



XML Common Biometric Format

Working Draft 01, Sunday 17 November 2002

Document identifier:

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

Location:

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

Editor:

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

Contributors:

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

Abstract:

Biometrics are measurable physical characteristics or personal behavioral traits that can be used to recognize the identity of an individual, or to verify a claimed identity. This specification defines a common set of secure XML encodings for the patron formats specified in CBEFF, the Common Biometric Exchange File Format (NISTIR 6529) [17]. These XML encodings are based on the ASN.1 schema defined in ANS X9.84:2002 Biometrics Information Management and Security [14]. They conform to the canonical variant of the XML Encoding Rules (XER) for ASN.1 defined in ITU-T Rec. X.693, and will rely on the security and processing requirements specified in X9.96 XML Cryptographic Message Syntax (XCMS) [15] and X9.73 Cryptographic Message Syntax (CMS) [13] standards .

Status:

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

Copyright   2002 The Organization for the Advancement of Structured Information Standards [OASIS]

Table of Contents

40	1	Introduction	4
41	2	Terminology	5
42	3	Acronyms and Abbreviations	6
43	4	Glossary	8
44	5	X9.84 and BioAPI 1.1 Interoperability	10
45	5.1	BiometricSyntaxSets	10
46	5.1.1	BiometricObjects	11
47	5.1.2	IntegrityObjects	29
48	5.1.3	PrivacyObjects	36
49	5.1.4	PrivacyAndIntegrityObjects	47
50	6	References	50
51	6.1	Normative	50
52	7	XCBF Schema	52
53	7.1	X9-84-Biometrics Module	52
54	7.2	X9-84-CMS Module	57
55	7.3	X9-84-Identifiers Module	61
56	8	Examples	70
57	8.1	BiometricSyntaxSets (cXER, DER)	70
58	8.2	SignedData	71
59	8.3	EncryptedData (fixedKey)	75
60		Appendix A. Acknowledgments	79
61		Appendix B. Revision History	80
62		Appendix C. Notices	81
63			

1 Introduction

64

65 Biometrics are measurable physical characteristics or personal behavioral traits that can be used
66 to recognize the identity of an individual, or to verify a claimed identity. This specification defines
67 a common set of secure XML encodings for the patron formats specified in CBEFF, the Common
68 Biometric Exchange File Format (NISTIR 6529). These CBEFF formats currently include the
69 binary biometric objects and information records in two ANSI standards.

70

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

79

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

85 **2 Terminology**

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

3 Acronyms and Abbreviations

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

91
92

4 Glossary

Term	Definition
Attacker	Any individual who is attempting to subvert the operation of the biometric system. The intention may be either to subsequently gain illegal entry to the portal or to deny entry to legitimate users.
Biometric	A measurable, physical characteristic or personal behavioral trait used to authenticate the claimed identity of an enrollee.
Biometric Authentication	The one-to-one process of comparing a submitted biometric sample against the biometric reference template of a single enrollee whose identity is being claimed, to determine whether it matches the enrollee's template. Contrast with Biometric "Identification".
Biometric Data	The extracted information taken from the biometric sample and used either to build a reference template or to compare against a previously created reference template.
Biometric Identification	The one-to-many process of comparing a submitted biometric sample against all of the biometric reference templates on file to determine whether it matches any of the templates and, if so, the identity of the enrollee whose template was matched. The biometric system using the one-to-many approach is seeking to find an identity amongst a database rather than authenticate a claimed identity. Contrast with "Authentication".
Biometric Object	A data record taken from a biometric source or a logical piece of biometric information, which may stand for either a template, or one or more samples. The header is a set of associate attributes that belong with the opaque data, and can include additional information about the purpose, quality, etc. This must be in line with the information content in X9.84 BiometricObject type.
Biometric Sample	Captured data that represents a biometric characteristic of a user of a biometric system.
Biometric System ???	An automated system capable of capturing a biometric sample from a user, extracting biometric data from that sample, comparing that biometric data with data contained in one or more reference templates, deciding how well the sample and reference templates match, and indicating whether or not an attempted authentication

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

93

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

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 used for the secure collection, distribution, and processing, of biometric information. All of the cryptographic operations provided in this standard are applied to a set of values of the ASN.1 type **BiometricObject** defined in the ANS X9.84 standard.

99

100

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

101

102

103

104

105

106

5.1 BiometricSyntaxSets

107

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

108

109

110

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

111

112

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

113

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.

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

173

```
174 typedef struct bioapi_bir {  
175     BioAPI_BIR_HEADER          Header;  
176     BioAPI_BIR_BIOMETRIC_DATA_PTR  BiometricData;  
177     BioAPI_DATA_PTR            Signature;  
178 } BioAPI_BIR, *BioAPI_BIR_PTR ;
```

179

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

183

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

186

```
187 <biometricObjects>  
188   <BiometricObjects>  
189     <BiometricObject>  
190       <biometricHeader>  
191         ...  
192       </biometricHeader>  
193       <biometricData>  
194         ...  
195       </biometricData>  
196     </BiometricObject>  
197   </BiometricObjects>  
198 </biometricObjects>
```

199

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

202

203 5.1.1.1 BiometricHeader

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

206

```
207 BiometricHeader ::= SEQUENCE {  
208   version          BiometricVersion  DEFAULT hv1,  
209   recordType       RecordType  OPTIONAL,  
210   dataType         DataType  OPTIONAL,  
211   purpose          Purpose  OPTIONAL,
```

```

212     quality          Quality OPTIONAL,
213     validityPeriod  ValidityPeriod OPTIONAL,
214     format          Format OPTIONAL
215 }

```

216

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

219

```

220 typedef struct bioapi_bir_header {
221     uint32 Length;
222     BioAPI_BIR_VERSION          HeaderVersion;
223     BioAPI_BIR_DATA_TYPE       Type;
224     BioAPI_BIR_BIOMETRIC_DATA_FORMAT Format;
225     BioAPI_QUALITY             Quality;
226     BioAPI_BIR_PURPOSE         Purpose;
227     BioAPI_BIR_AUTH_FACTORS    FactorsMask;
228 } BioAPI_BIR_HEADER, *BioAPI_BIR_HEADER_PTR ;

```

229

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

235

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

241

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

244

```

245 <biometricHeader>
246   <version> 0 </version>
247   <recordType> <id> 6 </id> </recordType>
248   <dataType> <processed/> </dataType>
249   <purpose> <audit/> </purpose>
250   <quality> 100 </quality>
251   <validityPeriod>
252     <notBefore> 1980.10.4 </notBefore>
253     <notAfter> 2015.10.3.23.59.59 </notAfter>
254   </validityPeriod>
255   <format>

```

```
256     <formatOwner>
257         <oid> 2.23.42.9.10.4.2.0 </oid>
258     </formatOwner>
259     <formatType>
260         <BlindedPrimaryAccountNumber>
261             A23D552FB4490281C1F6683163D9CCB2
262         </BlindedPrimaryAccountNumber>
263     </formatType>
264 </format>
265 </biometricHeader>
```

266

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

269 **5.1.1.1.1 BiometricVersion**

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

272

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

274

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

277

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

282

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

285

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

287

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

289 **5.1.1.1.2 RecordType**

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

291

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

293

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

296

```
297 BiometricTypes BIOMETRIC ::= {  
298   { BIOMETRIC id : unknown-Type      } |  
299   { BIOMETRIC id : body-Odor         } |  
300   { BIOMETRIC id : dna               } |  
301   { BIOMETRIC id : ear-Shape         } |  
302   { BIOMETRIC id : facial-Features  } |  
303   { BIOMETRIC id : finger-Image     } |  
304   { BIOMETRIC id : finger-Geometry  } |  
305   { BIOMETRIC id : hand-Geometry    } |  
306   { BIOMETRIC id : iris-Features    } |  
307   { BIOMETRIC id : keystroke-Dynamics } |  
308   { BIOMETRIC id : palm              } |  
309   { BIOMETRIC id : retina            } |  
310   { BIOMETRIC id : signature        } |  
311   { BIOMETRIC id : speech-Pattern   } |  
312   { BIOMETRIC id : thermal-Image    } |  
313   { BIOMETRIC id : vein-Pattern     } |  
314   { BIOMETRIC id : thermal-Face-Image } |  
315   { BIOMETRIC id : thermal-Hand-Image } |  
316   { BIOMETRIC id : lip-Movement     } |  
317   { BIOMETRIC id : gait              } |  
318  
319   ... -- expect additional biometric types --  
320 }
```

321

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

325

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

328

```
329 typedef sint8 BioAPI_BIR_AUTH_FACTORS;
```

330

```
331 #define BioAPI_FACTOR_MULTIPLE (0x00000001)  
332 #define BioAPI_FACTOR_FACIAL_FEATURES (0x00000002)  
333 #define BioAPI_FACTOR_VOICE (0x00000004)  
334 #define BioAPI_FACTOR_FINGERPRINT (0x00000008)  
335 #define BioAPI_FACTOR_IRIS (0x00000010)  
336 #define BioAPI_FACTOR_RETINA (0x00000020)  
337 #define BioAPI_FACTOR_HAND_GEOMETRY (0x00000040)  
338 #define BioAPI_FACTOR_SIGNATURE_DYNAMICS (0x00000080)
```

{Working Draft}-*{XML Common Biometric Format}*-*{XCBF}*-*{01}*

339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351

```
#define BioAPI_FACTOR_KEYSTROKE_DYNAMICS (0x00000100)
#define BioAPI_FACTOR_LIP_MOVEMENT (0x00000200)
#define BioAPI_FACTOR_THERMAL_FACE_IMAGE (0x00000400)
#define BioAPI_FACTOR_THERMAL_HAND_IMAGE (0x00000800)
#define BioAPI_FACTOR_GAIT (0x00001000)
#define BioAPI_FACTOR_PASSWORD (0x80000000)
```

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

RecordType	Value	BioAPI FactorsMask	Value
unknownType	0	BioAPI_FACTOR_MULTIPLE	0x00000001
body-Odor	1		
Dna	2		
Ear-Shape	3		
facial-Features	4	BioAPI_FACTOR_FACIAL_FEATURES	0x00000002
finger-Image	5	BioAPI_FACTOR_FINGERPRINT	0x00000008
finger-Geometry	6		
hand-Geometry	7	BioAPI_FACTOR_HAND_GEOMETRY	0x00000040
iris-Features	8	BioAPI_FACTOR_IRIS	0x00000010
keystroke-Dynamics	9	BioAPI_FACTOR_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

352

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

357

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

362

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

365

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

367

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

370 5.1.1.1.3 DataType

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

372

```
373 DataType ::= ENUMERATED {  
374     raw          (0),  
375     intermediate (1),  
376     processed   (2)  
377 }
```

378

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

381

```
382 typedef uint8 BioAPI_BIR_DATA_TYPE;
```

383

```
384 #define BioAPI_BIR_DATA_TYPE_RAW (0x01)
```

{Working Draft}-XML Common Biometric Format}-XCBF}-01}

```

385 #define BioAPI_BIR_DATA_TYPE_INTERMEDIATE (0x02)
386 #define BioAPI_BIR_DATA_TYPE_PROCESSED (0x04)

```

387

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

393

```

394 #define BioAPI_BIR_DATA_TYPE_ENCRYPTED (0x10)
395 #define BioAPI_BIR_DATA_TYPE_SIGNED (0x20)

```

396

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

397

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

400

```

401 <dataType> 2 </dataType>

```

402

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

404 5.1.1.1.4 Purpose

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

406

```

407 Purpose ::= ENUMERATED {
408     verify          (1),
409     identify        (2),
410     enroll          (3),
411     enrollVerify    (4),
412     enrollIdentify  (5),
413     audit           (6),
414

```

```

415     ... -- expect others --
416 }

```

417

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

420

```

421 typedef uint8 BioAPI_BIR_PURPOSE;

```

422

```

423 #define BioAPI_PURPOSE_VERIFY (1)
424 #define BioAPI_PURPOSE_IDENTIFY (2)
425 #define BioAPI_PURPOSE_ENROLL (3)
426 #define BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY (4)
427 #define BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY (5)
428 #define BioAPI_PURPOSE_AUDIT (6)

```

429

430

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

431

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

434

```

435 <purpose> 6 </purpose>

```

436

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

438 5.1.1.1.5 Quality

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

440

441
442
443
444
445
446
447

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

448

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

451

452

```
typedef sint8 BioAPI_QUALITY;
```

453

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

457

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

458

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

461

462

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

463

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

465 **5.1.1.1.6 ValidityPeriod**

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

{Working Draft}-XML Common Biometric Format}-XCBF}-01}

468

469
470
471
472
473

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

474

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

477

478

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

479

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

484

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

487

488
489
490
491

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

492

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

495

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

499 **5.1.1.1.7 Format**

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

501

502
503
504
505

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

506

507 A value of this type corresponds closely to the BIR *Format* field in the BioAPI
508 *bioapi_bir_biometric_data_format* structure and is defined as

509

```
510 BioAPI bioapi_bir_biometric_data_format  
511 typedef struct bioapi_bir_biometric_data_format {  
512     uint16 FormatOwner;  
513     uint16 FormatID;  
514 } BioAPI_BIR_BIOMETRIC_DATA_FORMAT,  
515 *BioAPI_BIR_BIOMETRIC_DATA_FORMAT_PTR;
```

516

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

520

```
521 BIOMETRIC ::= CLASS {  
522     &name BIOMETRIC-IDENTIFIER UNIQUE,  
523     &Type OPTIONAL  
524 }  
525 WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
```

526

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

531

```
532 BIOMETRIC-IDENTIFIER ::= CHOICE {  
533     oid OBJECT IDENTIFIER, -- complete object identifier  
534     id RELATIVE-OID -- object identifier fragment  
535 }
```

536

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

539

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

542

```
543 <format>  
544   <formatOwner>  
545     <oid> 2.23.42.9.10.4.2 </oid>  
546   </formatOwner>  
547   <formatType>
```

{Working Draft}-*{XML Common Biometric Format}*-*{XCBF}*-*{01}*

22

548
549
550

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

551

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

557

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

563

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

567 **5.1.1.1.7.1 Biometric Format Registration**

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

571

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

575

576
577
578
579
580
581
582

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

583

584 **5.1.1.1.7.2 CBEFF-Formats**

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

{Working Draft}-*{XML Common Biometric Format}*-*{XCBF}*-*{01}*

587

588

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

589

590

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

591

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

596

597

```
CBEFF-Formats BIOMETRIC ::= {  
598   CBEFFoidFormats | -- Complete object identifiers  
599   CBEFFidFormats,  -- Object identifier fragments  
600  
601   ... -- Expect additional CBEFF vendor specific formats --  
602 }  
603  
604 CBEFFoidFormats BIOMETRIC ::= {  
605   { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },  
606  
607   ... -- Expect other objects --  
608 }  
609  
610 CBEFFidFormats BIOMETRIC ::= {  
611   { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },  
612  
613   ... -- Expect other objects --  
614 }
```

615

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

617

618

```
BiometricInformationSets ::=  
619   SEQUENCE SIZE(1..MAX) OF BiometricInformation  
620  
621 BiometricInformation ::= SEQUENCE {  
622   processingAlgorithms ProcessingAlgorithms OPTIONAL,  
623   matchingMethods      MatchingMethods OPTIONAL  
624 }  
625 (ALL EXCEPT({ -- none; at least one component is present -- })))
```

