INTERNATIONAL STANDARD

ISO/IEC 24761

Second edition
2019-10

Information technology — Security techniques — Authentication context for biometrics

Technologies de l'information — Techniques de sécurité — Contexte d'authentification biométrique

Reference number
ISO/IEC 24761:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword ... **iv**
Introduction ... **v**
1 Scope ... **1**
2 Normative references ... **1**
3 Terms and definitions ... **1**
4 Symbols and abbreviated terms ... **5**
5 Model and framework of ACBio ... **6**
5.1 Biometric enrolment and verification process model and Biometric Processing Unit ... 6
5.2 BPU role and biometric capability class ... 8
5.2.1 Overview ... 8
5.2.2 BPU role ... 9
5.2.3 Biometric capability class ... 12
5.3 Framework for use of ACBio ... 17
5.3.1 General ... 17
5.3.2 Preparation in the production process ... 18
5.3.3 Preparation in the subject enrolment process ... 20
5.3.4 ACBio instance generation in the biometric verification process ... 21
5.3.5 Validation of biometric verification process with ACBio instances ... 23
6 ACBio instance ... **23**
6.1 General ... 23
6.2 BPU information block ... 25
6.3 Biometric process block ... 26
6.4 BRT certificate information ... 27
7 Definition of components in BPUInformationBlock ... **28**
7.1 BPU certificate ... 28
7.2 BPUReportInformation ... 29
7.2.1 General ... 29
7.2.2 BPUFunctionReport ... 30
7.2.3 BPUSecurityReport ... 36
8 BRT certificate ... **38**
8.1 General ... 38
8.2 BRTCContentInformation ... 39
8.3 Format Owner and Format Type values ... 41
Annex A (normative) **ASN.1 module for ACBio** ... **42**
Annex B (informative) **Implementation examples** ... **50**
Bibliography ... **75**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

This second edition cancels and replaces the first edition (ISO/IEC 24761:2009), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 24761:2009/Cor.1:2013. The main changes compared to the previous edition are as follows:

— extension of data types which reflects the progress in biometric technology for protection of biometric data such as renewable biometrics and others,

— introduction of a new biometric capability model which makes the validation of ACBio instances simpler, and

— changes to the ASN.1 module as a result of the above changes.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

A biometric verification process executed at a remote site is exposed to many risks, for example, falsified reference, forged captured biometric data, and unreliable biometric products, etc. How can the validator check whether a biometric verification process, carried out in a remote site, is trustworthy? This document gives a mechanism to cope with this problem.

In general, the reliability of the result of a biometric verification process is dependent both on the security of the process executed and on the functional performance of the biometric products used. If products offering a better functional performance are used, the result will be more reliable. If the products are not secure or the process has been executed in an insecure environment, then the result will not be reliable.

In the Internet environment, the validator of a biometric verification process usually does not directly know about the biometric products used or about the process(es) executed at a remote site. Authentication Context for Biometrics (ACBio) provides a solution to the above problem and mitigates security risks of biometric authentication, by sending information about the products used and the processes executed at the remote site to the validator if the biometric processing has resulted successfully.

ACBio defines data formats for evidence data generated by biometric processing units (BPUs), such as a sensor, smartcard or comparison device, which are carried in data structures called ACBio instances. ACBio specifies a trust and assurance mechanism based on digital signature technology to provide assured information about the BPU and its execution of the biometric enrolment and verification processes where the assured information about the BPU is provided as BPU report issued by the vendor of the BPU. This is based on the Public Key Infrastructure (PKI) technology and PKIX (see ISO/IEC 9594-8 and RFC 3852). An ACBio instance carries information about the biometric processing units (BPUs), biometric reference and biometric verification results that together characterise a biometric verification transaction. Assurance of the information in an ACBio instance is provided by digital certificates associated with the relevant elements of the information. These certificates are issued by trusted certification authorities in registration processes which gather evidence about the BPUs and their verification performance capabilities, and the biometric reference and the binding to a known subject. The certificates serve two purposes. Firstly, to provide assurance of the identity of the source of the biometric transaction (the BPUs) and the biometric reference, and secondly to provide assurance for the biometric verification result contained in the transaction. With all the ACBio instances sent to the validator, it can check the integrity of the data transmitted between BPUs. The real-time information of presentation attack detection is not provided with this document. The BPU report may, however, contain the assurance information of the PAD mechanism. ACBio recognizes that privacy requirements concerned with the storage of biometric data must comply with local laws and legislation on data privacy. ACBio ensures that the validator can validate the result of the biometric verification process without receiving private data, such as the biometric sample acquired from the claimant or the biometric reference used for comparison.

An ACBio instance is a report that is encoded using the Basic Encoding Rules (BER) of ASN.1 [see ISO/IEC 8824 (all parts)], commonly supported by cryptographic tool kit vendors. The syntax is algorithm independent and supports provision of data integrity and data origin authentication. In regard to cryptographic algorithms, those specified by ISO/IEC JTC 1/SC 27 are recommended, though any algorithm appropriate for use by a given community may be used.

This document reflects the progress in biometric technology for protection of biometric data such as renewable biometrics specified in ISO/IEC 24745 and others by extending the variation of biometric data types transmitted between biometric subprocesses, and in addition establishes a new biometric capability model which makes the validation of ACBio instance(s) simpler. This has resulted in some changes to the ASN.1 module which will give rise to inter-operational incompatibilities between systems implementing different versions of the ASN.1 modules.

Information technology — Security techniques — Authentication context for biometrics

1 Scope

This document defines the structure and the data elements of Authentication Context for Biometrics (ACBio), which is used for checking the validity of the result of a biometric enrolment and verification process executed at a remote site. This document allows any ACBio instance to accompany any biometric processes related to enrolment and verification. The specification of ACBio is applicable not only to single modal biometric enrolment and verification but also to multimodal fusion. The real-time information of presentation attack detection is not provided in this document. Only the assurance information of presentation attack detection (PAD) mechanism can be contained in the BPU report.

Biometric identification is out of the scope of this document.

This document specifies the cryptographic syntax of an ACBio instance. The cryptographic syntax of an ACBio instance is defined in this document applying a data structure specified in Cryptographic Message Syntax (CMS) schema whose concrete values can be represented using a compact binary encoding. This document does not define protocols to be used between entities such as BPUs, claimant, and validator. Its concern is entirely with the content and encoding of the ACBio instances for the various processing activities.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 2382-37, *Information technology — Vocabulary — Part 37: Biometrics*

ISO/IEC 9594-2, *Information Technology — Open Systems Interconnection — The Directory: Models*

ISO/IEC 9594-8, *Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 24745, *Information technology — Security techniques — Biometric information protection*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 2382-37, ISO/IEC 9594-8, ISO/IEC 24745, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

NOTE Terminology in this document has been updated where possible to conform to ISO/IEC 2382-37:2017. However, to maintain maximum compatibility with the terminology and type names in ASN.1 module in the first edition of this document, certain terms have been carried over from the previous edition.

3.1
ACBio instance
report generated by a BPU compliant to this document to show the validity of the execution result of one or more subprocesses executed in the BPU

3.2
biometric capability class
class of configurations for how the biometric enrolment/verification can be divided into single or multiple BPU roles

Note 1 to entry: There are four biometric capability classes defined in this document:

— all-in-one (biometric capability) class;

— sensor-and-comparator (biometric capability) class;

— storage-and-others (biometric capability) class; and

— sensor-only (biometric capability) class.

Note 2 to entry: To express each biometric capability class, "biometric capability" is usually omitted.

3.3
biometric processing unit
BPU
trusted implementation of a collection of biometric subprocesses implemented in a single physical unit

Note 1 to entry: A BPU commonly comprises biometric subprocesses that are sequential in the process flow for a biometric verification.

Note 2 to entry: Application/service requirements typically require BPU subprocesses to meet a uniform level of security assurance. In ACBio, assurance is achieved through a BPU evaluation process that is authenticated by means of an X.509 certificate embedded in an ACBio instance.

3.4
biometric processing unit certificate
BPU certificate
X.509 certificate that is issued to a BPU by a certification authority

3.5
biometric processing unit certification authority
BPU certification authority
X.509 certification authority which issues BPU certificates

3.6
biometric processing unit function report
BPU function report
report on the function of the BPU generated by the manufacturer of the BPU

3.7
biometric processing unit IO Index
BPU IO Index
integer number uniquely assigned to each biometric data stream between BPUs by the subject which utilizes the function of the BPU (e.g. a software) and used by the validator to reconstruct the data flow among BPUs

3.8
biometric processing unit report
BPU report
report on a BPU, which consists of a BPU function report and a BPU security report

3.9
biometric processing unit security report
BPU security report
report on the security of a BPU, which contains an assurance information of the security of the BPU

Note 1 to entry: The assurance information on PAD mechanism, if present in the BPU report, shall be included in the BPU security report.

3.10
biometric reference
one or more stored biometric samples, biometric templates or biometric models attributed to a biometric data subject and used as the object of biometric comparison

EXAMPLE Face image stored digitally on a passport, fingerprint minutiae template on a National ID card or Gaussian Mixture Model for speaker recognition, in a database.

Note 1 to entry: A biometric reference may be created with implicit or explicit use of auxiliary data, such as Universal Background Models.

Note 2 to entry: The subject/object labelling in a *comparison*might be arbitrary. In some comparisons, a biometric reference might be used as the subject of the comparison with other biometric references or incoming samples and input to an algorithm for biometric comparison. For example, in a duplicate enrolment check a biometric reference will be used as the subject for comparison against all other biometric references in the database.

Note 3 to entry: A term "biometric reference template" is also used to mean biometric reference partly in this document. See NOTE at the beginning of Clause 3.

[SOURCE: ISO/IEC 2382-37:2017, 3.3.16, modified — Note 3 to entry has been added.]

3.11
biometric reference template certificate
BRT certificate
certificate that is issued to a biometric reference by a BRT certification organization and enables the validator to determine the authenticity of the biometric reference

3.12
biometric reference template certification organization
BRT certification organization
organization which issues BRT certificates

3.13
biometric subprocess
subprocess
part of an overall biometric enrolment or verification process performing a specific function

Note 1 to entry: Biometric subprocess functions is one of the following: data capture, intermediate signal processing, final signal processing, storage, comparison, and decision.

3.14
BPU role
combination of biometric functionalities provided by a BPU whose combination of functionalities is specified in this document

Note 1 to entry: A BPU role can be referenced by a name that describes the biometric functionality provided by the BPU: all BPU role, sensor BPU role, comparator-with-storage BPU role, comparator BPU role, and storage BPU role.

3.15
captured biometric reference
captured biometric sample or combination of captured biometric samples used as a biometric reference

3.16
control value
random number provided by a validator by which the validator can check whether an ACBio instance is generated at the validator's request or not

3.17
declaration expression
explicit expression of BPU function report in which information on the subprocesses and data flows are contained

3.18
enrolment organization
organization which handles enrolment and creates and stores biometric references

3.19
evaluation organization
organization which evaluates a BPU function or security

3.20
final signal processing
signal processing stage immediately preceding biometric comparison

Note 1 to entry: See NOTE at the beginning of Clause 3.

3.21
intermediate biometric reference
intermediate biometric sample or combination of intermediate biometric samples used as a biometric reference

Note 1 to entry: An intermediate biometric reference is processed through biometric final processing and then compared against a biometric sample.

3.22
intermediate signal processing
any manipulation of a biometric sample that does not produce biometric features

Note 1 to entry: In ISO/IEC 2382-37, "intermediate biometric sample processing" (ISO/IEC 2382-37:2017, 3.5.9) is defined in place of "intermediate signal processing". See NOTE at the beginning of Clause 3.

Note 2 to entry: The term "biometric feature" is defined as numbers or labels extracted from biometric samples and used for comparison in ISO/IEC 2382-37:2017, 3.3.11.

3.23
on-card biometric comparison
OCBC
performing comparison and decision making on an IC card where the biometric reference is retained on-card in order to enhance security and privacy

3.24
processed biometric reference
processed biometric sample or combination of processed biometric samples used as a biometric reference

3.25
processed biometric sample
biometric sample or biometric feature set input to an algorithm for biometric comparison to a biometric reference(s)

Note 1 to entry: ISO/IEC 2382-37, a term "biometric probe" (ISO/IEC 2382-37:2017, 3.3.14) is defined in place of "processed biometric sample". See NOTE at the beginning of Clause 3.

3.26
renewable biometric sample
biometric sample which has a property of a transform or process to create multiple, independent transformed biometric data derived from one or more biometric samples obtained from the same data subject and which can be used to recognize the individual while not revealing information about the generative biometric data

Note 1 to entry: Renewablity is defined in ISO/IEC 24745:2011 as: property of a transform or process to create multiple, independent transformed biometric references derived from one or more biometric samples obtained from the same data subject and which can be used to recognize the individual while not revealing information about the original reference. This definition is only applicable to biometric reference. The definition in this document is slightly modified from that in ISO/IEC 24745:2011 so that it can be applied to biometric sample as well as biometric reference because the biometric sample in renewable biometrics should be so transformed that it can be compared with a renewable biometric reference.

3.27
role expression
implicit expression of BPU function report for BPU which has a BPU role

3.28
subprocess index
integer uniquely assigned to each subprocess within a BPU by the manufacturer of the BPU

3.29
subprocess IO index
unique integer assigned to each data stream between subprocesses in a BPU so that the validator can reconstruct the data flow between subprocesses in the BPU

3.30
validator
<of biometric verification> entity which makes a decision on whether the result of a biometric verification process is acceptable or not, based on the policy, using one or more comparison decisions and possibly other information, supported by ACBio instances

4 Symbols and abbreviated terms

ACBio	Authentication Context for Biometrics
ASN.1	Abstract Syntax Notation One as defined in the ISO/IEC 8824 series
BER	Basic Encoding Rules (of ASN.1)
BIR	Biometric Information Record
BRT	Biometric Reference Template
CBEFF	Common Biometric Exchange Formats Framework as defined in ISO/IEC 19785-1
CMS	Cryptographic Message Syntax as defined in RFC 3852 and RFC 5911
FAR	False Acceptance Rate
FRR	False Rejection Rate
PAD	Presentation Attack Detection

PKI Public Key Infrastructure

STOC Store On Card

URI Uniform Resource Identifier

5 Model and framework of ACBio

5.1 Biometric enrolment and verification process model and Biometric Processing Unit

This document defines the structure and the data elements of Authentication Context for Biometrics (ACBio), which is used for checking the validity of the result of a biometric enrolment and verification process executed at a remote site. An ACBio instance shall be accompanied witha biometric data output processesed by a BPU for biometric enrolment and verification.

ACBio's design is based on the following biometric subprocesses:

a) data capture: this subprocess captures biometric information from a claimant and converts it to a captured biometric sample. The captured biometric sample is transmitted to the intermediate signal processing subprocess for further processing;

b) intermediate signal processing: this subprocess receives a captured biometric sample and transforms it into an intermediate biometric sample. The intermediate biometric sample is transmitted to the final signal processing subprocess for further processing;

c) final signal processing: this subprocess receives an intermediate biometric sample and transforms it into a processed biometric sample. The processed biometric sample through the final signal processing is transmitted either to the comparison subprocess (for verification) or to the storage subprocess (for enrolment as the biometric reference). There is a variation of the output of the final signal processing, a renewable biometric sample;

d) storage: this subprocess stores one of three types of biometric reference; captured biometric reference (Key item 1 in Figure 1 and Figure 2), intermediate biometric reference (Key item 2 in Figure 1 and Figure 2), or processed biometric reference (Key item 3 in Figure 1 and Figure 2). One of the three types of biometric reference will be compared with a biometric sample for verification. As in c), there is a variation for processed biometric reference, renewable biometric reference;

e) comparison: this subprocess receives a biometric sample, which is acquired originally from a claimant, and can be further processed or not, and a biometric reference. This subprocess compares the biometric sample and the processed biometric reference, and calculates the similarity, which is called a comparison score. The comparison score is transmitted to the decision subprocess.

NOTE The processing of comparing a renewable biometric sample with a renewable biometric reference is different from that of comparing a processed biometric sample with a processed biometric reference. However, they are not distinguished in this document. They are distinguished only by the data input into the subprocess. If a BPU A executes the comparision subprocess and another BPU B executes the final signal processing subprocess or the storage subprocess, the BPU B does not know whether the BPU A can process the data submitted from the BPU B. In the ACBio framework, only the application system knows the compatibility of the data exchanged between the BPUs through the negotiation between them in advance. See B.1.2.5 for details.

f) decision: this subprocess receives a comparison score from the comparison subprocess, evaluates the score under rules determined by the security policy in use, decides the validity of the claimant's identity, and outputs the comparison decision, match or non-match, which is sent to the validator.

Besides the above six subprocesses biometric products generally have PAD mechanism. Figure 1 shows a general model of biometric verification including PAD mechanism. However, PAD is not discussed further in this document because it is dealt with only in the assurance information in the BPU report in this document. For more details of PAD, see ISO/IEC 30107-1.

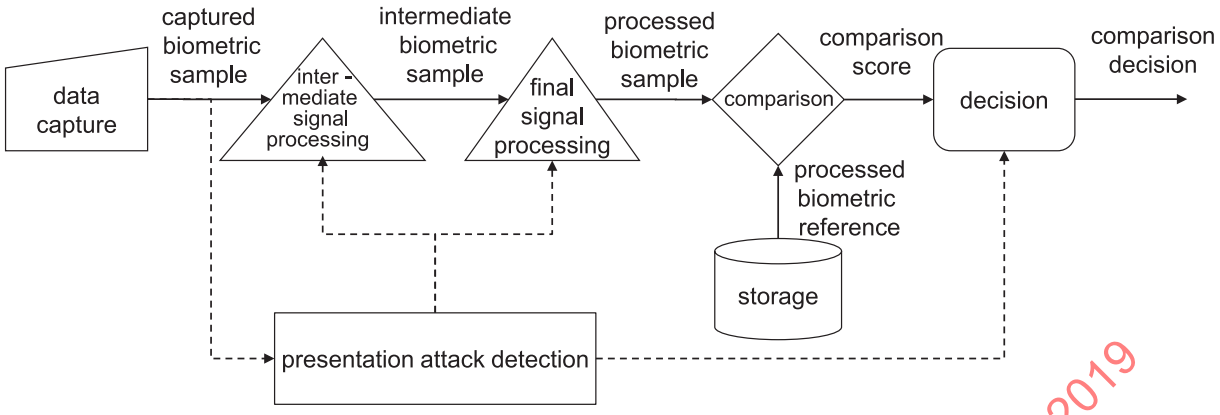
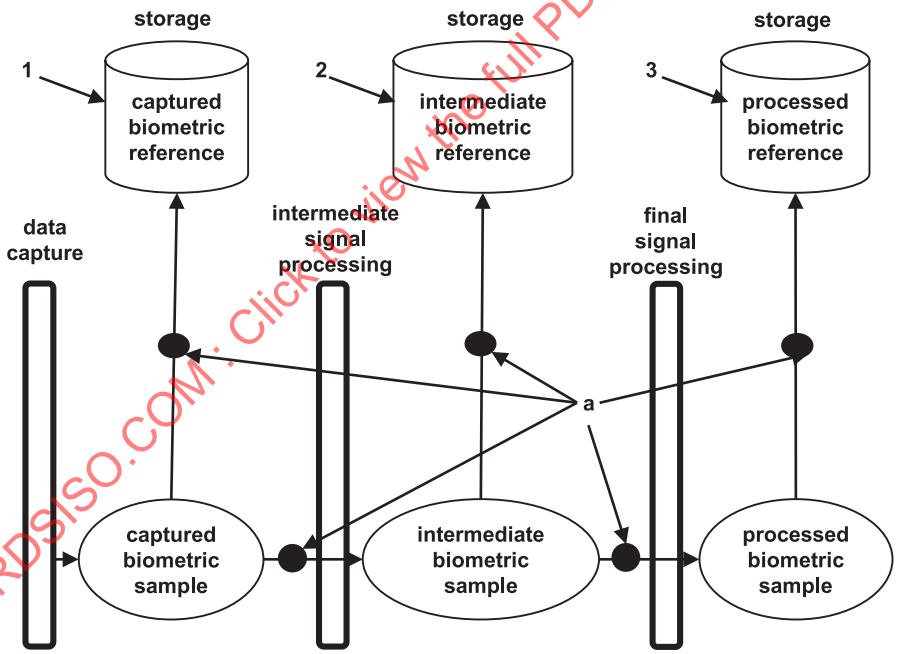


Figure 1 — General model of biometric verification including PAD mechanism

Figure 2 shows three cases of biometric enrolment process where the storage subprocess stores:

1) captured biometric samples,
2) intermediate biometric samples, and
3) processed biometric samples.



