

I n t e r n a t i o n a l T e l e c o m m u n i c a t i o n U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.1084

(05/2008)

SERIES X: DATA NETWORKS, OPEN SYSTEM
COMMUNICATIONS AND SECURITY

Telecommunication security

**Telebiometrics system mechanism – Part 1:
General biometric authentication protocol and
system model profiles for telecommunications
systems**

Recommendation ITU-T X.1084

ITU-T X-SERIES RECOMMENDATIONS
DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY

PUBLIC DATA NETWORKS	X.1–X.199
OPEN SYSTEMS INTERCONNECTION	X.200–X.299
INTERWORKING BETWEEN NETWORKS	X.300–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	X.600–X.699
OSI MANAGEMENT	X.700–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	X.850–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999
INFORMATION AND NETWORK SECURITY	
General security aspects	X.1000–X.1029
Network security	X.1030–X.1049
Security management	X.1050–X.1069
Telebiometrics	X.1080–X.1099
SECURE APPLICATIONS AND SERVICES	
Multicast security	X.1100–X.1109
Home network security	X.1110–X.1119
Mobile security	X.1120–X.1139
Web security	X.1140–X.1149
Security protocols	X.1150–X.1159
Peer-to-peer security	X.1160–X.1169
Networked ID security	X.1170–X.1179
IPTV security	X.1180–X.1199
CYBERSPACE SECURITY	
Cybersecurity	X.1200–X.1229
Countering spam	X.1230–X.1249
Identity management	X.1250–X.1279
SECURE APPLICATIONS AND SERVICES	
Emergency communications	X.1300–X.1309
Ubiquitous sensor network security	X.1310–X.1339

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T X.1084

Telebiometrics system mechanism – Part 1: General biometric authentication protocol and system model profiles for telecommunications systems

Summary

Biometric technologies are developed in various products and populated in application systems such as border control, physical access control, etc., for identity verification. These technologies are also expected to be applied to open network systems for reliable user authentication. However, open network systems need to manage risks in biometric products and system configurations for secure remote services. Recommendation ITU-T X.1084 specifies biometric authentication protocols and profiles for telecommunication systems in open networks.

Source

Recommendation ITU-T X.1084 was approved on 29 May 2008 by ITU-T Study Group 17 (2005-2008) under Recommendation ITU-T A.8 procedures.

Keywords

Telebiometric authentication profiles, telebiometric authentication protocol, telebiometric system mechanism, transport layer security.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2009

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

	Page
1 Scope	1
2 References.....	1
3 Definitions	2
3.1 Vocabulary definitions within ISO/IEC JTC 1/SC 37 [b-SC37SD2V8].....	2
3.2 Terms defined in this Recommendation.....	2
4 Abbreviations.....	3
5 Conventions	4
6 Prerequisites.....	4
7 Authentication models	5
8 Security threats for each models.....	10
9 General requirements.....	12
10 General protocol	13
10.1 Requirement of the biometrics handshake protocol	13
10.2 Alert protocol for biometric handshake.....	16
10.3 Implementation of the extended protocol.....	17
11 Requirements of the biometric transportation stage for each model	18
11.1 Local model.....	19
11.2 Download model	20
11.3 Attached model.....	20
11.4 Centre model	21
11.5 Reference management on TTP for local model	21
11.6 Reference management on TTP for centre model	22
11.7 Comparison outsourcing by client model.....	23
11.8 Comparison outsourcing by server model	25
11.9 Storage and comparison outsourcing model.....	26
Annex A – ASN.1 definitions for modified TLS extension protocol	30
Appendix I – Telebiometrics system mechanism definitions by TLS extension.....	41
I.1 Extensions for biometric transfer protocol	41
I.2 Biometrics Verify	43
I.3 Biometrics Retry Request.....	45
I.4 Finished Biometrics.....	45
I.5 Biometrics TTP Request.....	46
I.6 Biometrics TTP response	47
I.7 Extension alert protocol.....	47

	Page
Appendix II – Implementation example of the biometric transfer protocol using BIP	50
II.1 Local model.....	51
II.2 Download model	52
II.3 Attached model.....	52
II.4 Centre model	53
II.5 Comparison outsourcing by client model.....	53
II.6 Reference management on TTP for local model	54
II.7 Reference management on TTP for centre model	55
II.8 Comparison outsourcing by server model	56
II.9 Storage and comparison outsourcing model.....	57
Appendix III – Template registration and updating process for this Recommendation	59
III.1 Registration process.....	59
III.2 Updating or revocation process	60
Appendix IV – ASN.1 definitions for the protocol of TSM based on Appendix I.....	63
Appendix V – ECN modules for Appendix IV	75
V.1 EDM module	75
V.2 ELM module.....	97
Bibliography.....	100

Introduction

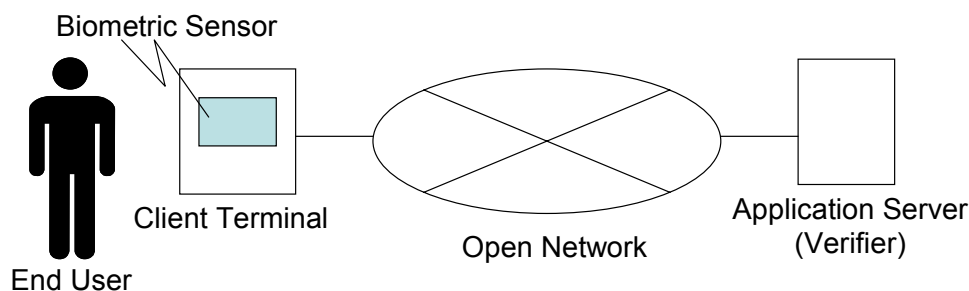
With the rapid and widespread diffusion of the Internet, various network services are now in operation. In high value services, such as Internet banking, Internet shopping, Internet trading, etc., illegal trading by obtaining a personal identification number (PIN) by means, such as phishing, are occurring with increasing regularity. Therefore, high security authentication mechanisms are increasingly required, such as can be provided by biometrics.

We have the following problems in standardizing biometric authentication on the Internet:

- Service providers do NOT have any information regarding what biometric devices are in use at the end-user's end, what security level is this device set at, or how it is operated.
- According to each biometric product, the accuracy (False Accept Rate) determined by the threshold parameter differs between different biometric products. Therefore, the service provider can NOT claim to maintain a uniform accuracy level.
- The accuracy of biometric verification may decline with the aging of end-users because biometrics uses features of the human body.

To solve these problems, protocols for biometric authentication between unspecified end-users and service providers on open networks are greatly required.

The figure below illustrates the environment of this Recommendation for a biometric security mechanism that authenticates a user via a non-face-to-face open network.



Environment of this Recommendation

The meaning of the open network:

- Many unspecified verifiers connect to the network and use varying biometric methods.
 - High value service provider
 - Efficient government service provider
 - Online shopping provider.
- A large number of unspecified end-users also connect to the network, and their identity is verified through biometric authentication in order to use services from the aforementioned providers.

The verifier here is "open" in the following sense. The purpose of biometric authentication is different for each verifier, and the risk/value for the verifier is also different for each. Therefore, each verifier has a different authentication security policy.

The user here is "open" in the sense that each user uses different biometric authentication methods. Each user can select any biometric authentication method to use, according to the acceptability or privacy policy they follow.

Recommendation ITU-T X.1084

Telebiometrics system mechanism – Part 1: General biometric authentication protocol and system model profiles for telecommunications systems

1 Scope

This Recommendation specifies the biometric authentication protocols and profiles for telecommunication systems. It defines the protocols for biometric authentication of unspecified end-users and service providers on open networks. In the open network, there are a range of biometric communication devices for the end-users. There are also a variety of security policies for network services for the providers. This Recommendation defines nine telebiometrics authentication models depending on the configuration of the client, the server, and the trusted third party. It also defines the negotiation protocol for the policies and the device environments using the models. Furthermore, it specifies the requirements of biometric transportation data for each model.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T X.509] Recommendation ITU-T X.509 (2005) | ISO/IEC 9594-8:2005, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- [ITU-T X.1089] Recommendation ITU-T X.1089 (2008), *Telebiometrics authentication infrastructure (TAI)*.
- [ISO/IEC 15408-1] ISO/IEC 15408-1:2005, *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*.
- [ISO/IEC 19784-1] ISO/IEC 19784-1:2006, *Information technology – Biometric application programming interface – Part 1: BioAPI specification*.
- [ISO/IEC 19785-1] ISO/IEC 19785-1:2006, *Information technology – Common Biometric Exchange Formats Framework – Part 1: Data element specification*.
- [ISO/IEC 19795-1] ISO/IEC 19795-1:2006, *Information technology – Biometric performance testing and reporting – Part 1: Principles and framework*.
- [ISO/IEC 24761] ISO/IEC 24761:2009, *Information technology – Security techniques – Authentication context for biometrics*.
- [IETF RFC 3986] IETF RFC 3986 (2005), *Uniform Resource Identifier: Generic Syntax*.
- [IETF RFC 4346] IETF RFC 4346 (2006), *The Transport Layer Security (TLS) Protocol Version 1.1*.
- [IETF RFC 4366] IETF RFC 4366 (2006), *Transport Layer Security (TLS) Extensions*.
- [X9.84-CMS] OASIS X9.84-CMS (2003), *XML Common Biometric Format*.

3 Definitions

3.1 Vocabulary definitions within ISO/IEC JTC 1/SC 37 [b-SC37SD2V8]

This Recommendation uses the following terms defined elsewhere:

3.1.1 biometric (adjective): Of or having to do with biometrics.

3.1.2 biometrics (noun): An automated recognition of individuals based on their behavioural and biological characteristics.

3.1.3 biometric template: A set of stored biometric features comparable directly to biometric features of a recognition biometric sample.

3.1.4 biometric reference: One or more stored biometric samples, biometric templates or biometric models attributed to a biometric data subject and used for comparison.

3.1.5 biometric sample: Analogue or digital representation of biometric characteristics prior to biometric feature extraction process, and obtained from a biometric capture device or biometric capture subsystem.

3.1.6 comparison (match/matching): Estimation, calculation or measurement of similarity or dissimilarity between recognition biometric sample(s)/biometric features/biometric models and biometric reference(s).

3.1.7 comparison decision: Determination of whether the recognition biometric sample(s) and biometric reference(s) have the same biometric source, based on a comparison score(s), a decision policy(ies), including a threshold, and possibly other inputs.

3.1.8 comparison score: Numerical value (or set of values) resulting from a comparison.

3.1.9 false match: Comparison decision of "match" for a recognition biometric sample and a biometric reference that are not from the same source.

3.1.10 false non-match: Comparison decision of "non-match" for a recognition biometric sample and a biometric reference that are from the same source.

3.1.11 match: Decision that the recognition biometric sample(s) and the biometric reference are from the same source.

3.1.12 non-match: Decision that the recognition biometric sample(s) and the biometric reference are not from the same source.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 biometric authentication: The process of confirming an individual's identity, either by verification or by identification.

3.2.2 decision policy: Logic through which a biometric system provides match/no match decisions, for example, through the setting of the threshold and number of attempts.

3.2.3 false accept rate (FAR): Measure of the probability that a biometric comparison process will incorrectly identify an individual or will fail to reject an impostor.

3.2.4 false reject rate (FRR): Measure of the probability that a biometric comparison process will incorrectly fail to identify an individual.

