3BFECF492A03818400  
028180476ACACB486186A153E25AE0E243FAAF0CD9105CF4  
DCF63412F36ABF671F53637E5F9FA7C5ADC78288FDB9FA3C  
FAFDEBFDD7A7C3FF2BD63D32F4773413EBD9EAB3CA03BA2D  
ED583187763181CB376954FD13F1F8E046D4E3D40652CA8D  
4645439A3ADB8D964F98F81E57147BDF4C009885CAD55D13  
B38DBAA2F9CBF13DC525F6
```

2720

2721 The biometric objects to be signed with the signer's private key associated with the public key
2722 provided above shown in the canonical form used as input to the message digest process used to
2723 create the digital signature on the signed content:

2724

```
<BiometricObjects><BiometricObject><biometricHeader><version>0</version>  
><recordType><id>4</id></recordType><dataType>2</dataType><purpose>6</p  
urpose><quality>-1</quality><validityPeriod><notBefore>1998.10.1</notBe  
fore><notAfter>2008.10.1</notAfter></validityPeriod><format><formatOwn  
er><oid>2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><  
biometricData>0102030405060708090A0B0C0D0E0F11</biometricData></Biometr  
icObject><BiometricObject><biometricHeader><version>0</version><recordT  
ype><id>4</id></recordType><dataType>1</dataType><purpose>3</purpose>  
<quality>50</quality><validityPeriod><notBefore>1998.10.2</notBefore><n  
otAfter>2008.10.2</notAfter></validityPeriod><format><formatOwner><oid>  
2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><biometr  
icData>0102030405060708090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11  
</biometricData></BiometricObject><BiometricObject><biometricHeader><ve
```

```

2738     rsion>0</version><recordType><id>4</id></recordType><dataType>0</dataTy
2739     pe><purpose>3</purpose><quality>100</quality><validityPeriod><notBefore
2740     >1998.10.3</notBefore><notAfter>2008.10.3</notAfter></validityPeriod><f
2741     ormat><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner></format></
2742     biometricHeader><biometricData>0102030405060708090A0B0C0D0E0F1101020304
2743     05060708090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11</biometricData
2744     ></BiometricObject></BiometricObjects>
2745

```

2746 The complete integrity object represented as an XML document:

```

2747
2748     <?xml version="1.0" encoding="UTF-8"?>
2749
2750     <BiometricSyntaxSets>
2751         <BiometricSyntax>
2752             <integrityObjects>
2753                 <biometricObjects>
2754                     <BiometricObjects>
2755
2756                     <BiometricObject>
2757                         <biometricHeader>
2758                             <version> 0 </version>
2759                             <recordType> <id> 4 </id> </recordType>
2760                             <dataType> 2 </dataType>
2761                             <purpose> 6 </purpose>
2762                             <quality> -1 </quality>
2763                             <validityPeriod>
2764                             <notBefore> 1998.10.1 </notBefore>
2765                             <notAfter> 2008.10.1 </notAfter>
2766                             </validityPeriod>
2767                         <format>
2768                             <formatOwner>
2769                             <oid> 2.23.42.9.10.4.2 </oid>
2770                             </formatOwner>
2771                         </format>
2772                     </biometricHeader>
2773                     <biometricData>
2774                         0102030405060708090A0B0C0D0E0F11
2775                     </biometricData>
2776                 </BiometricObject>
2777
2778                 <BiometricObject>
2779                     <biometricHeader>
2780                         <version> 0 </version>
2781                         <recordType> <id> 4 </id> </recordType>
2782                         <dataType> 1 </dataType>
2783                         <purpose> 3 </purpose>
2784                         <quality> 50 </quality>
2785                         <validityPeriod>
2786                         <notBefore> 1998.10.2 </notBefore>
2787                         <notAfter> 2008.10.2 </notAfter>
2788                         </validityPeriod>
2789                         <format>
2790                             <formatOwner>
2791                             <oid> 2.23.42.9.10.4.2 </oid>
2792                             </formatOwner>
2793                         </format>
2794                     </biometricHeader>
2795                     <biometricData>
2796                         0102030405060708090A0B0C0D0E0F11

```