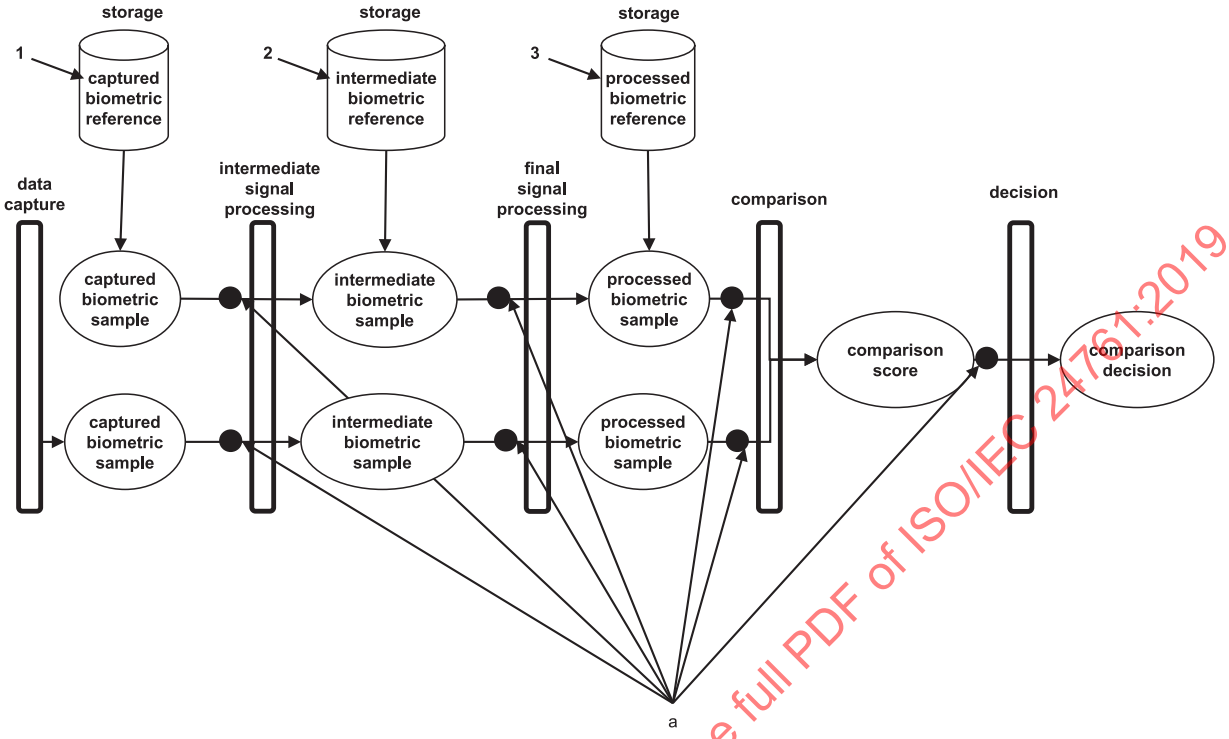
Key

1 storage subprocess which stores captured biometric samples
2 storage subprocess which stores intermediate biometric samples
3 storage subprocess which stores processed biometric samples
[a] Possible untrusted network.

Figure 2 — Biometric enrolment process model

The small black disks with Key a in Figure 2 mean that an untrusted network may intervene at the points, i.e., a captured biometric sample, an intermediate biometric sample, and a processed biometric sample can be transferred through untrusted networks.

Figure 3 shows three cases of biometric verification process model where the storage subprocess stores a captured biometric reference, an intermediate biometric reference, and a processed biometric reference.



Key

1 storage subprocess which stores captured biometric samples

2 storage subprocess which stores intermediate biometric samples

3 storage subprocess which stores processed biometric samples

[a] Possible untrusted network.

Figure 3 — Biometric verification process model

The small black disks with Key a in Figure 3 mean the same as in Figure 2.

This document considers only biometric data flows in Figure 3 and does not consider any other non-biometric data flows such as auxiliary data in renewable biometrics.

5.2 BPU role and biometric capability class

5.2.1 Overview

When implementing biometric systems, the various biometric subprocesses that compose the system are often grouped together into components called biometric processing units (BPUs). The groupings reflect the physical and architectural details and the grouping of biometric subprocesses in the system implementation. Commonly used combinations of BPU functionalities are described and defined as BPU roles in this document. In principle, a BPU can include any grouping of biometric subprocesses but, in practice, BPUs commonly comprise biometric subprocesses that are sequential in the overall process flow for a biometric verification. BPU roles that equate to commonly used combinations of BPU functionality are defined in 5.2.2.

NOTE Only commonly used combinations of BPU functionalities are expressed as BPU role, i.e. there are BPUs whose functionalities cannot be expressed as a BPU role. For example, a BPU does not have an expression of a BPU role if it consists only of the decision subprocess. In such cases, declaration expression is used for the BPU function report. In policy-based authorization where authorization is done based on defined policies which include those on authentication, a service consuming the result of authentication can have a preference for the configuration for biometric verification. Biometric capability classes are specified for this purpose. For the benefits of BPU roles and biometric capability classes, see NOTE in 7.2.2.3, B.1.2.5, and B.1.3.5. See also B.3.

5.2.2 BPU role

5.2.2.1 General

Biometric products and systems are often implemented using one or more BPUs. The grouping of subprocesses into BPUs typically matches the physical structure of the product or system, e.g. a physical sensor unit may contain the sensor itself plus signal processing elements. The typical grouping of biometric subprocesses into a BPU defines the BPU role. The BPU roles described in this document reflect subprocess groupings that are commonly found in biometric products and systems. Although the concept of BPUs and BPU roles can be applied to multimodal and multi-biometric systems, the concept of BPU roles is not extended to multimodal and multi-biometric systems in this document.

5.2.2.2 BPU role for biometric enrolment

The all-BPU-enrolment role is a BPU role which contains all the subprocesses used in biometric enrolment, that is data capture, intermediate signal processing, final signal processing, and storage subprocesses. The all-BPU-enrolment role is presented in Figure 4. There is a variation where processed biometric sample is replaced with renewable biometric sample.

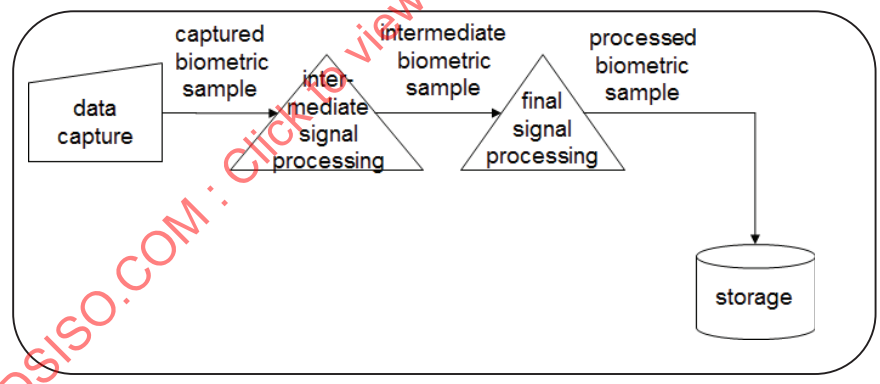


Figure 4 — All-BPU-enrolment role

The sensor BPU role is a BPU role which contains the data capture subprocess. It may contain the final signal processing subprocess and the intermediate signal processing subprocess. If it contains the final signal processing subprocess, then it shall also contain the intermediate signal processing subprocess. There are three patterns of sensor BPU role which are shown in Figure 5. There is a variation where processed biometric sample is replaced with renewable biometric sample.

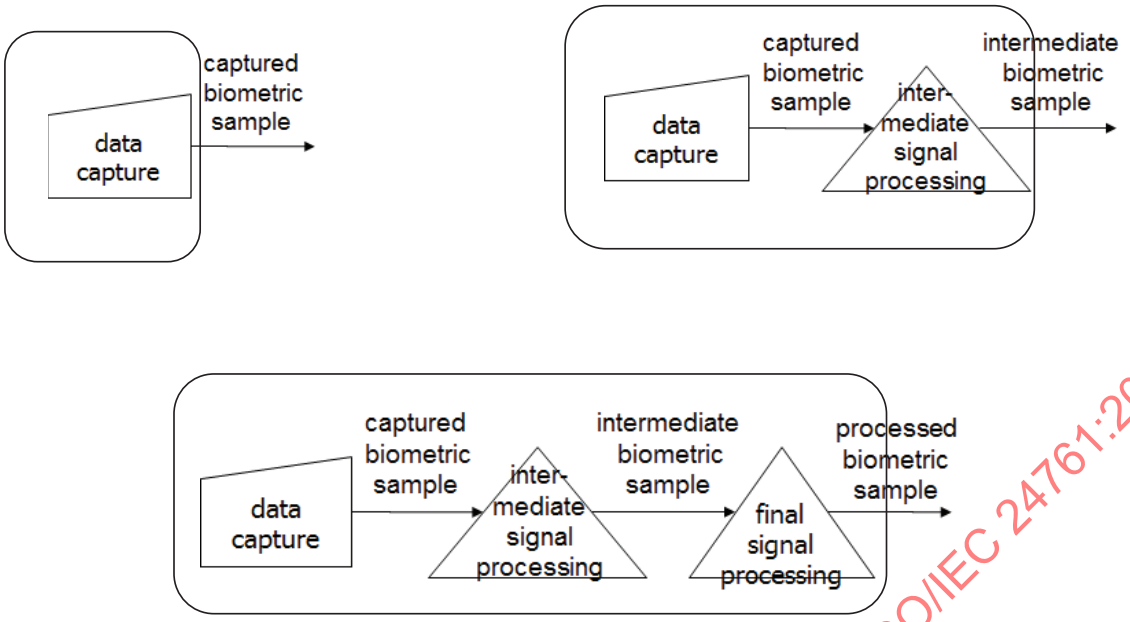


Figure 5 — Patterns of sensor BPU role

The storage-and-others-if-any BPU role is a BPU role which shall contain the storage subprocess but shall not contain the capture subprocess. It may contain the intermediate signal processing and the final signal processing subprocess. If it contains the intermediate signal processing subprocess, then it shall also contain the final signal processing subprocess. There are three patterns of storage-and-others-if-any BPU role which are depicted in Figure 6. There are variations where processed biometric sample is replaced with renewable biometric sample.

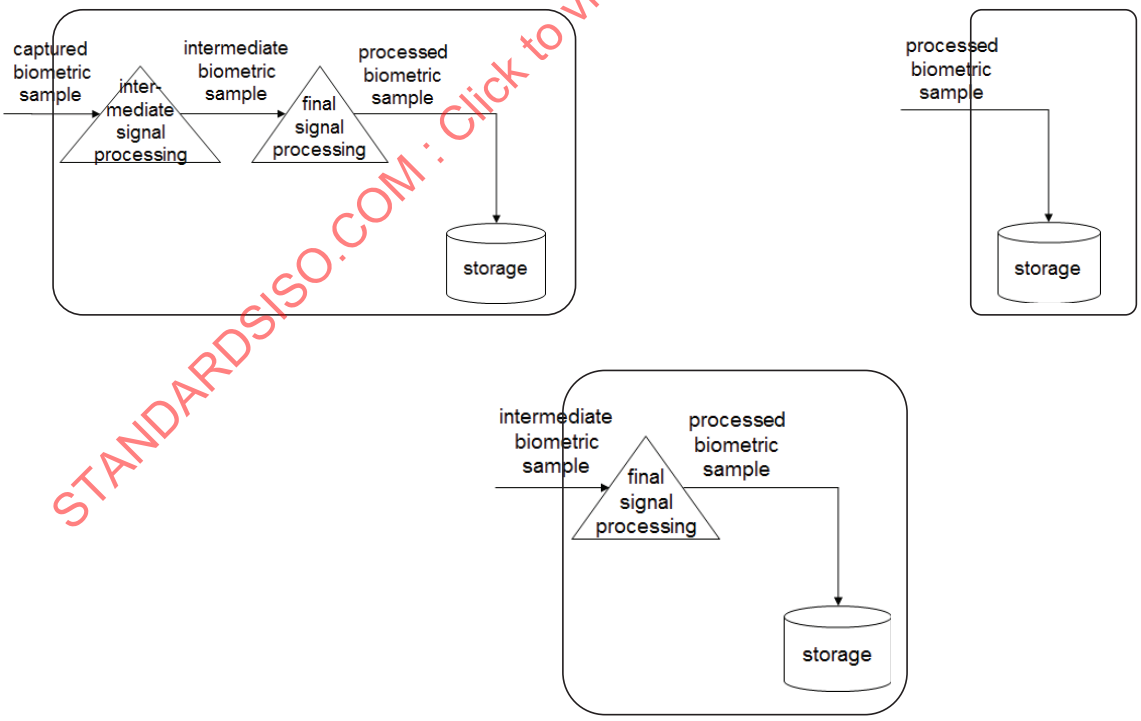


Figure 6 — Patterns of storage-and-others-if-any BPU role

5.2.2.3 BPU role for biometric verification

The all-BPU-verification role is a BPU role which contains all the subprocesses used in biometric verification. The all-BPU–verification role is presented in Figure 7. There are variations where processed biometric sample is replaced with renewable biometric sample.

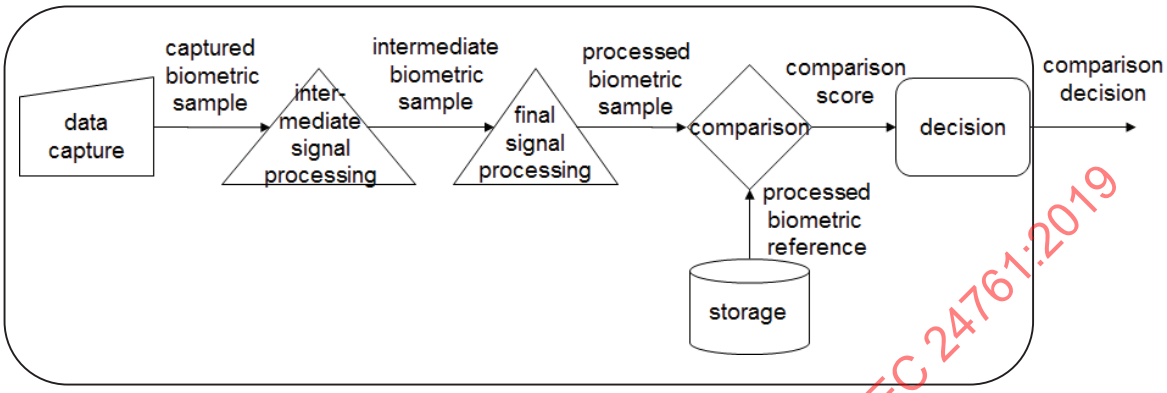


Figure 7 — All-BPU-verification role

The sensor BPU role, which is defined in 5.2.2.2, is used also in biometric verification.

The comparator-with-storage BPU role is a BPU role which shall contain the storage, comparison, and decision subprocesses. But it shall not contain the capture subprocess. It may contain the intermediate signal processing and the final signal processing subprocess. If it contains the intermediate signal processing subprocess, then it shall also contain the final signal processing subprocess. There are three patterns of comparator-with-storage BPU role which are depicted in Figure 8. There are variations where processed biometric sample is replaced with renewable biometric sample.

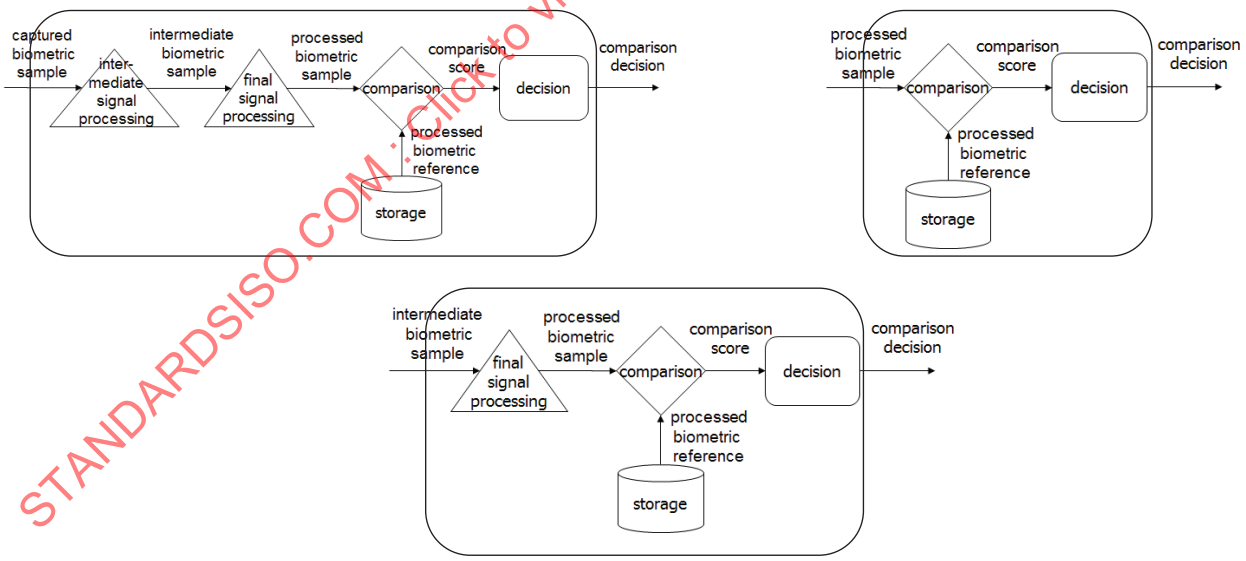


Figure 8 — Patterns of comparator-with-storage BPU role

The comparator BPU role is a BPU role which shall contain the comparison and decision subprocessess but shall not contain the storage subprocess. It may contain the data capture, intermediate signal processing, and final signal processing subprocess. If it contains a subprocess of the three, then it shall also contain the latter subprocess(es). There are four patterns of comparator BPU role which are shown in Figure 9. There are variations where processed biometric sample is replaced with renewable biometric sample.

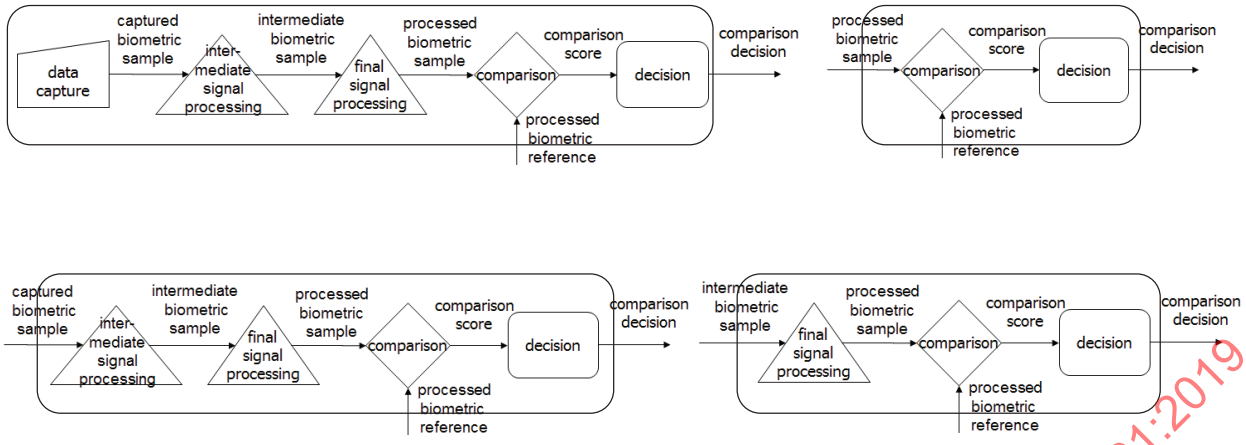


Figure 9 — Patterns of comparator BPU role

The storage BPU role is a BPU role which consists only of the storage subprocess. There is only one pattern for the storage BPU role as shown in Figure 10. There is a variation where processed the biometric sample is replaced with renewable biometric sample.

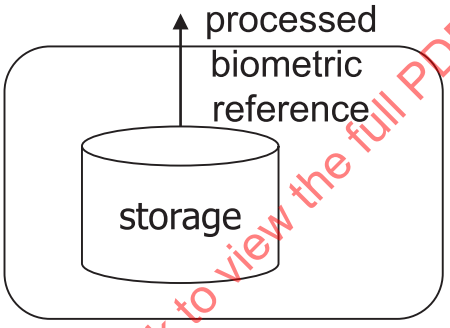


Figure 10 — Storage BPU role

5.2.3 Biometric capability class

5.2.3.1 General

In this document, BPU roles can be combined into a higher order structure representing the functional capability of the combination. This construct is named biometric capability class. The biometric capability classes defined in this document are described in 5.2.3.2 and 5.2.3.3.

5.2.3.2 Biometric capability classes for biometric enrolment

There are three classes. In the first and the second classes, all the subprocesses are executed at the client side while the biometric reference is stored at the server side in the last class. The all-in-one enrolment class is a biometric capability class which consists of an all BPU enrolment role.

Combination of sensor BPU role and sensor-and-others-if-any BPU role makes sensor-and-storage class. There are three cases of sensor-and-storage class as depicted in Figure 11. There are variations where processed biometric sample is replaced with renewable biometric sample.