626

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

630


```

631 ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation
632
633 ProcessingInformation ::= SEQUENCE {
634     id      BIOMETRIC.&name({ProcessingAIDs}),
635     parms  BIOMETRIC.&Type({ProcessingAIDs}{@id})  OPTIONAL
636 }

```

637

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

642

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

645

```

646 ProcessingAIDs BIOMETRIC ::= {
647     ... -- Expect CBEFF assignments in BiometricInformationSets --
648 }
649

```

650

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

654

```

655 MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation
656
657 MatchingInformation ::= SEQUENCE {
658     id      BIOMETRIC.&name({MatchingAIDs}),
659     parms  BIOMETRIC.&Type({MatchingAIDs}{@id})  OPTIONAL
660 }

```

661

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

666

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

669

```

670 MatchingAIDs BIOMETRIC ::= {
671     ... -- Expect CBEFF assignments in BiometricInformationSets --
672 }

```

673

```
}
}
```

674

675 5.1.1.1.7.3 IBIA-Formats

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

677

678

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

679

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

684

685

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

686

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

689

690

690	ibia-SAFLINK	RelOID ::= { 1 }
691	ibia-Bioscrypt	RelOID ::= { 2 }
692	ibia-Visionics	RelOID ::= { 3 }
693	ibia-InfineonTechnologiesAG	RelOID ::= { 4 }
694	ibia-IridianTechnologies	RelOID ::= { 5 }
695	ibia-Veridicom	RelOID ::= { 6 }
696	ibia-CyberSIGN	RelOID ::= { 7 }
697	ibia-eCryp	RelOID ::= { 8 }
698	ibia-FingerprintCardsAB	RelOID ::= { 9 }
699	ibia-SecuGen	RelOID ::= { 10 }
700	ibia-PreciseBiometric	RelOID ::= { 11 }
701	ibia-Identix	RelOID ::= { 12 }
702	ibia-DERMALOG	RelOID ::= { 13 }
703	ibia-LOGICO	RelOID ::= { 14 }
704	ibia-NIST	RelOID ::= { 15 }
705	ibia-A3Vision	RelOID ::= { 16 }
706	ibia-NEC	RelOID ::= { 17 }
707	ibia-STMicroelectronics	RelOID ::= { 18 }
708		

709

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

711

712

```
IBIAidFormats BIOMETRIC ::= {
```

```

713 { BIOMETRIC id : ibia-SAFLINK DATA BirInt16 } |
714 { BIOMETRIC id : ibia-Bioscrypt DATA BirInt16 } |
715 { BIOMETRIC id : ibia-Visionics DATA BirInt16 } |
716 { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
717 { BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
718 { BIOMETRIC id : ibia-Veridicom DATA BirInt16 } |
719 { BIOMETRIC id : ibia-CyberSIGN DATA BirInt16 } |
720 { BIOMETRIC id : ibia-eCryp DATA BirInt16 } |
721 { BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
722 { BIOMETRIC id : ibia-SecuGen DATA BirInt16 } |
723 { BIOMETRIC id : ibia-PreciseBiometric DATA BirInt16 } |
724 { BIOMETRIC id : ibia-Identix DATA BirInt16 } |
725 { BIOMETRIC id : ibia-DERMALOG DATA BirInt16 } |
726 { BIOMETRIC id : ibia-LOGICO DATA BirInt16 } |
727 { BIOMETRIC id : ibia-NIST DATA BirInt16 } |
728 { BIOMETRIC id : ibia-A3Vision DATA BirInt16 } |
729 { BIOMETRIC id : ibia-NEC DATA BirInt16 } |
730 { BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 },
731
732 ... -- Expect others --
733 }

```

734

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

737

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

741

```

742 IBIAoidFormats BIOMETRIC ::= {
743 { BIOMETRIC oid : id-ibia-SAFLINK DATA Any } |
744 { BIOMETRIC oid : id-ibia-Bioscrypt DATA Any } |
745 { BIOMETRIC oid : id-ibia-Visionics DATA Any } |
746 { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG DATA Any } |
747 { BIOMETRIC oid : id-ibia-IridianTechnologies DATA Any } |
748 { BIOMETRIC oid : id-ibia-Veridicom DATA Any } |
749 { BIOMETRIC oid : id-ibia-CyberSIGN DATA Any } |
750 { BIOMETRIC oid : id-ibia-eCryp DATA Any } |
751 { BIOMETRIC oid : id-ibia-FingerprintCardsAB DATA Any } |
752 { BIOMETRIC oid : id-ibia-SecuGen DATA Any } |
753 { BIOMETRIC oid : id-ibia-PreciseBiometric DATA Any } |
754 { BIOMETRIC oid : id-ibia-Identix DATA Any } |
755 { BIOMETRIC oid : id-ibia-DERMALOG DATA Any } |
756 { BIOMETRIC oid : id-ibia-LOGICO DATA Any } |
757 { BIOMETRIC oid : id-ibia-NIST DATA Any } |
758 { BIOMETRIC oid : id-ibia-A3Vision DATA Any } |
759 { BIOMETRIC oid : id-ibia-NEC DATA Any } |
760 { BIOMETRIC oid : id-ibia-STMicroelectronics DATA Any },
761

```

```
762     ... -- Expect additional vendor specific formats --
763 }
764
765 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
```

766

767 5.1.1.1.7.4 X9-Formats

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

769

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

771

772 Under the X9 arc, both complete and relative object identifiers can be registered for use by an
773 biometric application vendors. PHG – To be completed.

774

```
775 X9-Formats BIOMETRIC ::= {
776     X9oidFormats |
777     X9idFormats,
778
779     ... -- Expect additional X9 vendor specific formats --
780 }
781
782 X9oidFormats BIOMETRIC ::= {
783     ... -- Expect X9 assigned objects --
784 }
785
786 X9idFormats BIOMETRIC ::= {
787     ... -- Expect X9 assigned objects of the form { 2 x } --
788 }
```

789

790 5.1.1.2 BiometricData

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

793

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

795

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

798

```
799 typedef uint8 BioAPI_BIR_BIOMETRIC_DATA;
```

{Working Draft}-~~{XML Common Biometric Format}~~-~~{XCBF}~~-~~{01}~~

800

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

803

804 5.1.2 IntegrityObjects

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

808

```

809 IntegrityObjects ::= SEQUENCE {
810     biometricObjects  EncodedBiometricObjects,
811     integrityBlock    IntegrityBlock
812 }

```

813

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

818

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

820

```

821 IntegrityBlock ::= CHOICE {
822     digitalSignature      DigitalSignature,
823     messageAuthenticationCode  MessageAuthenticationCode,
824     signedData           SignedData,
825     authenticatedData   AuthenticatedData
826 }
827

```

828

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

830

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

831

832 5.1.2.1 DigitalSignature

833 The **digitalSignature** choice alternative of the **integrityBlock** component of type
834 **IntegrityObjects** is a value of type **DigitalSignature**. This type is a sequence of two
835 components, an algorithm identifier and a digital signature. Type **DigitalSignature** is defined as
836

```
837 DigitalSignature ::= SIGNATURE { EncodedBiometricObjects }
```

838

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

842

```
843 SIGNATURE { ToBeSigned } ::= SEQUENCE {  
844     algorithmID  SignatureAlgorithmIdentifier,  
845     signature    BIT STRING( CONSTRAINED BY {  
846                 -- signature on a value of -- ToBeSigned })  
847 }
```

848

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

850

```
851 DigitalSignature ::= SEQUENCE {  
852     algorithmID  SignatureAlgorithmIdentifier,  
853     signature    BIT STRING  
854                 ( CONSTRAINED BY { EncodedBiometricObjects })  
855 }
```

856

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

860

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

863

```
864 <integrityBlock>  
865   <digitalSignature>  
866     <algorithmID>  
867       <algorithm>1.2.840.10040.4.3</algorithm>  
868       <parameters><NoIV></parameters>  
869     </algorithmID>  
870     <signature>  
871       ...  
872     </signature>
```

873
874
875

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

876

877 This markup uses the digital signature choice alternative of the integrity block, a value of type
878 **DigitalSignature**. This type provides a simple digital signature on a value of type
879 **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) [8] with Secure Hash
880 Algorithm (SHA1) [9] and its associated parameters, an initialization vector, <IV> is used for
881 signing a value of **EncodedBiometricObjects**. An ellipsis is used as a placeholder and the
882 signature is not shown.

883 5.1.2.1.1 Digital Signature Process

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

888

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

895

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

900

901 5.1.2.2 MessageAuthenticationCode

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

906

907

```
MessageAuthenticationCode ::= MAC { EncodedBiometricObjects }
```

908

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

{Working Draft}-XML Common Biometric Format-XCBF-01}

912

913

```
MAC { ToBeMACed } ::= SEQUENCE {
914   keyName      OCTET STRING OPTIONAL,
915   algorithmID  MACAlgorithmIdentifier,
916   mac          OCTET STRING (CONSTRAINED BY {
917               -- MAC or HMAC on a value of -- ToBeMACed })
918 }
```

919

920

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

921

922

923

```
MessageAuthenticationCode ::= SEQUENCE {
924   keyName      OCTET STRING OPTIONAL,
925   algorithmID  MACAlgorithmIdentifier,
926   mac          OCTET STRING
927               ( CONSTRAINED BY { EncodedBiometricObjects })
928 }
```

929

930

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

931

932

933

934

935

936

For both MAC and HMAC, cryptographic keys shall be chosen at random, protected and kept

937

938

939

940

941

942

A value of the **messageAuthenticationCode** choice alternative of the **integrityBlock**

943

component of type **IntegrityObjects** can be represented in XML markup as

944

945

```
<integrityBlock>
946   <messageAuthenticationCode>
947     <keyName> ... </keyName>
948     <algorithmID>
949       <algorithm>1.3.6.1.5.5.8.1.2</algorithm>
950     </algorithmID>
951     <mac>
952       ...
953     </mac>
954   </messageAuthenticationCode>
955 </integrityBlock>
```

950

951

952

953

954

955

{Working Draft}-XML Common Biometric Format-XCBF-01}

956

957 This markup uses the message authentication code choice alternative of the integrity block, a
958 value of type **MessageAuthenticationCode**. Here this type is used to provide a hashed MAC on
959 a value of type **EncodedBiometricObjects**. The HMAC with SHA1 algorithm and a shared
960 secret key are used to compute an HMAC on a value of **EncodedBiometricObjects**. An ellipsis
961 is used as a placeholder for the shared secret key and the HMAC results, and neither of these
962 values is shown.

963

964 **5.1.2.2.1 Message Authentication Process**

965

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

971

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

975 **5.1.2.3 SignedData**

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

978

```
979 SignedData ::= SEQUENCE {  
980     version          CMSVersion,  
981     digestAlgorithms DigestAlgorithmIdentifiers,  
982     encapContentInfo EncapsulatedContentInfo,  
983     certificates     [0] CertificateSet OPTIONAL,  
984     crls             [1] CertificateRevocationLists OPTIONAL,  
985     signerInfos      SignerInfos  
986 }
```

987

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

989

- | | |
|-------------------------|--|
| version | An integer version number of the syntax definition. The version shall be 84 in this standard. |
| digestAlgorithms | The set of message digest algorithms used by the signers. This set contains only one element since there is only one signer of the content in this standard. |

encapContentInfo	An identifier of the type of content signed and optionally, the signed content. In this standard the signed content is not present. The type of content is always ordinary data as the nesting of cryptographic types is neither required nor supported.
certificates	An optional set of one or more X.509 [1] or X9.68 [11] certificates. When present, this set may contain more certificates or less than needed to verify the signature on the signed data or less.
crls	An optional set of one or more X.509 certificate revocation lists (CRLs). When present, there may be more CRLs or less than needed to determine whether or not the certificate of the signer is valid.
signerInfos	A set of information for each signer of the content. This set contains only one element since there is only one signer of the content in this standard.

990

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

993

994

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

995

996

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

1003

1004 The **SignerIdentifier** type is used to identify the public key certificate associated with the private
 1005 key used to create the **signature** component of **SignerInfo**. This type is defined in XCBF as a
 1006 choice type having only one alternative:

1007

1008

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

1011

1012 The **certHash** choice alternative of **SignerIdentifier** provides a single, simple mechanism that
 1013 allows any type of digital certificate to be identified.

1014 PHG: Will this be the hash be over the DER [6] or the cXER or DER encoding of X.509 and X9.68
 1015 certificates? Seems that if they are included in the SignedData message, then it should be over

1016 the hash of whatever they are encoded in. But if not included how do we know? Application
1017 dependent? Need words for this.

1018

```
1019 Hash ::= CHOICE {  
1020     ietf          CertHash, -- SHA-1 hash of entire certificate  
1021     withAlgID   DigestInfo  
1022 }  
1023  
1024 CertHash ::= OCTET STRING (ENCODED BY sha-1)  
1025  
1026 DigestInfo ::= SEQUENCE {  
1027     hashAlgorithm DigestAlgorithmIdentifier,  
1028     digest          OCTET STRING  
1029 }
```

1030

1031 5.1.2.3.1 Message digest and signature process

1032 A message digest is used to create the digital signature carried in the **SignerInfo** component of
1033 **SignedData**. The message digest is calculated using the algorithm and parameters specified in
1034 the **digestAlgorithm** component of **SignerInfo**, and the value of the **eContent** component of
1035 **EncapsulatedContentInfo**. The **eContentType** component of **EncapsulatedContentInfo**
1036 identifies the type of value being signed. This is always the object identifier value **id-data** in
1037 XCBF.

1038

1039 When a value of type **SignedData** is represented as XML markup, the starting and ending
1040 **eContent** tags are excluded from the message digest process. Only the "value" portion of the
1041 complete canonical XER encoding of **eContent** is digested. The **<eContent>** and **</eContent>**
1042 tags are excluded from the message digest process, and the digest is calculated starting with the
1043 **<BiometricObjects>** tag and ending with the **</BiometricObjects>** tag.

1044

1045 The result of the message digest process is then digitally signed using the signer's private key
1046 and the signature algorithm and parameters specified in the **signatureAlgorithm** component of
1047 **SignerInfo**. The result of the signature process becomes the value of the **signature** component
1048 of the **SignerInfo** component of **SignedData**.

1049

1050 5.1.2.4 AuthenticatedData

1051 Type **AuthenticatedData** is defined as

1052

```
1053 AuthenticatedData ::= SEQUENCE {  
1054     version          CMSVersion,  
1055     recipientInfos   RecipientInfos,  
1056     macAlgorithm     MACAlgorithmIdentifier,
```

{Working Draft}-XML Common Biometric Format-XCBF-01}

1057
1058
1059

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

1060

1061 5.1.3 PrivacyObjects

1062 The **privacyObjects** choice alternative of type **BiometricSyntax** is a value of type
1063 **PrivacyObjects**. This type is defined as a sequence of two components, **biometricHeaders** and
1064 **privacyBlock**.

1065

1066
1067
1068
1069

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

1070

1071 The **biometricHeaders** component is a series of one or more values of type **BiometricHeader**.

1072

1073

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

1074

1075 This optional **BiometricHeaders** component is not protected by encryption and should be
1076 present only when a privacy object is used in a secure environment, or when the information
1077 contained in the **biometricHeaders** component does not compromise security or assist an
1078 attacker. In a secure setting these biometric headers may be used as a convenience, to assist in
1079 searches of biometric information and in database management operations.