3.2.5 failure to acquire: Failure of a biometric system to capture a biometric sample, or to extract biometric data from a biometric sample, sufficient to generate a biometric template for matching.

3.2.6 failure to enrol: Failure of a biometric system to capture one or more biometric samples, or to extract data from one or more biometric samples, sufficient to generate a biometric template for successful enrolment.

3.2.7 registration: The process in which a person proves their identity by presenting credentials to the biometric service provider prior to being allowed to enrol and being assigned an electronic identifier.

3.2.8 risk management: Coordinated activities to direct and control an organization with regard to risk.

3.2.9 threshold: A predefined value which establishes the degree of similarity or correlation, (i.e., score), necessary for a biometric sample to be deemed an accurate comparison with a biometric reference template.

3.2.10 verifier: A service provider based on a biometric authentication on telecommunication environments.

3.2.11 client: A client terminal that provides a biometric function for telecommunication service applications.

3.2.12 biometric certification authority: One of third trusted party which certifies for an integrity of biometric reference and procedure of biometric product evaluation report.

3.2.13 user: The receiver of the telecommunication service using the client.

4 Abbreviations

This Recommendation uses the following abbreviations:

ACBio	Authentication Context for Biometrics
BC	Biometric Certificate
BCA	Biometric Certification Authority
BDC	Biometric Device Certificate
BFP	Biometric Function Provider
BioAPI	Biometric Application Programming Interface
BPU	Biometric Processing Unit
BSP	Biometric Service Provider
CA	Certificate Authority
CBEFF	Common Biometric Exchange File Format
CC	Common Criteria
CRL	Certificate Revocation List
EAL	Evaluation Assurance Level
PKI	Public Key Infrastructure
PP	Protection Profile
ST	Security Target
TLS	Transport Layer Security
TSM	Telebiometrics System Mechanism
TTP	Trusted Third Party

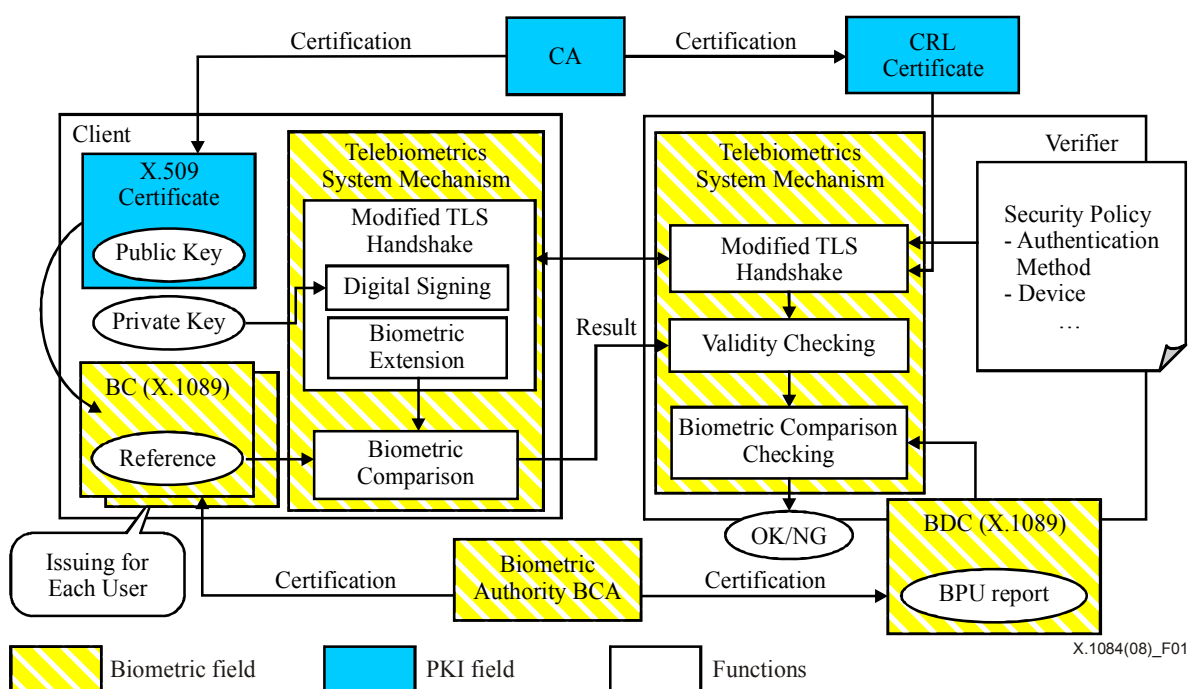
5 Conventions

None.

6 Prerequisites

This Recommendation contains the following prerequisites:

- A trusted third party (TTP) such as a certificate authority (CA) is required to authenticate a user's public key and to issue a certificate as specified in [ITU-T X.509].
- A TTP such as biometric certification authority (BCA) is required to authenticate user-registered biometric reference information, and to digitally sign that information using the biometric certificate (BC) in [ITU-T X.1089].
- A TTP such as BCA is required to authenticate both the threshold of each biometric technology in use and the accuracy evaluation result, using the biometric processing unit (BPU) report as specified in [ISO/IEC 24761]. BCA is required to grant the report by a digital signature using the biometric device certificate (BDC) in [ITU-T X.1089].
- A TTP is required to authenticate the security evaluation result based on the common criteria scheme for a biometric device, and to digitally sign the evaluation report using the biometric device certificate (BDC) in [ITU-T X.1089].
- Depending on the model, a TTP may also be required to manage biometric registration information.
- Depending on the model, a TTP may be required to perform biometric comparison.



NOTE – The yellow colour belongs to biometric field, the blue referring to the PKI field, the white boxes are functions.

Figure 1 – Outline system of this Recommendation

7 Authentication models

This Recommendation takes into account the two perspectives below, dividing models into nine categories (all combinations of the three items in the two lists below).

- Biometric reference template location
 - Client terminal-side (user) storage of template
 - Server-side (verifier) storage of template
 - TTP-side storage of template
- Biometric comparison location
 - Client terminal-side (user) comparison
 - Server-side (verifier) comparison
 - TTP-side comparison

Table 1 – Authentication models

Store \ Comparison	Client	Server	TTP
Client	Local	Download	Reference management on the TTP for client comparison
Server	Attached	Centre	Reference management on the TTP for server comparison
TTP	Comparison outsourcing by client	Comparison outsourcing by server	Storage & comparison outsourcing

1) *Local model*

Figure 2 illustrates a local model.

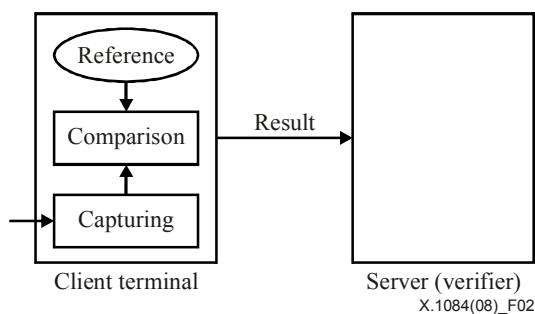


Figure 2 – Local model

The client terminal takes the request from the verifier; it acquires sample data, compares it with the registered user's template, and transfers the result to the verifier.

Template ID information is required, which is the comparison result.

For the local model, we assume that the server side cannot bear the biometric-processing load, and the user terminal-side is given sufficient processing resources. (The processing resources must be sufficient to acquire sample data and compare.)

This model can be used when the server side trusts the client-side processing procedures.

This is practical when put to use for access control systems that verify each process and transfer and save a log to the server.

2) *Download model*

Figure 3 illustrates a download model.

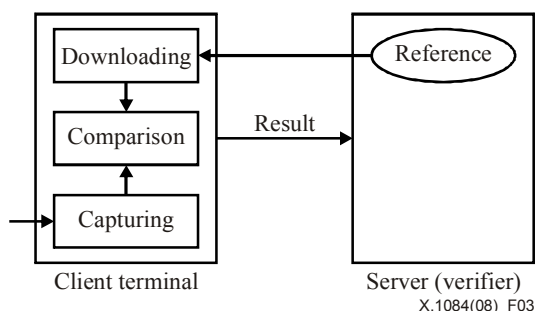


Figure 3 – Download model

The verifier sends the registered user's template and request of verification to the client terminal. It compares the acquired sample data with the received template, sending the result to the verifier.

The download model has the same sequence as the local model, excluding the transfer template from the verifier. Template ID information is required. This then forms the comparison result.

For the client terminal, we assume a temporary-use terminal (e.g., credit authorization terminal).

Under this structure, it is practical to use many terminals. This is suitable for the web-verification model, as users can connect to the server from any client terminal. This download model requires registration of the biometric reference template to the verifier.

This model also requires the server to trust the client-side processing procedure.

3) *Attached model*

Figure 4 illustrates an attached model.

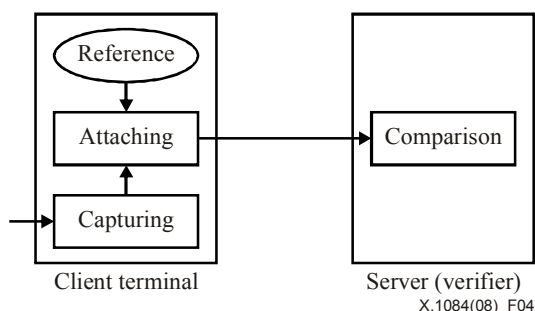


Figure 4 – Attached model

The verifier requires the comparison function, which is needed for template and captured sample data. The client terminal transfers the acquired sample data and template to the verifier, and the comparison is ultimately conducted within the verifier.

For an attached model, we assume each user utilizes a biometric algorithm that has a heavy import load (cost, private device). It is also possible for a user to utilize only a trusted verifier to judge and to send sensitive biometric data (which is similar to ID certification).

The server trusts client capture-data.

This is practical for use in credit authorization systems, as a comparison algorithm is not required for all client terminals.

4) *Centre model*

Figure 5 illustrates a centre model.

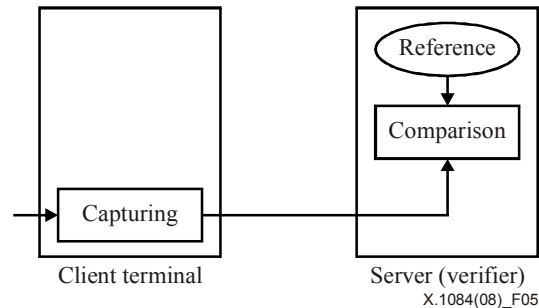


Figure 5 – Centre model

The verifier requests the sample data required for verification, the client terminal acquires sample data and transfers it to the verifier, and the verifier finally compares it with the user's template.

For the centre model, we assume that the user-side terminal cannot guarantee sufficient resources in terms of processing power, memory, etc.

The user registers its biometric reference in advance with a verifier that has a trusted biometric reference template.

This model requires that the server trusts the data captured from a client.

5) *Reference management on TTP for local model*

Figure 6 illustrates a reference management on TTP for local model.