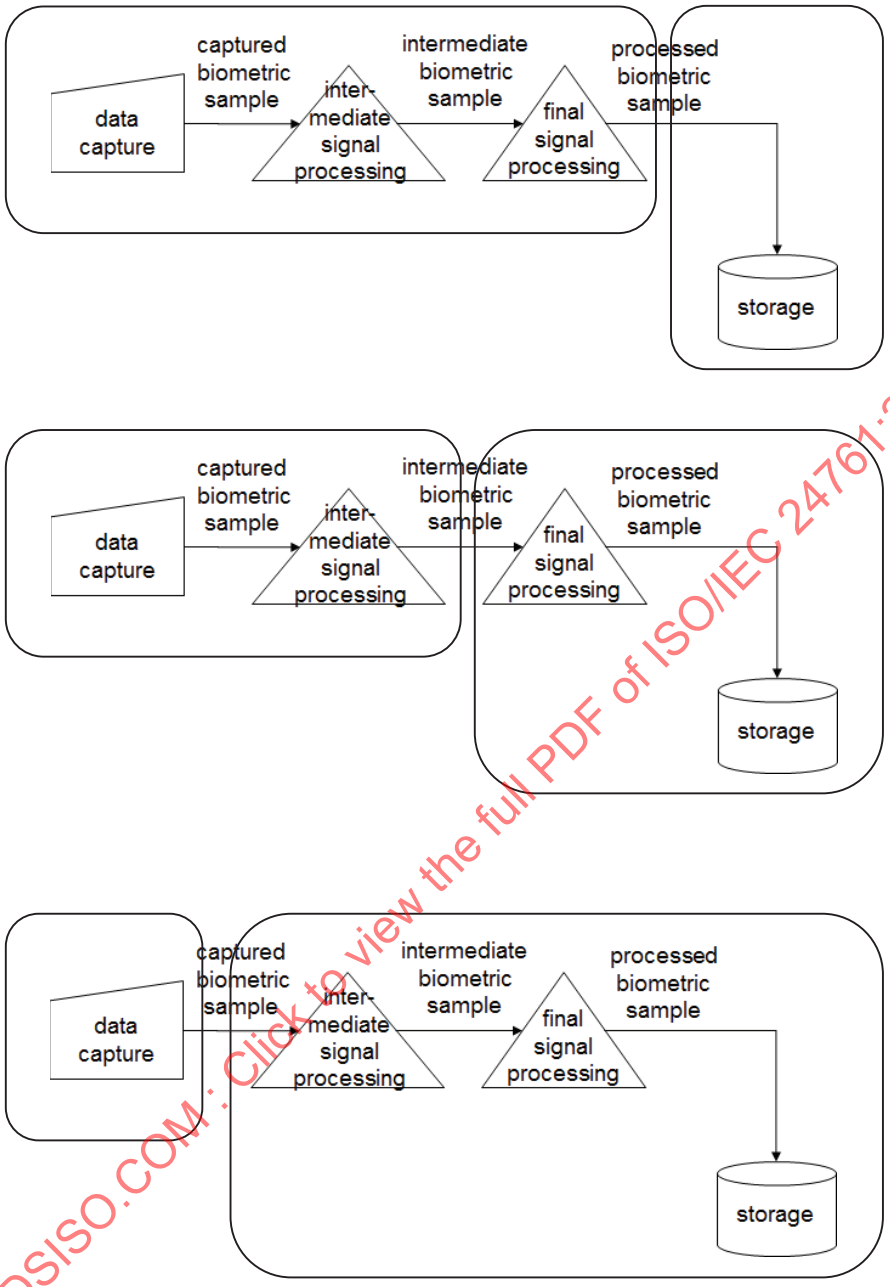


Figure 11 — Three cases of sensor-and-storage class

There are three cases for sensor-only-enrolment class as depicted in Figure 12 where the shaded parts are executed at the server side. There are variations where processed biometric sample is replaced with renewable biometric sample.

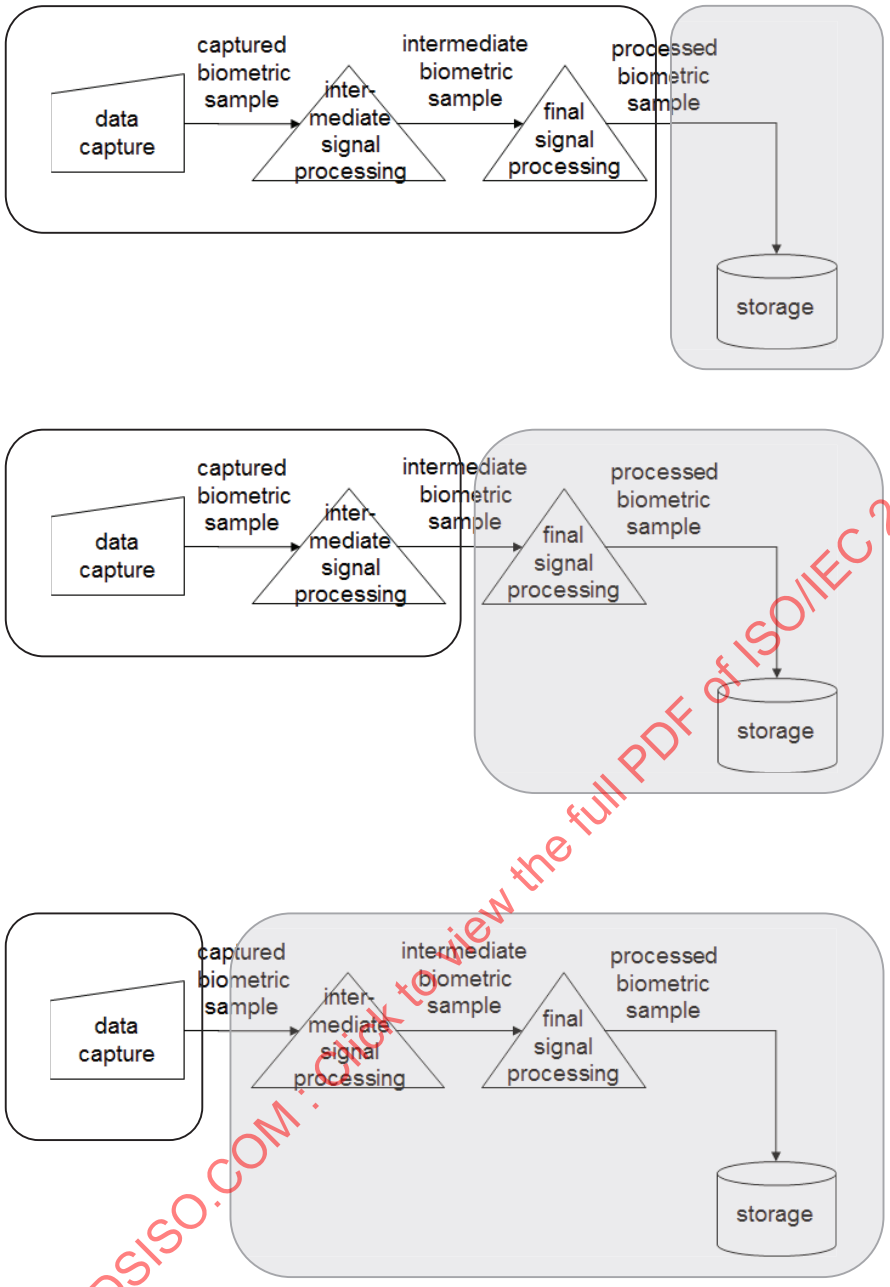


Figure 12 — Three cases of sensor-only-enrolment class

5.2.3.3 Biometric capability classes for biometric verification

There are four classes. In the first three classes, all the subprocesses are executed at the client side while the biometric comparison subprocesses is executed at the server side in the last class. The all-in-one verification class is a biometric capability class which consists of an all BPU verification role.

A combination of sensor BPU role and comparator-with-storage BPU role makes sensor-and-comparator class. There are three cases of sensor-and-comparator class as depicted in Figure 13. There are variations where processed biometric sample is replaced with renewable biometric sample.

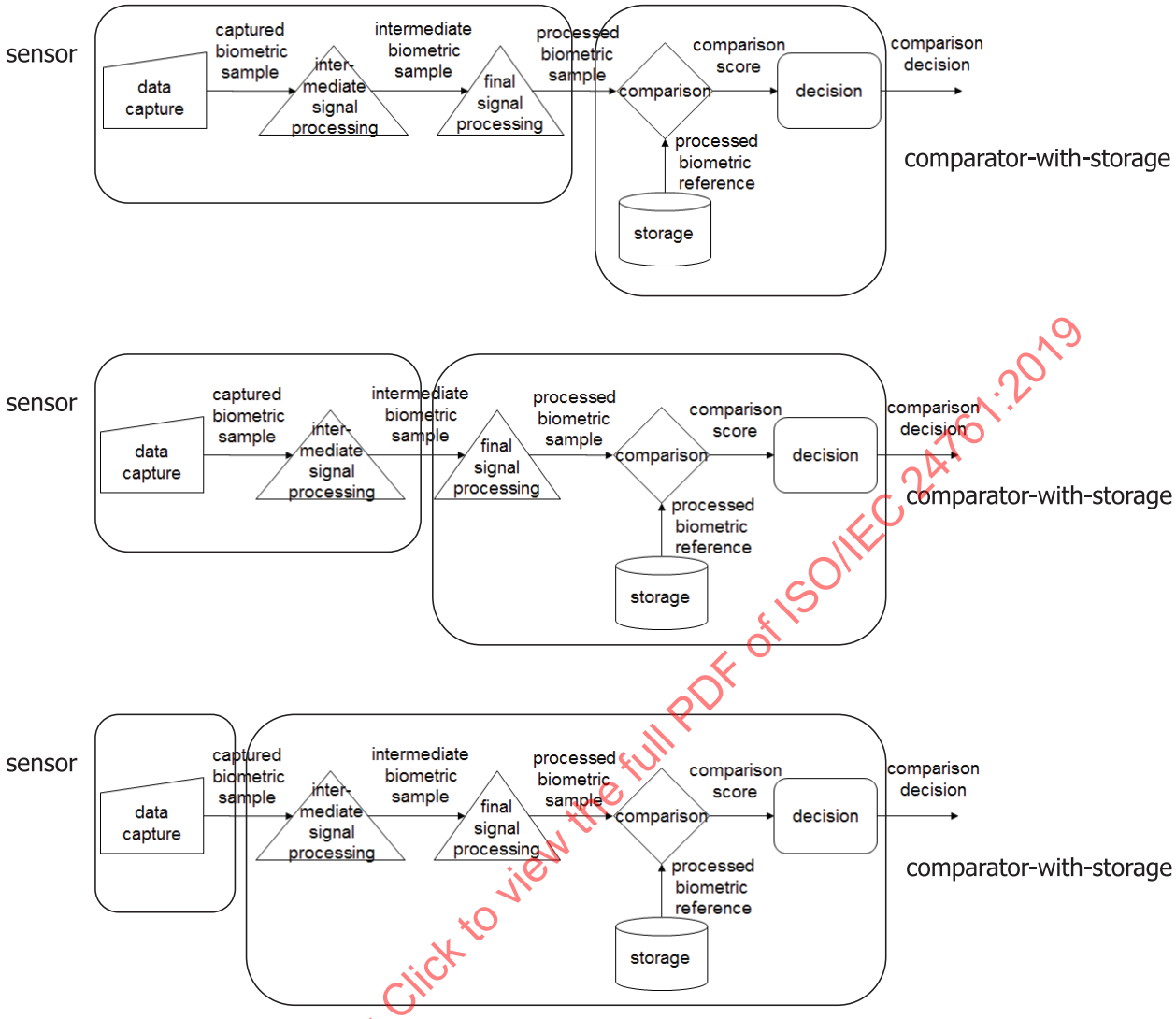


Figure 13 — Three cases of sensor-and-comparator class

A combination of storage BPU role and comparator BPU role makes storage-and-others class. There are four cases of storage-and-others class as shown in Figure 14. In three cases, the sensor BPU role also plays a role in the storage-and-others class. There are variations where processed the biometric sample is replaced with a renewable biometric sample.

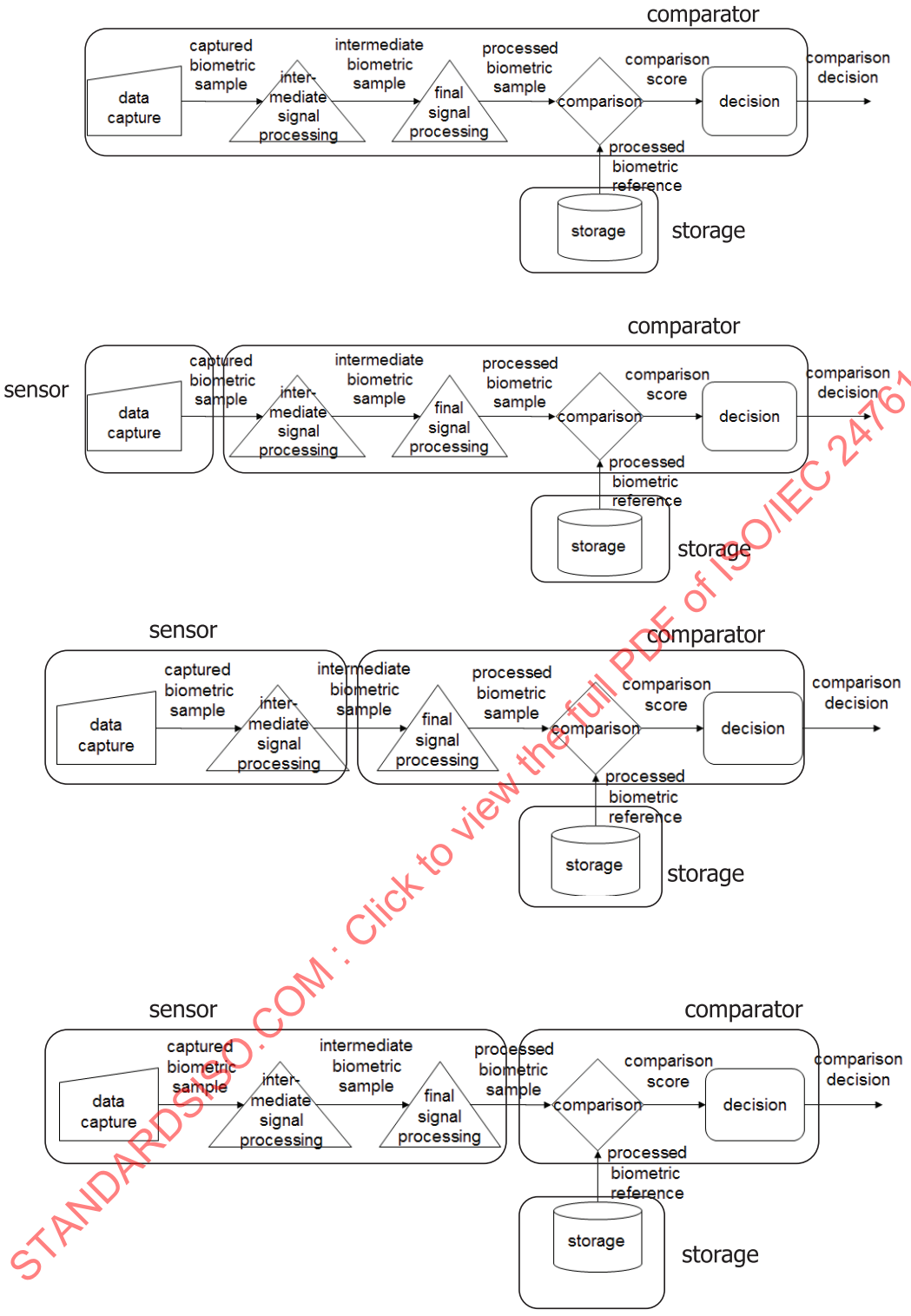


Figure 14 — Four cases of storage-and-others class

There are three cases for sensor-only-verification class as shown in Figure 15 where the shaded parts are executed at the server side. There are variations where the processed biometric sample is replaced with a renewable biometric sample.

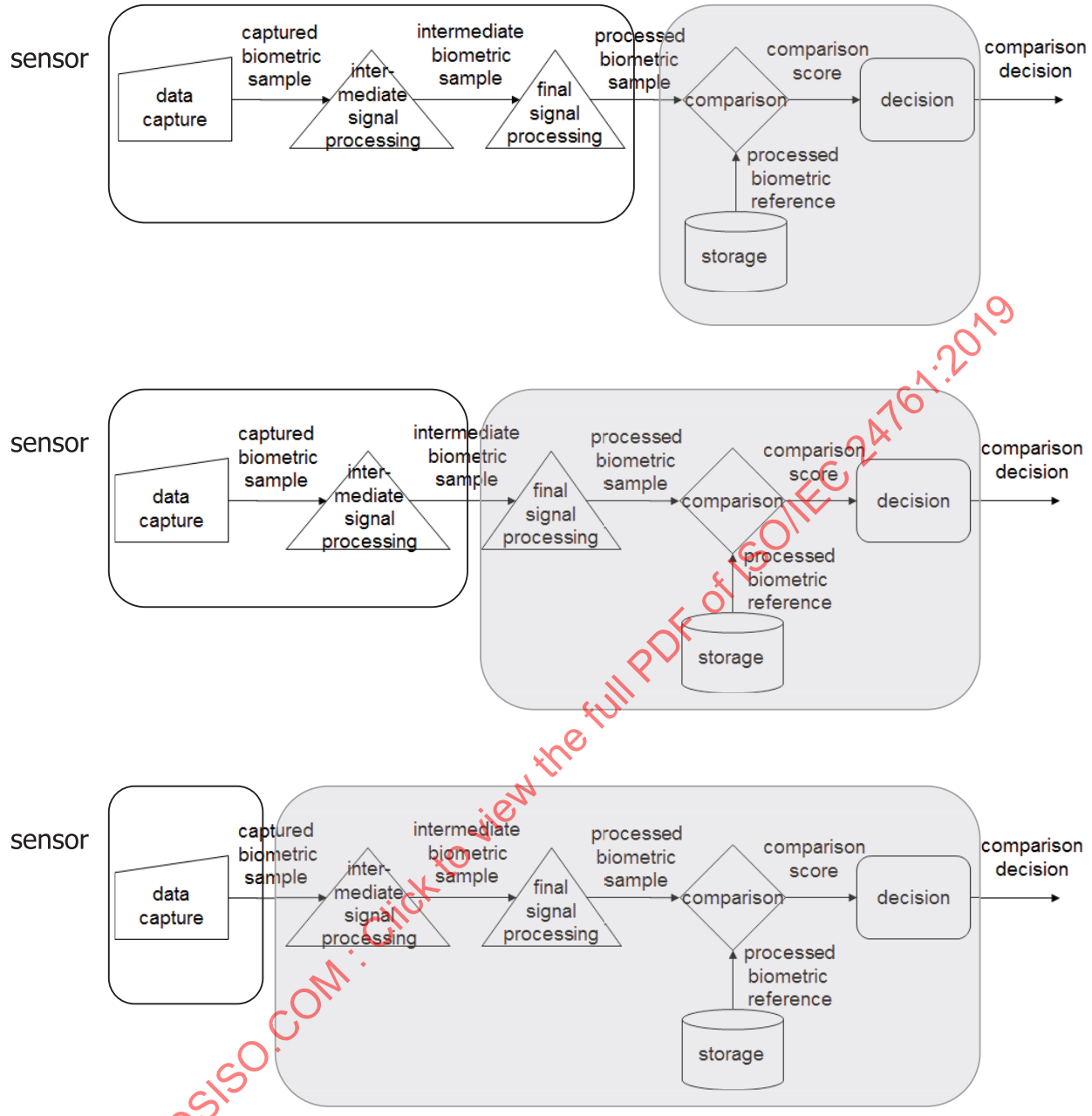


Figure 15 — Three cases of sensor-only- verification class

NOTE The scope of this document is restricted to biometric enrolment and verification. However, the technology can be applied to biometric identification where the configuration at the client side is one of the configurations in Figure 15. The processing at the server side of biometric identification can validate the processing at the client side with the ACBio instance generated at the client side.

5.3 Framework for use of ACBio

5.3.1 General

The technology in this document is designed to be used for remote biometric authentication for policy-based authorization, for example, in on-line bankings/shoppings (see B.3). In such cases, the provided service can change depending on how the authentication result can be trusted, that is the policy on authorization of the service provider. The policy may contain the configuration of biometric enrolment/verification process. In such a circumstance, the configuration is generally negotiated before the enrolment/authtentication itself, i.e., in the negotiation phase of enrolment/authentication, and also

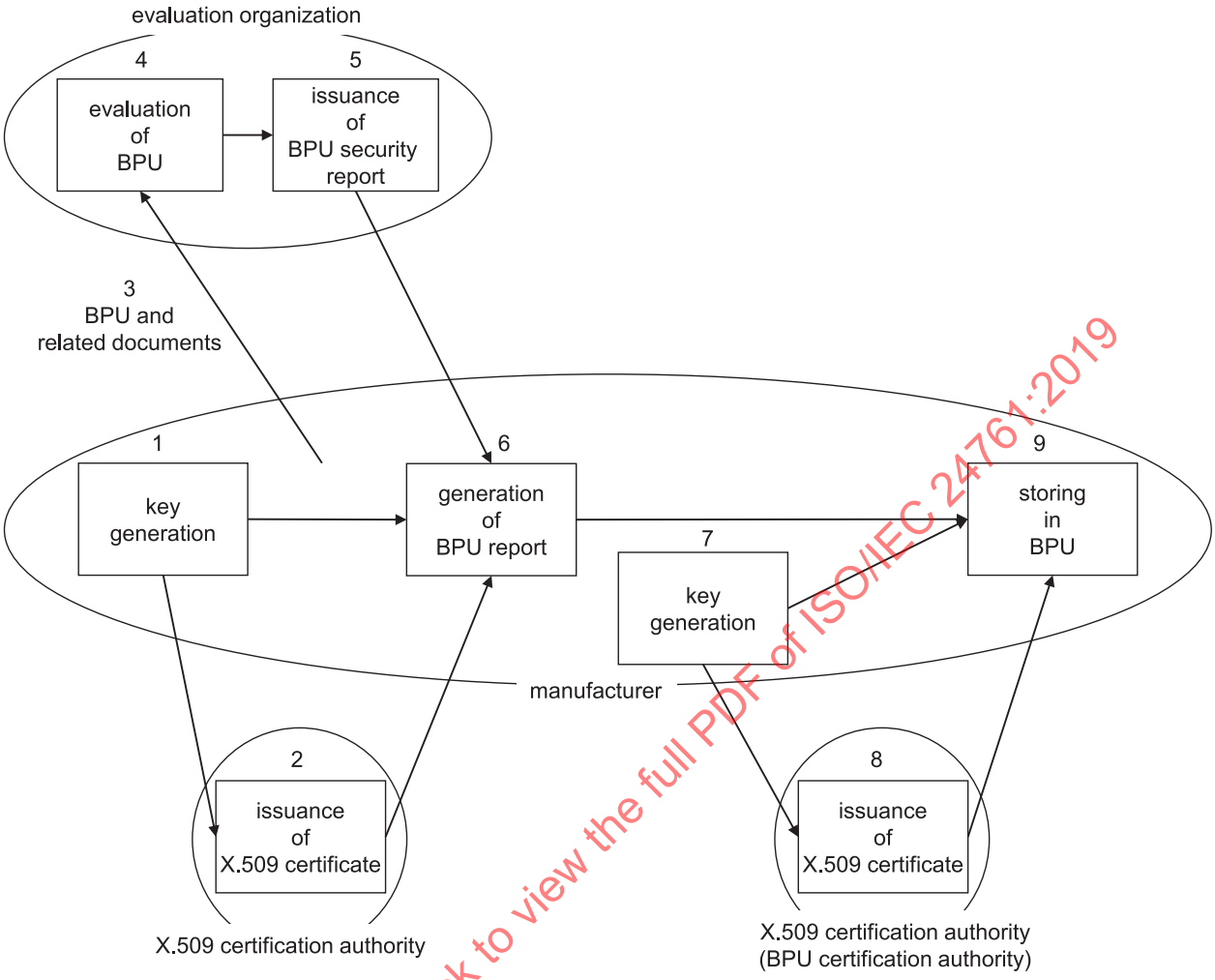
checked at the enrolment/authentication phase whether the configuration satisfies the condition which is already negotiated at the negotiation phase. The biometric capability classes are defined so that they are used in the enrolment/authentication phases. See B.1.2.5 and B.1.3.5 for examples of how biometric capability class is used.