1080

1081 The encrypted content in the privacy block contains a series of one or more values of type
1082 **BiometricObject**, including their biometric headers. To be useful, the **biometricHeaders**
1083 component need only provide an indication of the information contained in the encrypted privacy
1084 block. But this component need not contain exactly the same information as the headers in the
1085 encrypted privacy block, and may contain only a single **BiometricHeader** value when present.

1086

1087 The **privacyBlock** component of type **PrivacyObjects** offers three choice alternatives, **fixedKey**,
1088 **namedKey** and **establishedKey**.

1089

1090
1091
1092
1093
1094

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

1095

1096 The **fixedKey** and **namedKey** choice alternatives are based on the **EncryptedData** type. The
1097 **establishedKey** alternative is based on type **EnvelopedData**. Each of these alternatives has
1098 different characteristics, and the alternative chosen will depend upon application requirements
1099 and the key management scheme being used.

1100

1101 A value of the **privacyObjects** choice alternative of type **BiometricSyntax** can be represented
1102 in XML markup as

1103

```
1104 <privacyObjects>  
1105   <privacyBlock>  
1106     ...  
1107   </privacyBlock>  
1108 </privacyObjects>
```

1111

1112 This markup illustrates the wrapper for a typical privacy object. The optional privacy object
1113 biometric headers are not present. An ellipsis is used as a placeholder and the details of the
1114 privacy block choice alternative is not shown.

1115 5.1.3.1 Encrypted Content Information

1116 All three of the privacy block choice alternatives contain a value of type **EncryptedContentInfo**
1117 defined as

1118

```
1119 EncryptedContentInfo ::= SEQUENCE {  
1120   contentType           ContentType,  
1121   contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,  
1122   encryptedContent      [0] EncryptedContent  
1123 }
```

1124

1125 The **contentType** component identifies the type of encrypted content. In XCBF, the type of
1126 encrypted content is always a value of **EncodedBiometricObjects**, a series of one or more
1127 values of type **BiometricObject** encoded using the XML Encoding Rules. The type of encrypted
1128 content is identified as ordinary data by the information object identifier value **id-data**. This value
1129 is defined as

1130

```
1131 <contentType> 1.2.840.113549.1.7.1 </contentType>
```

1132

1133 The **contentEncryptionAlgorithm** component identifies the content encryption algorithm and
1134 any associated parameters used to encrypt and decrypt the **EncodedBiometricObjects**. This

1135 content encryption algorithm is a value of type **ContentEncryptionAlgorithmIdentifier** defined
1136 as

1137

```
1138 ContentEncryptionAlgorithmIdentifier ::=
1139     AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
```

1140

1141 The definition of type **ContentEncryptionAlgorithmIdentifier** is based on the parameterized
1142 type **AlgorithmIdentifier** {} and the information object set **ContentEncryptionAlgorithms**,
1143 defined as

1144

```
1145 ContentEncryptionAlgorithms ALGORITHM ::= {
1146     { OID des-ede3-cbc PARS IV },
1147     ... -- Expect other content encryption algorithms --
1148 }
1149
1150 IV ::= OCTET STRING (SIZE(8))
```

1151

1152 **ContentEncryptionAlgorithms** specifies an extensible set of **ALGORITHM** information objects.
1153 The fields of these information objects are used to constrain the valid values of the components
1154 of type **ContentEncryptionAlgorithmIdentifier**. Though only one content encryption algorithm
1155 object is defined explicitly in this set, implementations should expect additional algorithms.

1156

1157 The **ContentEncryptionAlgorithms** information object set contains a single object that identifies
1158 the encryption algorithm described in ANS X9.52 [10] as Triple DES (TDES) in CBC (cipher block
1159 chaining) mode. Only the two key and three key variants of TDES are supported in XCBF. The
1160 single key variant of TDES is simply the DES algorithm and is generally used only for backwards
1161 compatibility with existing DES based applications and is considered vulnerable to attack.

1162

1163 The Triple DES algorithm consists of three sequential DES operations, encrypt, decrypt, and
1164 encrypt. For three key TDES a different key is used for each DES operation. For two key TDES
1165 one key is used for both DES encrypt operations, and the second key is used for the DES decrypt
1166 operation.

1167

1168 The **encryptedContent** component contains a value of type **EncodedBiometricObjects**
1169 encrypted using the content encryption algorithm given in the **contentEncryptionAlgorithm**
1170 component. A value of **encryptedContent** is an opaque string of octets treated as having no
1171 discernable structure. This string is a value of type **EncryptedContent** defined as

1172

```
1173 EncryptedContent ::= OCTET STRING
```

1174

1175 A value of the **encryptedContentInfo** component of any of the privacy block choice alternative
1176 types can be represented in XML markup as

1177

```
1178 <encryptedContentInfo>  
1179   <contentType>1.2.840.113549.1.7.6</contentType>  
1180   <contentEncryptionAlgorithm>  
1181     <algorithm>1.2.840.113549.3.7</algorithm>  
1182     <parameters>  
1183       <IV>7EA13D6E143CB5C9</IV>  
1184     </parameters>  
1185   </contentEncryptionAlgorithm>  
1186   <encryptedContent>  
1187     ...  
1188   </encryptedContent>  
1189 </encryptedContentInfo>
```

1190

1191 This markup illustrates a typical value. The encrypted content type is identified as ordinary data.
1192 The Triple DES content encryption algorithm is identified along with its associated parameters, an
1193 initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not
1194 shown.

1195

1196 5.1.3.2 Fixed Key EncryptedData

1197 The **fixedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1198 value of type **EncryptedData**. This type is a sequence of two components, an integer version
1199 number and a value of type **EncryptedContentInfo**. Type **EncryptedData** is defined as

1200

```
1201 EncryptedData ::= SEQUENCE {  
1202   version          CMSVersion,  
1203   encryptedContentInfo EncryptedContentInfo  
1204 }
```

1205

1206 The **fixedKey** alternative assumes that the recipient of the **EncryptedData** value knows the key
1207 used to encrypt the biometric information, perhaps by prior agreement or as the result of a key
1208 exchange. The **version** component of type **EncryptedData** is always the integer value eighty-
1209 four. The components of type **EncryptedContentInfo** are described in section 4.1.3.1 *Encrypted*
1210 *Content Information*.

1211

1212 A value of the **fixedKey** choice alternative of the **privacyBlock** component of type
1213 **PrivacyObjects** can be represented in XML markup as

1214

```
1215 <fixedKey>
```

```

1216 <version>84</version>
1217 <encryptedContentInfo>
1218   <contentType>1.2.840.113549.1.7.6</contentType>
1219   <contentEncryptionAlgorithm>
1220     <algorithm>1.2.840.113549.3.7</algorithm>
1221     <parameters>
1222       <IV>7EA13D6E143CB5C9</IV>
1223     </parameters>
1224   </contentEncryptionAlgorithm>
1225   <encryptedContent>
1226     ...
1227   </encryptedContent>
1228 </encryptedContentInfo>
1229 </fixedKey>

```

1230

1231 This markup uses the fixed key choice alternative of the privacy block, a value of version number
1232 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
1233 identified as ordinary data. The Triple DES content encryption algorithm is identified along with its
1234 associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder and the
1235 encrypted content is not shown.

1236 5.1.3.2.1 Encryption Process

1237 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1238 **BiometricObject** in their encoded form using a content encryption algorithm and a fixed content
1239 encryption key known to the sender and recipient. The content to be encrypted is a value of type
1240 **EncodedBiometricObjects**. This value is always encoded using the XML Encoding Rules. The
1241 content encryption algorithm used to encrypt the biometric objects is one of the algorithms
1242 specified in the information object set **ContentEncryptionAlgorithms**.

1243

1244 The **contentType** component of type **EncryptedContentInfo** is set to indicate ordinary data. The
1245 associated **contentEncryptionAlgorithm** value is set to identify the algorithm used to encrypt
1246 the content, and the **encryptedContent** value is set to the results of encrypting the content using
1247 this content encryption algorithm.

1248 5.1.3.2.2 Decryption Process

1249 To decrypt a value of type **EncryptedData**, the content encryption algorithm specified in the
1250 **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1251 associated **encryptedContent** component using a known fixed key to recover a value of type
1252 **EncodedBiometricObjects**. This recovered value will contain one or more values of type
1253 **BiometricObject** encoded using the XML Encoding Rules.

1254 5.1.3.3 Named Key EncryptedData

1255 The **namedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects** is a
1256 value of type **NamedKeyEncryptedData**. This type is sequence with two components, **keyName**
1257 and **encryptedData**. Type **NamedKeyEncryptedData** is defined as

{Working Draft}-{XML Common Biometric Format}-{XCBF}-{01}

1258

1259
1260
1261
1262

```

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

```

1263

1264 The **keyName** component explicitly identifies the key used to encrypt and decrypt the content by
1265 name. The **encryptedData** component is a value of type **EncryptedData**. This type contains two
1266 components, an integer **version** number that is always eighty-four in this standard, and an
1267 **encryptedContentInfo** that is a value of type **EncryptedContentInfo** as described in section
1268 4.1.3.1 *Encrypted Content Information*.

1269

1270 A value of the **namedKey** choice alternative of the **privacyBlock** component of type
1271 **PrivacyObjects** can be represented in XML markup as

1272

1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290

```

<namedKey>
  <keyName>6AE173BF5A973D1E</keyName>
  <encryptedData>
    <version>84</version>
    <encryptedContentInfo>
      <contentType>1.2.840.113549.1.7.6</contentType>
      <contentEncryptionAlgorithm>
        <algorithm>1.2.840.113549.3.7</algorithm>
        <parameters>
          <IV>7EA13D6E143CB5C9</IV>
        </parameters>
      </contentEncryptionAlgorithm>
      <encryptedContent>
        ...
      </encryptedContent>
    </encryptedContentInfo>
  </encryptedData>
</namedKey>

```

1291

1292 This markup uses the named key choice alternative of the privacy block, a sequence of a key
1293 name and a value of version number eighty-four of the cryptographic message type
1294 **EncryptedData**. The encrypted content type is identified as ordinary data. The Triple DES
1295 content encryption algorithm is identified along with its associated parameters, an initialization
1296 vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not shown.

1297 5.1.3.3.1 Encryption Process

1298 A value of type **EncryptedData** is created by encrypting a series of one or more values of type
1299 **BiometricObject** in their encoded form using a content encryption algorithm and a named key
1300 that is known to the recipient of the encrypted biometric information. The content to be encrypted
1301 is a value of type **EncodedBiometricObjects**. This value is always encoded using the XML

{Working Draft}-XML Common Biometric Format-XCBF-01}

1302 Encoding Rules. The content encryption algorithm used to encrypt the biometric objects is one of
1303 the algorithms specified in the information object set **ContentEncryptionAlgorithms**.

1304

1305 The **keyName** component of type **NamedKeyEncryptedData** is set to the name of the content
1306 encryption key. The **contentType** component of type **EncryptedContentInfo** is set to indicate
1307 ordinary data. The associated **contentEncryptionAlgorithm** value is set to identify the algorithm
1308 used to encrypt the content, and the **encryptedContent** value is set to the results of encrypting
1309 the content using this content encryption algorithm.

1310 **5.1.3.3.2 Decryption Process**

1311 To decrypt a value of type **NamedKeyEncryptedData**, the content encryption algorithm specified
1312 in the **contentEncryptionAlgorithm** component of type **EncryptedContentInfo** is applied to the
1313 associated **encryptedContent** component using the key identified by the **keyName** component
1314 of type **NamedKeyEncryptedData** to recover a value of type **EncodedBiometricObjects**. This
1315 recovered value will contain one or more values of type **BiometricObject** encoded using the XML
1316 Encoding Rules.

1317 **5.1.3.4 Established Key EnvelopedData**

1318 The **establishedKey** choice alternative of the **privacyBlock** component of type **PrivacyObjects**
1319 is a value of type **EnvelopedData**. Using this type, a message sender can encrypt content that
1320 only the intended message recipient can decrypt.

1321

1322 **EnvelopedData** is defined as a sequence of four components, an integer version number,
1323 message sender information, message recipient information, and a value of type
1324 **EncryptedContentInfo** which is described in section 4.1.3.1 *Encrypted Content Information*.

1325 Type **EnvelopedData** is defined as

1326

```
1327 EnvelopedData ::= SEQUENCE {  
1328     version          CMSVersion,  
1329     originatorInfo   [0] OriginatorInfo OPTIONAL,  
1330     recipientInfos   RecipientInfos,  
1331     encryptedContentInfo EncryptedContentInfo  
1332 }
```

1333

1334 The combination of encrypted content and an encrypted content encryption key forms a “digital
1335 envelope”. The **establishedKey** alternative uses a randomly generated content encryption key to
1336 encrypt digital content. The same key is used to decrypt the content. The content encryption key
1337 shall be protected during transport, so the recipient’s public and private key pair is used to
1338 encrypt and decrypt the content encryption key.

1339

1340 The encrypted content is value of type **EncodedBiometricObjects**. This type is a series of one
1341 or more values of type **BiometricObject** in their encoded form. In XCBF these values are
1342 encoded using the XML Encoding Rules.

1343

1344 The **version** component of type **EnvelopedData** is the integer value eighty-four. The optional
1345 **originatorInfo** component facilitates distribution of digital certificates and certificate revocation
1346 lists. The **recipientInfos** component contains information needed to recover the encrypted
1347 content encryption key used to encrypt the biometric information. The **encryptedContentInfo**
1348 component is a value of type **EncryptedContentInfo**. This type is described in section 4.1.3.1
1349 *Encrypted Content Information*.

1350

1351 A value of the **establishedKey** choice alternative of the **privacyBlock** component of type
1352 **PrivacyObjects** can be represented in XML markup as

1353

```
1354 <establishedKey>  
1355   <version>84</version>  
1356   <recipientInfos>  
1357     <RecipientInfo>  
1358       <ktri>  
1359         <version>84</version>  
1360         <rid>  
1361           <certHash>  
1362             <ietf>6E143CF31A562FA9492681D27A22013D2AAD435D</ietf>  
1363           </certHash>  
1364         </rid>  
1365         <keyEncryptionAlgorithm>  
1366           <algorithm>  
1367             1.2.840.113549.1.1.1  
1368           </algorithm>  
1369           <parameters>  
1370             </NoIV>  
1371           </parameters>  
1372         </keyEncryptionAlgorithm>  
1373         <encryptedKey>  
1374           ...  
1375         </encryptedKey>  
1376       </ktri>  
1377     </RecipientInfo>  
1378   </recipientInfos>  
1379   <encryptedContentInfo>  
1380     <contentType>1.2.840.113549.1.7.6</contentType>  
1381     <contentEncryptionAlgorithm>  
1382       <algorithm>1.2.840.113549.3.7</algorithm>  
1383       <parameters>  
1384         <IV>7EA13D6E143CB5C9</IV>  
1385       </parameters>  
1386     </contentEncryptionAlgorithm>  
1387     <encryptedContent>  
1388       ...
```

1389
1390
1391

```
</encryptedContent>  
</encryptedContentInfo>  
</ establishedKey >
```

1392

1393 This markup uses the established key choice alternative of the privacy block, a value of version
1394 number eighty-four of the cryptographic message type **EnvelopedData**. The optional originator
1395 information is not present. The recipient information uses the key transport choice alternative.
1396 The recipient public and private key pair is indicated by a SHA1 hash of a public key certificate.
1397 The content encryption key is encrypted using the **rsaEncryption** algorithm, which has no
1398 associated parameters indicating that no initialization vector is required. The encrypted content
1399 type is identified as ordinary data. The Triple DES content encryption algorithm is identified along
1400 with its associated parameters, an initialization vector, <IV>. An ellipsis is used as a placeholder
1401 and the encrypted content encryption key and the encrypted content are not shown.