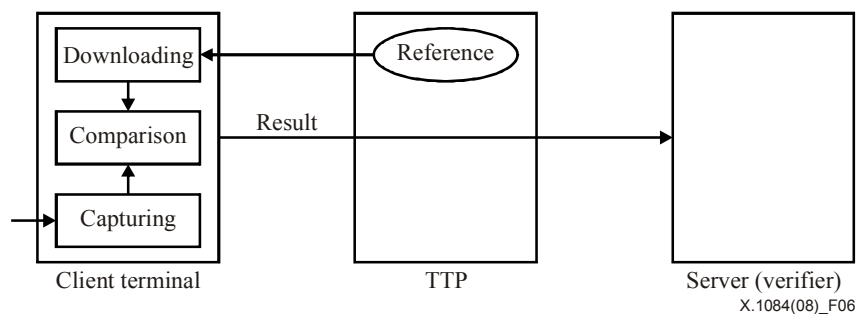


Figure 6 – Reference management on TTP for local model

The verifier requests verification, the client terminal acquires sample data, and requests the user's template to TTP. Then the client terminal compares the template which is transferred from TTP with the captured sample data, and transfers the result data to the verifier.

For this model, we assume that a user seeks to use multiple services on a temporary-use terminal, for example, a common credit card authorization terminal.

Template ID information is required. This forms the comparison result.

A user registers its biometric reference template in advance with TTP. This structure is the same as a local model, except that TTP manages the template.

The server is required to trust processing on the client side. The client and the verifier of the structure require guaranteed stability in the TTP process.

6) *Reference management on TTP for centre model*

Figure 7 illustrates a reference management on TTP for centre model.

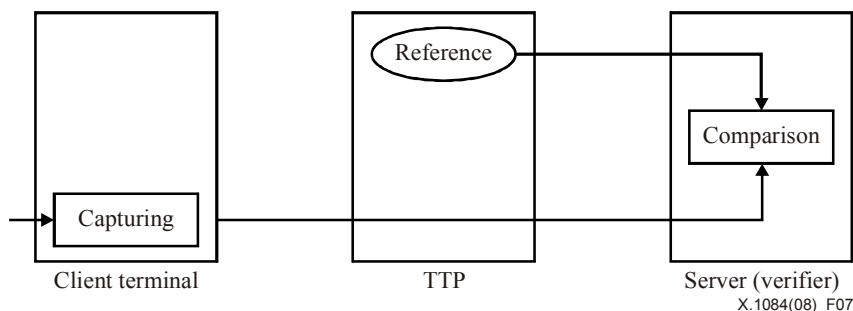


Figure 7 – Reference management on TTP for centre model

The verifier requests sample data to the client terminal and the user's template to the TTP. The client terminal captures the sample data and transfers it to the verifier, and TTP transfers the registered user's template to the verifier. Then, the verifier compares the received sample data with the received user's template.

For this model, users register their biometric reference template in advance with a TTP, and the verifier compares it with the reference and capturing data on the client terminal based on the verifier's policy.

The server is required to trust the client capture-data.

The client and the verifier are required to guarantee TTP process stability.

Taking account of privacy protection policies, a TTP is required to receive permission in advance to present both user data and verifier data.

This is the same as the centre model, except that the template is managed by TTP.

7) *Comparison outsourcing by client model*

Figure 8 illustrates a comparison outsourcing by client model.

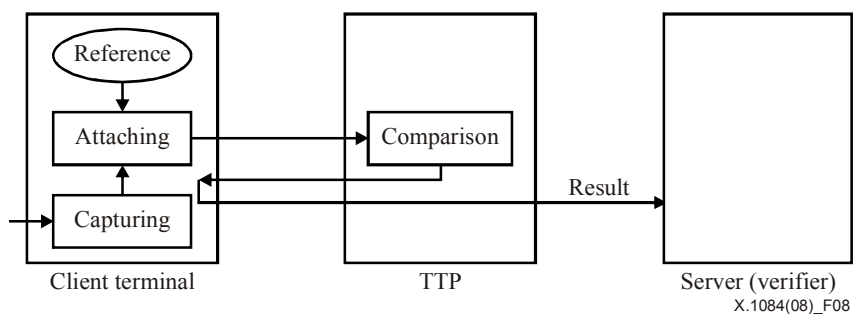


Figure 8 – Comparison outsourcing by client model

When the verifier requests verification, the client terminal first acquires sample data and requests a comparison with the user's template from a TTP. Therefore, the TTP transfers the result to the client terminal. It is then transferred from the client terminal to the verifier.

In this model, we assume that a client cannot trust the server, but a client can trust a third party. Accordingly, a client, as well as a server, request a TTP to perform comparison procedure.

Template ID information is required in the comparison result.

This model requires a TTP to process client capture-data.

8) *Comparison outsourcing by server model*

Figure 9 illustrates a comparison outsourcing by server model.

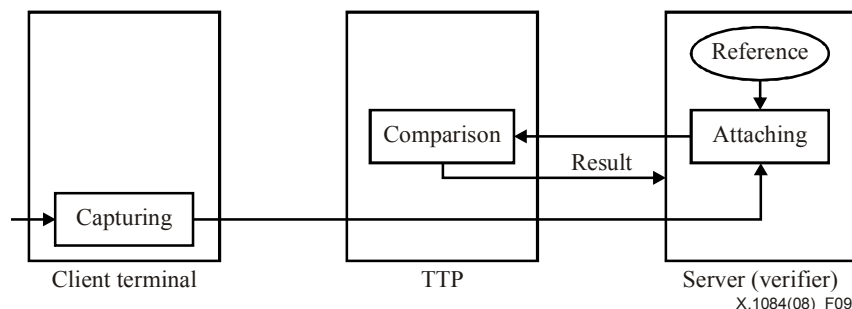


Figure 9 – Comparison outsourcing by server model

The verifier requests sample data from the client terminal, and verification with the user's template to TTP. TTP then achieves the comparison, and transfers the result to the verifier.

For this model, we assume a server cannot handle the cost of multiple biometric modes, so it requests a TTP to handle biometric modes.

This model requires trust in the data captured by a client.

Taking account of privacy protection, a server receives advance approval from a user to outsource the comparison to a TTP.

Template ID information is required in the comparison result.

9) *Storage and comparison outsourcing model*

Figure 10 illustrates the storage and comparison outsourcing model.

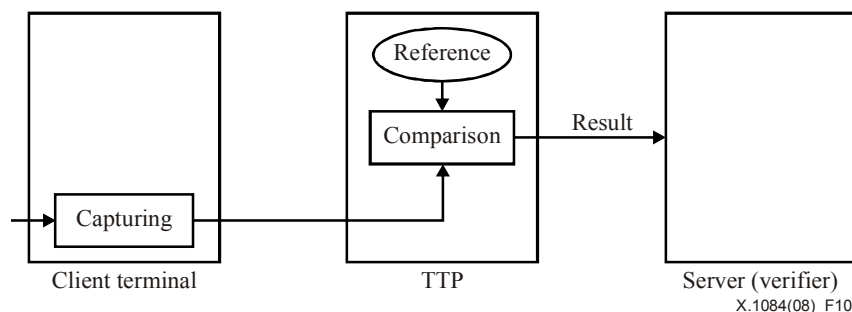


Figure 10 – Storage and comparison outsourcing model

For this model, we assume that client resources (memory, disk, etc.), are insufficient, and that the server-side has no comparison ability, meaning the client does not trust the server.

This can be compared with the outsourcing by client model and outsourcing by server model.

In the former case, the verifier requests verification from the client terminal. The client terminal captures the sample data, and requests the verification result from a TTP with the captured sample data. Therefore, the TTP compares sample data with the registered user's template, and transfers the result to the client terminal. The final result is transferred from the client terminal to the verifier.

In the latter case, on the other hand, the verifier requests sample data from the client terminal, and it obtains the captured sample data from the client terminal. The verifier requests, the verification result from a TTP. The result is transferred to the verifier.

A user registers its biometric reference template in advance with a TTP and outsources the comparison processing to the same TTP.

This model also requires the TTP, (server), to trust client capture-data. Template ID information is required in the comparison result.

8 Security threats for each models

Table 2 shows security threats allowing illegal use by an unauthorized user and possible counter measures for each model.

Table 2 – Security threats for each model

Model	Threats allowing illegal use by unauthorized users	Possible countermeasures
1) Local	– They may use illegal biometric reference template data – They may use an illegal comparison program – They may replace illegal capture data, such as stolen or altered data – They may replace illegal result data	– Reference with signature – Secure client – Client authentication
2) Download	– They may use an illegal comparison program – They may replace illegal capture data, such as stolen or altered data – They may replace illegal result data	– Secure client – Client authentication
3) Attached	– They may use illegal biometric reference template data – They may replace illegal capture data, such as stolen or altered data	– Reference with signature – Secure client – Client authentication
4) Centre	– They may replace illegal capture data, such as stolen or altered data	– Secure client – Client authentication
5) Reference management on TTP for local	– They may use an illegal comparison program – They may replace illegal capture data, such as stolen or altered data – They may replace illegal result data	– Secure client – Client authentication
6) Reference management on TTP for centre	– They may replace illegal capture data, such as stolen or altered data	– Secure client – Client authentication

Table 2 – Security threats for each model

Model	Threats allowing illegal use by unauthorized users	Possible countermeasures
7) Comparison outsourcing by client	– They may use illegal biometric reference template data – They may use an illegal comparison program – They may replace illegal capture data, such as stolen or altered data	– Reference with a signature – Secure client – Client authentication
8) Comparison outsourcing by server	– They may replace illegal capture data, such as stolen or altered data	– Secure client – Client authentication
9)-1 Storage and comparison outsourcing by client	– They may replace illegal capture data, such as stolen or altered data – They may use an illegal party	– Secure client – Client authentication – Validity check to the TTP of the client
9)-2 Storage and comparison outsourcing by server	– They may replace illegal capture data, such as stolen or altered data – Unauthorized user may request an illegal party	– Secure client – Client authentication – Validity check to the TTP of the server

In addition, Figure 11 illustrates specific threats for cooperation of PKI, and a biometric authentication, such as the environment of this Recommendation. For a model that incorporates PKI, there remains the threat of a private key leaking out, and that somebody may use it illegally, such as case a). Even without private key leakages, such as case b), the relation between PKI information and biometric template information may not be guaranteed. Therefore, the biometric template information should require the identifying information of the PKI certificate and the validity information for itself. This Recommendation assumes the use of a biometric certificate (BC) of [ITU-T X.1089] as the biometric template.

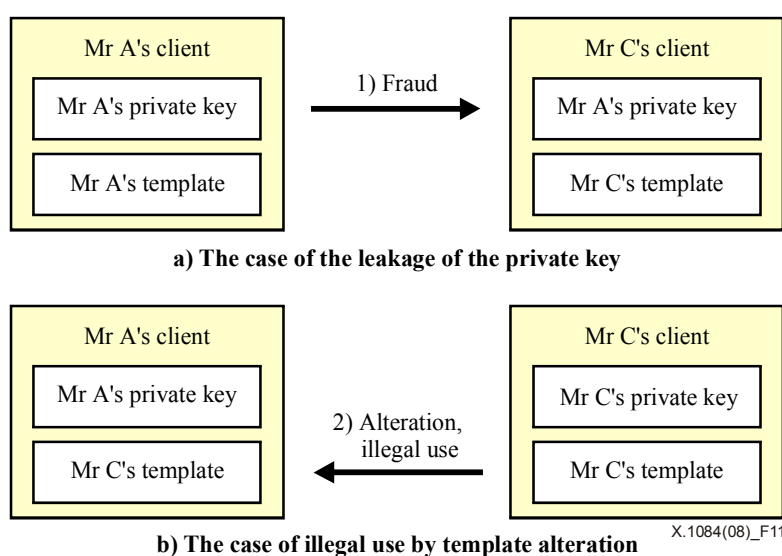


Figure 11 – Risk of illegal use by falsification of the template

Possible threats to private information protection on a server or with a TTP are listed below:

- Biometric and private information fraud through an illegal server.
- Biometric and private information fraud through an illegal TTP.
- Biometric and private information fraud through connecting to an illegal network.