ACBio validation information is provided by ACBio instances which are data structures that can provide assurance concerning the recognition performance and security capabilities of the BPU. ACBio instances contain X.509 certificates that authenticate the BPU and assure the information it provides. The X.509 certificates are provisioned through a production process conducted between the BPU manufacturer and the relevant certification authority.

ACBio gives information to the validator of a biometric verification process how it is done. For that to be done, in 5.3.2 to 5.3.5 give an overview of what shall be done in the production process of BPUs, the enrolment process of the biometric reference, the biometric verification process, and the validation process. The ASN.1 module specified in this document uses the X.509 certificate as public-key certificate but the concept itself is not limited to the X.509 certificate. In 5.3.2 to 5.3.5, the descriptions are generalized by using the term "public-key certificate" instead of "X.509 certificate".

5.3.2 Preparation in the production process

In the production process, the manufacturer of BPU prepares the BPU report and stores it into the BPU with the public key pair of the BPU and the public-key certificate of the public key. Figure 16 shows what shall be prepared in the production process of BPUs.



Key

1 key generation
2 Issuance of X.509 certificate
3 submission of BPU and related documents
4 evaluation of BPU
5 Issuance of BPU security report
6 generation of BPU report
7 key generation
8 issuance of X.509 certificate
9 storing of BPU report, key pairs, and X.509 certificate in BPU

Figure 16 — Preparation in the production process

The manufacturer of a BPU shall generate a public/private key pair (1 in Figure 16) and request the public-key certificate authority to issue a public-key certificate for the public key (2 in Figure 16) in advance. The private key is used to digitally sign the BPU report.

If security evaluation of the BPU is done by an evaluation organization (4 in Figure 16), a BPU security report (see 7.2.3) digitally signed using the private key of the evaluation organization is issued () in Figure 16), which is to be a part of the BPU report.

The manufacturer of the BPU shall generate the BPU report which is digitally signed with the private key of the manufacturer (6 in Figure 16). The BPU report provides trusted information about the BPU to

the validator when ACBio instance is checked. Next, the manufacturer shall generate a public key pair of the BPU (7 in Figure 16) and have a BPU certificate (see 7.1) issued by a BPU certification authority (8 in Figure 16). The BPU certificate is a public-key certificate of the public key of the BPU. The private key of the BPU is used later to digitally sign ACBio instances generated by the BPU.

The manufacturer should rotate keys to sign different BPU reports to anticipate the impact of any potential leakage of previously used keys, and it is additionally recommended to embed a different private key per product of the BPU, for example.

The BPU report or its referrer, the private key of the BPU, and the BPU certificate shall be stored in the BPU before shipping the product of the BPU. The BPU shall have a function to generate a digital signature for digitally signing ACBio instances using the BPU's private key.

Examples of preparation in the production process are provided in Annex B: B.1.2.2 for the STOC model and B.1.3.2 for the OCBC model.

5.3.3 Preparation in the subject enrolment process

In the enrolment process, the enrollee enrols his or her biometric reference, gets the BRT certificate corresponding to the biometric reference, and stores the biometric reference and the BRT certificate into a BPU.

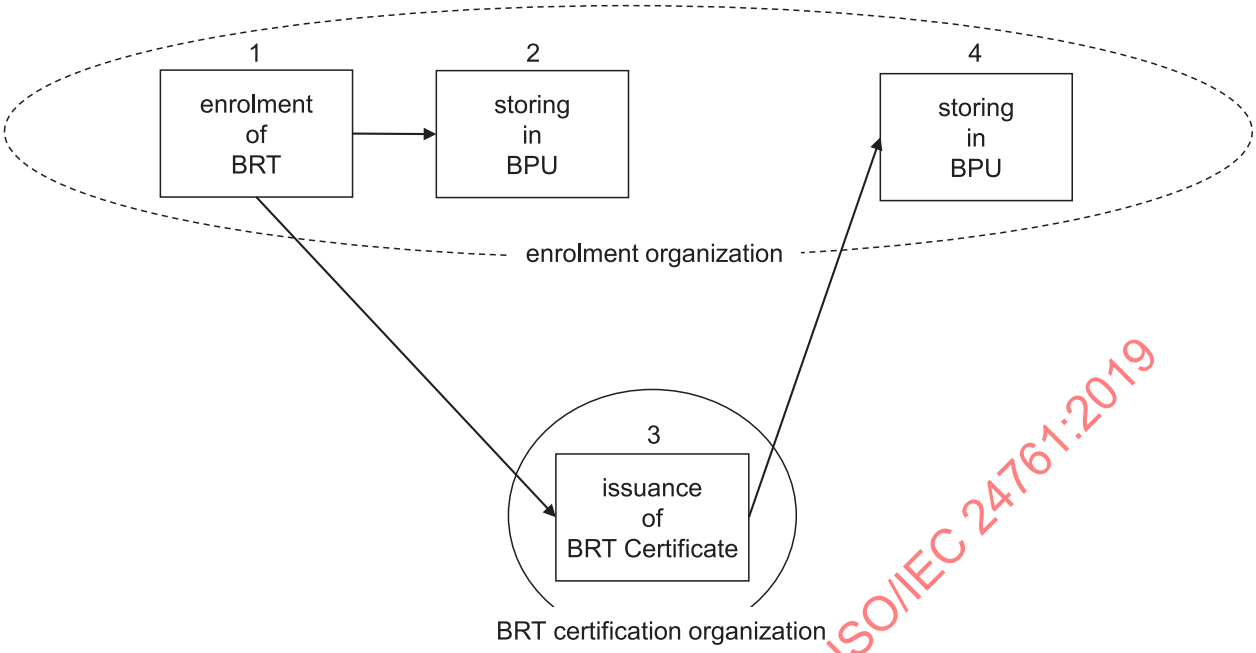
Figure 17 shows what shall be prepared in the enrolment process.

To use biometric verification, an enrollee shall enrol his or her biometric reference in advance under the control of an enrolment organization or without such a control depending on the use case (1 in Figure 17). Then the biometric reference is stored in a BPU with storage subprocess (2 in Figure 17).

To use ACBio for validation of a biometric verification process, a certificate of biometric reference called a BRT certificate (see Clause 8) shall be issued by a BRT certification organization (3 in Figure 17). The BRT certificate is digitally signed using the private key of the BRT certification organization. The BRT certificate or its referrer shall be stored in the BPU where the certified biometric reference is stored. The BRT and its BRT certificate shall be managed in the BPU as a couple.

During the enrolment process, ACBio instances should be generated to validate the reliability of the enrolment process. The validation with these ACBio instances may be done by the BRT certification organization to issue the BRT certificate as well as the validator of the biometric verification process.

NOTE When the ACBio instances are sent to the BRT certificate organization, the BRT certificate is not generated yet. Therefore, any ACBio instance sent to the BRT certificate contains no BRT certificate even if the BPU stores the biometric reference to which the BRT certificate is issued. Examples of preparation in the subject enrolment process are provided in B.1.2.3 for the STOC model and B.1.3.3 for the OCBC model.



Key

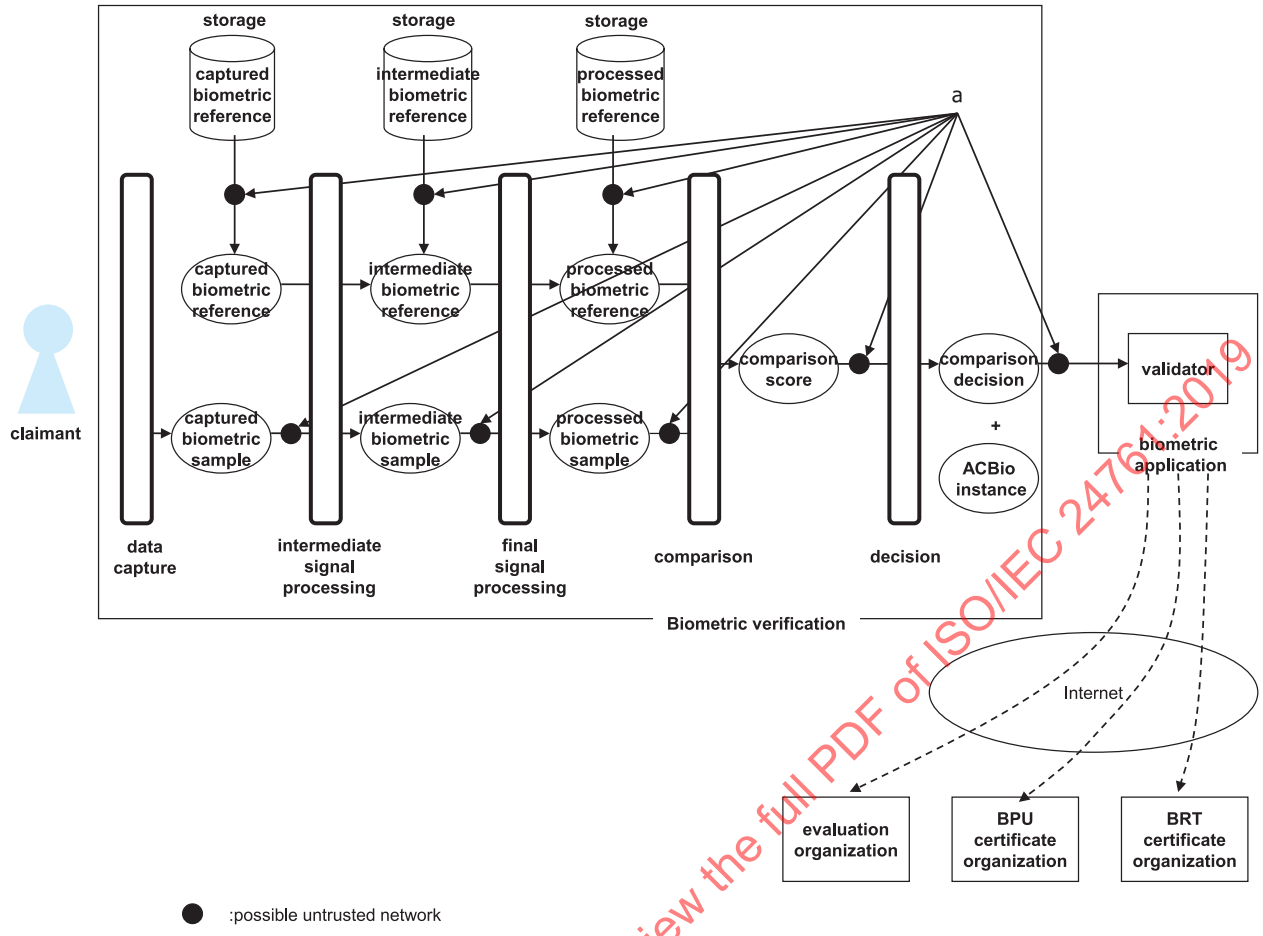
1 enrolment of BRT
2 storing of BRT in BPU
3 issuance of BRT certificate
4 storing of BRT certificate in BPU

Figure 17 — Preparation in the enrolment process

5.3.4 ACBio instance generation in the biometric verification process

ACBio applies the challenge-response mechanism to prevent the success of replay attackes. The validator shall send the challenge to the claimant and the claimant shall make all the BPUs that take a part in the biometric verification set the challenge into the ACBio instances they generate.

Figure 18 outlines the biometric verification process and its validation process when ACBio is used. Figure 18 shows three cases of biometric verification process models where the storage subprocess stores a captured biometric reference, an intermediate biometric reference, and a processed biometric reference.



a Possible untrusted network.

Figure 18 — Biometric verification process and its validation

The small black disk in Figure 18 means that an untrusted network may intervene at the points, i.e. biometric data in Figure 18 can be transferred through untrusted networks.

In a biometric verification process, each BPU shall pack the BPU certificate information (the BPU certificate itself or its referrer), the BPU report information (the BPU report itself or its referrer) into the data of type `ACBioContentInformation` (see Clause 6) to show later which subprocess(es) of the biometric verification are executed on which BPU. In addition, the BRT certificate information (the BRT certificate itself or its referrer) is also packed if the storage subprocess is in the BPU and stores the biometric reference used in the biometric verification process. The BRT certificate in an ACBio instance shows which biometric reference is used in the biometric verification. The data of type `ACBioContentInformation` shall also contain the challenge from the validator, called the control value, and the hash value(s) of the input/output biometric data to/from the BPU, which enables the validator to validate the consistency of the transmission of biometric data between BPUs.

By adding the digital signature to the data of type `ACBioContentInformation` with the private key of the BPU, the ACBio instance is generated.

Examples of ACBio instance generation are provided in B.1.2.4 for the STOC model and B.1.3.4 for the OCBC model.

5.3.5 Validation of biometric verification process with ACBio instances

In this ACBio framework, the validator receives not only the comparison decision, the result of biometric verification, but also the ACBio instance(s) with which the validator can validate the result of the executed biometric verification.

The validator can validate the authenticity and integrity of ACBio instance by verifying the digital signature with the BPU certificate. This shows also that the BPU has taken a part of the biometric verification process. The validator can obtain the result of security evaluation of the BPU by referring to the BPU report, and the authenticity of the biometric reference used in the biometric verification process by referring to the BRT certificate. The validator can also validate the consistency of the communication between BPUs and between the BPUs in the biometric verification process by checking the hash values in the biometric process blocks. By checking the control value in the ACBio instance(s), the validator can know whether there has been a replay attack or not. With all of these, the validator can decide the level of trust for the result of the executed biometric verification process.

NOTE In the cheking process, two types of indexes are used, one from BPU reports and the other given by the application program which calls BPUs. The integrity of the former is achieved by the digital signature given to BPU reports while that of the latter is achieved by the digital signature of ACBio instances.

If necessary, the validator can connect to relevant organizations such as the BPU certification authority, the evaluation organization, and the BRT certification organization, as shown in Figure 18.

Examples of validation of biometric verification process with ACBio instances are provided in B.1.2.5 for the STOC model and B.1.3.5 for the OCBC model.

6 ACBio instance

6.1 General

In Clauses 6 to 8, data structures are defined and explained. The definitions are specified in ASN.1 notation, which is in line with ISO/IEC 8824. The whole specification of the data structures is provided as an ASN.1 module in Annex A, which shall be used and encoded using BER in a system conforming to this document. Examples of ACBio instances are provided in B.1.2 for the STOC model and B.1.3 for the OCBC model.

An ACBio instance is data of ASN.1 type `ACBioInstance` as follows:

```
ACBioInstance ::= SEQUENCE {
        contentType CONTENT-TYPE.&id({ContentTypeACBio}),
        content CONTENT-TYPE.&Type
            ({ContentTypeACBio}{@contentType})}
```

The type `ACBioInstance` corresponds to the type `ContentInfo` of CMS. The latter is constrained by an extensible object set while the former is constrained by an object containing only `signedDataACBio`. The object of class `CONTENT-TYPE` is defined as follows:

```
ContentTypeACBio CONTENT-TYPE ::= {signedDataACBio }

signedDataACBio CONTENT-TYPE ::= {
        SignedDataACBio
        IDENTIFIED BY id-signedDataACBio }

id-signedDataACBio OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) signedDataACBio(1)}
```

`SignedDataACBio` is specified as follows:

```
SignedDataACBio ::= SIGNEDDATA { EncapsulatedContentInfoACBio }
```

The types `SignedDataACBio` specified above replaces the CMS types `SignedData` together with the following definitions:

```
SIGNEDDATA { EncapsulatedContentInfo } ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithms SET OF DigestAlgorithmIdentifier,
        encapContentInfo EncapsulatedContentInfo,
        certificates [0] IMPLICIT CertificateSet OPTIONAL,
        crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
        signerInfos SignerInfos}
```

The following types appeared in the above definition of `SIGNEDDATA` are imported from RFC 3852/5911: `CMSVersion`, `DigestAlgorithmIdentifier`, `SignerInfos`. `version` shall take the value as specified in RFC 3852/5911.

The type `EncapsulatedContentInfo` is a parameter in the above definition and is not imported from CMS. In the definition of the type `SignedDataACBio`, the following type replaces `EncapsulatedContentInfo`.

```
EncapsulatedContentInfoACBio ::= SEQUENCE {
        eContentType CONTENT-TYPE.&id({ContentTypeACBioContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
        ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypeACBioContentInfo}{@eContentType}))}
ContentTypeACBioContentInfo CONTENT-TYPE ::= {acbioContentInformation}
```

As in the above definition, the type `EncapsulatedContentInfoACBio` is constrained by an object containing a single object `acbioContentInformation` of the class `CONTENT-TYPE`. This object is defined as follows:

```
acbioContentInformation CONTENT-TYPE ::= {
       ACBioContentInformation
       IDENTIFIED BY id-acbioContentInformation }
id-acbioContentInformation OBJECT IDENTIFIER ::=
       {iso(1) standard(0) acbio(24761) contentType(2) acbioContent(3)}
```

Therefore, an ACBio instance is a data of type `ACBioInstance`, essentially the same as the CMS type `ContentInfo`, with the content of type `SignedDataACBio` on the content of type `ACBioContentInformation`.

Table 1 shows the structure of type `ACBioContentInformation`. `ACBioContentInformation` consists of five fields, version, BPU information block, control value, biometric process block, and BRT certificate information where the first four fields are mandatory. An ACBio instance shall have the last field if and only if the BPU contains the storage subprocess and stores the BRT used in the executed biometric verification. The digital signature of `SignedDataACBio` shall be generated with the private key of the BPU.

NOTE Any ACBio instance used in enrolment does not contain a BRT certificate because it is not issued yet.

Table 1 — ACBioContentInformation

ACBioContentInformation	
	Version
	BPU Information Block
	BPU Certificate Referrer Information
	BPU Report Information
	Control Value
	Biometric Process Block
	ProcessIndex[1]
	. .
	ProcessIndex[L]
	BPUIOExecutionInformation[1] (for input)
	. .
	BPUIOExecutionInformation[M] (for input)
	BPUIOExecutionInformation[1] (for output)
	. .
	BPUIOExecutionInformation[N] (for output)
	BRT Certificate Information

In ASN.1 notation, the type `ACBioContentInformation` is specified as follows:

```
ACBioContentInformation ::= SEQUENCE {
        version Version DEFAULT v2,
        bpuInformation BPUInformation,
        controlValue OCTET STRING (SIZE(16..256)),
        biometricProcess BiometricProcess,
        brtCertificateInformation BRTCertificateInformation OPTIONAL }
```

Version is the version of the format of `ACBioContentInformation`.

```
Version ::= INTEGER { v1(1), v2(2) }
```

The type `BPUInformation` is defined in 6.2. The detail of each type in `BPUInformation` is defined in Clause 7.

A control value is an octet string of 16 byte length which is sent from the validator and with which the validator can check to which validator's request the ACBio instance is generated to. It shall be set to `controlValue` field to make it infeasible to replay a biometric verification process.

The type `BiometricProcess` is defined in 6.3. The type `BRTCertificateInformation` is defined in 6.4. The detail of the type in `BRTCertificate`, which is used in `BRTCertificateInformation`, is defined in Clause 8.

6.2 BPU information block

BPU information block carries the static information of BPU, information which does not depend on each execution. This block is mandatory and consists of two components, BPU certificate referrer

information and BPU report information. ASN.1 type `BPUInformation` is defined for this block of information:

```
BPUInformation ::= SEQUENCE {

        bpuCertificateReferrerInformation BPUCertificateReferrerInformation OPTIONAL,

        bpuReportInformation BPUReportInformation}
```

BPU certificate referrer information of type `BPUCertificateReferrerInformation` is the referrer information to X.509 certificate for the public key of the BPU. If the ACBio instance contains the BPU certificate in the field of `certificates` in `SignedDataACBio`, BPU certificate referrer information can be omitted. BPU certificate is specified in 7.1.

```
BPUCertificateReferrerInformation ::= SEQUENCE {

        bpuCertificateReferrer URI,

        crlsReferrer URI OPTIONAL}


URI ::= VisibleString (SIZE(1..MAX))
```

BPU report information of type `BPUReportInformation` is the BPU report itself or its referrer. BPU report contains the information about functions implemented in the BPU and the information of the result of security evaluation of the BPU. BPU report is defined in 7.2.

```
BPUReportInformation ::= CHOICE {

        bpuReport  BPUReport,

        bpuReportReferrer URI}
```

6.3 Biometric process block

Biometric process block carries the runtime information of BPU, information which depends on each execution. This block consists of three components: `executedProcessIndexList`, `bpuInputExecutionInformationList`, and `bpuOuputExecutionInformationList`. `executedProcessIndexList` is the list of indexes of the subprocesses executed in the BPU when declaration expression is used. When role expression is used, `executedProcessIndexList` is the list of `executionIndex` (see 7.2.2.3) executed in the BPU. `bpuInputExecutionInformationList` contains the information on the input data to the BPU and `bpuOuputExecutionInformationList` contains the information on the output data from the BPU. If the BPU sends/receives data to/from other BPUs, then the corresponding components in this block are mandatory.