1402

1403 **5.1.3.4.1 Certificates and CRLs**

1404 Type **OriginatorInfo** is a sequence of two components that may contain sets of digital certificates
1405 and certificate revocation lists (CRLs). This type is defined as

1406

1407
1408
1409
1410
1411

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

1412

1413 Any combination of X9.68 domain certificates, X.509 certificates and attribute certificates may be
1414 included in the **CertificateSet** type in any order. There may be more or fewer certificates needed
1415 for any purpose. Certificates are provided as needed to support content key encryption in the key
1416 transport key management technique used in XCBF. Use of the **CertificateSet** type to distribute
1417 certificates is not required. They may be obtained by other means, or an online certificate
1418 validation service may be used instead. Only version one X9.68 domain certificates, version three
1419 X.509 certificates and version two attribute certificates are supported in this standard.

1420

1421 Any number of CRLs may be included in the **CertificateRevocationLists** type in any order.
1422 There may be more or fewer CRLs needed for any purpose. CRLs are provided as needed to
1423 support certificate validation. Use of the **CertificateRevocationLists** type to distribute CRLs is
1424 not required. CRLs may be obtained by other means, or an online certificate validation service
1425 may be used instead. Only version two certificate revocation lists are supported in this standard.

1426

1427 The certificates and certificate revocation lists used in XCBF are signed binary objects, whose
1428 digital signatures have been calculated on values encoded using the Distinguished Encoding
1429 Rules (DER) of ASN.1. In order to verify the signatures on these objects, their original encodings

1430 must be maintained. But these values must also be represented in XML encodings in a useful
1431 textual format. So the values in the **certs** and **crls** components of type **OriginatorInfo** I have
1432 been base64 armored to minimize their size when represented using XML markup while
1433 preserving their original encodings.

1434 5.1.3.4.2 Recipient Information

1435 Type **RecipientInfos** is a series of values of type **RecipientInfo**, one value for each recipient of a
1436 digital envelope in **EnvelopedData**. In XCBF there is always a single digital envelope recipient,
1437 and type **RecipientInfos** is constrained to a series of one **RecipientInfo** and defined as

1438

```
1439 RecipientInfos ::= SET SIZE(1) OF RecipientInfo
```

1440

1441 Several key management techniques can be used in **EnvelopedData**. In XCBF, only key
1442 transport is supported. Other techniques such as constructive key management may be
1443 employed by an application, but such use is not defined in this standard. Type **RecipientInfo** is
1444 restricted to a single choice alternative and defined as

1445

```
1446 RecipientInfo ::= CHOICE {  
1447     ktri KeyTransRecipientInfo  
1448 }
```

1449

1450 Key transport information is provided to the recipient of a digital envelope so that the envelope
1451 can be opened and the protected content encryption key recovered. The content encryption key
1452 may then be used to decrypt the content.

1453

1454 The information needed by the recipient to recover the content encryption key is contained in a
1455 value of type **KeyTransRecipientInfo** defined as

1456

```
1457 KeyTransRecipientInfo ::= SEQUENCE {  
1458     version          CMSVersion,  
1459     rid              RecipientIdentifier,  
1460     keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,  
1461     encryptedKey     EncryptedKey  
1462 }
```

1463

1464 This type is a sequence of four components. The integer **version** number is always set to eighty-
1465 four in XCBF. The **rid** component is used to identify the public key used to encrypt the content
1466 encryption key. This public key is bound to a key encryption algorithm in a public key certificate. It
1467 is associated with the recipient private key needed to decrypt the content encryption key used by
1468 the sender to encrypt the content.

1469

1470 A hash of the public key certificate uniquely identifies the recipient certificate.

1471

```
1472 RecipientIdentifier ::= CHOICE {  
1473     certHash [73] EXPLICIT Hash  
1474 }
```

1475

1476 The **keyEncryptionAlgorithm** component identifies the key encryption algorithm and any
1477 associated parameters used to encrypt the content encryption key.

1478

```
1479 KeyEncryptionAlgorithmIdentifier ::=  
1480     AlgorithmIdentifier {{KeyEncryptionAlgorithms}}  
1481  
1482 KeyEncryptionAlgorithms ALGORITHM ::= {  
1483     { OID rsaEncryption PARMS NoIV },  
1484  
1485     ... -- expect other key encryption algorithms --  
1486 }
```

1487

1488 The encrypted content encryption key is an opaque string, a value of type **EncryptedKey** defined
1489 as

1490

```
1491 EncryptedKey ::= OCTET STRING
```

1492

1493 5.1.3.4.3 Digital Envelope Processing

1494 To create a digital envelope, a content encryption algorithm is selected. The content encryption
1495 algorithm identifier and any associated parameters form the **contentEncryptionAlgorithm** value
1496 of the **encryptedContentInfo** component of type **EnvelopedData**. The recipient uses this value
1497 to recover the encrypted content.

1498

1499 The content encryption key is encrypted using the key encryption algorithm and public key from
1500 the recipient's public key certificate. The key encryption algorithm identifier and any associated
1501 parameters used to encrypt the content encryption key form the **keyEncryptionAlgorithm**
1502 component of type **KeyTransRecipientInfo**.

1503

1504 The result of encrypting the content encryption key forms the **encryptedKey** component of type
1505 **KeyTransRecipientInfo**. A hash of the complete DER encoding of the recipient's public key
1506 certificate is used to populate the **rid** component, and the **version** component is set to the integer
1507 eighty-four.

1508

1509 The content encryption key is used to encrypt a value of type **EncodedBiometricObjects**. This
1510 type is a series of one or more values of type **BiometricObject** in their encoded form. These
1511 values are encoded using the XML Encoding Rules.

1512

1513 To retrieve the encrypted content, the recipient first decrypts the value of the **encryptedKey**
1514 component of type **KeyTransRecipientInfo** to recover the content encryption key using the
1515 private key associated with the public key used to encrypt the content encryption key. This private
1516 key is indicated by the hash of the associated public key certificate in the **rid** component of type
1517 **KeyTransRecipientInfo**. The recovered content encryption key is then used to decrypt the
1518 content to recover a value of type **EncodedBiometricObjects**.

1519 **5.1.4 PrivacyAndIntegrityObjects**

1520 The **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** is a value of type
1521 **PrivacyAndIntegrityObjects**. This type is defined as a sequence of three components, optional
1522 **biometricHeaders**, a **privacyBlock**, and an **integrityBlock**.

1523

```
1524 PrivacyAndIntegrityObjects ::= SEQUENCE {  
1525     biometricHeaders BiometricHeaders OPTIONAL,  
1526     privacyBlock     PrivacyBlock,  
1527     integrityBlock   IntegrityBlock  
1528 }
```

1529

1530 The **biometricHeaders** component is optional and is composed of a series of one or more values
1531 of type **BiometricHeader**. The **privacyBlock** component is a value of type **PrivacyBlock**. The
1532 **BiometricHeader** and **PrivacyBlock** types are described in section 4.1.3. *PrivacyObjects*. The
1533 **integrityBlock** component is a value of type **IntegrityBlock**. This type is described in section
1534 4.1.2. *IntegrityObjects*.

1535

1536 The input to all cryptographic process is a value of type **EncodedBiometricObjects**, a series of
1537 one or more values of type **BiometricObject** in their encoded form based on the XML Encoding
1538 Rules. The order of the components in type **PrivacyAndIntegrityObjects** facilitates one pass
1539 processing for both sender and recipient.

1540

1541 For the sender, a value of type **EncodedBiometricObjects** is created and then used as input to
1542 the cryptographic processing of both the **privacyBlock** and **integrityBlock** components. For the
1543 recipient, the **privacyBlock** component is first processed to recover the encrypted content, a
1544 value of type **EncodedBiometricObjects**. This recovered value is then used to validate the
1545 signature in the **integrityBlock** component.

1546

1547 A value of the **privacyAndIntegrityObjects** choice alternative of type **BiometricSyntax** can be
1548 represented in XML markup as

1549

{Working Draft}-*{XML Common Biometric Format}*-*{XCBF}*-*{01}*

```

1550 <privacyAndIntegrityObjects>
1551   <privacyBlock>
1552     <namedKey>
1553       <keyName>6AE173BF5A973D1E</keyName>
1554       <encryptedData>
1555         <version>84</version>
1556         <encryptedContentInfo>
1557           <contentType>1.2.840.113549.1.7.6</contentType>
1558           <contentEncryptionAlgorithm>
1559             <algorithm>1.2.840.113549.3.7</algorithm>
1560             <parameters>
1561               <IV>7EA13D6E143CB5C9</IV>
1562             </parameters>
1563           </contentEncryptionAlgorithm>
1564           <encryptedContent>
1565             ...
1566           </encryptedContent>
1567         </encryptedContentInfo>
1568       </encryptedData>
1569     </namedKey>
1570   </privacyBlock>
1571   <integrityBlock>
1572     <digitalSignature>
1573       <algorithmID>
1574         <algorithm>1.2.840.10040.4.3</algorithm>
1575         <parameters><NoIV></parameters>
1576       </algorithmID>
1577       <signature>
1578         ...
1579       </signature>
1580     </digitalSignature>
1581   </integrityBlock>
1582 </privacyAndIntegrityObjects>

```

1583

1584 This markup combines a privacy block and integrity block. The named key choice alternative of
 1585 the privacy block and the digital signature choice alternative of the integrity block are used. The
 1586 optional biometric headers are not present.

1587

1588 The named key alternative is a sequence containing a key name and a value of version number
 1589 eighty-four of the cryptographic message type **EncryptedData**. The encrypted content type is
 1590 identified as ordinary data, and is computed on a value of type **EncodedBiometricObjects**. The
 1591 Triple DES content encryption algorithm is identified along with its associated parameters, an
 1592 initialization vector, <IV>. An ellipsis is used as a placeholder and the encrypted content is not
 1593 shown.

1594

1595 The digital signature choice alternative is a simple digital signature on a value of type
 1596 **EncodedBiometricObjects**. The Digital Signature Algorithm (DSA) with Secure Hash Algorithm
 1597 (SHA1) and its associated parameters, an initialization vector, <IV> are used for signing a value

1598 of **EncodedBiometricObjects**. An ellipsis is used as a placeholder and the signature is not
1599 shown.

6 References

1600

6.1 Normative

1601

1602

1603 1 ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open Systems
1604 Interconnection -- The Directory: Authentication framework”, International Organization
1605 for Standardization, Geneva, Switzerland, 2000.

1606

- 1607 [1] ISO/IEC 9594-8: Information technology | ITU-T Recommendation X.509, Open
1608 Systems Interconnection -- The Directory: Authentication framework.
- 1609 [2] ISO/IEC 8824-1:2001 | ITU-T Recommendation X.680 (2002), Information Technology
1610 - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation.
- 1611 [3] ISO/IEC 8824-2:2001 | ITU-T Recommendation X.681 (2002), Information Technology
1612 - Abstract Syntax Notation One (ASN.1): Information Object Specification.
- 1613 [4] ISO/IEC 8824-3:2001 | ITU-T Recommendation X.682 (2002), Information Technology
1614 - Abstract Syntax Notation One (ASN.1): Constraint Specification.
- 1615 [5] ISO/IEC 8824-4:2001 | ITU-T Recommendation X.683 (2002), Information Technology
1616 - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications.
- 1617 [6] ISO/IEC 8825-1:2001 | ITU-T Recommendation X.690 (2002), Information Technology
1618 - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical
1619 Encoding Rules (CER) and Distinguished Encoding Rules (DER).
- 1620 [7] ISO/IEC 8825-4:2001 | X.693 ITU-T Recommendation X.693 (2002) |, Information
1621 Technology - ASN.1 Encoding Rules: XML Encoding Rules (XER).
- 1622 [8] ANS X9.30-1997 Public Key Cryptography Using Irreversible Algorithms for the
1623 Financial Services Industry, Part 1: The Digital Signature Algorithm (DSA).
- 1624 [9] ANS X9.30-1997 Public Key Cryptography Using Irreversible Algorithms for the
1625 Financial Services Industry, Part 2: The Secure Hash Algorithm (SHA).
- 1626 [10] ANS X9.52-1999 Triple Data Encryption Algorithm Modes of Operation.
- 1627 [11] ANS X9.68:2001 Digital Certificates for Mobile/Wireless and High Transaction Volume
1628 Financial Services and Security For The Financial Systems: Part 2: Domain Certificate
1629 Syntax.
- 1630 [12] ANS X9.71-1999 Keyed Hash Message Authentication Code (HMAC).
- 1631 [13] ANS X9.73:2002 Cryptographic Message Syntax (CMS).
- 1632 [14] ANS X9.84:2002 Biometrics Information Management and Security For The Financial
1633 Services Industry.
- 1634 [15] ANS X9.96:2002 (draft) XML Cryptographic Message Syntax (XCMS).
- 1635 [16] ANSI/INCITS 358-2002 - Information technology - BioAPI Specification.
- 1636 [17] CBEFF Common Biometric Exchange File Format, NISTIR-6529, January 3, 2001.