Countermeasures are listed below:

- Authenticate a server using the PKI.
- Authenticate a TTP using the PKI.
- Encrypt the session key exchange by use of TLS protocol as specified in [IETF RFC 4346].

9 General requirements

1) *Requirement for authentication policies in the verifier*

The following are conditions for a security policy which is required to offset threats to server-side biometrics.

- A statement of the acceptable biometric technologies (modality, device, algorithm), with ordering by use of priorities
- A statement of the acceptable false acceptance level (threshold for each biometric technology, number of attempts)
- A statement of the acceptable client security level ([ISO/IEC 15408-1] authentication protection profile (PP), evaluation assurance level (EAL) for PP)
- A statement of the acceptable biometric system models (9 models – see clause 7) with priorities
- A statement of the acceptable cryptography procedures (method, key length) with priorities
- A statement of the acceptable biometric data quality (reference and captured sample quality)
- A statement of the acceptable biometric TTP site for the use of TTP models.

In addition to the security conditions listed above, the verifier should manage the following parameters:

- Profiles for suitable biometric technology when comparison is performed on an application server, such as the attached model, centre model, or storage outsourcing by server model
- Profiles of biometric technology (device, algorithm) which should be approved in advance.

2) *Authentication parameter in the client*

There are two types of parameter for the authentication parameter in the client. Those are the profile of the client terminal and the privacy settings by client user. The profile of the client terminal should be certified by TTP, with regard to its performance and security assurance level. This certificate or report is provided in BDC in [ITU-T X.1089].

- Regarding a profile of client terminal:
 - Profiles (quality as specified in [ISO/IEC 19785-1]) for collectable biometric sample using the client-side
 - Profiles (authentication accuracy as specified in [ISO/IEC 19795-1]) for suitable biometric modes using the client-side
 - Type of biometric reference template held by a client user
 - Client security guarantee (EAL, functional level as specified in [ISO/IEC 15408-1])
 - Client cryptography (method, key length).

- Regarding privacy settings by client user:
 - System model acceptable to a user (9 models) for each service provider
 - Application server, acceptable to a user, that presents biometric sample
 - TTP, acceptable to a user, that presents biometric sample.

10 General protocol

This Recommendation provides a handshake protocol as specified in [IETF RFC 4346] for a negotiation of the policies in the verifier and the functions and acceptable system model in the client. Furthermore, this Recommendation provides a transfer protocol for a biometric data for each negotiated system model (see Figure 12).

However, the transfer protocol for the biometric data has variable implementations. This clause describes the handshake protocol, and clause 11 describes the requirements for biometric transportation data for each system model. The transfer protocol is described in Appendices I and II.

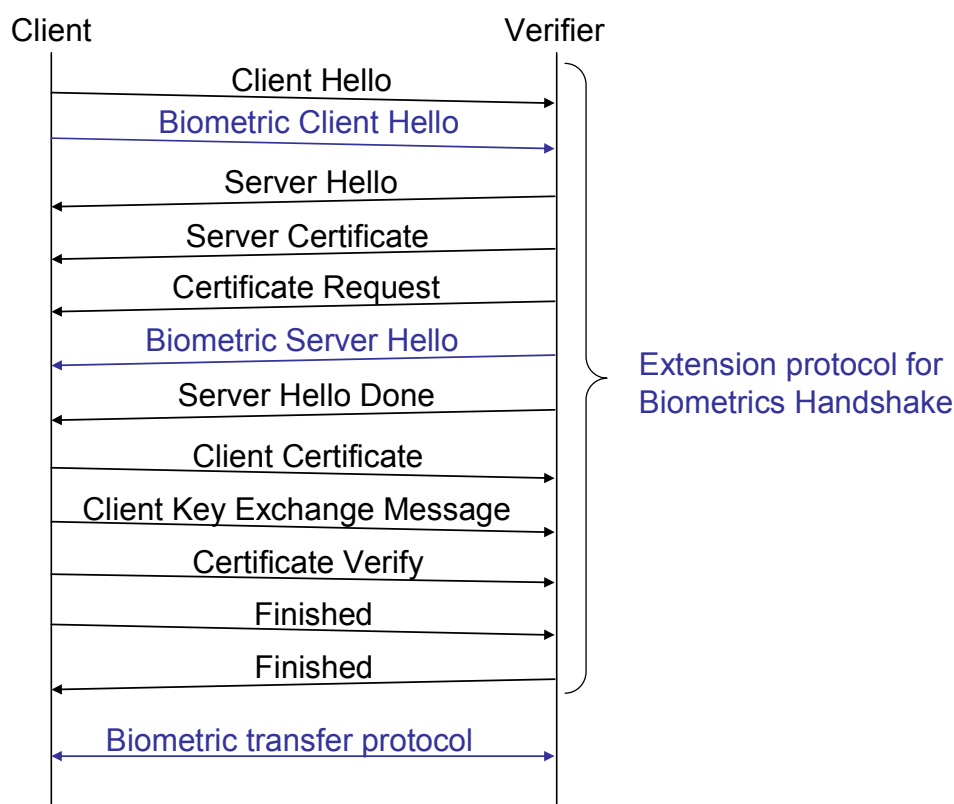


Figure 12 – Extended TLS protocol for biometrics handshake

The following subclauses describe the extended TLS protocol for a biometrics handshake, as specified in [IETF RFC 4366].

10.1 Requirement of the biometrics handshake protocol

10.1.1 Extended protocol for biometric handshake protocol

This Recommendation recommends the use of two extended handshake messages, namely, "Biometric Client Hello" and "Biometric Server Hello". These messages are described in clauses 10.1.2 and 10.1.3. The extended handshake message structures are as follows:

```

Handshake ::= SEQUENCE {
    type          HANDSHAKE.&id({Handshakes}),
    value         HANDSHAKE.&Type({Handshakes}){@type}
}
  
```

```

    }

    HANDSHAKE ::= CLASS {
        &Type,
        &id HandshakeType    UNIQUE
    }
    WITH SYNTAX {
        &Type IDENTIFIED BY &id
    }

    HandshakeType ::= INTEGER{
        hello-request          (0),
        client-hello           (1),
        server-hello           (2),
        certificate             (11),
        server-key-exchange     (12),
        certificate-request     (13),
        server-hello-done       (14),
        certificate-verify      (15),
        client-key-exchange     (16),
        finished                (20),
        biometric-client-hello   (101),
        biometric-server-hello   (102)
    } (0..255)

    Handshakes HANDSHAKE ::= {
        helloRequest |
        clientHello |
        serverHello |
        certificateList |
        serverKeyExchange |
        certificateRequest |
        serverHelloDone |
        certificateVerify |
        clientKeyExchange |
        finished |
        biometricClientHello |
        biometricServerHello,
        ...
    }

```

10.1.2 Biometric Client Hello

A list is sent from the client to the verifier. The list contains client-supported biometric models and user-approved biometric system models.

```

    biometricClientHello HANDSHAKE ::= {
        BiometricClientHello    IDENTIFIED BY biometric-client-hello
    }

    BiometricClientHello ::= SEQUENCE SIZE(1..MAX) OF BiometricMethod

```

Each biometric method contains the biometric type, a biometric function provider, a network authentication model and third-party information. The third-party information is required by the TTP models. It specifies the network address for the third party (URI as specified in [IETF RFC 3986]).

```

    BiometricMethod ::= SEQUENCE{
        biometricType           BiometricType,
        biometricFunctionProvider BSP-BFP-Schema,
        networkAuthenticationModel NetworkAuthenticationModel,
        thirdPartyInfo           UTF8String
    }

```

The biometric type and the BSP-BFP-Schema are defined in standard specifications of ISO/IEC JTC 1/SC 37 as specified in [ISO/IEC 19784-1]. This Recommendation complies with these reference standards for the following:

```

    BiometricType ::= BioAPI-BIR-BIOMETRIC-TYPE

    BioAPI-BIR-BIOMETRIC-TYPE ::= BIT STRING {
        typeMultipleBiometricTypes    (0),

```

```

typeFace                (1),
typeVoice                (2),
typeFinger               (3),
typeIris                 (4),
typeRetina               (5),
typeHandGeometry         (6),
typeSignatureSign        (7),
typeKeystroke             (8),
typeLipMovement          (9),
typeGait                 (12),
typeVein                 (13),
typeDNA                  (14),
typeEar                  (15),
typeFoot                 (16),
typeScent                 (17),
typeOther                 (30),
typePassword             (31)
} (SIZE(32))

BSP-BFP-Schema ::= CHOICE {
    bSPSchema    BioAPI-BSP-SCHEMA,
    bFPSchema    BioAPI-BFP-SCHEMA
}

BSP-BFP-Schemas ::= SEQUENCE(SIZE(1..MAX)) OF BSP-BFP-Schema

```

This Recommendation describes network authentication models in clause 7. Each model is identified by the following numbers:

```

NetworkAuthenticationModel ::= BIT STRING {
    no-value                (0), -- no selection --
    local-model             (1),
    download-model          (2),
    attached-model          (3),
    center-model            (4),
    ref-onttp-for-local-model (5),
    ref-onttp-for-center-model (6),
    comparison-outsourcing-by-client-model (7),
    comparison-outsourcing-by-server-model (8),
    storage-comparison-outsourcing-by-client-model (9),
    storage-comparison-outsourcing-by-server-model (10)
} (SIZE (0..10))

```

10.1.3 Biometric Server Hello

Based on the server's security policy (user authentication policy), a supported biometric model and an approved biometric system are selected from a client-user list, and sent to the client by the verifier.

```

biometricServerHello    HANDSHAKE ::= {
    BiometricServerHello IDENTIFIED BY biometric-server-hello
}

BiometricServerHello ::= SEQUENCE {
    request    BiometricAuthenticationRequest
}

```

If the Biometric Client Hello does not have permissive network authentication models or biometric types, the server sends an alert message to the client (refer to clause 10.2) using one of the following description values.

```

AlertDescription ::= ENUMERATED{
    ...,
    unacceptable-model                (115), -- Extension item
    unacceptable-biometrics           (116), -- Extension item
    ...
}

```

The Biometric Authentication Request message is filled out according to the server's biometric authentication policy. It lists the selected biometric method, requests the score level of result in biometric comparison (request FMR as specified in [ISO/IEC 19784-1]), requests the number of

trials for biometric comparison process (request trial number), and requests the sample quality for biometric data.

```

Quality ::= INTEGER(0..100)

BiometricAuthenticationRequest ::= SEQUENCE {
    biometricMethod      BiometricMethod,
    requestFMR           BioAPI-FMR, -- (32-bit integer value:requestFMR/231-1)
    requestTrialNumber   INTEGER(1..15),
    requestQuality       Quality,
    requestTemplateData  XtsmTemplate OPTIONAL
    -- for download model (no value available)
}

```

10.2 Alert protocol for biometric handshake

The following describes the extended alert protocol of TLS for this Recommendation. This Recommendation defines the alert description number, from 115 to 117, in order to avoid a conflict with the TLS Extension definition as specified in [IETF RFC 4366].