```

2797                               0102030405060708090A0B0C0D0E0F11
2798                         </biometricData>
2799                       </BiometricObject>
2800
2801                         <BiometricObject>
2802                           <biometricHeader>
2803                             <version> 0 </version>
2804                             <recordType> <id> 4 </id> </recordType>
2805                             <dataType> 0 </dataType>
2806                             <purpose> 3 </purpose>
2807                             <quality> 100 </quality>
2808                             <validityPeriod>
2809                               <notBefore> 1998.10.3 </notBefore>
2810                               <notAfter> 2008.10.3 </notAfter>
2811                           </validityPeriod>
2812                           <format>
2813                             <formatOwner>
2814                               <oid> 2.23.42.9.10.4.2 </oid>
2815                             </formatOwner>
2816                           </format>
2817                         </biometricHeader>
2818                         <biometricData>
2819                           0102030405060708090A0B0C0D0E0F110102030405060708
2820                           090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11
2821                         </biometricData>
2822                       </BiometricObject>
2823
2824                         </BiometricObjects>
2825                     </biometricObjects>
2826
2827                     <integrityBlock>
2828                       <signedData>
2829                         <version> 84 </version>
2830                         <digestAlgorithms>
2831                           <DigestAlgorithmIdentifier>
2832                             <algorithm> 1.3.14.3.2.26 </algorithm>
2833                             <parameters> <NoIV/> </parameters>
2834                           </DigestAlgorithmIdentifier>
2835                         </digestAlgorithms>
2836                         <encapContentInfo>
2837                           <eContentType> 1.2.840.113549.1.7.1 </eContentType>
2838                         </encapContentInfo>
2839                         <signerInfos>
2840                           <SignerInfo>
2841                             <version> 84 </version>
2842                             <sid>
2843                               <certHash>
2844                                 <withAlgID>
2845                                   <hashAlgorithm>
2846                                     <algorithm> 1.3.14.3.2.26 </algorithm>
2847                                     <parameters> <NoIV/> </parameters>
2848                                   </hashAlgorithm>
2849                                   <digest>
2850                                     DA9245BCD6E666749F43C1A1BD070BAF259B70AA
2851                                   </digest>
2852                                 </withAlgID>
2853                               </certHash>
2854                             <sid>
2855                               <digestAlgorithm>
2856                                 <algorithm> 1.3.14.3.2.26 </algorithm>

```

```

2857          <parameters> <NoIV/> </parameters>
2858      </digestAlgorithm>
2859      <signatureAlgorithm>
2860          <algorithm> 1.2.840.10040.4.3 </algorithm>
2861          <parameters> <NoIV/> </parameters>
2862      </signatureAlgorithm>
2863      <signature>
2864          302C02144F9CA4507E2638AE9B632A3698A7AE84858F13
2865          3802140BD484312B36B090D2DF8B8A4719353F9A1EFAA5
2866      </signature>
2867      </SignerInfo>
2868  </signerInfos>
2869  </signedData>
2870 </integrityBlock>
2871
2872 </integrityObjects>
2873 </BiometricSyntax>
2874 </BiometricSyntaxSets>
2875
2876

```

2877 8.3 EncryptedData (fixedKey)

2878 This example illustrates how to encode a series of one or more biometric samples or templates
 2879 for cryptographic enhancement. A group of three biometric objects is used here, though XCBF
 2880 allows any number of objects to be included. The optional, cleartext biometric headers are not
 2881 included in the example message. The group of three biometric objects is encrypted for privacy
 2882 using a fixed Triple DES key.

2883

2884 As shown in this example, XCBF users can create and exchange arbitrary collections of biometric
 2885 information to suit the needs of their security applications. This capability provides the application
 2886 designer complete flexibility. The order of the biometric objects is determined in the application by
 2887 the sender, allowing them to prioritize or order biometric information for purposes such as aging
 2888 of data, or grouping records by quality or data type or entity.

2889

2890 Collections of useful biometric information include:

- 2891 • pairs of iris image templates for an individual; one for each eye
- 2892 • collections of paired iris image templates for a group of individuals
- 2893 • collections of finger print image templates, one per digit for an individual
- 2894 • sets of individual finger print image template collections for a group of persons
- 2895 • a collection of mixed biometric templates for an individual; say, retina, hand geometry, and DNA
- 2896 • collections of templates for groups of individuals, such as:
 - 2897 – all employees at work today, or
 - 2898 – all of the passengers on Flight 12, or
 - 2899 – all of the finger print samples collected on Tuesday

2900

2902 The hex encoding of a two key Triple DES key used to encrypt a series of three values of type
 2903 **BiometricObject** is represented in hexadecimal notation as:

2904
 2905 **623131F21CA8B9B6** **707A7F3DB68A5401** **623131F21CA8B9B6**
 2906
 2907 The biometric objects to be encrypted are shown here for ease of reading, and are not in the
 2908 canonical form used as input to the encryption process in this example:

```

2909
2910 <BiometricObjects>
2911
2912   <BiometricObject>
2913     <biometricHeader>
2914       <version> 0 </version>
2915       <recordType> <id> 4 </id> </recordType>
2916       <dataType> 2 </dataType>
2917       <purpose> 6 </purpose>
2918       <quality> -1 </quality>
2919       <validityPeriod>
2920         <notBefore> 1998.10.1 </notBefore>
2921         <notAfter> 2008.10.1 </notAfter>
2922       </validityPeriod>
2923       <format>
2924         <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
2925       </format>
2926     </biometricHeader>
2927     <biometricData>
2928       0102030405060708090A0B0C0D0E0F11
2929     </biometricData>
2930   </BiometricObject>

2931
2932   <BiometricObject>
2933     <biometricHeader>
2934       <version> 0 </version>
2935       <recordType> <oid> 4 </oid> </recordType>
2936       <dataType> 1 </dataType>
2937       <purpose> 3 </purpose>
2938       <quality> 50 </quality>
2939       <validityPeriod>
2940         <notBefore> 1998.10.2 </notBefore>
2941         <notAfter> 2008.10.2 </notAfter>
2942       </validityPeriod>
2943       <format>
2944         <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
2945       </format>
2946     </biometricHeader>
2947     <biometricData>
2948       0102030405060708090A0B0C0D0E0F11
2949       0102030405060708090A0B0C0D0E0F11
2950     </biometricData>
2951   </BiometricObject>

2952
2953   <BiometricObject>
2954     <biometricHeader>
2955       <version> 0 </version>
2956       <recordType> <id> 4 </id> </recordType>
2957       <dataType> 0 </dataType>
2958       <purpose> 3 </purpose>
2959       <quality> 100 </quality>
2960     <validityPeriod>
```

```

2961             <notBefore> 1998.10.3 </notBefore>
2962             <notAfter> 2008.10.3 </notAfter>
2963         </validityPeriod>
2964         <format>
2965             <formatOwner> <oid> 2.23.42.9.10.4.2 </oid> </formatOwner>
2966             </format>
2967         </biometricHeader>
2968         <biometricData>
2969             0102030405060708090A0B0C0D0E0F110102030405060708
2970             090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11
2971         </biometricData>
2972     </BiometricObject>
2973
2974 </BiometricObjects>

```

2975

2976 The complete XCBF privacy message formed after cryptographic processing of the series of
2977 three biometric objects:

2978