1637
1638

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

1639 7 XCBF Schema

1640 The following ASN.1 modules provide the schema for all of the XML markup defined in this
1641 standard.

1642

1643 7.1 X9-84-Biometrics Module

1644

```
1645 X9-84-Biometrics {
1646     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
1647         x9Standards(9) x9-84(84) module(0) biometrics(1) rev(1) }
1648     DEFINITIONS AUTOMATIC TAGS ::= BEGIN
1649
1650     -- EXPORTS All;
1651
1652     IMPORTS
1653
1654         -- X9.84 Biometrics Information Management and Security IDs --
1655
1656         BiometricTypes, CBEFF-Formats, IBIA-Formats, MatchingAIDs,
1657         ProcessingAIDs, X9-Formats
1658         FROM X9-84-Identifiers {
1659             iso(1) identified-organization(3) tc68(133) country(16)
1660                 x9(840) x9Standards(9) x9-84(84) module(0)
1661                 ids(3) rev(1) }
1662
1663         -- X9.84 Biometrics Information Management and Security CMS --
1664
1665         AuthenticatedData, EncryptedData, EnvelopedData,
1666         MACAlgorithmIdentifier, SignatureAlgorithmIdentifier,
1667         SignedData
1668         FROM X9-84-CMS {
1669             iso(1) identified-organization(3) tc68(133) country(16)
1670                 x9(840) x9Standards(9) x9-84(84) module(0)
1671                 cms(2) rev(1) } ;
1672
1673
1674
1675     BiometricSyntaxSets ::= SEQUENCE SIZE(1..MAX) OF BiometricSyntax
1676
1677     BiometricSyntax ::= CHOICE {
1678         biometricObjects          BiometricObjects,
1679         integrityObjects          IntegrityObjects,
1680         privacyObjects           PrivacyObjects,
1681         privacyAndIntegrityObjects PrivacyAndIntegrityObjects
1682     }
1683
```

```

1684 BiometricObjects ::= SEQUENCE SIZE(1..MAX) OF BiometricObject
1685
1686 BiometricObject ::= SEQUENCE {
1687     biometricHeader  BiometricHeader,
1688     biometricData    BiometricData
1689 }
1690
1691 --
1692 -- All of the cryptographic processing in this standard is performed
1693 -- on a value of type EncodedBiometricObjects. This is a sequence of
1694 -- one or more values of type BiometricObject in its encoded form.
1695 --
1696
1697 EncodedBiometricObjects ::= BIOMETRIC.&Type( BiometricObjects )
1698
1699 BiometricHeader ::= SEQUENCE {
1700     version          BiometricVersion  DEFAULT hv1,
1701     recordType       RecordType  OPTIONAL,
1702     dataType         DataType  OPTIONAL,
1703     purpose          Purpose  OPTIONAL,
1704     quality          Quality  OPTIONAL,
1705     validityPeriod   ValidityPeriod  OPTIONAL,
1706     format           Format  OPTIONAL
1707 }
1708
1709 BiometricVersion ::= INTEGER { hv1(0) } (0..MAX)
1710
1711 RecordType ::= BIOMETRIC.&name({BiometricTypes})
1712
1713 DataType ::= ENUMERATED {
1714     raw              (0),
1715     intermediate    (1),
1716     processed        (2)
1717 }
1718
1719 Purpose ::= ENUMERATED {
1720     verify           (1),
1721     identify         (2),
1722     enroll           (3),
1723     enrollVerify     (4),
1724     enrollIdentity   (5),
1725     audit            (6),
1726
1727     ... -- Expect other values --
1728 }
1729
1730 Quality ::= INTEGER {
1731     lowest           ( 0),
1732     highest          (100),
1733     notSet           ( -1),
1734     notSupported     ( -2)
1735 } (-2..100,...)
1736

```

```

1737 ValidityPeriod ::= SEQUENCE {
1738     notBefore  DateTime OPTIONAL,
1739     notAfter   DateTime  OPTIONAL
1740 }
1741 (ALL EXCEPT({ -- none; at least one component is present -- })))
1742
1743 DateTime ::= RELATIVE-OID -- { yyyy mm dd hh mm ss z } --
1744
1745 Format ::= SEQUENCE {
1746     formatOwner  BIOMETRIC.&name({Owner}),
1747     formatType   BIOMETRIC.&Type({Owner}{@formatOwner})  OPTIONAL
1748 }
1749
1750 Owner BIOMETRIC ::= {
1751     CBEFF-Formats | -- http://www.nist.gov --
1752     IBIA-Formats  | -- http://www.ibia.org --
1753     X9-Formats,   -- http://www.x9.org --
1754
1755     ... -- expect additional vendor specific formats --
1756 }
1757
1758 -- Integrity --
1759
1760 IntegrityObjects ::= SEQUENCE {
1761     biometricObjects  EncodedBiometricObjects,
1762     integrityBlock    IntegrityBlock
1763 }
1764
1765 IntegrityBlock ::= CHOICE {
1766     digitalSignature      DigitalSignature,
1767     messageAuthenticationCode  MessageAuthenticationCode,
1768     signedData            SignedData,
1769     authenticatedData    AuthenticatedData
1770 }
1771
1772 DigitalSignature ::= SIGNATURE { EncodedBiometricObjects }
1773
1774 MessageAuthenticationCode ::= MAC { EncodedBiometricObjects }
1775
1776 -- Privacy --
1777
1778 PrivacyObjects ::= SEQUENCE {
1779     biometricHeaders  BiometricHeaders  OPTIONAL,
1780     privacyBlock      PrivacyBlock
1781 }
1782
1783 BiometricHeaders ::= SEQUENCE SIZE(1..MAX) OF BiometricHeader
1784
1785 PrivacyBlock ::= CHOICE {
1786     fixedKey          EncryptedData,
1787     namedKey          NamedKeyEncryptedData,
1788     establishedKey   EnvelopedData
1789 }

```

```

1790
1791 NamedKeyEncryptedData ::= SEQUENCE {
1792     keyName          OCTET STRING (SIZE(1..MAX)),
1793     encryptedData    EncryptedData
1794 }
1795
1796 -- Privacy and integrity --
1797
1798 PrivacyAndIntegrityObjects ::= SEQUENCE {
1799     biometricHeaders BiometricHeaders OPTIONAL,
1800     privacyBlock     PrivacyBlock,
1801     integrityBlock   IntegrityBlock
1802 }
1803
1804 -- Authentication Information (AI) --
1805
1806 BiometricInformationSets ::=
1807     SEQUENCE SIZE(1..MAX) OF BiometricInformation
1808
1809 BiometricInformation ::= SEQUENCE {
1810     processingAlgorithms ProcessingAlgorithms OPTIONAL,
1811     matchingMethods      MatchingMethods OPTIONAL
1812 }
1813 (ALL EXCEPT({ -- none; at least one component is present -- }))
1814
1815 -- Biometric processing algorithms --
1816
1817 ProcessingAlgorithms ::= SEQUENCE SIZE(1..MAX) OF ProcessingInformation
1818
1819 ProcessingInformation ::= SEQUENCE {
1820     id      BIOMETRIC.&name({ProcessingAIDs}),
1821     parms  BIOMETRIC.&Type({ProcessingAIDs}{@id}) OPTIONAL
1822 }
1823
1824 -- Biometric matching methods --
1825
1826 MatchingMethods ::= SEQUENCE SIZE(1..MAX) OF MatchingInformation
1827
1828 MatchingInformation ::= SEQUENCE {
1829     id      BIOMETRIC.&name({MatchingAIDs}),
1830     parms  BIOMETRIC.&Type({MatchingAIDs}{@id}) OPTIONAL
1831 }
1832
1833 BiometricData ::= OCTET STRING(SIZE(1..MAX))
1834
1835 -- Biometrics information object class --
1836
1837 BIOMETRIC ::= CLASS {
1838     &name BIOMETRIC-IDENTIFIER UNIQUE,
1839     &Type OPTIONAL
1840 }
1841 WITH SYNTAX { BIOMETRIC &name [ DATA &Type ] }
1842

```

```

1843 BIOMETRIC-IDENTIFIER ::= CHOICE {
1844     oid OBJECT IDENTIFIER, -- complete object identifier
1845     id  RELATIVE-OID      -- object identifier fragment
1846 }
1847
1848 -- Parameterized types --
1849
1850 SIGNATURE { ToBeSigned } ::= SEQUENCE {
1851     algorithmID SignatureAlgorithmIdentifier,
1852     signature    BIT STRING( CONSTRAINED BY {
1853         -- signature on a value of -- ToBeSigned })
1854 }
1855
1856 MAC { ToBeMACed } ::= SEQUENCE {
1857     keyName      OCTET STRING OPTIONAL,
1858     algorithmID  MACAlgorithmIdentifier,
1859     mac          OCTET STRING (CONSTRAINED BY {
1860         -- MAC or HMAC on a value of -- ToBeMACed })
1861 }
1862
1863 -- Biometric certificate extension --
1864
1865 --
1866 -- A biometricTemplates information object can be used to extend the
1867 -- information bound to a public key in an value of types Certificate
1868 -- or AttributeCertificate as defined in The Directory series of
1869 -- standards, to include biometric identity information.
1870 --
1871
1872 biometricTemplates EXTENSION ::= {
1873     SYNTAX          EncodedBiometricObjects -- DER or cXER --
1874     IDENTIFIED BY  x509-biometricTemplates
1875 }
1876
1877 EXTENSION ::= CLASS {
1878     &id            OBJECT IDENTIFIER UNIQUE,
1879     &ExtnType
1880 }
1881 WITH SYNTAX { SYNTAX &ExtnType IDENTIFIED BY &id }
1882
1883 --
1884 -- A domainBiometricTemplates information object can be used to
1885 -- extend the information bound to a public key in an value of
1886 -- ASN.1 type DomainCertificate as defined in the X9.68 Domain
1887 -- Certificate Syntax standard, to include biometric identity
1888 -- information.
1889 --
1890
1891 domainBiometricTemplates PRIVATE ::= {
1892     NAME oid : x968-biometricTemplates
1893     TYPE EncodedBiometricObjects -- DER or cXER --
1894 }
1895

```



```

1896 PRIVATE ::= CLASS {
1897     &name Identifier UNIQUE,
1898     &Type OPTIONAL
1899 }
1900 WITH SYNTAX { NAME &name [TYPE &Type] }
1901
1902 Identifier ::= CHOICE {
1903     oid OBJECT IDENTIFIER, -- complete object identifier
1904     id RELATIVE-OID -- object identifier fragment
1905 }
1906
1907 END -- X9-84-Biometrics --

```

1908

1909 7.2 X9-84-CMS Module

1910

```

1911 X9-84-CMS {
1912     iso(1) identified-organization(3) tc68(133) country(16) x9(840)
1913     x9Standards(9) x9-84(84) module(0) cms(2) rev(1) }
1914 DEFINITIONS IMPLICIT TAGS ::= BEGIN
1915
1916 -- EXPORTS All;
1917
1918 IMPORTS
1919
1920     -- ANS X9.84 Biometric Information Management & Security IDs --
1921
1922     des-ede3-cbc, dsa-with-sha1, ecdsa-with-SHA1, hmac-with-SHA1,
1923     id-data, NoIV, rsaEncryption, SHA-Algorithms, sha1WithRSAEncryption
1924     FROM X9-84-Identifiers {
1925         iso(1) identified-organization(3) tc68(133) country(16)
1926         x9(840) x9Standards(9) x9-84(84) module(0)
1927         ids(3) rev(1) };
1928
1929 SignedData ::= SEQUENCE {
1930     version CMSVersion,
1931     digestAlgorithms DigestAlgorithmIdentifiers,
1932     encapContentInfo EncapsulatedContentInfo,
1933     certificates [0] CertificateSet OPTIONAL,
1934     crls [1] CertificateRevocationLists OPTIONAL,
1935     signerInfos SignerInfos
1936 }
1937
1938 CMSVersion ::= INTEGER { v84(84) } (v84,...)
1939
1940 DigestAlgorithmIdentifiers ::=
1941     SET SIZE(1) OF DigestAlgorithmIdentifier
1942
1943 DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{DigestAlgorithms}}
1944

```

```

1945 DigestAlgorithms ALGORITHM ::= {
1946   SHA-Algorithms,
1947
1948   ... -- Expect other digest algorithms --
1949 }
1950
1951 EncapsulatedContentInfo ::= SEQUENCE {
1952   eContentType ContentType,
1953   eContent      [0] EXPLICIT OCTET STRING OPTIONAL
1954 }
1955
1956 ContentType ::= CONTENTS.&id({Contents})
1957
1958 CONTENTS ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1998, Annex A
1959
1960 Contents CONTENTS ::= {
1961   { Data IDENTIFIED BY id-data }
1962 }
1963
1964 Data ::= OCTET STRING
1965
1966 CertificateSet ::= OCTET STRING
1967
1968 CertificateRevocationLists ::= OCTET STRING
1969
1970 SignerInfos ::= SET SIZE(1) OF SignerInfo
1971
1972 SignerInfo ::= SEQUENCE {
1973   version          CMSVersion,
1974   sid              SignerIdentifier,
1975   digestAlgorithm  DigestAlgorithmIdentifier,
1976   signatureAlgorithm SignatureAlgorithmIdentifier,
1977   signature        SignatureValue
1978 }
1979
1980 SignerIdentifier ::= CHOICE {
1981   certHash [1] EXPLICIT Hash
1982 }
1983
1984 Hash ::= CHOICE {
1985   ietf      CertHash, -- SHA-1 hash of entire certificate
1986   withAlgID DigestInfo
1987 }
1988
1989 CertHash ::= OCTET STRING (ENCODED BY sha-1)
1990
1991 DigestInfo ::= SEQUENCE {
1992   hashAlgorithm DigestAlgorithmIdentifier,
1993   digest        OCTET STRING
1994 }
1995
1996 SignatureAlgorithmIdentifier ::=
1997   AlgorithmIdentifier {{SignatureAlgorithms}}

```

```

1998
1999 SignatureAlgorithms ALGORITHM ::= {
2000   { OID dsa-with-sha1          PARMS NoIV } |
2001   { OID ecdsa-with-SHA1      PARMS NoIV } |
2002   { OID sha1WithRSAEncryption PARMS NoIV },
2003
2004   ... -- Expect other signature algorithms --
2005 }
2006
2007 SignatureValue ::= OCTET STRING
2008
2009 EncryptedData ::= SEQUENCE {
2010   version          CMSVersion,
2011   encryptedContentInfo EncryptedContentInfo
2012 }
2013
2014 EncryptedContentInfo ::= SEQUENCE {
2015   contentType          ContentType,
2016   contentEncryptionAlgorithm ContentEncryptAlgorithmIdentifier,
2017   encryptedContent     [0] EncryptedContent
2018 }
2019
2020 ContentEncryptAlgorithmIdentifier ::=
2021   AlgorithmIdentifier {{ContentEncryptionAlgorithms}}
2022
2023 ContentEncryptionAlgorithms ALGORITHM ::= {
2024   { OID des-ede3-cbc PARMS IV },
2025
2026   ... -- Expect other content encryption algorithms --
2027 }
2028
2029 IV ::= OCTET STRING (SIZE(8))
2030
2031 EncryptedContent ::= OCTET STRING
2032
2033 EnvelopedData ::= SEQUENCE {
2034   version          CMSVersion,
2035   originatorInfo   [0] OriginatorInfo OPTIONAL,
2036   recipientInfos   RecipientInfos,
2037   encryptedContentInfo EncryptedContentInfo
2038 }
2039
2040 OriginatorInfo ::= SEQUENCE {
2041   certs [0] CertificateSet OPTIONAL,
2042   crls  [1] CertificateRevocationLists OPTIONAL
2043 }
2044 (ALL EXCEPT({ -- none; at least one component is present -- }))
2045
2046 RecipientInfos ::= SET SIZE(1) OF RecipientInfo
2047
2048 RecipientInfo ::= CHOICE {
2049   ktri KeyTransRecipientInfo
2050 }