The ASN.1 type `BiometricProcess` is defined as follows:

```
BiometricProcess ::= SEQUENCE {

        executedProcessIndexList ProcessIndexList,

        bpuInputExecutionInformationList BPUIOExecutionInformationList OPTIONAL,

        bpuOuputExecutionInformationList BPUIOExecutionInformationList }

ProcessIndexList ::= SEQUENCE SIZE(1..MAX) OF ProcessIndex

BPUIOExecutionInformationList ::= SEQUENCE OF BPUIOExecutionInformation
```

`executedProcessIndexList` is a list of data of type `ProcessIndex`. When declaration expression is applied, this type is also used for `subprocessIndex` in type `FunctionDefinition` (See 7.2.2.2.2) which describes the function of a subprocess in a BPU. A BPU report in an ACBio instance contains as many

data of type FunctionDefinition as the number of subprocesses in the BPU. The subprocessIndex in the type FunctionDefinition which corresponds to the executed subprocess shall be set to the above executedProcessIndexList. When role expression is applied, this type ProcessIndex is used for executionIndex in type ExecutionInformation (See 7.2.2.3) which describes an execution pattern of a BPU. A BPU report in an ACBio instance contains as many data of type ExecutionInformation as the number of execution patterns in the BPU. The executionIndex in the type ExecutionInformation which corresponds to the executed execution pattern shall be set to the above executedProcessIndexList.

bpuInputExecutionInformationList consists of the elements of type BPUIOExecutionInformation as many as the input data to the BPU.

bpuOuputExecutionInformationList consists of the elements of type BPUIOExecutionInformation as many as the output data from the BPU.

For example, in case of a BPU which contains only the storage subprocess such as a STOC card, there is no bpuInputExecutionInformationList but bpuOuputExecutionInformationList with one element corresponding to the biometric reference from the storage subprocess.

The definition for type BPUIOExecutionInformation is given as follows:

```
BPUIOExecutionInformation ::= SEQUENCE {

        dataType DataType,

        bpuIOIndex IOIndex,

        subprocessIOIndex IOIndex,

        hash Hash}

Hash ::= SEQUENCE {

        algorithmIdentifier AlgorithmIdentifier,

        hashValue OCTET STRING}
```

BPUIOInformation consists of four components, dataType, bpuIOIndex, subprocessIOIndex, and hash.

dataType indicates the type of the input/output data to/from the BPU. The type DataType is defined in 7.2.2.2.3.

On execution, the application program, which utilizes the function of the BPU, shall uniquely assign an integer to each biometric data stream from/to BPUs. Such an integer given by the application program shall be set to bpuIOIndex. If another BPU generates an ACBio instance with the same integer in bpuIOIndex in the biometric process block, it means that there was a communication between these two BPUs In this way, the validator can reconstruct the data flow among BPUs.

The subprocessIOIndex of the corresponding element of the bpuInputStaticInformationList/ bpuOutputStaticInformationList in the BPU information block shall be set to subprocessIOIndex of BPUIOExecutionInformation. The combination of bpuIOIndex and subprocessIOIndex makes the connection between the data flow inside BPU and the data flow in the whole biometric verification process.

hash contains the hash value of the input/output data to/from the BPU and the identifier of the hash algorithm. The type AlgorithmIdentifier is imported from ISO/IEC 9594-2.

6.4 BRT certificate information

BRTcertificateInformation contains a list of the BRT certificates or the list of the referrer to each of the BRT certificate, as the following ASN.1 notation. A BRT certificate contains information about the biometric reference stored in BPU. An ACBio instance generated in biometric verification shall contain the BRT certificate Information if and only if the BPU contains the storage subprocess while an ACBio

instance generated in enrolment contains no BRT certificate even if the BPU contains the storage subprocess. A list of more than one element is used if multi-modal fusion biometric verification is used. BRT certificate is specified in Clause 8.

```
BRTCertificateInformation ::= CHOICE {

       brtCertificateList BRTCertificateList,

       brtCertificateReferrerList BRTCertificateReferrerList}

BRTCertificateList ::= SEQUENCE SIZE(1..MAX) OF BRTCertificate


BRTCertificateReferrerList ::= SEQUENCE SIZE(1..MAX) OF URI
```

7 Definition of components in BPUInformationBlock

7.1 BPU certificate

A BPU certificate is an X.509 certificate for the (public) key of BPU. The structure of BPU certificate is described in Table 2.

Table 2 — BPU certificate

field		content
tbsCertificate	version	as ordinary
	serialNumber	as ordinary
	signature	as ordinary
	validity	as ordinary
	issuer	a trusted third party or a public CA in the vendor which produces/sells the product of the BPU
	subject	identifier of the subject including the serial number of product, the product name and version of the product, and the name of the product vendor
	subjectPublicKeyInfo	as ordinary
	issuerUniqueIdentifier	as ordinary
	subjectUniqueIdentifier	as ordinary
	extensions	
signatureAlgorithm		as ordinary
signatureValue		as ordinary

The basic part of BPU certificate consists of nine fields; `version`, `serialNumber`, `signature`, `validity`, `issuer`, `subject`, `subjectPublicKeyInfo`, `issuerUniqueIdentifier`, and `subjectUniqueIdentifier`, all of which are the subfields of the field `tbsCertificate` of the type `Certificate` for X.509 certificate which is defined in ISO/IEC 9594-8. Here the field **issuer** is a trusted third party or a public CA in the vendor which produces/sells the product of the BPU. The field `subject` is the identifier whose description shall conform to ISO/IEC 9594-2 and shall include the serial number of product, the product name and version of the product, and the name of the product vendor. The serial number of product in the subject field shall be the leaf entry of the identifier. The product name and version shall be the entry next to the leaf. Other seven attributes in the basic field are used as ordinary.

The BPU certificate shall be stored in the `certificates` field of `SignedDataACBio` or `AuthenticatedDataACBio` type field of the ACBio instance, or the referrer to the BPU certificate shall be stored in `bpuCertificateReferrerInformation`.

BPUCertificateReferrerInformation consists of two components, bpuCertificateReferrer and crlsReferrer. In ASN.1 notation, BPUCertificateReferrerInformation is described as follows:

```
BPUCertificateReferrerInformation ::= SEQUENCE {

       bpuCertificateReferrer URI,

       crlsReferrer URI OPTIONAL}
```

7.2 BPUReportInformation

7.2.1 General

BPU report information contains information about function(s) implemented in the BPU and information on the security of the BPU. Either the BPU report itself or the referrer information to it shall be set in BPUReportInformation. In ASN.1 notation, BPUReportInformation is described as follows:

```
BPUReportInformation ::= CHOICE {

       bpuReport BPUReport,

       bpuReportReferrer URI}
```

BPUReport is defined in a similar way to ACBioInstance. BPUReport consists of two fields; the first field of fixed value of id-contentBPUReport and the second of type ContentBPUReport, which is a type of parameterized SIGNEDDATA with encapsulated content of type BPUReportContentInformation, which consists of two components, bpuFunctionReport and bpuSecurityReport. The signature shall be generated using the private key of the product vendor of the BPU.

NOTE The functions of and data flow in a BPU in enrolment mcanay be different from those in biometric verification. In such a case, two BPUReports maeany be prepared, one for enrolment, another for biometric verification. Otherwise, one BPUReport can be prepared for both enrolment and biometric verification. The latter case is noted in 7.2.2.2.1.

In ASN.1 notation, BPUReport is described as follows:

```
BPUReport ::= SEQUENCE {

       contentType CONTENT-TYPE.&id({ContentTypeBPUReport}),

       content CONTENT-TYPE.&Type

           ({ContentTypeBPUReport}{@contentType})}

ContentTypeBPUReport CONTENT-TYPE ::= { contentBPUReport }

ContentBPUReport ::= SIGNEDDATA { EncapsulatedContentInfoBPUReport }

EncapsulatedContentInfoBPUReport ::= SEQUENCE {

       eContentType CONTENT-TYPE.&id({ContentTypeBPUReportContentInfo}),

       eContent [0] EXPLICIT OCTET STRING

                ( CONTAINING CONTENT-TYPE.&Type

           ({ContentTypeBPUReportContentInfo}{@eContentType}))}

ContentTypeBPUReportContentInfo CONTENT-TYPE ::= { bpuReportContentInformation }

BPUReportContentInformation ::= SEQUENCE {

       bpuFunctionReport BPUFunctionReport,
```

```
        bpuSecurityReport BPUSecurityReport}
```

`BPUFunctionReport` and `BPUSecurityReport` are defined in 7.2.2 and 7.2.3.

The types `BPUReport` and `BPUReportContentInformation` are constrained with object sets containing a single object of class `CONTENT-TYPE`. These objects are defined as follows:

```
contentBPUReport CONTENT-TYPE ::= {

        ContentBPUReport

        IDENTIFIED BY id-contentBPUReport }

bpuReportContentInformation CONTENT-TYPE ::= {

        BPUReportContentInformation

        IDENTIFIED BY id-bpuReportContentInformation }
```

7.2.2 BPUFunctionReport

7.2.2.1 General

BPU function report contains information about function(s) implemented in the BPU and input/output data to/from the BPU. There are two ways of expression for BPU function report: declaration expression and role expression. The role expression is a new expression introduced in this document while the declaration expression is almost the same as in the first edition of this document. BPU function report for a BPU of BPU role may be expressed in both expressions. If a BPU is not of any BPU role, the BPU function report can be expressed only in declaration expression. In ASN.1 notation, `BPUFunctionReport` is described as follows:

```
BPUFunctionReport ::= CHOICE {

        bpuFunctionReportDeclaration BPUFunctionReportDeclaration,

        bpuFunctionReportRole BPUFunctionReportRole }
```

7.2.2.2 BPUFunctionReportDeclaration

7.2.2.2.1 General

`BPUFunctionReportDeclaration` includes the definition of function of BPU and may include functional performance level (quality) of the function. In ASN.1 notation, `BPUFunctionReportDeclaration` is described as follows:

```
BPUFunctionReportDeclaration ::= SEQUENCE {

        bpuSubprocessInformationList BPUSubprocessInformationList,

        bpuInputStaticInformationList BPUIOStaticInformationList OPTIONAL,

        bpuOutputStaticInformationList BPUIOStaticInformationList }

BPUSubprocessInformationList ::= SEQUENCE SIZE(1..MAX) OF BPUSubprocessInformation

BPUIOStaticInformationList ::= SEQUENCE SIZE(1..MAX) OF BPUIOStaticInformation
```

NOTE The specification of `BPUFunctionReportDeclaration` in this document is the same as that of `BPUFunctionReport` in ISO/IEC 24761:2009.

bpuSubprocessInformationList is a list of elements of type BPUSubprocessInformation as many as the number of the subprocesses implemented in the BPU. The type BPUSubprocessInformation is defined in 7.2.2.2.2.

bpuInputStaticInformationList is a list of elements of type BPUIOStaticInformation as many as the number of the input data to the BPU. bpuOutputStaticInformationList is a list of elements of type BPUIOStaticInformation as many as the number of the output data from the BPU. The type BPUIOStaticInformation is defined in 7.2.2.2.3.

In enrolment, storage subprocess shall output the hash value of the input of biometric sample which is to be stored as the biometric reference, and the hash value is to be set in the BRT certificate. Therefore, bpuOutputStaticInformationList shall have such a member if it is an expression for a BPU with storage subprocess in enrolment.

When the function of and data flow in a BPU in enrolment are different from those in biometric verification, the number of the elements in bpuSubprocessInformationList might not be equal to the number of the subprocesses in the BPU. It might be the sum of the number of the subprocesses in enrolment and that in biometric verification. In this case, bpuSubprocessInformationList is divided into two groups, one for enrolment and another for biometric verification. subprocessName of functionDefinition in a member of a group of bpuSubprocessInformationList might have the same value as the value of subprocessName of functionDefinition in a member in the other group but the value of the field subprocessIndex shall be different from that of the corresponding member of the list. If the bpuSubprocessInformaitonList is expressed as above, so are bpuInputStaticInformationList and bpuOutputStaticInformationList expressed in a similar way: there might be two members in the list where the value of subprocessIOIndex of one member is different from that of the other while the values of dataType are the same.

7.2.2.2.2 BPUSubprocessInformation

BPUSubprocessInformation contains information about the function and result of biometric performance evaluation for the subprocess, of type FunctionDefinition and PerformanceReport defined in 7.2.2.2.2 and 7.2.2.2.3 respectively.

```
BPUSubprocessInformation ::= SEQUENCE {

        functionDefinition FunctionDefinition,

        performanceReport  PerformanceReport OPTIONAL
```

FunctionDefinition

FunctionDefinition consists of seven components; subprocessName, subprocessIndex, inputIndexList, outputIndexList, and functionDescription.

subprocessName is of type SubprocessName and takes a value which represents the name of the subprocess.

To each subprocess in the BPU, the vendor of the product of the BPU shall assign a unique integer. subprocessIndex is such an index given to the subprocess.

A pair of components biometricType and biometricSubtype indicates the modality of biometric data processed in the subprocess. The types BiometricType and BiometricSubType are defined in ISO/IEC 19785-3. biometricType is mandatory if subprocessName does not take the value decision.

To each data stream which comes into or goes from any subprocess in the BPU, the vendor of the product of the BPU shall assign an integer. These integers shall be assigned uniquely within the BPU. If an input/output to/from a subprocess is given, then it is on one of the streams and is given the integer assigned to the data stream naturally. Each member of inputIndexList and outputIndexList are given in this way. Any subprocess except data capture shall have inputIndexList. Comparison subprocess shall have two members in inputIndexList. In the case of multibiometrics, inputIndexList/outputIndexList may have more than two members.

descriptionFunction is for supplementary description of the function of the subprocess.

The ASN.1 notation for type FunctionDefinition is given as follows:

```
FunctionDefinition ::= SEQUENCE {
        subprocessName SubprocessName,
        subprocessIndex ProcessIndex,
        biometricType BiometricType OPTIONAL,
        biometricSubtype BiometricSubtype OPTIONAL,
        inputIndexList IOIndexList OPTIONAL,
        outputIndexList IOIndexList,
        functionDescription OCTET STRING (SIZE(1..MAX)) OPTIONAL}
SubprocessName ::= ENUMERATED {
        data-capture(1),
        intermediate-signal-processing(2),
        final-signal-processing(3),
        storage(4),
        comparison(5),
        decision(6),
        sample-fusion(7),
        feature-fusion(8),
        score-fusion(9),
        decision-fusion(10),
        ...}
ProcessIndex ::= INTEGER (0..65535)
IOIndexList ::= SEQUENCE SIZE(1..MAX) OF IOIndex
IOIndex ::= INTEGER (0..65535)
```


PerformanceReport

Performance report contains information about performance evaluation of the BPU and is given with an ASN.1 type PerformanceReport. This type is defined as parameterized SIGNEDDATA with encapsulated content of type PerpformanceReportContentInformation, which consists of two components nameProduct of type Name and resultPerformanceTest of type ResultPerformanceTest. The value of nameProduct shall take the same as subject in the BPU certificate. Type ResultPerformanceTest has four optional components, testResultEnrol, testResultAcquire, testResultVerify and testResultExtension. Appropriate components for the BPU shall be set. If testResultVerify is set, testResultEnrol should be also set to show the performance of enrolment under which the performance of verification is accomplished. The last field is for extension. The types of other fields are defined in ISO/IEC 29120-1.

The signature in PerformanceReport shall be generated using the private key of the organization which evaluated the performance of the BPU.

NOTE ISO/IEC 29120-1:2015 only specifies machine readable data formats for testing and reporting of biometric recognition performance. The requirements are defined in the ISO/IEC 19795 series.

In ASN.1 notation, PerformanceReport is described as follows:

```
PerformanceReport::= SIGNEDDATA { EncapsulatedContentInfoPerformanceReport }

EncapsulatedContentInfoPerformanceReport ::= SEQUENCE {

        eContentType CONTENT-TYPE.&id({ContentTypePerformanceReportContentInfo}),

        eContent [0] EXPLICIT OCTET STRING

                ( CONTAINING CONTENT-TYPE.&Type

            ({ContentTypePerformanceReportContentInfo}{@eContentType}))}

ContentTypePerformanceReportContentInfo CONTENT-TYPE ::= { performanceReportContentInfor
mation }

PerformanceReportContentInformation ::=  SEQUENCE {

        nameProduct Name,

        resultPerformanceTest ResultPerformanceTest }

ResultPerformanceTest ::=  SEQUENCE {

        testResultEnrol   TestResultEnrol OPTIONAL,

        testResultAcquire TestResultAcquire OPTIONAL,

        testResultVerify  TestResultVerify OPTIONAL,

        testResultExtension   TestResultExtension OPTIONAL }

TestResultExtension ::= OCTET STRING (SIZE(1..MAX)) -- For extension
```

The type PerformanceReportContentInformation is constrained with an object set containing a single object of class CONTENT-TYPE. The object is defined as follows:

```
performanceReportContentInformation CONTENT-TYPE ::= {

        PerformanceReportContentInformation

        IDENTIFIED BY id-performanceReportContentInformation }
```

7.2.2.2.3 BPUIOStaticInformation

BPUIOStaticInformation is a data type which gives information about input/output to/from the BPU, and consists of two components; dataType and ioIndex.

```
BPUIOStaticInformation ::= SEQUENCE {

        dataType DataType,

        ioIndex IOIndex}
```

A pair of components biometricType and biometricSubtype indicates the modality of biometric data of input/output to/from the BPU. biometricType is mandatory if processedLevel of dataType

field does not take either `comparison-score` or `comparison-decision`. The types `BiometricType` and `BiometricSubType` are defined in ISO/IEC 19785-3.

`dataType` is of type `DataType` which consists of two components, `processedLevel` and `purpose`. The former takes a value which corresponds to one of captured data, intermediate data, processed data, comparison score, or comparison decision. The latter takes a value which corresponds to biometric reference or biometric sample.

There shall be the component `purpose` if the first component `processedLevel` takes the value `raw-data`, `intermediate-data`, `processed-data`, or `renewable-data`. The type `renewable-data` shall be applied to renewable biometric sample and renewable biometric reference. There shall not be the component `purpose` if the `processedLevel` takes the value `comparison-score`, `comparison-decision` or `hashed-data`.

NOTE Raw data is the old term for captured data used in ISO/IEC 24761: 2009.

An input/output to/from a BPU is one of input/output to/from a subprocess in the BPU. `ioIndex` shall be the value of the corresponding member of a certain data of type `FunctionDefinition` in BPU subprocess information.

```
DataType ::= SEQUENCE {
        processedLevel ProcessedLevel,
        purpose Purpose OPTIONAL}
ProcessedLevel ::= ENUMERATED {
        raw-data(1),
        intermediate-data(2),
        processed-data(3),
        comparison-score(4),
        comparison-decision (5),
        hashed-data(6),
        renewable-data(7),
        ...}
Purpose ::= ENUMERATED {
        reference(1),
        sample(2)}
```


7.2.2.3 BPUFunctionReportRole

Type `BPUFunctionReportRole` is a sequence of type `BPUFunctionReportRoleSingle`:

```
BPUFunctionReportRole ::= SEQUENCE OF BPUFunctionReportRoleSingle
```

Type `BPUFunctionReportRoleSingle` consists of two fields as follows:

```
BPUFunctionReportRoleSingle::= SEQUENCE {
        nameRole NameRole,
        executionInformationList ExecutionInformationList }
```

The first field `nameRole` of type `NameRole` is to show of which BPU role the BPU is, namely all–BPU-enrolment role, all–BPU-verification role, sensor BPU role, storage-and-others-if-any-BPU-role, comparator-with-capture BPU role, comparator BPU role, or storage BPU role. Type `NameRole` is defined as follows:

```
NameRole  ::= ENUMERATED {

        all-BPU-enrolment-role(1),

        all-BPU-verification-role(2),

        sensor-BPU-role(3),

        storage-and-others-if-any-BPU-role(4),

        comparator-with-storage-BPU-role(5),

        comparator-BPU-role(6),

        storage-BPU-role(7) }
```

The second field `executionInformationList` of type `ExecutionInformationList` in `BPUFunctionReportRoleSingle` shows the execution patterns of the BPU. If the BPU does not support multibiometrics, the list consists of only one member. If the BPU is of sensor BPU role which capture fingerprint and finger vein, then the list consists of two members: one for fingerprint, the other for finger vein. Each member of `ExecutionInformationList` is of type `ExecutionInformation`.

```
ExecutionInformationList ::= SEQUENCE SIZE(1..MAX) OF ExecutionInformation
```

Type `ExecutionInformation` consists of six fields. The first field `executionIndex` of type `ProcessIndex` (see 7.2.2.2.2.2) is the index assigned to this execution of all the execution patterns of the BPU. The second field `biometricType` together with the third field `biometricSubtype` shows the modality the execution processes. The fourth field `performanceReport` is information for performance evaluation. The fifth and sixth fields are information for the input and output of the BPU. They are the same as for `BPUFunctionReportDeclaration`.

```
ExecutionInformation ::= SEQUENCE {

        executionIndex ProcessIndex,

        biometricType BiometricType,

        biometricSubtype BiometricSubtype,

        performanceReport PerformanceReport OPTIONAL,

        bpuInputStaticInformationList BPUIOStaticInformationList OPTIONAL,

        bpuOutputStaticInformationList BPUIOStaticInformationList }
```

NOTE Role expression of BPU function report using `BPUFunctionReportRole` is a black-box approach while declaration expression using `BPUFunctionReportDeclaration` is a white-box approach. In role expression, there is no explicit information about subprocess(es) in the BPU. The left part of Figure 19 illustrates role expression of BPU function report for BPU of comparator-with-storage BPU role. The right part contains more information than on the left but the information contained on the left is equivalent to that on the right since there are more implicit information in role expression than in declaration expression. Accordingly, the validation of ACBio instances in role expression becomes simpler than that in declaration expression.