```

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

```

```

3019 CD8DF2C9CF9D497C960FD1DD99D3EAF22442F2B73F59E194
3020 50B102C848A220764AFCAF9B182C619F505166760BC7D63E
3021 09F16BE521C5DB9BFB2784A6BAD69C23899486381A1DEB14
3022 31EECAAB4BD02DDAD00F181AF069D48977F2E01D0184A387
3023 1F132654651E3965B0076238C200390187FAA213F3A6012F
3024 7D9AECD95292B9069FA1409DFB3C32AF19656D9DDD83F07F
3025 D634E9EFAFFF7D7236BC08B4417B4618EF767790E6AE653D
3026 27F8C797F9132E497B20E5A13B2BBF4F64187BE84B61C9C5
3027 CD25ACCE38E3D22F678C79E4532A0B5C5D66EF557D829CCD
3028 466B1607AF462163ED07032474DA7240360A6BF0354795EE
3029 01CF95A8776E9D16717CF140A72804C53F9619FE6B96141E
3030 C97DBB495FFBDC0194665D2A1196EAEC814307CB2FAAA8C8
3031 2476B11EA01B42E2C40C7306710323E288EC1A9166BF9D85
3032 2E3976257315ED6BB70F6933017072830DD1CCC399380114
3033 8935F5CFB19C84B8FE753AE5FFE1D89A2872C5675C4E8E8A
3034 A3A9C4ACBA749E84CBF3969D72C6B84299E7B5F7062C797F
3035 9F057370183C468111522ACBC84DC410900395DCB25E81EA
3036 66B6D20E0AE5D5B7206EABD451539A7EBB2FBDDFCE2B06EE
3037 09E5D5A16904CB2496EDCAE16EFC4B0FAF107B074A06CBE4
3038 5747D8CA1FA0F2F2E8BE5BF27D9F248DF87D351F3EC6EAE7
3039 36A8A6BE99811373480040117B6A78C5C0EA97A524FCD3BA
3040 5F2A4BD3265CE1CA1A94ACA5FE0E6A69FDB6102AD97FA2EE
3041 4DBF6F399075B3AD9339304E16D25959506E90AE771D6CF5
3042 C1C7F793A77D52121C152F66B173A04A851C596090354504
3043 785DBA06B8C49E4DC5E47734DC95451E8D1A58098888F505
3044 E6CA12EA4C8DE8C08102C088A66FE42E38F17694166B7841
3045 CDAB5E46DA0C9ACDBC92840450AA547491CCD82242E32A02
3046 6D1BDB1C6B35B47BCC672E4ED8702DCDC804B20692508747
3047 5C939872694192D6088E0353460C875C0801464AA20AFC71
3048 8CD61A72FDC730609AAA1A20952F3D8790C489B20B3FF6A3
3049 2346B544B18A91FCF485E06F85A58B310746507330CF3376
3050 27875EE54C22CD489BFDD89F7020BD59C8AD15D403DE2E78
3051 75B5571A5EBF80C78B8990448F5B71144FFEBE4B5726BCD1
3052      </encryptedContent>
3053      </encryptedContentInfo>
3054      </fixedKey>
3055      </privacyBlock>
3056      </privacyObjects>
3057  </BiometricSyntax>
3058 </BiometricSyntaxSets>

```

3059 Appendix A. Acknowledgments

3060 The following individuals were members of the committee during the development of this
3061 specification:

- 3062
- 3063 • Lieutenant John Aerts, LASD
 - 3064 • Tyky Aichelen, IBM
 - 3065 • Karl Best, OASIS
 - 3066 • Taylor Boon, BioNetrix Systems
 - 3067 • Robin Cover, Isogen
 - 3068 • Ed Day, Objective Systems
 - 3069 • Dr. Paul Gérôme, AULM
 - 3070 • Phillip H. Griffin (chair), Griffin Consulting
 - 3071 • Todd Harbour, FGM
 - 3072 • William Koenig, Bank of America
 - 3073 • John Larmouth, Larmouth T&PDS Ltd
 - 3074 • Monica Martin, Drake Certivo
 - 3075 • Rob Philpott, RSA Security
 - 3076 • Stefan Ri, Microsoft
 - 3077 • Darran Rolls, Waveset
 - 3078 • Bancroft Scott, OSS Nokalva
 - 3079 • Clifford Thompson, TGI Solutions
 - 3080 • Paul Thorpe, OSS Nokalva
 - 3081 • Alessandro Triglia, OSS Nokalva
 - 3082 • Per Vorm, BEA

3083

Appendix B. Revision History

Rev	Date	By Whom	What
CS-1.0	2003-01-20	Phil Griffin	Committee Specification

3084

3085 Appendix C. Notices

3086 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
3087 that might be claimed to pertain to the implementation or use of the technology described in this
3088 document or the extent to which any license under such rights might or might not be available;
3089 neither does it represent that it has made any effort to identify any such rights. Information on
3090 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
3091 website. Copies of claims of rights made available for publication and any assurances of licenses
3092 to be made available, or the result of an attempt made to obtain a general license or permission
3093 for the use of such proprietary rights by implementers or users of this specification, can be
3094 obtained from the OASIS Executive Director.

3095 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
3096 applications, or other proprietary rights, which may cover technology that may be required to
3097 implement this specification. Please address the information to the OASIS Executive Director.

3098 Copyright © 2003 OASIS Open, Inc. All Rights Reserved.

3099 This document and translations of it may be copied and furnished to others, and derivative works
3100 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
3101 published and distributed, in whole or in part, without restriction of any kind, provided that the
3102 above copyright notice and this paragraph are included on all such copies and derivative works.
3103 However, this document itself does not be modified in any way, such as by removing the
3104 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
3105 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
3106 Property Rights document must be followed, or as required to translate it into languages other
3107 than English.

3108 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
3109 successors or assigns.

3110 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3111 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
3112 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
3113 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
3114 PARTICULAR PURPOSE.