```

```

2051
2052 KeyTransRecipientInfo ::= SEQUENCE {
2053     version          CMSVersion,
2054     rid              RecipientIdentifier,
2055     keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
2056     encryptedKey     EncryptedKey
2057 }
2058
2059 RecipientIdentifier ::= CHOICE {
2060     certHash  [73] EXPLICIT Hash
2061 }
2062
2063 KeyEncryptionAlgorithmIdentifier ::=
2064     AlgorithmIdentifier {{KeyEncryptionAlgorithms}}
2065
2066 KeyEncryptionAlgorithms ALGORITHM ::= {
2067     { OID rsaEncryption PARMS NoIV },
2068
2069     ... -- expect other key encryption algorithms --
2070 }
2071
2072 EncryptedKey ::= OCTET STRING
2073
2074 AuthenticatedData ::= SEQUENCE {
2075     version          CMSVersion,
2076     recipientInfos  RecipientInfos,
2077     macAlgorithm    MACAlgorithmIdentifier,
2078     encapContentInfo EncapsulatedContentInfo,
2079     mac             MessageAuthenticationCode
2080 }
2081
2082 MACAlgorithmIdentifier ::= AlgorithmIdentifier {{MACAlgorithms}}
2083
2084 MACAlgorithms ALGORITHM ::= {
2085     { OID hmac-with-SHA1 },
2086
2087     ... -- expect other MAC or HMAC algorithms --
2088 }
2089
2090 MessageAuthenticationCode ::= OCTET STRING
2091
2092
2093 -- Cryptographic algorithm identification --
2094
2095 ALGORITHM ::= CLASS {
2096     &id    OBJECT IDENTIFIER UNIQUE,
2097     &Type  OPTIONAL
2098 }
2099     WITH SYNTAX { OID &id [PARMS &Type] }
2100
2101 AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
2102     algorithm  ALGORITHM.&id( {IOSet} ),
2103     parameters ALGORITHM.&Type( {IOSet}{@algorithm} ) OPTIONAL

```

2104
2105
2106
2107

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

2108

2109 7.3 X9-84-Identifiers Module

2110

```
2111 X9-84-Identifiers {  
2112     iso(1) identified-organization(3) tc68(133) country(16) x9(840)  
2113     x9Standards(9) x9-84(84) module(0) ids(3) rev(1) }  
2114     DEFINITIONS AUTOMATIC TAGS ::= BEGIN  
2115  
2116     -- EXPORTS All;  
2117  
2118     IMPORTS  
2119  
2120     -- X9.84 Biometrics Information Management and Security --  
2121  
2122     BIOMETRIC, BiometricInformationSets  
2123     FROM X9-84-Biometrics {  
2124         iso(1) identified-organization(3) tc68(133) country(16)  
2125         x9(840) x9Standards(9) x9-84(84) module(0)  
2126         biometrics(1) rev(1) }  
2127  
2128     -- X9.84 Biometrics Information Management and Security CMS --  
2129  
2130     ALGORITHM  
2131     FROM X9-84-CMS {  
2132         iso(1) identified-organization(3) tc68(133) country(16)  
2133         x9(840) x9Standards(9) x9-84(84) module(0)  
2134         cms(2) rev(1) };  
2135  
2136  
2137     OID ::= OBJECT IDENTIFIER -- Alias  
2138  
2139     ReOID ::= RELATIVE-OID -- Alias  
2140  
2141     -- x9-84 { 1 3 133 16 840 9 84 }  
2142     -- x9-84-Module { 1 3 133 16 840 9 84 0 }  
2143     -- x9-84-Biometrics { 1 3 133 16 840 9 84 0 1 }  
2144     -- x9-84-CMS { 1 3 133 16 840 9 84 0 2 }  
2145     -- x9-84-Identifiers { 1 3 133 16 840 9 84 0 3 }  
2146     -- biometric { 1 3 133 16 840 9 84 1 }  
2147     -- id-unknown-Type { 1 3 133 16 840 9 84 1 0 }  
2148     -- id-body-Odor { 1 3 133 16 840 9 84 1 1 }  
2149     -- id-dna { 1 3 133 16 840 9 84 1 2 }  
2150     -- id-ear-Shape { 1 3 133 16 840 9 84 1 3 }  
2151     -- id-facial-Features { 1 3 133 16 840 9 84 1 4 }  
2152     -- id-finger-Image { 1 3 133 16 840 9 84 1 5 }
```

```

2153 -- id-finger-Geometry { 1 3 133 16 840 9 84 1 6 }
2154 -- id-hand-Geometry { 1 3 133 16 840 9 84 1 7 }
2155 -- id-iris-Features { 1 3 133 16 840 9 84 1 8 }
2156 -- id-keystroke-Dynamics { 1 3 133 16 840 9 84 1 9 }
2157 -- id-palm { 1 3 133 16 840 9 84 1 10 }
2158 -- id-retina { 1 3 133 16 840 9 84 1 11 }
2159 -- id-signature { 1 3 133 16 840 9 84 1 12 }
2160 -- id-speech-Pattern { 1 3 133 16 840 9 84 1 13 }
2161 -- id-thermal-Image { 1 3 133 16 840 9 84 1 14 }
2162 -- id-vein-Pattern { 1 3 133 16 840 9 84 1 15 }
2163 -- id-thermal-Face-Image { 1 3 133 16 840 9 84 1 16 }
2164 -- id-thermal-Hand-Image { 1 3 133 16 840 9 84 1 17 }
2165 -- id-lip-Movement { 1 3 133 16 840 9 84 1 18 }
2166 -- id-gait { 1 3 133 16 840 9 84 1 19 }
2167 -- processing-algorithm { 1 3 133 16 840 9 84 2 }
2168 -- matching-method { 1 3 133 16 840 9 84 3 }
2169 -- format-Owner { 1 3 133 16 840 9 84 4 }
2170 -- cbeff-Owner { 1 3 133 16 840 9 84 4 0 }
2171 -- ibia-Owner { 1 3 133 16 840 9 84 4 1 }
2172 -- id-ibia-SAFLINK { 1 3 133 16 840 9 84 4 1 1 }
2173 -- id-ibia-Bioscrypt { 1 3 133 16 840 9 84 4 1 2 }
2174 -- id-ibia-Visionics { 1 3 133 16 840 9 84 4 1 3 }
2175 -- id-ibia-InfineonTechnologiesAG { 1 3 133 16 840 9 84 4 1 4 }
2176 -- id-ibia-IridianTechnologies { 1 3 133 16 840 9 84 4 1 5 }
2177 -- id-ibia-Veridicom { 1 3 133 16 840 9 84 4 1 6 }
2178 -- id-ibia-CyberSIGN { 1 3 133 16 840 9 84 4 1 7 }
2179 -- id-ibia-eCryp { 1 3 133 16 840 9 84 4 1 8 }
2180 -- id-ibia-FingerprintCardsAB { 1 3 133 16 840 9 84 4 1 9 }
2181 -- id-ibia-SecuGen { 1 3 133 16 840 9 84 4 1 10 }
2182 -- id-ibia-PreciseBiometric { 1 3 133 16 840 9 84 4 1 11 }
2183 -- id-ibia-Identix { 1 3 133 16 840 9 84 4 1 12 }
2184 -- id-ibia-DERMALOG { 1 3 133 16 840 9 84 4 1 13 }
2185 -- id-ibia-LOGICO { 1 3 133 16 840 9 84 4 1 14 }
2186 -- id-ibia-NIST { 1 3 133 16 840 9 84 4 1 15 }
2187 -- id-ibia-A3Vision { 1 3 133 16 840 9 84 4 1 16 }
2188 -- id-ibia-NEC { 1 3 133 16 840 9 84 4 1 17 }
2189 -- id-ibia-STMicroelectronics { 1 3 133 16 840 9 84 4 1 18 }
2190 -- x9-Owner { 1 3 133 16 840 9 84 4 2 }
2191 -- certificate-Extensions { 1 3 133 16 840 9 84 5 }
2192 -- x968-biometricTemplates { 1 3 133 16 840 9 84 5 0 }
2193 -- x509-biometricTemplates { 1 3 133 16 840 9 84 5 1 }
2194
2195 -- X9.84 arc; base object identifier --
2196
2197 x9-84 OID ::= {
2198     iso(1) identified-organization(3) tc68(133) country(16)
2199     x9(840) x9Standards(9) x9-84(84)
2200 }
2201
2202 -- X9.84 ASN.1 modules --
2203
2204 x9-84-Module OID ::= { x9-84 modules(0) }
2205

```

```

2206 x9-84-Biometrics   OID ::= { x9-84-Module biometrics(1) rev(1) }
2207
2208 x9-84-CMS          OID ::= { x9-84-Module cms(2) rev(1) }
2209
2210 x9-84-Identifiers OID ::= { x9-84-Module ids(3) rev(1) }
2211
2212 -- X9.84 biometric technologies --
2213
2214 biometric OID ::= { x9-84 biometrics(1) }
2215
2216 id-unknown-Type   OID ::= { biometric unknownType(0) }
2217 id-body-Odor      OID ::= { biometric bodyOdor(1) }
2218 id-dna            OID ::= { biometric dna(2) }
2219 id-ear-Shape      OID ::= { biometric ear-Shape(3) }
2220 id-facial-Features OID ::= { biometric facialFeatures(4) }
2221 id-finger-Image   OID ::= { biometric fingerImage(5) }
2222 id-finger-Geometry OID ::= { biometric fingerGeometry(6) }
2223 id-hand-Geometry  OID ::= { biometric handGeometry(7) }
2224 id-iris-Features  OID ::= { biometric irisFeatures(8) }
2225 id-keystroke-Dynamics OID ::= { biometric keystrokeDynamics(9) }
2226 id-palm           OID ::= { biometric palm(10) }
2227 id-retina         OID ::= { biometric retina(11) }
2228 id-signature      OID ::= { biometric signature(12) }
2229 id-speech-Pattern OID ::= { biometric speech-Pattern(13) }
2230 id-thermal-Image  OID ::= { biometric thermalImage(14) }
2231 id-vein-Pattern   OID ::= { biometric veinPattern(15) }
2232 id-thermal-Face-Image OID ::= { biometric thermalFaceImage(16) }
2233 id-thermal-Hand-Image OID ::= { biometric thermalHandImage(17) }
2234 id-lip-Movement   OID ::= { biometric lipMovement(18) }
2235 id-gait           OID ::= { biometric gait(19) }
2236
2237 -- X9.84 biometric technology object identifier fragments --
2238
2239 unknown-Type      RelOID ::= { unknownType(0) }
2240 body-Odor         RelOID ::= { bodyOdor(1) }
2241 dna              RelOID ::= { dna(2) }
2242 ear-Shape         RelOID ::= { earShape(3) }
2243 facial-Features   RelOID ::= { facialFeatures(4) }
2244 finger-Image      RelOID ::= { fingerImage(5) }
2245 finger-Geometry   RelOID ::= { fingerGeometry(6) }
2246 hand-Geometry     RelOID ::= { handGeometry(7) }
2247 iris-Features     RelOID ::= { irisFeatures(8) }
2248 keystroke-Dynamics RelOID ::= { keystrokeDynamics(9) }
2249 palm             RelOID ::= { palm(10) }
2250 retina           RelOID ::= { retina(11) }
2251 signature         RelOID ::= { signature(12) }
2252 speech-Pattern    RelOID ::= { speechPattern(13) }
2253 thermal-Image     RelOID ::= { thermalImage(14) }
2254 vein-Pattern      RelOID ::= { veinPattern(15) }
2255 thermal-Face-Image RelOID ::= { thermalFaceImage(16) }
2256 thermal-Hand-Image RelOID ::= { thermalHandImage(17) }
2257 lip-Movement      RelOID ::= { lipMovement(18) }
2258 gait             RelOID ::= { gait(19) }

```

```

2259
2260 BiometricTypes BIOMETRIC ::= {
2261   { BIOMETRIC id : unknown-Type           } |
2262   { BIOMETRIC id : body-Odor              } |
2263   { BIOMETRIC id : dna                    } |
2264   { BIOMETRIC id : ear-Shape              } |
2265   { BIOMETRIC id : facial-Features        } |
2266   { BIOMETRIC id : finger-Image          } |
2267   { BIOMETRIC id : finger-Geometry       } |
2268   { BIOMETRIC id : hand-Geometry         } |
2269   { BIOMETRIC id : iris-Features          } |
2270   { BIOMETRIC id : keystroke-Dynamics    } |
2271   { BIOMETRIC id : palm                  } |
2272   { BIOMETRIC id : retina                 } |
2273   { BIOMETRIC id : signature              } |
2274   { BIOMETRIC id : speech-Pattern        } |
2275   { BIOMETRIC id : thermal-Image         } |
2276   { BIOMETRIC id : vein-Pattern          } |
2277   { BIOMETRIC id : thermal-Face-Image    } |
2278   { BIOMETRIC id : thermal-Hand-Image    } |
2279   { BIOMETRIC id : lip-Movement          } |
2280   { BIOMETRIC id : gait                   } |
2281
2282   ... -- expect additional biometric types --
2283 }
2284
2285
2286 -- X9.84 biometric processing algorithms --
2287
2288 processing-algorithm OID ::= { x9-84 processingAlgorithms(2) }
2289
2290 -- X9.84 biometric matching methods --
2291
2292 matching-method OID ::= { x9-84 matchingMethods(3) }
2293
2294 -- X9.84 vendor specific formats --
2295
2296 format-Owner OID ::= { x9-84 format-owners(4) }
2297
2298 cbeff-Owner OID ::= { format-Owner cbeff(0) }
2299
2300 ibia-Owner  OID ::= { format-Owner ibia(1) }
2301
2302 x9-Owner    OID ::= { format-Owner x9(2) }
2303
2304 -- IBIA vendor specific formats registered at http://www.ibia.org
2305
2306 id-ibia-SAFLINK          OID ::= { ibia-Owner 1 }
2307 id-ibia-Bioscrypt       OID ::= { ibia-Owner 2 }
2308 id-ibia-Visionics       OID ::= { ibia-Owner 3 }
2309 id-ibia-InfineonTechnologiesAG  OID ::= { ibia-Owner 4 }
2310 id-ibia-IridianTechnologies  OID ::= { ibia-Owner 5 }
2311 id-ibia-Veridicom       OID ::= { ibia-Owner 6 }

```



```

2312 id-ibia-CyberSIGN          OID ::= { ibia-Owner 7 }
2313 id-ibia-eCryp            OID ::= { ibia-Owner 8 }
2314 id-ibia-FingerprintCardsAB  OID ::= { ibia-Owner 9 }
2315 id-ibia-SecuGen          OID ::= { ibia-Owner 10 }
2316 id-ibia-PreciseBiometric  OID ::= { ibia-Owner 11 }
2317 id-ibia-Identix          OID ::= { ibia-Owner 12 }
2318 id-ibia-DERMALOG        OID ::= { ibia-Owner 13 }
2319 id-ibia-LOGICO           OID ::= { ibia-Owner 14 }
2320 id-ibia-NIST             OID ::= { ibia-Owner 15 }
2321 id-ibia-A3Vision         OID ::= { ibia-Owner 16 }
2322 id-ibia-NEC              OID ::= { ibia-Owner 17 }
2323 id-ibia-STMicroelectronics  OID ::= { ibia-Owner 18 }
2324
2325
2326 -- When represented as values of type OBJECT IDENTIFIER, these
2327 -- IBIA vendor specific formats may be associated with any ASN.1
2328 -- type.
2329
2330 IBIAoidFormats BIOMETRIC ::= {
2331   { BIOMETRIC oid : id-ibia-SAFLINK          DATA Any } |
2332   { BIOMETRIC oid : id-ibia-Bioscrypt        DATA Any } |
2333   { BIOMETRIC oid : id-ibia-Visionics        DATA Any } |
2334   { BIOMETRIC oid : id-ibia-InfineonTechnologiesAG  DATA Any } |
2335   { BIOMETRIC oid : id-ibia-IridianTechnologies  DATA Any } |
2336   { BIOMETRIC oid : id-ibia-Veridicom        DATA Any } |
2337   { BIOMETRIC oid : id-ibia-CyberSIGN        DATA Any } |
2338   { BIOMETRIC oid : id-ibia-eCryp            DATA Any } |
2339   { BIOMETRIC oid : id-ibia-FingerprintCardsAB  DATA Any } |
2340   { BIOMETRIC oid : id-ibia-SecuGen          DATA Any } |
2341   { BIOMETRIC oid : id-ibia-PreciseBiometric  DATA Any } |
2342   { BIOMETRIC oid : id-ibia-Identix          DATA Any } |
2343   { BIOMETRIC oid : id-ibia-DERMALOG        DATA Any } |
2344   { BIOMETRIC oid : id-ibia-LOGICO           DATA Any } |
2345   { BIOMETRIC oid : id-ibia-NIST             DATA Any } |
2346   { BIOMETRIC oid : id-ibia-A3Vision         DATA Any } |
2347   { BIOMETRIC oid : id-ibia-NEC              DATA Any } |
2348   { BIOMETRIC oid : id-ibia-STMicroelectronics  DATA Any },
2349
2350   ... -- Expect additional vendor specific formats --
2351 }
2352
2353 Any ::= TYPE-IDENTIFIER.&Type -- Application constrained
2354
2355 -- Relative object identifier representations of the identical
2356 -- IBIA vendor specific formats defined as OBJECT IDENTIFIER
2357 -- values above are used to identify these formats when they must
2358 -- comply with the fixed format requirements of the BioAPI 1.1
2359 -- specification and are associated with a two byte integer value.
2360
2361 ibia-SAFLINK              RelOID ::= { 1 }
2362 ibia-Bioscrypt            RelOID ::= { 2 }
2363 ibia-Visionics            RelOID ::= { 3 }
2364 ibia-InfineonTechnologiesAG  RelOID ::= { 4 }