- unsupported-extension (110)

This alert is the message of TLS extension [IETF RFC 4366]. This alert may be returned if the verifier received this extended biometrics handshake protocol in an unsupported fashion. This message is always fatal.

- unacceptable-model (115)

This alert may be returned if the verifier received the unacceptable models only (those that do not conform to the verification policy of the verifier). This message is always fatal.

- unacceptable-biometrics (116)

This alert may be returned if the verifier received unacceptable biometrics, modalities, biometric algorithms, or biometric devices only, which do not conform to the verification policy of the verifier. This message is always fatal.

- unsupported-biometrics (117)

In the server comparison models, the server received unsupported biometric algorithms only, which are not supported by the verifier. This message is always fatal.

```

Alert ::= SEQUENCE {
    level      AlertLevel,
    description AlertDescription
}

AlertLevel ::= ENUMERATED {
    warning (1),
    fatal (2)
}

AlertDescription ::= ENUMERATED {
    close-notify           (0),
    unexpected-message     (10),
    bad_record-mac         (20),
    decryption-failed      (21),
    record-overflow        (22),
    decompression-failure  (30),
    handshake-failure      (40),
    --41 is not defined, for historical reasons
    bad-certificate        (42),
    unsupported-certificate (43),
    certificate-revoked     (44),
    certificate-expired     (45),
    certificate-unknown     (46),
    illegal-parameter      (47),
    unknown-ca             (48),
    access-denied          (49),
}

```

```

decode-error                (50),
decrypt-error               (51),
export-restriction          (60),
protocol-version            (70),
insufficient-security       (71),
internal-error              (80),
user-canceled               (90),
no-renegotiation            (100),
unsupported-extension        (110),
certificate-unobtainable    (111),
unrecognized-name           (112),
bad-certificate-status-response (113),
bad-certificate-hash_value  (114),
unacceptable-model          (115),      -- Extension item for TSM
unacceptable-biometrics     (116),      -- Extension item for TSM
unsupported-biometrics        (117),      -- Extension item for TSM
}

```

10.3 Implementation of the extended protocol

10.3.1 Modification of the TLS record layer protocol

TLS protocol is able to be extended by using [IETF RFC 4366], with IETF consensus.

This Recommendation is based on [IETF RFC 4366] for this biometrics handshake protocol. [IETF RFC 4366] extended the message contents of Client Hello and Server Hello for TLS Extension protocol. However, this Recommendation does not comply with [IETF RFC 4366] without IETF consensus. Therefore, this Recommendation modifies the Record Layer of TLS as an original protocol, as outlined below:

```

TSMPlainText ::= SEQUENCE {
    protocolID ProtocolIdentifier,
    version    ProtocolVersion,
    fragment   CHOICE {
        change-cipher-spec-opaque ChangeCipherSpec,
        alert-opaque               Alert,
        biometric-handshake-opaque Handshake,
        application-data-opaque    ApplicationData
    }
}

ProtocolIdentifier ::= UINT8

ProtocolVersion ::= SEQUENCE {
    major  UINT8,
    minor  UINT8
}

ChangeCipherSpec ::= ENUMERATED {
    change-cipher-spec(1),
    ...
}

ApplicationData ::= Opaque

TSMCipherText ::= SEQUENCE {
    protocolID ProtocolIdentifier,
    type       ContentType,
    version    ProtocolVersion,
    fragment   CHOICE {
        stream GenericStreamCipher,
        block  GenericBlockCipher
    }
}

ContentType ::= ENUMERATED {
    change-cipher-spec (20),
    alert               (21),
    handshake           (22),
    application-data    (23),
    ...
}

```

10.3.2 General extension mechanism for the biometrics handshake protocol

These extended messages are described as follow:

1) Extended Client Hello (Client to Server)

As with the TLS Extension handshake protocol, a client sends a list of supported encryption methods and cipher suites (challenge code 1).

Extended Client Hello is given as follows:

```
clientHello HANDSHAKE ::= {
    ClientHello IDENTIFIED BY client-hello
}

ClientHello ::= SEQUENCE{
    client-version      ProtocolVersion,      -- Same as TLS protocol
    random              Random,                -- Same as TLS protocol
    session-id          SessionID,             -- Same as TLS protocol
    cipher-suites       CipherSuite,           -- Same as TLS protocol
    compression-methods CompressionMethod,    -- Same as TLS protocol
    ...,
    ...,
    client-hello-extension-list ExtensionValues -- Same as TLS Extension
}

EXTENSION ::= CLASS {
    &id ExtensionType UNIQUE,
    &Type
}
WITH SYNTAX {
    &Type IDENTIFIED BY &id
}

ExtensionType ::= INTEGER(0..66535)

Extensions EXTENSION ::= {
    ...
}

ExtensionValues ::= SEQUENCE OF ExtensionValue

ExtensionValue ::= SEQUENCE {
    extension-type      EXTENSION.&id({Extensions}),
    extension-data      EXTENSION.&Type({Extensions}){@extension-type}
}
```

2) Extended Server Hello (Server to Client)

As with the TLS Extension handshake protocol, a server selects (and sends) an encryption method from the list of encryption methods from the client, generates a cipher suite (challenge code 2), and then sends back a server hello message.

```
serverHello HANDSHAKE ::= {
    ServerHello IDENTIFIED BY server-hello
}

ServerHello ::= SEQUENCE{
    server-version      ProtocolVersion,      -- Same as TLS protocol
    random              Random,                -- Same as TLS protocol
    session-id          SessionID,             -- Same as TLS protocol
    cipher-suites       CipherSuite,           -- Same as TLS protocol
    compression-methods CompressionMethod,    -- Same as TLS protocol
    ...,
    ...,
    server-hello-extension-list ExtensionValues -- Same as TLS Extension
}
```

11 Requirements of the biometric transportation stage for each model

The following clauses describe the requirements of the exchange of data contents for all the models.

11.1 Local model

Figure 2 outlines a local model. Table 3 lists four items required for the result data.

Table 3 – Items required for local model

#	Items	Details
1	Information on biometric service/function providers (BSPs) for client processes	The verifier needs to know how the biometric process operates for the client. It divides some functions: capturing, pre-processing, and comparing. In BioAPI as specified in [ISO/IEC 19784-1], the services and functions define the BSPs (or BFPs). There are varying types of BSPs that provide all functions or only one function, or pre-processing and comparison functions. If the client anticipates all types of functions to be provided, it uses various BSPs together. Therefore, this item should be made able to define various BSPs. The verifier can then check whether the security level and the performance of all BSPs for the verifier's service are sufficiently high.
2	Reference template information for client processes	The verifier needs to know whose reference template is being used. However, this model is of the privacy protection type for the reference template. Therefore, this item should at minimum include reference template ID information. The verifier can then check the revocation template.
3	Sample data quality and comparison score (similarity) for client processes	The verifier needs sample data quality for input data of a comparison process as well as the comparison results in order to enable appropriate risk management of the verifier's service.
4	Security code of client processes	The verifier requires the assurance of the integrity of the client processes.

The following are examples for this message:

```

BDforLocalModel ::= SEQUENCE {
    biometricClientProcess      BiometricClientProcess,
    digitalSignature            SignedData,
    acforBioOnClient            ACBioInformation OPTIONAL
    -- see ISO/IEC 24761
}

BiometricClientProcess ::= SEQUENCE {
    bFPSchemaForClientProcess  SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID                 TemplateID,
    sampleQuality               Quality, -- specified in ISO/IEC 19784-1
    score                       BioAPI-FMR -- specified in ISO/IEC 19784-1
}

TemplateID ::= SEQUENCE {
    certificateIssuerName      Name, -- see Rec. ITU-T X.509
    serialNumber               CertificateSerialNumber, -- see Rec. ITU-T X.509
    templateinfo               TemplateInfo
}

TemplateInfo ::= SEQUENCE {
    biometricType              BiometricType,
    creator                    UTF8String,
    createdBFPSchema           BSP-BFP-Schema, -- specified in ISO/IEC 19784-1
    templateID                 CertificateIDInformation
    -- such as CertificateSerialNumber (no value available)
}

```

CertificateIDInformation ::= CertificateSerialNumber

11.2 Download model

Figure 3 outlines a download model.

It contains the same items required for the results data as the local model (see Table 3).

The following is a sample of Biometric Data for the download model on Biometric Verify. The content is the same as for the local model.

```
BDforDownloadModel ::= SEQUENCE {
    biometricClientProcess    BiometricClientProcess,
    digitalSignature          SignedData,
    aCforBioOnClient          ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}
```

Table 4 lists the items required for download data from the server in the download model. The verifier requires the template ID (=Certificate ID) in order to retrieve the reference template DB for this message (Client to Server). However, the verifier already has the certificate ID on the TLS handshake.

Table 4 – Items required for download data for download model

#	Item	Details
1	Reference template	The client requires the reference template for the comparison process (Server to Client).

This message includes the message of the Biometric Server Hello.

```
BiometricServerHello ::= SEQUENCE {
    request    BiometricAuthenticationRequest
}

BiometricAuthenticationRequest ::= SEQUENCE {
    biometricMethod    BiometricMethod,
    requestFMR          BioAPI-FMR, -- (32-bit integer value:requestFMR/231-1)
    requestTrialNumber  INTEGER(1..15),
    requestQuality       Quality,
    requestTemplateData  XtsmTemplate OPTIONAL
    -- for download model (no value available)
}
```

11.3 Attached model

Figure 4 outlines an attached model.

Table 5 lists the items required for the attached data for the attached model.

Table 5 – Items required for attached data for attached model

#	Items	Details
1	Reference template	The verifier requires the reference template for the comparison process in the server.
2	Sample data	The verifier requires sampling data and sampling device information. If the sampling data complies with [ISO/IEC 19784-1], then the data (BIR) contains BSP information. The verifier can then check the security level and the quality of the sampling data from the sensor device for the comparison process.
3	Security code of client process	The verifier requires the assurance of the integrity of the client processes.

The following is an example of this message:

```

BDforAttachedModel ::= SEQUENCE {
    templateData      XtsmTemplate,
    sampleData        SampleData,          -- BIR: BioAPI defined format --
    digitalSignature   SignedData,
    aCforBioOnClient  ACBioContentInformation OPTIONAL
                    -- see ISO/IEC 24761
}

```

11.4 Centre model

Figure 5 outlines a centre model. Table 6 lists the items required for the centre model.

Table 6 – Items required for the centre model

#	Item	Details
1	Sample data	The verifier requires sampling data and sampling device information. If the sampling data complies with [ISO/IEC 19784-1], then the data (BIR) contains BSP information. The verifier can then check the security level and the quality of the sampling data from the sensor device for the comparison process.
2	Security code of client process	The verifier requires the assurance of the integrity of the client processes.

The following is an example of this message:

```

BDforCenterModel ::= SEQUENCE {
    sampleData        SampleData,          -- BIR: BioAPI defined format --
    digitalSignature   SignedData,
    aCforBioOnClient  ACBioContentInformation OPTIONAL
                    -- see ISO/IEC 24761
}

```

11.5 Reference management on TTP for local model

Figure 6 outlines the reference management on TTP for the local model.

It has the same items required for the result data as the local model, except for the addition of reference registered third-party information and the process information on TTP.