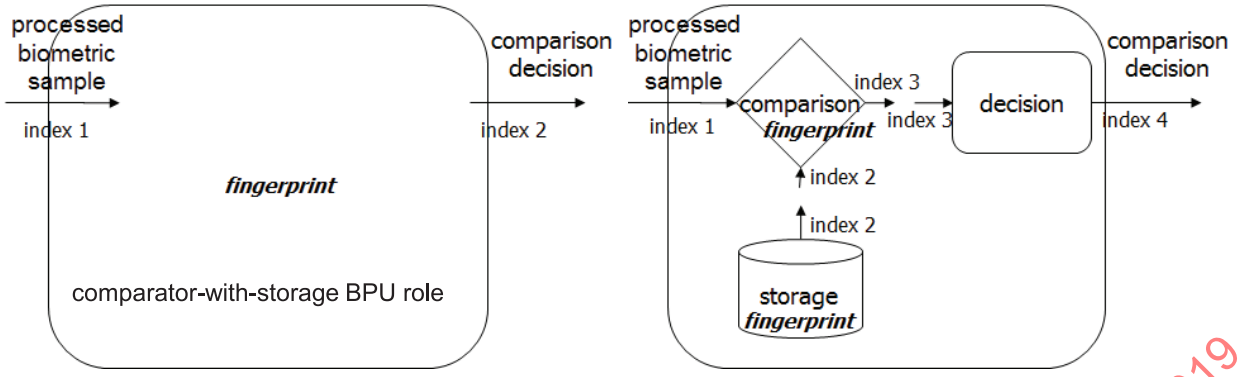


Figure 19 — Comparison between role expression and declaration expression for BPU of comparator-with-storage BPU role

7.2.3 BPUSecurityReport

`BPUSecurityReport` consists of three components, `cmSecurityReport`, `bpSecurityReport`, and `securityReportExtension`.

```
BPUSecurityReport ::= SEQUENCE {

        cmSecurityReport CMSecurityReport OPTIONAL,

        bpSecurityReport BPSecurityReport OPTIONAL,

        securityReportExtension SecurityReportExtension OPTIONAL}
```

`CMSecurityReport` carries information about security evaluation of the cryptographic module in the BPU. This type is defined as parameterized `SIGNEDDATA` with encapsulated content of type `CMSecurityReportContentInformation`, which consists of two components `nameProduct` of type `Name`, `level19790` of type `Level19790`. The value of `nameProduct` shall take the same value as **subject** in the BPU certificate. The second component `level19790` is to show the level in ISO/IEC 19790 which the BPU satisfies. The signature in `BPSecurityReport` shall be generated using the private key of the organization which evaluated the security of cryptographic module of the BPU.

In ASN.1 notation, `CMSecurityReport` is described as follows:

```
CMSecurityReport ::= SIGNEDDATA { EncapsulatedContentInfoCMSecurityReport }

EncapsulatedContentInfoCMSecurityReport::= SEQUENCE {

        eContentType CONTENT-TYPE.&id({ContentTypePerformanceCMSecurityReportContentInfo}),

        eContent [0] EXPLICIT OCTET STRING

                ( CONTAINING CONTENT-TYPE.&Type

            ({ContentTypePerformanceCMSecurityReportContentInfo }{@eContentType}))}

ContentTypePerformanceCMSecurityReportContentInfo CONTENT-TYPE ::= { cmSecurityReportConte
ntInformation }

CMSecurityReportContentInformation ::=  SEQUENCE {

        nameProduct Name,

        level19790 Level19790 }

Level19790 ::= ENUMERATED {
```

```
        level1 (1),

        level2 (2),

        level3 (3),

        level4 (4)  }
```

NOTE The four security levels of cryptographic modules and evaluation requirements for them are defined in ISO/IEC 19790.

The type `CMSecurityReportContentInformation` is constrained with an object set containing a single object of class `CONTENT-TYPE`. The object is defined as follows:

```
cmSecurityReportContentInformation CONTENT-TYPE ::= {

        CMSecurityReportContentInformation

    IDENTIFIED BY id-cmSecurityReportContentInformation }

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

Type `BPSecurityReport` carries information about security evaluation of the biometric processing in the BPU. This type is defined as parameterized `SIGNEDDATA` with encapsulated content of type `BPSecurityReportContentInformation`, which consists of three components `nameProduct` of type `Name`, `requirements` of type `Requirements`, and optional `resultPerformanceTest` of type `ResultPerformanceTest` defined in 7.2.2.2.3. The value of `nameProduct` shall take the same value as `subject` in the BPU certificate. The second component `requirements` is to show a set of security requirements which the BPU satisfies. The third component is used if the performance evaluation was done together with security evaluation of the BPU. The signature in `BPSecurityReport` shall be generated using the private key of the organization which evaluated the security of the BPU.

In ASN.1 notation, `BPSecurityReport` is described as follows:

```
BPSecurityReport ::= SIGNEDDATA { EncapsulatedContentInfoBPSecurityReport }

EncapsulatedContentInfoBPSecurityReport::= SEQUENCE {

        eContentType CONTENT-TYPE.&id({ContentTypePerformanceBPSecurityReportContentInfo}),

        eContent [0] EXPLICIT OCTET STRING

                ( CONTAINING CONTENT-TYPE.&Type

            ({ContentTypePerformanceBPSecurityReportContentInfo }{@eContentType}))}

ContentTypePerformanceBPSecurityReportContentInfo CONTENT-TYPE ::= { bpSecurityReportConte
ntInformation }

BPSecurityReportContentInformation ::=  SEQUENCE {

        nameProduct Name,

        requirements Requirements,

        resultPerformanceTest ResultPerformanceTest OPTIONAL }

Requirements ::= SEQUENCE OF Requirement

Requirement ::= OBJECT IDENTIFIER
```

NOTE An example of `Requirement` is an identifier assigned to a Protection Profile which is a set of security requirements specified to the product category (see ISO/IEC 15408-1). At the time of making this document, there is a trend to specify Protection Profiles for biometric products. The ISO/IEC 19989 series is also standardized to make security evaluation of biometric products based on ISO/IEC 15408 possible.

The type BPSecurityReportContentInformation is constrained with an object set containing a single object of class CONTENT-TYPE. The object is defined as follows:

```
bpSecurityReportContentInformation CONTENT-TYPE ::= {

        BPSecurityReportContentInformation

    IDENTIFIED BY id-bpSecurityReportContentInformation }
```

The last component SecurityReportExtension is for extension.

```
SecurityReportExtension ::= OCTET STRING (SIZE(1..MAX)) -- For extension
```

8 BRT certificate

8.1 General

BRT certificate is a non-X.509 certificate to the biometric reference issued by a certain BRT certification organization. It contains information about the biometric reference stored in a BPU, such as the issuer and validity period, etc.

Type BRTCertificate is defined similarly to BPUReport. BRTCertificate consists of two fields; the first field of fixed value of id-contentBRTCertificate and the second of type ContentBRTCertificate, which is a type of parameterized SIGNEDDATA with encapsulated content of type BRTCContentInformation. The signature shall be generated using the private key of the BRT certification organization.

In ASN.1 notation, BRTCertificate is described as follows:

```
BRTCertificate ::= SEQUENCE {
        contentType CONTENT-TYPE.&id({ContentTypeBRTCertificate}),
        content         CONTENT-TYPE.&Type({ContentTypeBRTCertificate}{@contentType})}
ContentTypeBRTCertificate CONTENT-TYPE ::= { contentBRTCertificate }
ContentBRTCertificate ::= SIGNEDDATA { EncapsulatedContentInfoBRTCertificate }
EncapsulatedContentInfoBRTCertificate ::= SEQUENCE {
        eContentType CONTENT-TYPE.&id({ContentTypeBRTCertificateContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
        ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypeBRTCertificateContentInfo}{@eContentType}))}
ContentTypeBRTCertificateContentInfo CONTENT-TYPE ::= { brtcContentInformation }
```

The following attributes bind the type ContentBRTCertificate to id-contentBRTCertificate, and the type BRTCContentInformation to id-brtcContentInformation.

The types BRTCertificate and EncapsulatedContentInfoBRTCertificate are constrained with object sets containing a single object of class CONTENT-TYPE defined as follows:

```
contentBRTCertificate CONTENT-TYPE ::= {

        ContentBRTCertificate

        IDENTIFIED BY id-contentBRTCertificate   }

id-contentBRTCertificate  OBJECT IDENTIFIER ::=

        {iso(1) standard(0) acbio(24761) contentType(2) brtCertificate(6)}
```

```
brtcContentInformation CONTENT-TYPE ::= {

        BRTCContentInformation

        IDENTIFIED BY id-brtcContentInformation  }

id-brtcContentInformation OBJECT IDENTIFIER ::=

        {iso(1) standard(0) acbio(24761) contentType(2) brtcContent(7)}
```

8.2 BRTCContentInformation

`BRTCContentInformation` is expressed with CBEFF BIR (Biometric Information Record) which is specified in ISO/IEC 19785-1. `BRTCContentInformation` consists of two parts, `sbhForBRTC` and `bdbForBRTC`. To express the former, SBH (Standard Biometric Header) of CBEFF is applied. The latter is a newly defined BDB (Biometric Data Block) format for BRT certificate.

Type `BRTCContentInformation` is described as follows:

```
BRTCContentInformation ::= SEQUENCE {

        sbhForBRTC SBHForBRTC,

        bdbForBRTC BDBForBRTC}
```

`sbhForBRTC` is of type `SBHForBRTC` and has nine elements: `version`, `brtcIndex`, `brtcValidityPeriod`, `biometricType`, `biometricSubtype`, `brtQuality`, `bdbEncryptionOptions`, `bdbIntegrityOptions`, and `bdbFormatForBRC`. The types for these elements are specified in ISO/IEC 19785-3 besides `version`, `bdbEncryptionOptions` and `bdbIntegrityOptions`.

`version` is used for specifying the version of format of `SHBForBRTC`.

`brtcIndex` indicates the index of the BRT certificate.

`brtcValidityPeriod` contains the validity period of the BRT certificate.

`biometricType` together with `biometricSubtype` shows the modality of the biometric reference.

`brtQuality` contains the quality of the biometric reference.

`bdbEncryptionOptions` and `bdbIntegrityOptions` are encryptions option and integrity options of type `BOOLEAN`, and shall be set to `FALSE`. `bdbFormatForBRTC` indicates the format owner and format type of `BDBForBRTC`.

In ASN.1 notation, `SBHForBRTC` is described as follows:

```
SBHForBRTC ::= SEQUENCE {

        version Version DEFAULT v1,

        brtcIndex BIRIndex,

        brtcValidityPeriod BDBValidityPeriod,

        biometricType BiometricType,

        biometricSubtype BiometricSubtype OPTIONAL,

        brtQuality Quality OPTIONAL,

        bdbEncryptionOptions BOOLEAN(FALSE),

        bdbIntegrityOptions BOOLEAN(FALSE),
```

```
        bdbFormatForBRTC BDBFormat}
```

`bdbForBRTC` is of type `BDBForBRTC` and has nine elements; `version`, `issuerAndSerialNumberBRTC`, `originalBDBHashList`, `originalBIRReferrer`, `originalBIRpatronFormat`, `originalBDBPosition`, `userInformation`, `pkiCertificateInformation`, and `enrolmentACBioInstances`.

`version` is the version of the format `BDBForBRTC`.

Optional field `issuerAndSerialNumberBRTC` of type `IssuerAndSerialNumberBRTC` imported from RFC 3852/5911 is a pair of information, the issuer of the BRT certificate and the unique serial number issued by the issuer. OCSP may be applied to check the validity of the BRT certificate.

`originalBDBHashList` is a list of `Hash`. `Hash` contains two fields, hash value and algorithm identifier of hash algorithm. The former is the hash value of the biometric reference. If `originalBDBHashList` contains more than one element, they are of a single biometric reference and of different hash algorithm.

`originalBIRReferrer` is the referrer to the original BIR.

`originalBIRpatronFormat` is the patron format of the original BIR, which is of type `PatronFormat`.

`originalBDBPosition` indicates the position of the biometric reference corresponding to this BRT certificate in the original BDB.

`userInformation` is an optional field of type `UserInformation` which contains identifier, name, and unique identifier of a person whose biometric reference is the object of the BRT certificate.

`pkiCertificateInformation` is an optional field and contains information about X.509 public-key certificate of the user, the serial number of the certificate, the name of issuer, and the unique identifier of the certificate. This field links BRT certificate to X.509 certificate.

`enrolmentACBioInstances` is an optional list of ACBio instances generated at the enrolment of the biometric reference.

In ASN.1 notation, `BDBForBRTC` is described as follows:

```
BDBForBRTC ::= SEQUENCE {
        version Version DEFAULT v1,
        issuerAndSerialNumberBRTC IssuerAndSerialNumber OPTIONAL,
        originalBDBHashList HashList,
        originalBIRReferrer URI OPTIONAL,
        originalBIRPatronFormat PatronFormat,
        originalBDBPosition INTEGER,
        userInformation UserInformation OPTIONAL,
        pkiCertificateInformation PKICertificateInformation OPTIONAL,
        enrolmentACBioInstances SequenceOfACBioInstances OPTIONAL}
HashList ::= SEQUENCE SIZE(1..MAX)  OF Hash
UserInformation ::= SEQUENCE {
        userIdentifier OCTET STRING,
        userName Name OPTIONAL,
        userUniqueIdentifier UniqueIdentifier OPTIONAL}
```

```
PKICertificateInformation ::= SEQUENCE {
       pkiCertificateSerialNumber CertificateSerialNumber,
       pkiCertificateIssuerName Name OPTIONAL,
       pkiCertificateIssuerUniqueIdentifier UniqueIdentifier OPTIONAL}
SequenceOfACBioInstances ::= SEQUENCE SIZE(1..MAX) OF ACBioInstance
```

8.3 Format Owner and Format Type values

The Format Owner for the `BDBForBRTC` shall use the value 0102 hex (258 decimal). The Format Type for the `BDBForBRTC` shall use the value 0001 hex (1 decimal), which has been registered as the value for the `BDBForBRTC`.

Thus, te resulting ASN.1 Object Identifier value for the `BDBForBRTC` is:

```
{iso registration-authority cbeff(19785) organization(0) iso-iec-jtc1-SC27 (258)
       bdbs(0) biometric-reference-template-certificate (1)}
```

Annex A
(normative)

ASN.1 module for ACBio

```
AuthenticationContextForBiometrics {iso(1) standard(0) acbio(24761) module(1) acbio(2) version2(2)}
DEFINITIONS AUTOMATIC TAGS ::= BEGIN
IMPORTS
-- ASN.1 Module AlgorithmInformation in RFC 5912
       AlgorithmIdentifier
       FROM AlgorithmInformation-2009 {iso(1) identified-organization(3) dod(6) internet(1) security(5)
            mechanisms(5) pkix(7) id-mod(0) id-mod-algorithmInformation-02(58)}
-- RFC 5280  revised as RFC 5912
       Certificate, CertificateList, CertificateSerialNumber, Name, UniqueIdentifier
       FROM PKIX1Explicit-2009 { iso(1) identified-organization(3) dod(6) internet(1)
           security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51) }

-- RFC 5755  revised as RFC 5912
       AttributeCertificate
       FROM PKIXAttributeCertificate-2009 { iso(1) identified-organization(3) dod(6) internet(1)
          security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-attribute-cert-02(47) }
-- ISO/IEC 19785 Common Biometric Exchange Formats Framework
       BiometricType, BiometricSubtype, BIRIndex,
       BDBValidityPeriod,Quality, BDBFormat, PatronFormat
       FROM CBEFF-DATA-ELEMENTS {iso standard 19785 modules(0)
           types-for-cbeff-data-elements(1) }
-- ISO/IEC 29120-1 Machine readable test data for biometric testing and reporting
       TestResultEnrol, TestResultAcquire, TestResultVerify
       FROM MachineReadableBiometricTestingAndReportingTestReport {
       iso(1) standard(0) MRTDBTR(29120) testReport(1) module(1) rev(0) }
-- RFC 3852 Cryptographic Message Syntax revised as RFC 5911
       CMSVersion, DigestAlgorithmIdentifier, SignerInfos,
       IssuerAndSerialNumber, CertificateSet, RevocationInfoChoices,
       CONTENT-TYPE
       FROM CryptographicMessageSyntax2009{
           iso(1) member-body(2) us(840) rsadsi(113549)
           pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004-02(41) };
ACBioInstance ::= SEQUENCE {
       contentType CONTENT-TYPE.&id({ContentTypeACBio}),
       content CONTENT-TYPE.&Type
           ({ContentTypeACBio}{@contentType})}
```

```
ContentTypeACBio CONTENT-TYPE ::= {signedDataACBio}
SignedDataACBio ::= SIGNEDDATA { EncapsulatedContentInfoACBio }
EncapsulatedContentInfoACBio ::= SEQUENCE {
       eContentType CONTENT-TYPE.&id({ContentTypeACBioContentInfo}),
       eContent [0] EXPLICIT OCTET STRING
        ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypeACBioContentInfo}{@eContentType}))}
ContentTypeACBioContentInfo CONTENT-TYPE ::= {acbioContentInformation}
ACBioContentInformation ::= SEQUENCE {
       version Version DEFAULT v2,
       bpuInformation BPUInformation,
       controlValue OCTET STRING (SIZE(16..256)),
       biometricProcess BiometricProcess,
       brtCertificateInformation BRTCertificateInformation OPTIONAL}
Version ::= INTEGER { v1(1), v2(2) }
BPUInformation ::= SEQUENCE {
       bpuCertificateReferrerInformation BPUCertificateReferrerInformation
           OPTIONAL,
       bpuReportInformation BPUReportInformation}
BPUCertificateReferrerInformation ::= SEQUENCE {
       bpuCertificateReferrer URI,
       crlsReferrer URI OPTIONAL}
URI ::= VisibleString (SIZE(1..MAX))
BPUReportInformation ::= CHOICE {
       bpuReport BPUReport,
       bpuReportReferrer URI}
BPUReport ::= SEQUENCE {
       contentType CONTENT-TYPE.&id({ContentTypeBPUReport}),
       content CONTENT-TYPE.&Type
           ({ContentTypeBPUReport}{@contentType})}
ContentTypeBPUReport CONTENT-TYPE ::= {contentBPUReport }
ContentBPUReport ::= SIGNEDDATA { EncapsulatedContentInfoBPUReport }
EncapsulatedContentInfoBPUReport ::= SEQUENCE {
       eContentType CONTENT-TYPE.&id({ContentTypeBPUReportContentInfo}),
       eContent [0] EXPLICIT OCTET STRING
            ( CONTAINING CONTENT-TYPE.&Type
           ({ContentTypeBPUReportContentInfo}{@eContentType}))}
ContentTypeBPUReportContentInfo CONTENT-TYPE ::= { bpuReportContentInformation }
BPUReportContentInformation ::= SEQUENCE {
    bpuFunctionReport BPUFunctionReport,
    bpuSecurityReport BPUSecurityReport}
BPUFunctionReport ::= CHOICE {
       bpuFunctionReportDeclaration BPUFunctionReportDeclaration,
       bpuFunctionReportRole BPUFunctionReportRole }
BPUFunctionReportDeclaration  ::= SEQUENCE {
```

```
        bpuSubprocessInformationList BPUSubprocessInformationList,
        bpuInputStaticInformationList BPUIOStaticInformationList OPTIONAL,
        bpuOutputStaticInformationList BPUIOStaticInformationList }
BPUSubprocessInformationList ::= SEQUENCE SIZE(1..MAX) OF BPUSubprocessInformation
BPUSubprocessInformation ::= SEQUENCE {
        functionDefinition FunctionDefinition,
        performanceReport PerformanceReport OPTIONAL}
FunctionDefinition ::= SEQUENCE {
        subprocessName SubprocessName,
        subprocessIndex ProcessIndex,
        biometricType BiometricType OPTIONAL,
        biometricSubtype BiometricSubtype OPTIONAL,
        inputIndexList IOIndexList OPTIONAL,
        outputIndexList IOIndexList,
        functionDescription OCTET STRING (SIZE(1..MAX)) OPTIONAL}
SubprocessName ::= ENUMERATED {
        data-capture(1),
        intermediate-signal-processing(2),
        final-signal-processing(3),
        storage(4),
        comparison(5),
        decision(6),
        sample-fusion(7),
        feature-fusion(8),
        score-fusion(9),
        decision-fusion(10),
        ...}
ProcessIndex ::= INTEGER (0..65535)
IOIndexList ::= SEQUENCE SIZE(1..MAX) OF IOIndex
IOIndex ::= INTEGER (0..65535)

PerformanceReport ::= SIGNEDDATA { EncapsulatedContentInfoPerformanceReport }
EncapsulatedContentInfoPerformanceReport ::= SEQUENCE {
        eContentType CONTENT-TYPE.&id({ContentTypePerformanceReportContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
                ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypePerformanceReportContentInfo}{@eContentType}))}
ContentTypePerformanceReportContentInfo CONTENT-TYPE ::= { performanceReportContentInformation }
PerformanceReportContentInformation ::=  SEQUENCE {
        nameProduct Name,
        resultPerformanceTest ResultPerformanceTest }
ResultPerformanceTest ::=  SEQUENCE {
        testResultEnrol   TestResultEnrol OPTIONAL,
        testResultAcquire TestResultAcquire OPTIONAL,
        testResultVerify  TestResultVerify OPTIONAL,
```