```

```

2365 ibia-IridianTechnologies RelOID ::= { 5 }
2366 ibia-Veridicom RelOID ::= { 6 }
2367 ibia-CyberSIGN RelOID ::= { 7 }
2368 ibia-eCryp RelOID ::= { 8 }
2369 ibia-FingerprintCardsAB RelOID ::= { 9 }
2370 ibia-SecuGen RelOID ::= { 10 }
2371 ibia-PreciseBiometric RelOID ::= { 11 }
2372 ibia-Identix RelOID ::= { 12 }
2373 ibia-DERMALOG RelOID ::= { 13 }
2374 ibia-LOGICO RelOID ::= { 14 }
2375 ibia-NIST RelOID ::= { 15 }
2376 ibia-A3Vision RelOID ::= { 16 }
2377 ibia-NEC RelOID ::= { 17 }
2378 ibia-STMicroelectronics RelOID ::= { 18 }
2379
2380
2381 IBIAidFormats BIOMETRIC ::= {
2382     { BIOMETRIC id : ibia-SAFLINK DATA BirInt16 } |
2383     { BIOMETRIC id : ibia-Bioscrypt DATA BirInt16 } |
2384     { BIOMETRIC id : ibia-Visionics DATA BirInt16 } |
2385     { BIOMETRIC id : ibia-InfineonTechnologiesAG DATA BirInt16 } |
2386     { BIOMETRIC id : ibia-IridianTechnologies DATA BirInt16 } |
2387     { BIOMETRIC id : ibia-Veridicom DATA BirInt16 } |
2388     { BIOMETRIC id : ibia-CyberSIGN DATA BirInt16 } |
2389     { BIOMETRIC id : ibia-eCryp DATA BirInt16 } |
2390     { BIOMETRIC id : ibia-FingerprintCardsAB DATA BirInt16 } |
2391     { BIOMETRIC id : ibia-SecuGen DATA BirInt16 } |
2392     { BIOMETRIC id : ibia-PreciseBiometric DATA BirInt16 } |
2393     { BIOMETRIC id : ibia-Identix DATA BirInt16 } |
2394     { BIOMETRIC id : ibia-DERMALOG DATA BirInt16 } |
2395     { BIOMETRIC id : ibia-LOGICO DATA BirInt16 } |
2396     { BIOMETRIC id : ibia-NIST DATA BirInt16 } |
2397     { BIOMETRIC id : ibia-A3Vision DATA BirInt16 } |
2398     { BIOMETRIC id : ibia-NEC DATA BirInt16 } |
2399     { BIOMETRIC id : ibia-STMicroelectronics DATA BirInt16 },
2400
2401     ... -- Expect others --
2402 }
2403
2404 BirInt16 ::= INTEGER (0..65535)
2405
2406
2407 IBIA-Formats BIOMETRIC ::= {
2408     IBIAidFormats | -- Complete object identifiers
2409     IBIAidFormats, -- Object identifier fragments
2410
2411     ... -- Expect additional IBIA vendor specific formats --
2412 }
2413
2414 id-x984BioInfo OID ::= { cbeff-Owner x984BioInfo(0) }
2415
2416
2417 CBEFFoidFormats BIOMETRIC ::= {

```

```

2418     { BIOMETRIC oid : id-x984BioInfo DATA BiometricInformationSets },
2419
2420     ... -- Expect other objects --
2421 }
2422
2423 x984BioInfo RelOID ::= { x984BioInfo(0) } -- CBEFF owner
2424
2425 CBEFFidFormats BIOMETRIC ::= {
2426     { BIOMETRIC id : x984BioInfo DATA BiometricInformationSets },
2427
2428     ... -- Expect other objects --
2429 }
2430
2431 CBEFF-Formats BIOMETRIC ::= {
2432     CBEFFoidFormats | -- Complete object identifiers
2433     CBEFFidFormats,   -- Object identifier fragments
2434
2435     ... -- Expect additional CBEFF vendor specific formats --
2436 }
2437
2438 MatchingAIDs BIOMETRIC ::= {
2439
2440     ... -- Expect CBEFF assignments in BiometricInformationSets --
2441 }
2442
2443 ProcessingAIDs BIOMETRIC ::= {
2444
2445     ... -- Expect CBEFF assignments in BiometricInformationSets --
2446 }
2447
2448 X9-Formats BIOMETRIC ::= {
2449     X9oidFormats |
2450     X9idFormats,
2451
2452     ... -- Expect additional X9 vendor specific formats --
2453 }
2454
2455 X9oidFormats BIOMETRIC ::= {
2456     ... -- Expect X9 assigned objects --
2457 }
2458
2459 X9idFormats BIOMETRIC ::= {
2460     ... -- Expect X9 assigned objects of the form { 2 x } --
2461 }
2462
2463 certificate-Extensions OID ::= { x9-84 certificate-Extensions(5) }
2464
2465 x968-biometricTemplates OID ::= { certificate-Extensions 0 }
2466 x509-biometricTemplates OID ::= { certificate-Extensions 1 }
2467
2468 -- RSA PKCS #7 Content type
2469
2470 id-data OID ::= {

```

```

2471     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
2472         pkcs7(7) data(1)
2473     }
2474
2475     -- Security object identifiers
2476
2477
2478     -- FIPS 180-1 and FIPS 180-2 Secure Hash Algorithm --
2479
2480     id-sha1 OID ::= {
2481         iso(1) identified-organization(3) oiw(14) secsig(3)
2482         algorithm(2) 26
2483     }
2484
2485     sha2Algorithm OID ::= {
2486         joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
2487         csor(3) nistAlgorithm(4) hashAlgs(2)
2488     }
2489
2490     id-sha256 OID ::= { sha2Algorithm sha256(1) }
2491
2492     id-sha384 OID ::= { sha2Algorithm sha384(2) }
2493
2494     id-sha512 OID ::= { sha2Algorithm sha512(3) }
2495
2496     SHA-Algorithms ALGORITHM ::= {
2497         { OID id-sha1     PARMS NoIV } |
2498         { OID id-sha256   } |
2499         { OID id-sha384   } |
2500         { OID id-sha512   },
2501
2502         ... -- Expect others --
2503     }
2504
2505     NoIV ::= NULL -- No initialization vector
2506
2507     -- X9.57 DSA signature generated with SHA-1 hash (DSA X9.30)
2508
2509     dsa-with-sha1 OID ::= {
2510         iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }
2511
2512     -- X9.71 HMAC with SHA-1 algorithm
2513
2514     hmac-with-SHA1 OID ::= {
2515         iso(1) identified-organization(3) dod(6)
2516         internet(1) security(5) mechanisms(5) 8 1 2 }
2517
2518     -- RSA PKCS #1 signature generated with SHA-1 hash & encryption scheme
2519
2520     sha1WithRSAEncryption OID ::= {
2521         iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 5 }
2522
2523     rsaEncryption OBJECT IDENTIFIER ::= {

```

```
2524     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 1 }
2525
2526 -- ANS X9.52 Triple DES Modes of Operation --
2527
2528 des-ede3-cbc OBJECT IDENTIFIER ::= {
2529     iso(1) member-body(2) us(840) rsadsi(113549)
2530         encryptionAlgorithm(3) 7
2531 }
2532
2533 -- X9.62 ECDSA signature with SHA-1
2534
2535 ecdsa-with-SHA1 OID ::= {
2536     iso(1) member-body(2) us(840) ansi-x962(10045) signatures(4) 1 }
2537
2538
2539 END -- X9-84-Identifiers --
2540
```

8 Examples

2541

2542

2543 Need some introductory text here

2544

2545 8.1 BiometricSyntaxSets (cXER, DER)

2546

2547 This example illustrates a value of type **BiometricSyntaxSets** encoded in XML markup using the
2548 basic XML Encoding Rules (XER), a canonical variant of the XML Encoding Rules (cXER) and a
2549 compact, canonical binary format using the ASN.1 Distinguished Encoding Rules (DER). The
2550 XER, cXER and DER representations use exactly the same abstract values, based on the same
2551 XCBF ASN.1 schema.

2552 Two representations are well-formed XML markup. The third representation is a compact, binary
2553 DER encoding. Both cXER and DER are suitable for use in cryptographic applications involving
2554 digital signatures, since these encoding rules provide one and only one way to encode any given
2555 value.

2556 The BiometricSyntaxSets value encoded in basic XER:

2557

2558

2559

2560

2561

2562

2563

2564

2565

2566

2567

2568

2569

2570

2571

2572

2573

2574

2575

2576

2577

2578

2579

2580

2581

2582

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

2583 The same abstract value encodes in 517 octets using canonical XER:

```
2584 <BiometricSyntaxSets><BiometricSyntax><biometricObjects><BiometricObjec
2585 t><biometricHeader><version>0</version><recordType><id>4</id></recordTy
2586 pe><dataType>2</dataType><purpose>6</purpose><quality>-1</quality><vali
2587 dityPeriod><notBefore>1980.10.4</notBefore><notAfter>2003.10.3.23.59.59
2588 </notAfter></validityPeriod><format><formatOwner><oid>2.23.42.9.10.4.2<
2589 /oid></formatOwner><format></biometricHeader><biometricData>0A0B0C0D</b
2590 iometricData></BiometricObject></biometricObjects></BiometricSyntax></B
2591 iometricSyntaxSets>
```

2592 The same abstract value encodes in 57 octets using DER:

```
2593 3037A0353033A02BA103810104820102830106
2594 8401FFA50F80048F3C0A0481078F530A03173B
2595 3BA60AA0088006672A090A040281040A0B0C0D
```

2596

2597 8.2 SignedData

2598

2599 This example illustrates how to encode one or more biometric samples or templates for
2600 cryptographic enhancement to provide authentication of origin and data integrity services for
2601 biometric samples or templates.

2602

2603 The **SubjectPublicKeyInfo** value from the signer's X.509 Public Key Certificate is shown here in
2604 the hexadecimal representation of a DER encoded sequence:

2605

```
2606 308201B73082012C06072A8648CE3804013082011F028181
2607 00FD7F53811D75122952DF4A9C2EECE4E7F611B7523CEF44
2608 00C31E3F80B6512669455D402251FB593D8D58FABFC5F5BA
2609 30F6CB9B556CD7813B801D346FF26660B76B9950A5A49F9F
2610 E8047B1022C24FBBA9D7FEB7C61BF83B57E7C6A8A6150F04
2611 FB83F6D3C51EC3023554135A169132F675F3AE2B61D72AEF
2612 F22203199DD14801C70215009760508F15230CCB292B982
2613 A2EB840BF0581CF502818100F7E1A085D69B3DDECBBCAB5C
2614 36B857B97994AFBBFA3AEA82F9574C0B3D0782675159578E
2615 BAD4594FE67107108180B449167123E84C281613B7CF0932
2616 8CC8A6E13C167A8B547C8D28E0A3AE1E2BB3A675916EA37F
2617 0BFA213562F1FB627A01243BCCA4F1BEA8519089A883DFE1
2618 5AE59F06928B665E807B552564014C3BFECF492A03818400
2619 028180476ACACB486186A153E25AE0E243FAAF0CD9105CF4
2620 DCF63412F36ABF671F53637E5F9FA7C5ADC78288FDB9FA3C
2621 FAFDEBFDD7A7C3FF2BD63D32F4773413EBD9EAB3CA03BA2D
2622 ED583187763181CB376954FD13F1F8E046D4E3D40652CA8D
2623 4645439A3ADB8D964F98F81E57147BDF4C009885CAD55D13
2624 B38DBAA2F9CBF13DC525F6
```

2625

2626 The biometric objects to be signed with the signer's private key associated with the public key
2627 provided above shown in the canonical form used as input to the message digest process used to
2628 create the digital signature on the signed content:

2629

```
2630 <BiometricObjects><BiometricObject><biometricHeader><version>0</version>  
2631 ><recordType><id>4</id></recordType><dataType>2</dataType><purpose>6</p  
2632 urpose><quality>-1</quality><validityPeriod><notBefore>1998.10.1</notBe  
2633 fore><notAfter>2008.10.1</notAfter></validityPeriod><format><formatOwne  
2634 r><oid>2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><  
2635 biometricData>0102030405060708090A0B0C0D0E0F11</biometricData></Biometr  
2636 icObject><BiometricObject><biometricHeader><version>0</version><recordT  
2637 ype><oid>4</oid></recordType><dataType>1</dataType><purpose>3</purpose>  
2638 <quality>50</quality><validityPeriod><notBefore>1998.10.2</notBefore><n  
2639 otAfter>2008.10.2</notAfter></validityPeriod><format><formatOwner><oid>  
2640 2.23.42.9.10.4.2</oid></formatOwner></format></biometricHeader><biometr  
2641 icData>0102030405060708090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11  
2642 </biometricData></BiometricObject><BiometricObject><biometricHeader><ve  
2643 rsion>0</version><recordType><id>4</id></recordType><dataType>0</dataTy  
2644 pe><purpose>3</purpose><quality>100</quality><validityPeriod><notBefore  
2645 >1998.10.3</notBefore><notAfter>2008.10.3</notAfter></validityPeriod><f  
2646 ormat><formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner></format></  
2647 biometricHeader><biometricData>0102030405060708090A0B0C0D0E0F1101020304  
2648 05060708090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11</biometricData  
2649 ></BiometricObject></BiometricObjects>  
2650
```