The following is an example of this message:

```

BDforRefOnTTPforLocalModel ::= SEQUENCE {
    thirdPartyInfo      UTF8String,
    biometricClientProcess BiometricClientProcess,
}

```

```

aCforBioOnTTP          ACBioInformation,
digitalSignaturebyClient SignedData,
aCforBioOnClient        ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

```

Table 7 lists the items required for the download data from TTP for this model. The TTP requires a template ID (=Certificate ID) in order to retrieve a reference template DB for this message (Client to TTP). However, it should maintain the secure transportation between TTP and the client. Therefore, TTP already includes the certificate ID on the TLS handshake.

Table 7 – Items required for download data from TTP

#	Item	Details
1	Reference template	The client requires the reference template in TTP for the comparison process on the client terminal (TTP to client).
2	Security code of TTP process	The client and the verifier require an assurance of the integrity of the TTP processes.

The following is an example of this message:

```

BiometricTTPProcess ::= SEQUENCE {
    templateData      XtsmTemplate,
    digitalSignature   SignedData,
    aCforBioOnTTP      ACBioContentInformation OPTIONAL
                      -- see ISO/IEC 24761
}

```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP, or no reference template in TTP.

11.6 Reference management on TTP for centre model

Figure 7 outlines reference management on TTP for the centre model.

It has the same items as required for the captured data as the centre model, except for the addition of the reference registered third-party information.

The following is an example of this message:

```

BDforRefOnTTPforCenterModel ::= SEQUENCE {
    thirdPartyInfo    UTF8String,
    sampleData        SampleData,      -- BIR: BioAPI defined format --
    digitalSignature   SignedData,
    aCforBioOnClient  ACBioContentInformation OPTIONAL
                      -- see ISO/IEC 24761
}

```

Tables 8 and 9 list the items required for the download data from TTP for this model.

Table 8 – Items required for download request to TTP

#	Item	Details
1	Reference template ID	TTP requires the template ID (=Certificate ID) in order to retrieve a reference DB (Server to TTP). It should maintain the secure transportation between TTP and the server.

The following is an example of this message:

```

TTPRequestRefOnTTPforCenterModel ::= SEQUENCE {
    templateID        TemplateID
}

```

Table 9 – Items required for download data from TTP

#	Item	Details
1	Reference template	The verifier requires the reference template in TTP for the comparison process on the server (TTP to Server).
2	Security code of TTP process	The verifier requires an assurance of the integrity of the TTP processes.

The following is an example of this message:

```

TTPResponseRefOnTTPforCenterModel ::= SEQUENCE {
    templateData      XtsmTemplate,
    digitalSignature   SignedData,
    aCforBioOnTTP     ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP, or no reference template in TTP.

11.7 Comparison outsourcing by client model

Figure 8 outlines comparison outsourcing by the client model.

1) Client to TTP

**Table 10 – Items required for data attached to TTP
for comparison outsourcing by client model**

#	Items	Details
1	Reference template	TTP requires a reference template in the client terminal for the comparison process in TTP.
2	Sample data	TTP requires sampling data on the client terminal for the comparison process in TTP. The sampling data should comply with [ISO/IEC 19784-1] (BIR). The BIR format contains BSP information.

The following is an example of this message:

```

TTPRequestCobyClientModel ::= SEQUENCE {
    templateData      XtsmTemplate,
    sampleData        SampleData      -- BIR: BioAPI defined format --
}

```

2) TTP to Client

**Table 11 – Items required for comparison result data to client
for comparison outsourcing by client model**

#	Items	Details
1	Information on BSP(s) for TTP process	The verifier needs to know how the biometric process operates in TTP. This item clarifies information on BSPs in the TTP process.
2	Sample data quality and comparison score (similarity) for TTP process	The verifier requires sample data quality for input data of the comparison process, and the results of the comparison to enable appropriate risk management of the verifier's services.
3	Security code of TTP process	The client & the verifier require the assurance of the integrity of TTP processes.

The following is an example of this message:

```
TTPResponseCObyClientModel ::= SEQUENCE {
    bFPSchemaOnTTPProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID             TemplateID,
    sampleQuality           Quality,
    score                   BioAPI-FMR,
    digitalSignature        SignedData,
    aCforBioTTP             ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}
```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP.

3) Client to Server

Table 12 – Items required for local model

#	Items	Details
1	TTP information	The verifier needs to know who the outsourcer is, and who operates the biometric comparison process, so as to enable appropriate risk management of the verifier's services.
2	Information on BSP(s) for client & TTP processes	The verifier needs to know how the biometric process operates for the client. It divides some functions: capturing, pre-processing, and comparing. In [ISO/IEC 19784-1], the services and functions define the BSPs (or BFPs). There are varying types of BSPs that provide all functions or only one function, or pre-processing and comparison functions. If the client anticipates all types of functions to be provided, it uses various BSPs together. Therefore, this item should be made available to define various BSPs. The verifier can then check whether the security level and the performance of all BSPs for the verifier's service are sufficiently high.
3	Reference template information for client process	The verifier needs to know whose reference template is being used. However, this model is of the privacy protection type for the reference template. Therefore, this item should at minimum include reference template ID information. The verifier can then check the revocation template.
4	Sample data quality and comparison score (similarity) for TTP processes	The verifier needs sample data quality for input data of a comparison process as well as the comparison results in order to enable appropriate risk management of the verifier's service.
5	Security code of all processes	The verifier requires the assurance of the integrity of client and TTP processes.

The following is an example of this message:

```
BDforCObyClientModel ::= SEQUENCE {
    bFPSchemaForClientProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    thirdPartyInfo            UTF8String,
    bFPSchemaforTTPProcess    SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID                 TemplateID,
    sampleQuality              Quality,
    score                      BioAPI-FMR,
    digitalSignaturebyClient   SignedData,
    digitalSignaturebyTTP      SignedData,
    aCforBioOnClient           ACBioContentInformation OPTIONAL,
    aCforBioOnTTP              ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}
```

}

11.8 Comparison outsourcing by server model

Figure 9 outlines comparison outsourcing by the server model.

1) Client to Server

Table 13 – Items required for comparison outsourcing by server model

#	Item	Details
1	Sample data	The verifier requires sampling data and sampling device information. If the sampling data complies with [ISO/IEC 19784-1], then the data (BIR) contains BSP information. The verifier can then check the security level and the quality of the sampling data from the sensor device for the comparison process.
2	Security code of client processes	The verifier requires the assurance of the integrity of the client processes.

The following is an example of this message:

```
BDforCObyServerModel ::= SEQUENCE {
    sampleData      SampleData,      -- BIR: BioAPI defined format --
    digitalSignature SignedData,
    acforBiometrics ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}
```

2) Server to TTP

Table 14 – Items required for attached data for comparison outsourcing by server model

#	Items	Details
1	Reference template	TTP requires a reference template for the comparison process in TTP.
2	Sample data	TTP requires a sample BIR for the comparison process in TTP.

The following is an example of this message:

```
TTPRequestCObyServerModel ::= SEQUENCE {
    templateData XtsmTemplate,
    sampleData SampleData      -- BIR: BioAPI defined format --
}
```

3) TTP to Server

Table 15 – Items required for comparison outsourcing by server model

#	Items	Details
1	Information on BSP(s) for TTP process	The verifier needs to know how the biometric process operates in the client. TTP must obtain comparison results from the received data. This item clarifies information on BSPs in the TTP process.
2	Sample data quality and comparison score (similarity) for TTP process	The verifier requires sample data quality for input data of the comparison process and the comparison results, so as to enable appropriate risk management of the verifier's services.
3	Security code of TTP process	The verifier requires the assurance of the integrity of TTP processes.

The following is an example of this message:

```

TTPResponsebyServer ::= SEQUENCE {
    bFPSchema          SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID         TemplateID,
    sampleQuality       Quality,
    score              BioAPI-FMR,
    digitalSignaturebyTTP SignedData,
    aCforBioOnTTP      ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP, or no reference template in TTP.

11.9 Storage and comparison outsourcing model

Figure 10 outlines the storage and comparison outsourcing model.

This model has two further detailed versions. The first is outsourcing by the client. The second is outsourcing by the server. The following describes the complete data components for these detailed models.

11.9.1 Outsourcing by the client

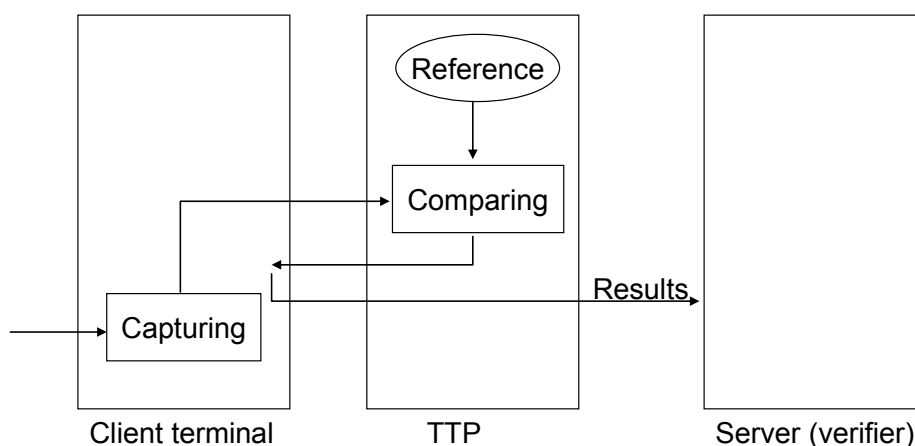


Figure 13 – Storage and comparison outsourcing by client model

- 1) Client to TTP

Table 16 – Items required for storage and comparison outsourcing by client model

#	Item	Details
1	Sample data	TTP requires sampling data and capturing device information for the comparison process in TTP. The BIR format as specified in [ISO/IEC 19784-1] contains the capturing BSP information.

The following is an example of this message:

```

TTPRequestSCobyClientModel ::= SEQUENCE {
    sampleData SampleData -- BIR: BioAPI defined format --
}

```

2) TTP to Client

Table 17 – Items required for storage and comparison outsourcing by client model

#	Items	Details
1	Information on BSP(s) for TTP processes	The verifier needs to know how the biometric process operates in the client. TTP must obtain comparison results from the received data. This item clarifies information on BSPs in the TTP process.
2	Sample data quality and comparison score (similarity) for TTP process	The verifier requires sample data quality for input data of the comparison process, and the comparison results so as to enable appropriate risk management of the verifier's services.
3	Security code of TTP process	The client & the verifier require the assurance of the integrity of TTP processes.