```
        testResultExtension    TestResultExtension OPTIONAL }
TestResultExtension ::= OCTET STRING (SIZE(1..MAX)) -- For extension

BPUIOStaticInformationList ::= SEQUENCE SIZE(1..MAX) OF BPUIOStaticInformation
BPUIOStaticInformation ::= SEQUENCE {
        dataType DataType,
        ioIndex IOIndex}
DataType ::= SEQUENCE {
        processedLevel ProcessedLevel,
        purpose Purpose OPTIONAL}
ProcessedLevel ::= ENUMERATED {
        raw-data(1),
        intermediate-data(2),
        processed-data(3),
        comparison-score(4),
        comparison-result(5),
        hashed-data(6),
        renewable-data(7),
        ...}
Purpose ::= ENUMERATED {
        reference(1),
        sample(2)}
BPUFunctionReportRole ::= SEQUENCE OF BPUFunctionReportRoleSingle
BPUFunctionReportRoleSingle::= SEQUENCE {
        nameRole NameRole,
        executionInformationList ExecutionInformationList }
NameRole  ::= ENUMERATED {
        all-BPU-enrolment-role (1),
        all-BPU-verification-role(2),
        sensor-BPU-role(3),
        storage-and-others-if-any-BPU-role(4),
        comparator-with-storage-BPU-role(5),
        comparator-BPU-role(6),
        storage-BPU-role(7) }
ExecutionInformationList ::= SEQUENCE SIZE(1..MAX) OF ExecutionInformation
ExecutionInformation ::= SEQUENCE {
        executionIndex ProcessIndex,
        biometricType BiometricType,
        biometricSubtype BiometricSubtype,
        performanceReport PerformanceReport OPTIONAL,
        bpuInputStaticInformationList BPUIOStaticInformationList OPTIONAL,
        bpuOutputStaticInformationList BPUIOStaticInformationList }
BPUSecurityReport ::= SEQUENCE {
        cmSecurityReport CMSecurityReport OPTIONAL,
        bpSecurityReport BPSecurityReport OPTIONAL,
        securityReportExtension SecurityReportExtension OPTIONAL}
CMSecurityReport ::= SIGNEDDATA { EncapsulatedContentInfoCMSecurityReport }
EncapsulatedContentInfoCMSecurityReport::= SEQUENCE {
```

```
        eContentType CONTENT-TYPE.&id({ContentTypePerformanceCMSecurityReportContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
                ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypePerformanceCMSecurityReportContentInfo }{@eContentType}))}
ContentTypePerformanceCMSecurityReportContentInfo CONTENT-TYPE ::= { cmSecurityReportContentInformation }
CMSecurityReportContentInformation ::=  SEQUENCE {
        nameProduct Name,
        level19790 Level19790 }
Level19790 ::= ENUMERATED {
        level1 (1),
        level2 (2),
        level3 (3),
        level4 (4)  }
BPSecurityReport ::= SIGNEDDATA { EncapsulatedContentInfoBPSecurityReport }
EncapsulatedContentInfoBPSecurityReport::= SEQUENCE {
        eContentType CONTENT-TYPE.&id({ContentTypePerformanceBPSecurityReportContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
                ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypePerformanceBPSecurityReportContentInfo }{@eContentType}))}
ContentTypePerformanceBPSecurityReportContentInfo CONTENT-TYPE ::= { bpSecurityReportContentInformation }
BPSecurityReportContentInformation ::=  SEQUENCE {
        nameProduct Name,
        requirements Requirements,
        resultPerformanceTest ResultPerformanceTest OPTIONAL }
Requirements ::= SEQUENCE OF Requirement
Requirement ::= OBJECT IDENTIFIER
SecurityReportExtension ::= OCTET STRING (SIZE(1..MAX)) -- For extension
BiometricProcess ::= SEQUENCE {
        executedProcessIndexList ProcessIndexList,
        bpuInputExecutionInformationList BPUIOExecutionInformationList OPTIONAL,
        bpuOuputExecutionInformationList BPUIOExecutionInformationList }
ProcessIndexList ::= SEQUENCE SIZE(1..MAX) OF ProcessIndex
BPUIOExecutionInformationList ::= SEQUENCE SIZE(1..MAX) OF BPUIOExecutionInformation
BPUIOExecutionInformation ::= SEQUENCE {
        dataType DataType,
        bpuIOIndex IOIndex,
        subprocessIOIndex IOIndex,
        hash Hash}
Hash ::= SEQUENCE {
        algorithmIdentifier AlgorithmIdentifier,
        hashValue OCTET STRING}
BRTCertificateInformation ::= CHOICE {
```

```
        brtCertificateList BRTCertificateList,
        brtCertificateReferrerList BRTCertificateReferrerList}
BRTCertificateList ::= SEQUENCE SIZE(1..MAX) OF BRTCertificate
BRTCertificateReferrerList ::= SEQUENCE SIZE(1..MAX) OF URI
BRTCertificate ::= SEQUENCE {
        contentType CONTENT-TYPE.&id({ContentTypeBRTCertificate}),
        content
           CONTENT-TYPE.&Type({ContentTypeBRTCertificate}{@contentType})}
ContentTypeBRTCertificate CONTENT-TYPE ::= { contentBRTCertificate }
ContentBRTCertificate ::= SIGNEDDATA { EncapsulatedContentInfoBRTCertificate }
EncapsulatedContentInfoBRTCertificate ::= SEQUENCE {
        eContentType CONTENT-TYPE.&id({ContentTypeBRTCertificateContentInfo}),
        eContent [0] EXPLICIT OCTET STRING
         ( CONTAINING CONTENT-TYPE.&Type
            ({ContentTypeBRTCertificateContentInfo}{@eContentType}))}
ContentTypeBRTCertificateContentInfo CONTENT-TYPE ::= { brtcContentInformation }
BRTCContentInformation ::= SEQUENCE {
        sbhForBRTC SBHForBRTC,
        bdbForBRTC BDBForBRTC}
SBHForBRTC ::= SEQUENCE {
        version  Version DEFAULT v1,
        brtcIndex BIRIndex,
        brtcValidityPeriod BDBValidityPeriod,
        biometricType BiometricType,
        biometricSubtype BiometricSubtype OPTIONAL,
        brtQuality Quality OPTIONAL,
        bdbEncryptionOptions BOOLEAN(FALSE),
        bdbIntegrityOptions BOOLEAN(FALSE),
        bdbFormatForBRTC BDBFormat}
BDBForBRTC ::= SEQUENCE {
        version Version DEFAULT v1,
        issuerAndSerialNumberBRTC IssuerAndSerialNumber OPTIONAL,
        originalBDBHashList HashList,
        originalBIRReferrer URI OPTIONAL,
        originalBIRPatronFormat PatronFormat,
        originalBDBPosition INTEGER,
```

```
        userInformation UserInformation OPTIONAL,
        pkiCertificateInformation PKICertificateInformation OPTIONAL,
        enrolmentACBioInstances SequenceOfACBioInstances OPTIONAL}
HashList ::= SEQUENCE SIZE(1..MAX) OF Hash
UserInformation ::= SEQUENCE {
        userIdentifier OCTET STRING,
        userName Name OPTIONAL,
        userUniqueIdentifier UniqueIdentifier OPTIONAL}
PKICertificateInformation ::= SEQUENCE {
        pkiCertificateSerialNumber CertificateSerialNumber,
        pkiCertificateIssuerName Name OPTIONAL,
        pkiCertificateIssuerUniqueIdentifier UniqueIdentifier OPTIONAL}
SequenceOfACBioInstances ::= SEQUENCE SIZE(1..MAX) OF ACBioInstance
-- Useful definitions
SIGNEDDATA { EncapsulatedContentInfo } ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithms SET OF DigestAlgorithmIdentifier,
        encapContentInfo EncapsulatedContentInfo,
        certificates [0] IMPLICIT CertificateSet OPTIONAL,
        crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
        signerInfos SignerInfos}
-- contentType object identifiers
id-signedDataACBio OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) signedDataACBio(1)}
id-acbioContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) acbioContent(3)}
id-contentBPUReport OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) bpuReport(4)}
id-bpuReportContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) bpuReportContent(5)}
id-contentBRTCertificate  OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) brtCertificate(6)}
id-brtcContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) brtcContent(7)}
id-performanceReportContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) brtcContent(8)}
id-cmSecurityReportContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) brtcContent(9)}
id-bpSecurityReportContentInformation OBJECT IDENTIFIER ::=
        {iso(1) standard(0) acbio(24761) contentType(2) brtcContent(10)}

-- ContentType objects
signedDataACBio CONTENT-TYPE ::= {
        SignedDataACBio
        IDENTIFIED BY id-signedDataACBio }
acbioContentInformation CONTENT-TYPE ::= {
        ACBioContentInformation
        IDENTIFIED BY id-acbioContentInformation }
```

```
contentBPUReport CONTENT-TYPE ::= {
        ContentBPUReport
        IDENTIFIED BY id-contentBPUReport }
bpuReportContentInformation CONTENT-TYPE ::= {
        BPUReportContentInformation
        IDENTIFIED BY id-bpuReportContentInformation }
contentBRTCertificate CONTENT-TYPE ::= {
        ContentBRTCertificate
        IDENTIFIED BY id-contentBRTCertificate }
brtcContentInformation CONTENT-TYPE ::= {
        BRTCContentInformation
        IDENTIFIED BY id-brtcContentInformation }
performanceReportContentInformation CONTENT-TYPE ::= {
        PerformanceReportContentInformation
    IDENTIFIED BY id-performanceReportContentInformation }
cmSecurityReportContentInformation CONTENT-TYPE ::= {
        CMSecurityReportContentInformation
    IDENTIFIED BY id-cmSecurityReportContentInformation }
bpSecurityReportContentInformation CONTENT-TYPE ::= {
        BPSecurityReportContentInformation
    IDENTIFIED BY id-bpSecurityReportContentInformation }