2651 The complete integrity object represented as an XML document:

2652

```
2653 <?xml version="1.0" encoding="UTF-8"?>  
2654  
2655 <BiometricSyntaxSets>  
2656 <BiometricSyntax>  
2657 <integrityObjects>  
2658 <biometricObjects>  
2659 <BiometricObjects>  
2660  
2661 <BiometricObject>  
2662 <biometricHeader>  
2663 <version>0</version>  
2664 <recordType><id>4</id></recordType>  
2665 <dataType>2</dataType>  
2666 <purpose>6</purpose>  
2667 <quality>-1</quality>  
2668 <validityPeriod>  
2669 <notBefore>1998.10.1</notBefore>  
2670 <notAfter>2008.10.1</notAfter>  
2671 </validityPeriod>  
2672 <format>  
2673 <formatOwner>  
2674 <oid>2.23.42.9.10.4.2</oid>  
2675 </formatOwner>
```


2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728

```
</format>
</biometricHeader>
<biometricData>
  0102030405060708090A0B0C0D0E0F11
</biometricData>
</BiometricObject>

<BiometricObject>
  <biometricHeader>
    <version>0</version>
    <recordType><id>4</id></recordType>
    <dataType>1</dataType>
    <purpose>3</purpose>
    <quality>50</quality>
    <validityPeriod>
      <notBefore>1998.10.2</notBefore>
      <notAfter>2008.10.2</notAfter>
    </validityPeriod>
    <format>
      <formatOwner>
        <oid>2.23.42.9.10.4.2</oid>
      </formatOwner>
    </format>
  </biometricHeader>
  <biometricData>
    0102030405060708090A0B0C0D0E0F11
    0102030405060708090A0B0C0D0E0F11
  </biometricData>
</BiometricObject>

<BiometricObject>
  <biometricHeader>
    <version>0</version>
    <recordType><id>4</id></recordType>
    <dataType>0</dataType>
    <purpose>3</purpose>
    <quality>100</quality>
    <validityPeriod>
      <notBefore>1998.10.3</notBefore>
      <notAfter>2008.10.3</notAfter>
    </validityPeriod>
    <format>
      <formatOwner>
        <oid>2.23.42.9.10.4.2</oid>
      </formatOwner>
    </format>
  </biometricHeader>
  <biometricData>
    0102030405060708090A0B0C0D0E0F110102030405060708
    090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11
  </biometricData>
</BiometricObject>
```

```

2729         </BiometricObjects>
2730     </biometricObjects>
2731
2732     <integrityBlock>
2733         <signedData>
2734             <version>84</version>
2735             <digestAlgorithms>
2736                 <DigestAlgorithmIdentifier>
2737                     <algorithm>1.3.14.3.2.26</algorithm>
2738                     <parameters><NoIV/></parameters>
2739                 </DigestAlgorithmIdentifier>
2740             </digestAlgorithms>
2741             <encapContentInfo>
2742                 <eContentType>1.2.840.113549.1.7.1</eContentType>
2743             </encapContentInfo>
2744             <signerInfos>
2745                 <SignerInfo>
2746                     <version>84</version>
2747                     <sid>
2748                         <certHash>
2749                             <withAlgID>
2750                                 <hashAlgorithm>
2751                                     <algorithm>1.3.14.3.2.26</algorithm>
2752                                     <parameters><NoIV/></parameters>
2753                                 </hashAlgorithm>
2754                                 <digest>
2755                                     DA9245BCD6E666749F43C1A1BD070BAF259B70AA
2756                                 </digest>
2757                             </withAlgID>
2758                         </certHash>
2759                     </sid>
2760                     <digestAlgorithm>
2761                         <algorithm>1.3.14.3.2.26</algorithm>
2762                         <parameters><NoIV/></parameters>
2763                     </digestAlgorithm>
2764                     <signatureAlgorithm>
2765                         <algorithm>1.2.840.10040.4.3</algorithm>
2766                         <parameters><NoIV/></parameters>
2767                     </signatureAlgorithm>
2768                     <signature>
2769                         302C02144F9CA4507E2638AE9B632A3698A7AE84858F13
2770                         3802140BD484312B36B090D2DF8B8A4719353F9A1EFAA5
2771                     </signature>
2772                 </SignerInfo>
2773             </signerInfos>
2774         </signedData>
2775     </integrityBlock>
2776
2777     </integrityObjects>
2778 </BiometricSyntax>
2779 </BiometricSyntaxSets>
2780
2781

```

2782 8.3 EncryptedData (fixedKey)

2783

2784 This example illustrates how to encode one or more biometric samples or templates for
2785 cryptographic enhancement. A group of three biometric objects is used for this purpose, though
2786 XCBF allows any number of objects to be used. The optional, cleartext biometric headers are not
2787 included in the example program message. The group of three biometric objects is encrypted for
2788 privacy using a fixed Triple DES key.

2789 As shown by this example, XCBF users can create and exchange arbitrary collections of
2790 biometric information to suit the needs of their security applications. This capability provides the
2791 application designer complete flexibility. The order of the biometric objects is determined in the
2792 application by the sender, allowing them to prioritize or order biometric information for purposes
2793 such as aging of data, or grouping records by quality or data type or entity.

2794

2795 Collections of useful biometric information include:

2796 pairs of iris image templates for an individual; one for each eye

2797 collections of paired iris image templates for a group of individuals

2798 collections of finger print image templates, one per digit for an individual

2799 sets of individual finger print image template collections for a group of persons

2800 a collection of mixed biometric templates for an individual; say, retina, hand geometry, and DNA

2801 collections of templates for groups of individuals, such as:

2802 - all employees at work today, or

2803 - all of the passengers on Flight 12, or

2804 - all of the finger print samples collected on Tuesday

2805

2806 The hex encoding of a two key Triple DES key used to encrypt a series of three values of type
2807 BiometricObject:

2808

2809 623131F21CA8B9B6 707A7F3DB68A5401 623131F21CA8B9B6

2810

2811 The biometric objects to be encrypted (shown here for ease of reading not in the canonical form
2812 used as input to the encryption process):

2813

2814 <BiometricObjects>

2815

2816 <BiometricObject>

2817 <biometricHeader>

2818 <version>0</version>

2819 <recordType><id>4</id></recordType>

2820 <dataType>2</dataType>

2821 <purpose>6</purpose>

2822 <quality>-1</quality>

2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834

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

2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875

```
<BiometricObject>  
  <biometricHeader>  
    <version>0</version>  
    <recordType><oid>4</oid></recordType>  
    <dataType>1</dataType>  
    <purpose>3</purpose>  
    <quality>50</quality>  
    <validityPeriod>  
      <notBefore>1998.10.2</notBefore>  
      <notAfter>2008.10.2</notAfter>  
    </validityPeriod>  
    <format>  
      <formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner>  
    </format>  
  </biometricHeader>  
  <biometricData>  
    0102030405060708090A0B0C0D0E0F11  
    0102030405060708090A0B0C0D0E0F11  
  </biometricData>  
</BiometricObject>  
  
<BiometricObject>  
  <biometricHeader>  
    <version>0</version>  
    <recordType><id>4</id></recordType>  
    <dataType>0</dataType>  
    <purpose>3</purpose>  
    <quality>100</quality>  
    <validityPeriod>  
      <notBefore>1998.10.3</notBefore>  
      <notAfter>2008.10.3</notAfter>  
    </validityPeriod>  
    <format>  
      <formatOwner><oid>2.23.42.9.10.4.2</oid></formatOwner>  
    </format>  
  </biometricHeader>  
  <biometricData>  
    0102030405060708090A0B0C0D0E0F110102030405060708  
    090A0B0C0D0E0F110102030405060708090A0B0C0D0E0F11  
  </biometricData>
```

2876
2877
2878
2879

```
</BiometricObject>  
</BiometricObjects>
```

2880 The complete XCBF privacy message:

2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926

```
<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>  
              F9C32CC99FA8CEA043B88D7A4981C32836A0D0444FECA6B  
              0B8CC83622CF4112F04B24710498D82996183196809FA353  
              39BABC6323F0509700F9ACE6CEA6B06033282486BB1E9AD8  
              FEFED445EFB00B103547F3B07E9CC839E636A10C780744B2  
              F0B8536926E38C2FB3F2F55882AFB8E1AA5CEF23B8C67F47  
              70171EB7DEA87772F8E4615C62AD06C71450D8569A3B69B4  
              828DA9D25F7B91E117A122C8025E29EE4947C39C22C2F299  
              EAEDC3AB24F39B307F84005D82502F972F2C5E1834ECF992  
              912CA6DCCF4B4FCBD84F019BD632A72A5513E3E57C697146  
              6CAA0F9BDADA69095876ADA8F4D3FB5C4F211C9915AFA096  
              63C657EDBAD1CDE85D24B2C0B4A97355D72507987D9F83AB  
              24A5E10FD08FBC362BBC54EC7B9E122A8457AFA68D25B82A  
              C897BDA198CB02D4E1A7E2A731908893A7B3286A5B567394  
              823E178F75F4597CA18B0019C99E6DA6D793D5A8284ECDB7  
              AA34C6C85B7C606370248CFB17C63642AA49FFE835C0EE33  
              8EAC25C98E9FC0095DB10F05B405C366F42594389CC5C178  
              9EA2CEC0F388AB2C4DA2006C491751005E55975FD3A6725C  
              7D608E41A688BD6E8DF552DEAAF6ADDE232E03D2B756D3CD  
              8A9C542017C41A2971914D999F4CF630B347F496AA1D5876  
              4CB805DF6ED506456017351ACA4BF7A80B7B6773359C2F03  
              BE0AC907A1951FE2FAC6589321D873BB752104180E7026F4  
              772570896C86E78303024F2EB0D62635E872901849952029  
              E178645E2A2A7E4F0A2255F65692DD66DA3E0CD8382EFA3F  
              3A1DBAC342DBC10D992E01E5A42E7993C29CF6BD71BECA6A  
              BE7CBE2DF2E9D91DA66FA51DF914BDA34A925B26EEF76F6F  
              CD8DF2C9CF9D497C960FD1DD99D3EAF22442F2B73F59E194  
              50B102C848A220764AFCAF9B182C619F505166760BC7D63E  
              09F16BE521C5DB9BFB2784A6BAD69C23899486381A1DEB14  
              31EECAAB4BD02DDAD00F181AF069D48977F2E01D0184A387  
              1F132654651E3965B0076238C200390187FAA213F3A6012F
```

```

2927 7D9AEC95292B9069FA1409DFB3C32AF19656D9DDD83F07F
2928 D634E9EFAFFF7D7236BC08B4417B4618EF767790E6AE653D
2929 27F8C797F9132E497B20E5A13B2BBF4F64187BE84B61C9C5
2930 CD25ACCE38E3D22F678C79E4532A0B5C5D66EF557D829CCD
2931 466B1607AF462163ED07032474DA7240360A6BF0354795EE
2932 01CF95A8776E9D16717CF140A72804C53F9619FE6B96141E
2933 C97BDB495FFBDC0194665D2A1196EAE814307CB2FAAA8C8
2934 2476B11EA01B42E2C40C7306710323E288EC1A9166BF9D85
2935 2E3976257315ED6BB70F6933017072830DD1CCC399380114
2936 8935F5CFB19C84B8FE753AE5FFE1D89A2872C5675C4E8E8A
2937 A3A9C4ACBA749E84CBF3969D72C6B84299E7B5F7062C797F
2938 9F057370183C468111522ACBC84DC410900395DCB25E81EA
2939 66B6D20E0AE5D5B7206EABD451539A7EBB2FBDDFCE2B06EE
2940 09E5D5A16904CB2496EDCAE16EFC4B0FAF107B074A06CBE4
2941 5747D8CA1FA0F2F2E8BE5BF27D9F248DF87D351F3EC6EAE7
2942 36A8A6BE99811373480040117B6A78C5C0EA97A524FCD3BA
2943 5F2A4BD3265CE1CA1A94ACA5FE0E6A69FDB6102AD97FA2EE
2944 4DBF6F399075B3AD9339304E16D25959506E90AE771D6CF5
2945 C1C7F793A77D52121C152F66B173A04A851C596090354504
2946 785DBA06B8C49E4DC5E47734DC95451E8D1A58098888F505
2947 E6CA12EA4C8DE8C08102C088A66FB42E38F17694166B7841
2948 CDAB5E46DA0C9ACDBC92840450AA547491CCD82242E32A02
2949 6D1BDB1C6B35B47BCC672E4ED8702DCDC804B20692508747
2950 5C939872694192D6088E0353460C875C0801464AA20AFC71
2951 8CD61A72FDC730609AAA1A20952F3D8790C489B20B3FF6A3
2952 2346B544B18A91FCF485E06F85A58B310746507330CF3376
2953 27875EE54C22CD489BFDD89F7020BD59C8AD15D403DE2E78
2954 75B5571A5EBF80C78B8990448F5B71144FFEBE4B5726BCD1
2955 </encryptedContent>
2956 </encryptedContentInfo>
2957 </fixedKey>
2958 </privacyBlock>
2959 </privacyObjects>
2960 </BiometricSyntax>
2961 </BiometricSyntaxSets>

```

2962 **Appendix A. Acknowledgments**

2963 The following individuals were members of the committee during the development of this
2964 specification:

2965

- 2966 • Tyky Aichelen, IBM
- 2967 • Karl Best, OASIS
- 2968 • Taylor Boon, BioNetrix Systems
- 2969 • Robin Cover, Isogen
- 2970 • Ed Day, Objective Systems
- 2971 • Dr. Paul G r me, AULM
- 2972 • Phillip H. Griffin (chair), Griffin Consulting
- 2973 • Todd Harbour, FGM
- 2974 • William Koenig, Bank of America
- 2975 • John Larmouth, Larmouth T&PDS Ltd
- 2976 • Monica Martin, Drake Certivo
- 2977 • Rob Philpott, RSA Security
- 2978 • Stefan Ri, Microsoft
- 2979 • Darran Rolls, Waveset
- 2980 • Bancroft Scott, OSS Nokalva
- 2981 • Clifford Thompson, TGI Solutions
- 2982 • Paul Thorpe, OSS Nokalva
- 2983 • Alessandro Triglia, OSS Nokalva
- 2984 • Per Vorm, BEA

2985

Appendix B. Revision History

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

2986

2987

Appendix C. Notices

2988 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2989 that might be claimed to pertain to the implementation or use of the technology described in this
2990 document or the extent to which any license under such rights might or might not be available;
2991 neither does it represent that it has made any effort to identify any such rights. Information on
2992 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
2993 website. Copies of claims of rights made available for publication and any assurances of licenses
2994 to be made available, or the result of an attempt made to obtain a general license or permission
2995 for the use of such proprietary rights by implementers or users of this specification, can be
2996 obtained from the OASIS Executive Director.

2997 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2998 applications, or other proprietary rights, which may cover technology that may be required to
2999 implement this specification. Please address the information to the OASIS Executive Director.

3000 Copyright © 2002 OASIS Open, Inc. All Rights Reserved.

3001 This document and translations of it may be copied and furnished to others, and derivative works
3002 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
3003 published and distributed, in whole or in part, without restriction of any kind, provided that the
3004 above copyright notice and this paragraph are included on all such copies and derivative works.
3005 However, this document itself does not be modified in any way, such as by removing the
3006 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
3007 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
3008 Property Rights document must be followed, or as required to translate it into languages other
3009 than English.

3010 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
3011 successors or assigns.

3012 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3013 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
3014 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
3015 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
3016 PARTICULAR PURPOSE.