The following is an example of this message:

```

BDforSCobyCModel2 ::= SEQUENCE {
    bFPSchemaForTTPProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID             TemplateID,
    sampleQuality           Quality,
    score                   BioAPI-FMR,
    digitalSignatureByTTP   SignedData,
    aCforBioOnTTP           ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP, or no reference template in TTP.

3) Client to Server

Table 18 – Items required for storage and comparison outsourcing by client model

#	Items	Details
1	Information on BSP(s) for client & TTP processes	The verifier needs to know how the biometric process operates for the client. It divides some functions: capturing, pre-processing, and comparing. In [ISO/IEC 19784-1], the services and functions define the BSPs (or BFPs). There are varying types of BSPs that provide all functions or only one function, or pre-processing and comparison functions. If the client anticipates all types of functions to be provided, it uses various BSPs together. Therefore, this item should be made available to define various BSPs. The verifier can then check whether the security level and the performance of all BSPs for the verifier's service are sufficiently high.
2	Reference template information for client process	The verifier needs to know whose reference template is being used. However, this model is of the privacy protection type for the reference template. Therefore, this item should at minimum include reference template ID information. The verifier can then check the revocation template.

Table 18 – Items required for storage and comparison outsourcing by client model

#	Items	Details
3	Sample data quality and comparison score (similarity) for TTP process	The verifier requires sample data quality for input data of the comparison process and the comparison results so as to enable appropriate risk management of the verifier's services.
4	Security code of all processes	The verifier requires the assurance of the integrity of client and TTP processes.

The following is an example of this message:

```

BDforSCobyCModel3 ::= SEQUENCE {
    bFPSchema
    thirdPartyInfo
    bFPSchemaForTTPProcess
    templateID
    sampleQuality
    score
    digitalSignatureByClient
    digitalSignatureByTTP
    aCforBioOnClient
    aCforBioOnTTP
    ForClientProcess SEQUENCE SIZE(1..MAX)
    OF BSP-BFP-Schema,
    UTF8String,
    SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    TemplateID,
    Quality,
    BioAPI-FMR,
    SignedData,
    SignedData,
    ACBioContentInformation OPTIONAL,
    ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

```

11.9.2 Outsourcing by server

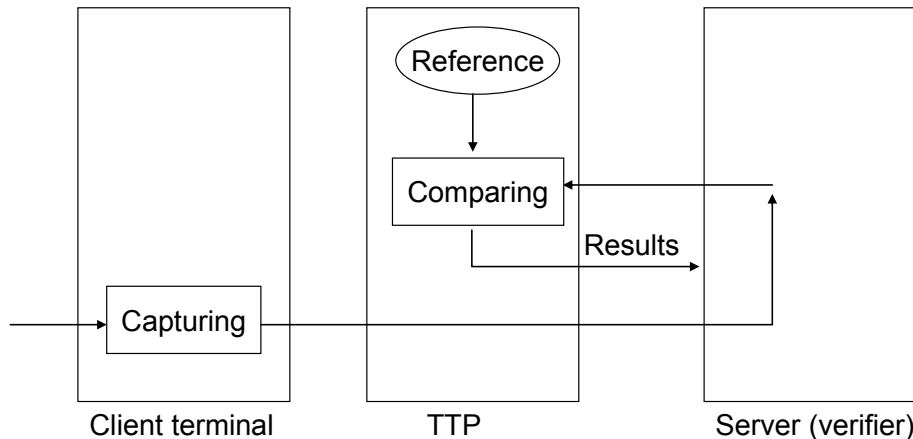


Figure 14 – Storage and comparison outsourcing by server model

1) Client to Server

Table 19 – Items required for storage and comparison outsourcing by server model

#	Item	Details
1	Sample data	The verifier requires sampling data and sampling device information. If the sampling data complies with [ISO/IEC 19784-1], then the data (BIR) contains BSP information. The verifier can then check the security level and the quality of the sampling data from the sensor device for the comparison process.
2	Security code of client process	The verifier requires assurance of the integrity of the client processes.

The following is an example of this message:

```

BDforSCobySModel ::= SEQUENCE {
    sampleData          SampleData, -- BIR: BioAPI defined format --
    digitalSignatureByClient SignedData,
    aCforBioOnClient    ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

```

2) Server to TTP

Table 20 – Items required for attached data for storage and comparison outsourcing by server model

#	Items	Details
1	Reference template ID	TTP requires the template ID (=Certificate ID) to retrieve a reference DB (Server to TTP). It should maintain the secure transportation between TTP and the server.
2	Sample data	TTP requires sampling data and capturing device information for the comparison process in TTP. The BIR format as specified in [ISO/IEC 19784-1] contains BSP information.

The following is an example of this message:

```

TTPRequestSCobyServerModel ::= SEQUENCE {
    templateID      TemplateID,
    sampleData      SampleData      -- BIR: BioAPI defined format --
}

```

3) TTP to Server

Table 21 – Items required for storage and comparison outsourcing by server model

#	Items	Details
1	Information on BSP(s) for TTP processes	The verifier needs to know how the biometric process operates in the client. TTP must obtain comparison results from the received data. This item clarifies information on BSPs in the TTP process.
2	Sample data quality and comparison score (similarity) for TTP process	The verifier requires sample data quality for input data of the comparison process and the comparison results so as to enable appropriate risk management of the verifier's services.
3	Security code of TTP process	The verifier requires the assurance of the integrity of TTP processes.

The following is an example of this message:

```

TTPResponseSCobyServer ::= SEQUENCE {
    bFPSchemaforTTPProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID             TemplateID,
    sampleQuality           Quality,
    score                   BioAPI-FMR,
    digitalSignatureByTTP   SignedData,
    aCforBioOnTTP           ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

```

Alert messages should be defined for illegal cases during the transmission between TTP, such as, for example, no response of TTP, or no reference template in TTP.

Annex A

ASN.1 definitions for modified TLS extension protocol

(This annex forms an integral part of this Recommendation)

```
TSM {itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0) protocol(0)
version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS
    BioAPI-BFP-SCHEMA, BioAPI-BSP-SCHEMA, BioAPI-FMR, BioAPI-BIR,
    BioAPI-BIR-BIOMETRIC-TYPE
        FROM BIP {joint-iso-itu-t(2) bip(41) modules(0) bip(0) version1(1)}
    BiometricCertificate
        FROM TAI {itu-t(0) recommendation(0) x(24) tai(1089) modules(0)
        framework(0) version1(1)}
    SignedData
        FROM X9-84-CMS {iso(1) identified-organization(3) tc68(133) country(16)
        x9(840) x9Standards(9) x9-84(84) module(0) cms(2) rev(1)}
    SignedDataACBio, ACBioContentInformation
        FROM AuthenticationContextForBiometrics
        {iso(1) standard(0) acbio(24761) module(1) acbio(2) version1(1)}
    DistinguishedName, Name
        FROM InformationFramework
        {joint-iso-itu-t ds(5) module(1) informationFramework(1) 5}
    Certificate, CertificateSerialNumber
        FROM AuthenticationFramework
        {joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 5};

UINT8 ::= INTEGER(0..255)

UINT16 ::= INTEGER(0..65535)

UINT24 ::= INTEGER(0..16777215)

UINT32 ::= INTEGER(0..4294967295)

UINT64 ::= INTEGER(0..18446744073709551615)

Opaque ::= OCTET STRING

BiometricType ::= BioAPI-BIR-BIOMETRIC-TYPE

SampleData ::= BioAPI-BIR

HandshakeType ::= INTEGER {
    hello-request (0),
    client-hello (1),
    server-hello (2),
    certificate-list (11),
    server-key-exchange (12),
    certificate-request (13),
    server-hello-done (14),
    certificate-verify (15),
    client-key-exchange (16),
    finished (20),
    biometric-client-hello (100),
    biometric-server-hello (101)
} (0..255)

HANDSHAKE ::= CLASS {
    &Type,
    &id HandshakeType UNIQUE
}
WITH SYNTAX {
    &Type IDENTIFIED BY &id
}
```

```

Handshake ::= SEQUENCE {
    type HANDSHAKE.&id({Handshakes}),
    value HANDSHAKE.&Type({Handshakes}){@type}}
}

Handshakes HANDSHAKE ::= {
    helloRequest |
    clientHello |
    serverHello |
    certificateList |
    serverKeyExchange |
    certificateRequest |
    serverHelloDone |
    certificateVerify |
    clientKeyExchange |
    finished |
    biometricClientHello |
    biometricServerHello,
    ...
}

biometricClientHello HANDSHAKE ::= {
    BiometricClientHello IDENTIFIED BY biometric-client-hello
}

BiometricClientHello ::= SEQUENCE(SIZE(1..MAX)) OF BiometricMethod

BiometricMethod ::= SEQUENCE {
    biometricType                BiometricType,
    biometricFunctionProvider     BSP-BFP-Schema,
    networkAuthenticationModel   NetworkAuthenticationModel,
    thirdPartyInfo               UTF8String
}

BSP-BFP-Schema ::= CHOICE {
    bSPSchema      BioAPI-BSP-SCHEMA,
    bFPSchema      BioAPI-BFP-SCHEMA
}

BSP-BFP-Schemas ::= SEQUENCE(SIZE(1..MAX)) OF BSP-BFP-Schema

NetworkAuthenticationModel ::= ENUMERATED {
    no-value                (0), -- no selection --
    local-model             (1),
    download-model          (2),
    attached-model          (3),
    center-model            (4),
    ref-onhttp-for-local-model (5),
    ref-onhttp-for-center-model (6),
    comparison-outsourcing-by-client-model (7),
    comparison-outsourcing-by-server-model (8),
    storage-comparison-outsourcing-by-client-model (9),
    storage-comparison-outsourcing-by-server-model (10),
    ...
}

biometricServerHello HANDSHAKE ::= {
    BiometricServerHello IDENTIFIED BY biometric-server-hello
}

BiometricServerHello ::= SEQUENCE {
    request      BiometricAuthenticationRequest
}

Quality ::= INTEGER(0..100)

BiometricAuthenticationRequest ::= SEQUENCE {
    biometricMethod BiometricMethod,
    requestFMR      BioAPI-FMR,
    -- (32-bit integer value:requestFMR/231-1)
    requestTrialNumber INTEGER(1..15),
    requestQuality      Quality,
    requestTemplateData XtsmTemplate OPTIONAL
    -- for download model (no value available)
}

```

```

Alert ::= SEQUENCE {
    level          AlertLevel,
    description    AlertDescription
}

AlertLevel ::= ENUMERATED {
    warning (1),
    fatal (2)
}

AlertDescription ::= ENUMERATED {
    close-notify                (0),
    unexpected-message          (10),
    bad-record-mac              (20),
    decryption-failed           (21),
    record-overflow             (22),
    decompression-failure       (30),
    handshake-failure           (40),
    -- 41 is not defined, for historical reasons
    bad-certificate             (42),
    unsupported-certificate      (43),
    certificate-revoked          (44),
    certificate-expired          (45),
    certificate-unknown          (46),
    illegal-parameter           (47),
    unknown-ca                  (48),
    access-denied               (49),
    decode-error                (50),
    decrypt-error               (51),
    export-restriction          (60),
    protocol-version            (70),
    insufficient-security       (71),
    internal-error              (80),
    user-canceled               (90),
    no-renegotiation            (100),
    unsupported-extension       (110),
    certificate-unobtainable    (111),
    unrecognized-name           (112),
    bad-certificate-status-response (113),
    bad-certificate-hash-value  (114),
    unacceptable-model          (115), -- Extension item for TSM
    unacceptable-biometrics     (116), -- Extension item for TSM
    unsupported-biometrics       (117) -- Extension item for TSM
}

TSMPlainText ::= SEQUENCE {
    protocolID ProtocolIdentifier,
    version    ProtocolVersion,
    fragment   CHOICE {
        change-cipher-spec-opaque    ChangeCipherSpec,
        alert-opaque                  Alert,
        biometric-handshake-opaque    Handshake,
        application-data-opaque       ApplicationData
    }
}