END -- AuthenticationContextForBiometrics
```

Annex B
(informative)

Implementation examples

B.1 Examples of the implementation of ACBio

B.1.1 General

In this document, the protocol for ACBio is not specified. In this Annex, two examples of implementation of ACBio are given including protocols; one for a case of STOC (STore On Card) model, the other for a case of OCBC (On Card Biometric Comparison) model.

B.1.2 An Example of the implementation of STOC Model

B.1.2.1 General

In this example, the STOC model of a biometric verification process means a biometric system consisting of two BPUs: one is a biometric device which has the functions of data capture, intermediate signal processing, final signal processing, comparison and decision, and the other is a STOC card which stores the processed biometric reference. This example is mainly focused on STOC card.

B.1.2.2 In production process

B.1.2.2.1 Case of declaration expression

Products of BPUs, i.e. STOC cards and biometric devices used in a biometric verification process, should be evaluated at a certain evaluation organization and issued their BPU security reports.

B.1.2.2.2 Case of declaration expression

Vendors of BPUs indexes every subprocess and stream in accordance with the rule in 7.2.2. If the subprocesses and streams in the biometric device and those in the STOC card are indexed as in Figure B.1 and Figure B.2, then the BPUFunctionReport of the biometric device and that of the STOC card are as shown in Figure B.3. In Figure B.1 and Figure B.2, SIndex means subprocess index and IOIndex means subprocess IO index.

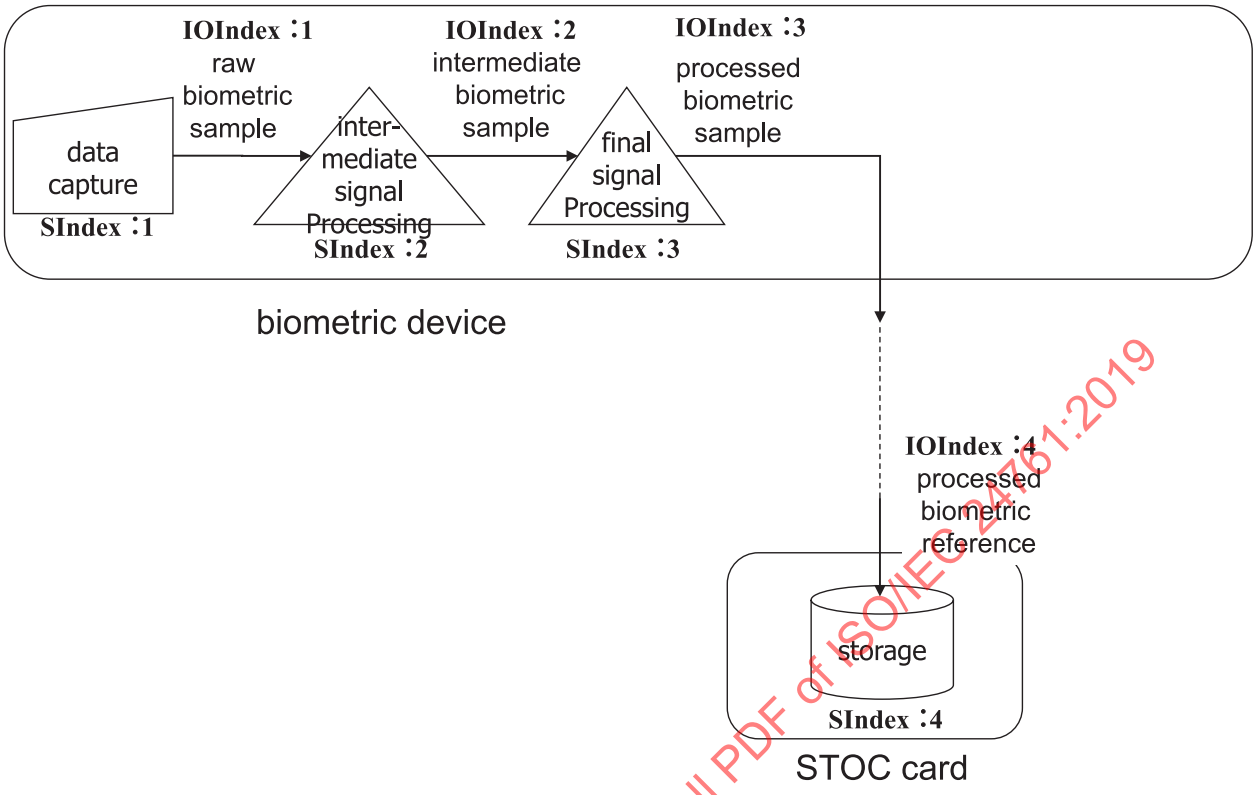


Figure B.1 — A biometric enrolment process of a STOC model and an example of indexing for declaration expression

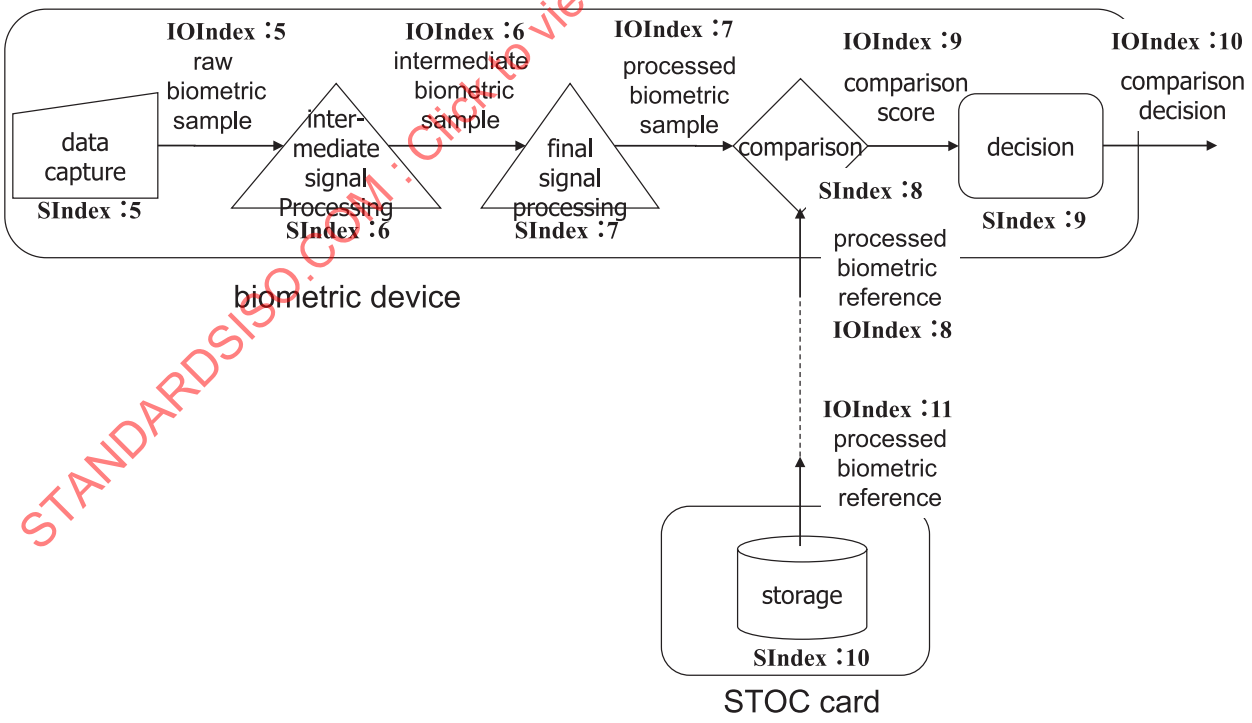


Figure B.2 — A biometric verification process of a STOC model and an example of indexing for declaration expression

As written in the NOTE in 7.2.2.2, `bpuSubprocessInformationList`, `bpuInputStaticInformationList`, and `bpuOutputStaticInformationList` are divided into two groups: one for enrolment and another

for verification. Figure B.3 is an example for BPUFunctionReports. The whole of BPUFunctionReport for the STOC card is shown in Figure B.3. BPUSubprocessInformation and BPUIOInformation in BPUFunctionReport for the biometric device are depicted in Figure B.4 and Figure B.5 respectively.

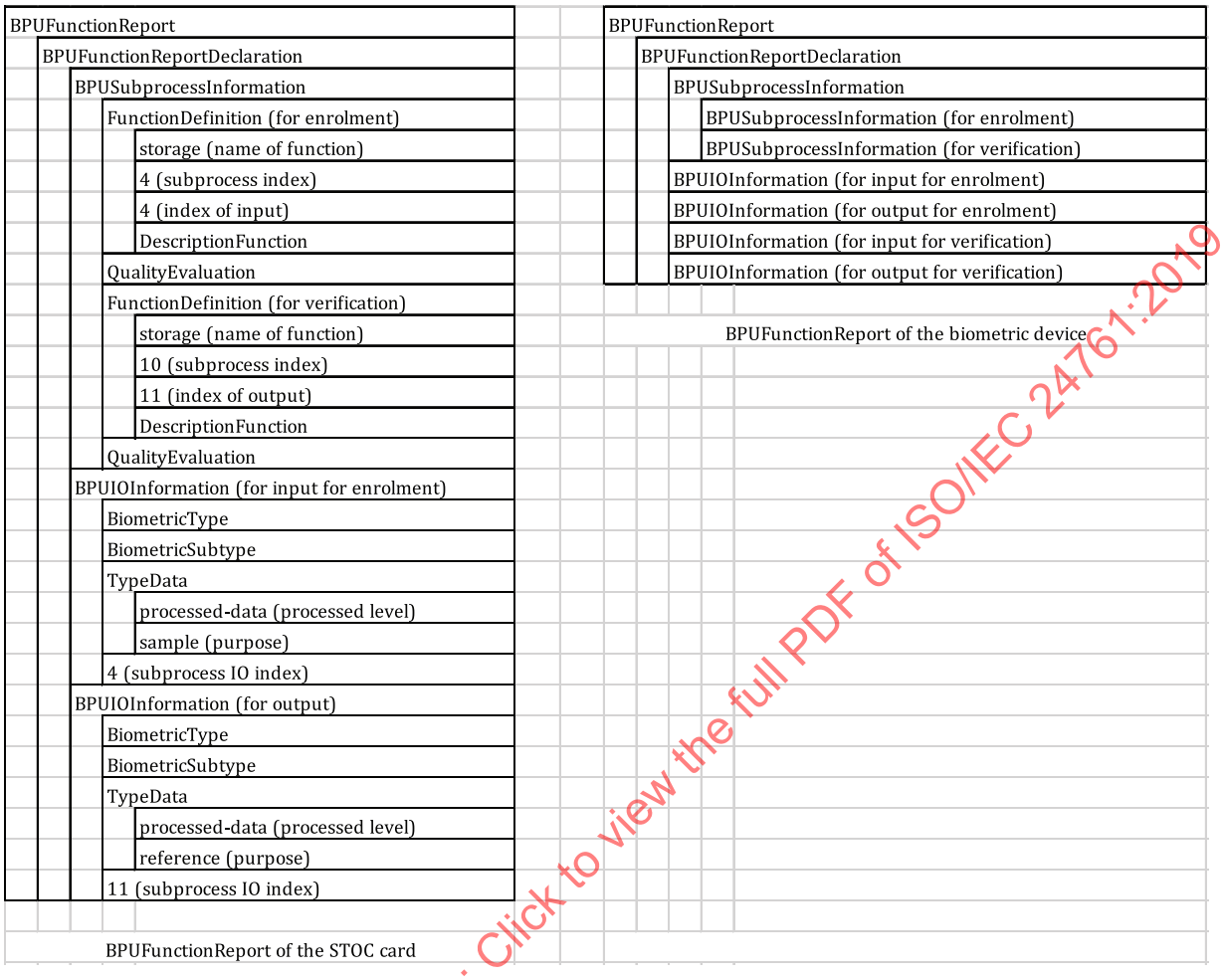


Figure B.3 — Examples of BPUFunctionReports for a STOC model in declaration expression

FunctionDefinition
data-capture (name of function)
1 (subprocess index)
1 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
intermediate-signal-processing (name of function)
2 (subprocess index)
1 (index of input1)
2 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
final-signal-processing (name of function)
3 (subprocess index)
2 (index of input1)
3 (index of output)
DescriptionFunction
QualityEvaluation

BPUSubprocessInformation (for enrolment)

FunctionDefinition
data-capture (name of function)
5 (subprocess index)
5 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
intermediate-signal-processing (name of function)
6 (subprocess index)
5 (index of input1)
6 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
final-signal-processing (name of function)
7 (subprocess index)
6 (index of input1)
7 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
comparison (name of function)
8 (subprocess index)
7 (index of input1)
8 (index of input2)
9 (index of output)
DescriptionFunction
QualityEvaluation
FunctionDefinition
decision (name of function)
9 (subprocess index)
9 (index of input1)
10 (index of output)
DescriptionFunction
QualityEvaluation

BPUSubprocessInformation (for verification)

Figure B.4 — Examples of BPUSubprocessInformation for a biometric device

BPUIOInformation (for output)	
BiometricType	
BiometricSubtype	
TypeData	
	processed-data (processed level)
	sample (purpose)
3 (subprocess IO index)	

BPUIOInformation (for input)	
BiometricType	
BiometricSubtype	
TypeData	
	processed-data (processed level)
	reference (purpose)
8 (subprocess IO index)	

BPUIOInformation (for enrolment)

BPUIOInformation (for output)	
BiometricType	
BiometricSubtype	
TypeData	
	comparison-decision (processed level)
10 (subprocess IO index)	

BPUIOInformation (for verification)

Figure B.5 — Examples of BPUIOInformation for a biometric device

B.1.2.2.3 Case of role expression

Compared with declaration expression, role expression is more implicit but simpler. Vendors of BPUs do not have to index every subprocess and stream in BPUs but should index streams from/to the BPUs. If the streams from/to the biometric device and that from the STOC card are indexed as in Figure B.6 for enrolment and Figure B.7 for verification, then the BPUFunctionReport of the biometric device and that of the STOC card are expressed as in Figure B.8. In Figure B.6 and Figure B.7, IOIndex means subprocess IO index.

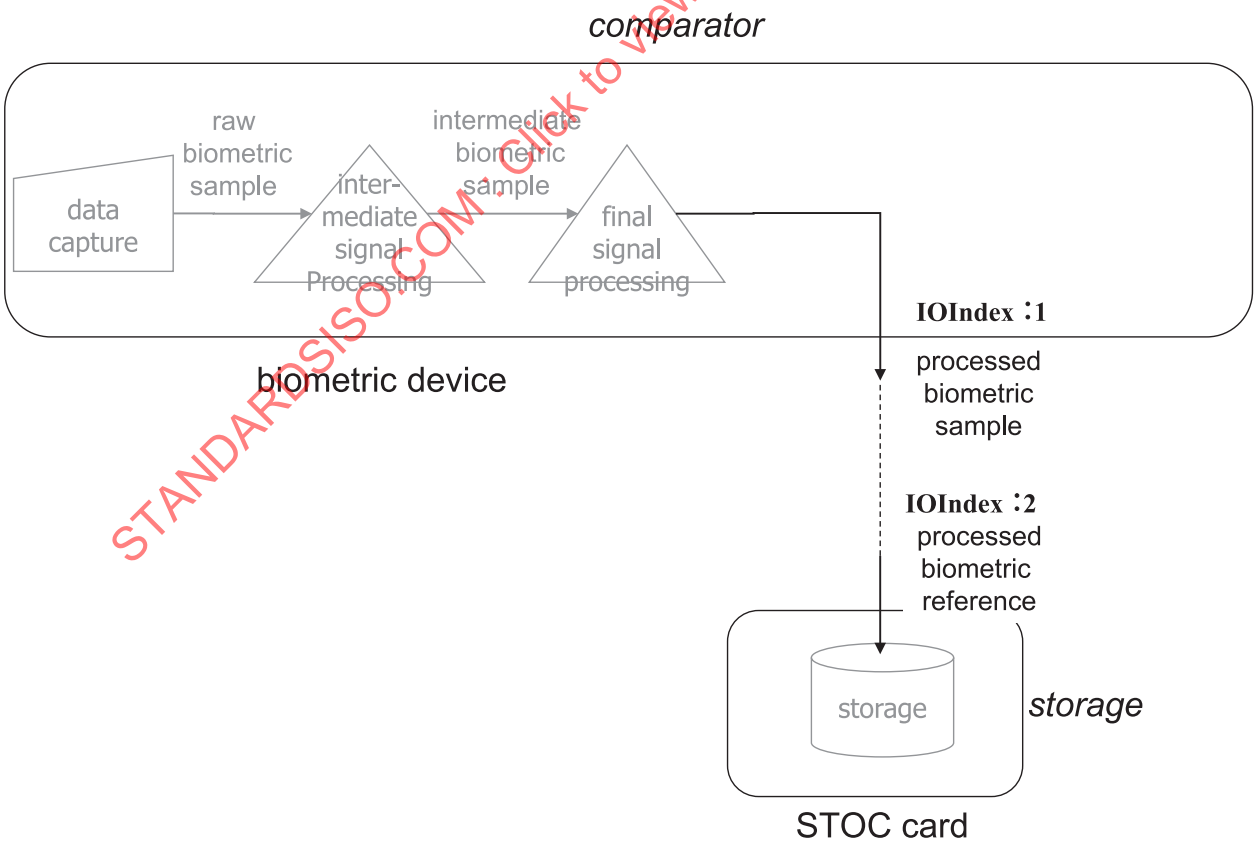


Figure B.6 — A biometric enrolment process of a STOC model and an example of indexing for role expression

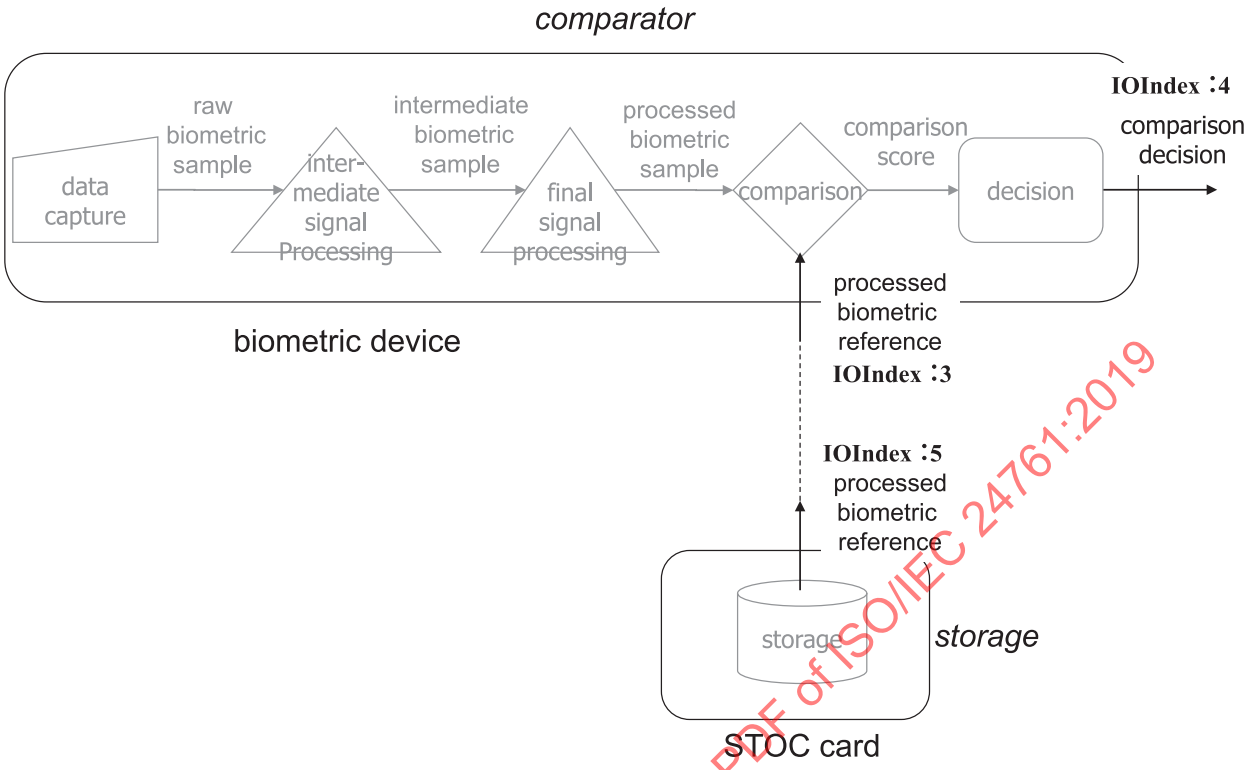


Figure B.7 — A biometric verification process of a STOC model and an example of indexing for role expression

Figure B.8 — Examples of BPUFunctionReports for a STOC model in role expression

B.1.2.3 In enrolment process

Biometric reference is stored to a STOC card in this process. A BRT certificate is issued to the biometric reference. It or its referrer is to be stored in `brtCertificateInformation` of `ACBioContentInformation`.

B.1.2.4 In execution process

On an execution of biometric verification, two inputs are given to a STOC card; the first is a control value from the validator, the second is the BPU IO index to the output from the STOC card. Called with PERFORM BIOMETRIC OPERATION command, which is specified in ISO/IEC 7816-11, with parameters including the above two, the STOC card digitally signs the whole field of the type `ACBioContentInformation` to get the ACBio instance and returns it together with the processed biometric reference, as depicted in Figure B.9 when declaration expression is used. When role expression is used, 10, index of executed subprocess, is replaced with 2 of `executionIndex`, and 11, the subprocess IO index for output, is replaced with 5 (see Figure B.7 and Figure B.8).

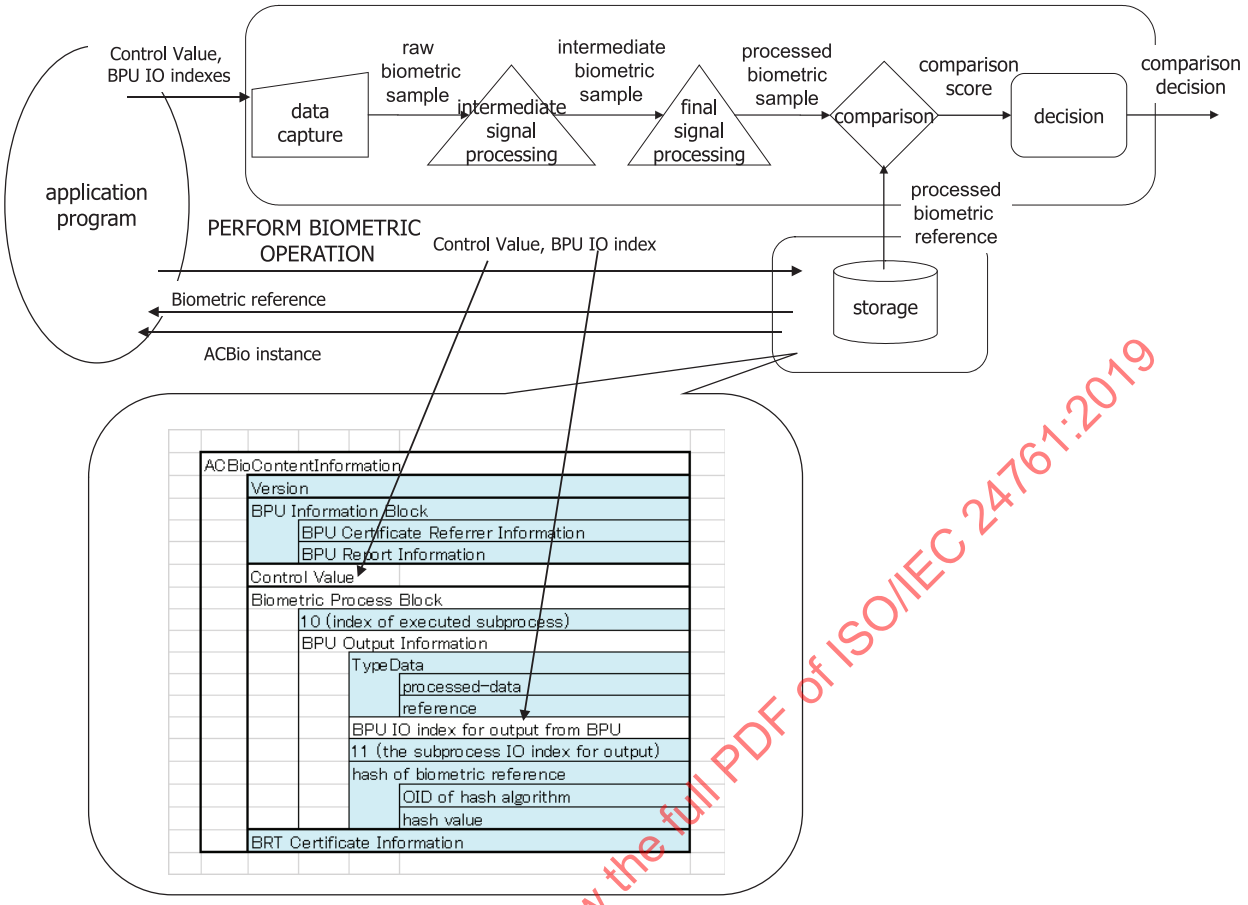
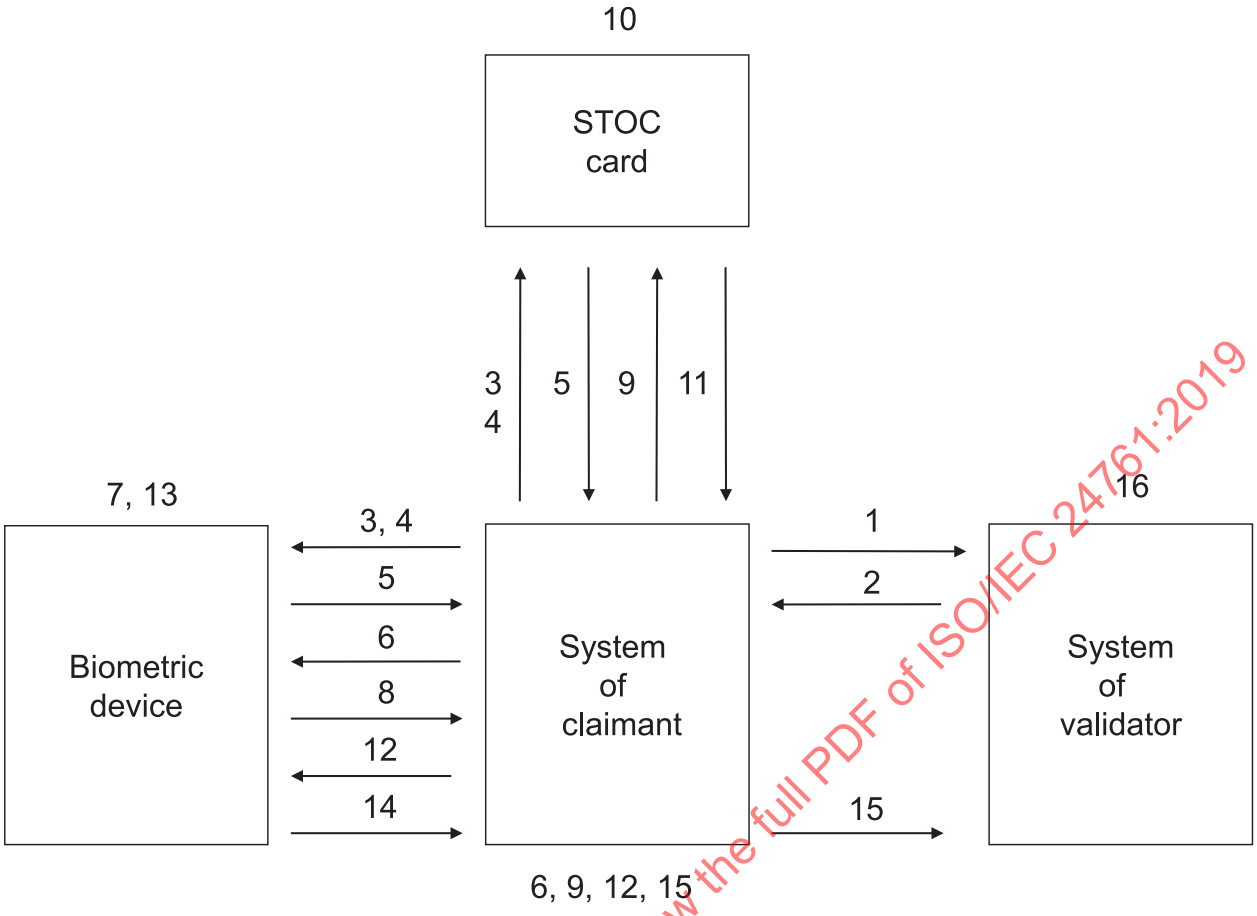


Figure B.9 — ACBio instance generation on a STOC card on an execution of biometric verification

B.1.2.5 An example of protocol

In this example, it is assumed that all of the subprocesses of biometric verification are done at the claimant and also that there is a system at the claimant which sends and receives messages to/from the system of the validator, biometric device, and STOC card, as shown in Figure B.10.



Key

1 to 15 order of protocol execution

Figure B.10 — An example of protocol for a STOC model

The following shows an example of the protocol for this STOC model:

1) The claimant requests authentication to the validator, via the system of claimant.

2) The system of validator sends the control value and the candidate list of hash algorithms, digital signature algorithms, modalities, and biometric capability classes according to the ACBio validation policy, and requests the execution of biometric verification to the system of claimant. Here in this example, storage-and-others class is assumed to be contained in the candidate list of biometric capability classes.

3) The system of the claimant checks if any of the list of modalities and biometric capability classes are satisfied at the claimant. In this case, the biometric device and STOC card can configure storage-and-others class in the list of biometric capability classes. If there is a set of BPUs which satisfy the modalities and biometric capabilitiy classes, the system of the claimant also checks if the data types of the data flows between the BPUs are consistent.

4) The system of the claimant inquires available hash algorithms and digital signature algorithms to the biometric device and the STOC card.

5) The biometric device and the STOC card return available hash algorithms and digital signature algorithms to the system of claimant side.

6) The system of claimant decides the hash algorithm and the signature algorithm, and sends the validator's control value, the hash algorithm, and the signature algorithm to the biometric device to request execution of data capture, intermediate signal processing, and final signal processing.

7) The biometric device captures the biometric information from the claimant and generates the processed biometric sample through data capture, intermediate signal processing, and final signal processing subprocesses.

8) The biometric device sends the termination message of final signal processing to the system of the claimant.

9) The system of the claimant calls the STOC card to get the processed biometric reference with the validator's control value, the hash algorithm and the signature algorithm selected in step (6), and a BPU IO Index for the output data (processed biometric reference) of the STOC card.

10) The STOC card generates an ACBio instance (see B.1.2.4 for details).

11) The STOC card returns the processed biometric reference with the ACBio instance to the system of the claimant.

12) The system of claimant calls the biometric device for execution of comparison and decision with the processed biometric reference received from the STOC card, the BPU IO index assigned to the processed biometric reference, and the BPU IO Index assigned to the comparison decision of the biometric device.

13) The biometric device receives the processed biometric reference and executes comparison and decision subprocesses. The biometric device also generates an ACBio instance with the following procedures.

    a) Restore the BPU Information block and set it to `bpuInformation` of `ACBioContentInformation`.

    b) Set the validator's control value to `controlValue` of `ACBioContentInformation`.

    c) Generate the biometric process block as follows;

        1a) (The case of declaration expression used for BPU report) Set the subprocess indexes corresponding to the subprocesses executed at the biometric devices to `executedProcessIndexList`. In this case, the subprocess indexes corresponding to data capture, intermediate signal processing, final signal processing, comparison, and decision are set in `executedProcessIndexList`.

        1b) (The case of role expression used for BPU report) Set the `executionIndex` in `executionInformationList` in `BPUFunctionReportRoleSingle` corresponding to the execution of the functions at the biometric device to `executedProcessIndexList`. If `BPUFunctionReportRoleSingle` is expressed as in Figure B.8, 2 is set as the only one member of `executedProcessIndexList`.

        2) To make `bpuInputExecutionInformationList` field, the following should be done. Set the value `processed-data` and the value `reference` respectively to `processedLevel` and `purpose` of `dataType`. Set the BPU IO index, assigned to the input of the biometric device, to `bpuIOIndex`. Set the subprocess IO index of the input data (processed biometric reference) to the comparison subprocess to `subprocessIOIndex`. Set the pair of the hash value of the processed biometric reference received and the hash algorithm to `hash`.

        3) To make `bpuOutputExecutionInformationList` field, the following should be done. Set the value `comparison-decision` to `processedLevel` of `dataType`. Set the BPU IO index, assigned to the output data of the biometric device, to `bpuIOIndex`. Set the subprocess IO index of the output data (comparison decision) of the decision subprocess, which was assigned by the product vendor of the biometric device, to `subprocessIOIndex`. Set the pair of the hash value of the comparison decision and the hash algorithm into `hash`.

    d) Generate `SignedDataACBio` with the data generated as in a), b) and c), using the digital signature algorithm selected in (6).

14) The biometric device sends the result of decision subprocess and the ACBio instance to the system of the claimant.