ProtocolIdentifier ::= UINT8

ProtocolVersion ::= SEQUENCE {
    major    UINT8,
    minor    UINT8
}

ChangeCipherSpec ::= ENUMERATED {
    change-cipher-spec (1),
    ...
}

ApplicationData ::= Opaque

TSMCipherText ::= SEQUENCE {
    protocolID ProtocolIdentifier,
    type        ContentType,
    version     ProtocolVersion,

```

```

        fragment CHOICE {
            stream GenericStreamCipher,
            block GenericBlockCipher
        }
    }

ContentType ::= ENUMERATED {
    change-cipher-spec (20),
    alert (21),
    handshake (22),
    application-data (23),
    ...
}

GenericStreamCipher ::= SEQUENCE {
    content Opaque(SIZE(0..65535)),
    mAC HASH{Opaque}
}

GenericBlockCipher ::= SEQUENCE {
    content Opaque(SIZE(0..65535)),
    mAC HASH{Opaque},
    padding Opaque(SIZE(0..255))
        (CONSTRAINED BY {-- each octet contains the number of
            -- padding octets minus 1 to obtain
            -- a length multiple of block length
            GenericBlockCipher})
}

HASH{ToBeHashed} ::= Opaque(SIZE(0..255))
    (CONSTRAINED BY {ToBeHashed})

helloRequest HANDSHAKE ::= {
    HelloRequest IDENTIFIED BY hello-request
}

HelloRequest ::= NULL

clientHello HANDSHAKE ::= {
    ClientHello IDENTIFIED BY client-hello
}

ClientHello ::= SEQUENCE {
    client-version ProtocolVersion,
    random ClientRandom,
    session-id SessionID,
    cipher-suites CipherSuites,
    compression-methods CompressionMethods,
    ...,
    ...,
    client-hello-extension-list ExtensionValues
}

EXTENSION ::= CLASS {
    &id ExtensionType UNIQUE,
    &Type
}
WITH SYNTAX {
    &Type IDENTIFIED BY &id
}

ExtensionType ::= INTEGER(0..65535)

Extensions EXTENSION ::= {
    ...
}

ExtensionValues ::= SEQUENCE OF ExtensionValue

ExtensionValue ::= SEQUENCE {
    extension-type EXTENSION.&id({Extensions}),
    extension-data EXTENSION.&Type({Extensions}{@extension-type})
}

```

```

ClientRandom ::= SEQUENCE {
    gmt-unix-time    UINT32,
    random-bytes     Opaque(SIZE(28))
}

SessionID ::=    UINT32

CipherSuites ::= SEQUENCE(SIZE(1..32767)) OF CipherSuite

CipherSuite ::=  ENUMERATED {
    tls-null-with-null-null                (0),
    tls-rsa-with-null-md5                  (1),
    tls-rsa-with-null-sha                   (2),
    tls-rsa-export-with-rc4-40-md5          (3),
    tls-rsa-with-rc4-128-md5                (4),
    tls-rsa-with-rc4-128-sha                (5),
    tls-rsa-export-with-rc2-cbc-40-md5       (6),
    tls-rsa-with-idea-cbc-sha                (7),
    tls-rsa-export-with-des40-cbc-sha        (8),
    tls-rsa-with-des-cbc-sha                 (9),
    tls-rsa-with-3des-edc-cbc-sha            (10),
    tls-dh-dss-export-with-des40-cbc-sha     (11),
    tls-dh-dss-with-des-cbc-sha              (12),
    tls-dh-dss-with-3des-edc-cbc-sha         (13),
    tls-dh-rsa-export-with-des40-cbc-sha     (14),
    tls-dh-rsa-with-des-cbc-sha              (15),
    tls-dh-rsa-with-3des-edc-cbc-sha         (16),
    tls-dhe-dss-export-with-des40-cbc-sha    (17),
    tls-dhe-dss-with-des-cbc-sha              (18),
    tls-dhe-dss-with-3des-edc-cbc-sha        (19),
    tls-dhe-rsa-export-with-des40-cbc-sha    (20),
    tls-dhe-rsa-with-des-cbc-sha              (21),
    tls-dhe-rsa-with-3des-edc-cbc-sha        (22),
    tls-dh-anon-export-rc4-40-md5            (23),
    tls-dh-anon-with-rc4-128-md5             (24),
    tls-dh-anon-export-with-des40-cbc-sha    (25),
    tls-dh-anon-with-des-cbc-sha             (26),
    tls-dh-anon-with-3des-edc-cbc-sha        (27),
    -- numbers 28 and 29 are reserved to prevent confusion with SSLv3
    tls-krb5-with-des-cbc-sha                (30),
    tls-krb5-with-3des-edc-cbc-sha            (31),
    tls-krb5-with-rc4-128-sha                 (32),
    tls-krb5-with-idea-cbc-sha                (33),
    tls-krb5-with-des-cbc-md5                 (34),
    tls-krb5-with-3des-edc-cbc-md5            (35),
    tls-krb5-with-rc4-128-md5                 (36),
    tls-krb5-with-idea-cbc-md5                (37),
    tls-krb5-export-with-des-cbc-40-sha        (38),
    tls-krb5-export-with-rc2-cbc-40-sha        (39),
    tls-krb5-export-with-rc4-40-sha           (40),
    tls-krb5-export-with-des-cbc-40-md5        (41),
    tls-krb5-export-with-rc2-cbc-40-md5        (42),
    tls-krb5-export-with-rc4-40-md5           (43),
    tls-psk-with-null-sha                     (44),
    tls-dhe-psk-with-null-sha                 (45),
    tls-rsa-psk-with-null-sha                 (46),
    tls-rsa-with-aes-128-cbc-sha              (47),
    tls-dh-dss-with-aes-128-cbc-sha           (48),
    tls-dh-rsa-with-aes-128-cbc-sha           (49),
    tls-dhe-dss-with-aes-128-cbc-sha          (50),
    tls-dhe-rsa-with-aes-128-cbc-sha          (51),
    tls-dh-anon-with-aes-128-cnc-sha          (52),
    tls-rsa-with-aes-256-cbc-sha              (53),
    tls-dh-dss-with-aes-256-cbc-sha           (54),
    tls-dh-rsa-with-aes-256-cbc-sha           (55),
    tls-dhe-dss-with-aes-256-cbc-sha          (56),
    tls-dhe-rsa-with-aes-256-cbc-sha          (57),
    tls-dh-anon-with-aes-256-cbc-sha          (58),
    -- numbers 59 to 64 are not allocated --
    tls-rsa-with-camellia-128-cbc-sha         (65),
    tls-dh-dss-with-camellia-128-cbc-sha      (66),
    tls-dh-rsa-with-camellia-128-cbc-sha      (67),

```

```

tls-dhe-dss-with-camellia-128-cbc-sha    (68),
tls-dhe-rsa-with-camellia-128-cbc-sha    (69),
tls-dh-anon-with-camellia-128-cbc-sha    (70),
-- numbers 71 to 131 are reserved or used by some implementations --
tls-rsa-with-camellia-256-cbc-sha        (132),
tls-dh-dss-with-camellia-256-cbc-sha    (133),
tls-dh-rsa-with-camellia-256-cbc-sha    (134),
tls-dhe-dss-with-camellia-256-cbc-sha    (135),
tls-dhe-rsa-with-camellia-256-cbc-sha    (136),
tls-dh-anon-with-camellia-256-cbc-sha    (137),
tls-psk-with-rc4-128-sha                (138),
tls-psk-with-3des-ede-cbc-sha            (139),
tls-psk-with-aes-128-cbc-sha             (140),
tls-psk-with-aes-256-cbc-sha             (141),
tls-dhe-psk-with-rc4-128-sha             (142),
tls-dhe-psk-with-3des-ede-cbc-sha        (143),
tls-dhe-psk-with-aes-128-cbc-sha         (144),
tls-dhe-psk-with-aes-256-cbc-sha         (145),
tls-rsa-psk-with-rc4-128-sha             (146),
tls-rsa-psk-with-3des-ede-cbc-sha        (147),
tls-rsa-psk-with-aes-128-cbc-sha         (148),
tls-rsa-psk-with-aes-256-cbc-sha         (149),
tls-rsa-with-seed-cbc-sha                (150),
tls-dh-dss-with-seed-cbc-sha             (151),
tls-dh-rsa-with-seed-cbc-sha             (152),
tls-dhe-dss-with-seed-cbc-sha            (153),
tls-dhe-rsa-with-seed-cbc-sha            (154),
tls-dh-anon-with-seed-cbc-sha            (155),
-- unallocated numbers --
tls-ecdh-ecdsa-with-null-sha             (49153),
tls-ecdh-ecdsa-with-rc4-128-sha          (49154),
tls-ecdh-ecdsa-with-3des-ede-cbc-sha     (49155),
tls-ecdh-ecdsa-with-aes-128-cbc-sha      (49156),
tls-ecdh-ecdsa-with-aes-256-cbc-sha      (49157),
tls-ecdhe-ecdsa-with-null-sha            (49158),
tls-ecdhe-ecdsa-with-rc4-128-sha         (49159),
tls-ecdhe-ecdsa-with-3des-ede-cbc-sha    (49160),
tls-ecdhe-ecdsa-with-aes-128-cbc-sha     (49161),
tls-ecdhe-ecdsa-with-aes-256-cbc-sha     (49162),
tls-ecdh-rsa-with-null-sha               (49163),
tls-ecdh-rsa-with-rc4-128-sha            (49164),
tls-ecdh-rsa-with-3des-ede-cbc-sha       (49165),
tls-ecdh-rsa-with-aes-128-cbc-sha        (49166),
tls-ecdh-rsa-with-aes-256-cbc-sha        (49167),
tls-ecdhe-rsa-with-null-sha              (49168),
tls-ecdhe-rsa-with-rc4-128-sha           (49169),
tls-ecdhe-rsa-with-3des-ede-cbc-sha      (49170),
tls-ecdhe-rsa-with-aes-128-cbc-sha       (49171),
tls-ecdhe-rsa-with-aes-256-cbc-sha       (49172),
tls-ecdh-anon-with-null-sha              (49173),
tls-ecdh-anon-with-rc4-128-sha           (49174),
tls-ecdh-anon-with-3des-ede-cbc-sha      (49175),
tls-ecdh-anon-with-aes-128-cbc-sha       (49176),
tls-ecdh-anon-with-aes-256-cbc-sha       (49177),
...
}

CompressionMethods ::= SEQUENCE(SIZE(1..255)) OF CompressionMethod

CompressionMethod ::= ENUMERATED {
    null,
    ...
}

serverHello HANDSHAKE ::= {
    ServerHello IDENTIFIED BY server-hello
}

ServerHello ::= SEQUENCE {
    server-version      ProtocolVersion,
    random              ServerRandom,
    session-id          SessionID,

```

```

        cipher-suite          CipherSuite,
        compression-method    CompressionMethod,
        ...,
        ...,
        server-hello-extension-list  ExtensionValues
    }

ServerRandom ::= SEQUENCE {
    gmt-unix-time    UINT32,
    random-bytes     Opaque(SIZE(57))
}

certificateList    HANDSHAKE ::= {
    CertificateList IDENTIFIED BY certificate-list
}

CertificateList ::= SEQUENCE {
    certificates     Certificates
}

Certificates ::= SEQUENCE OF X509Certificate

X509Certificate ::= OCTET STRING(CONTAINING Certificate ENCODED BY der)

der OBJECT IDENTIFIER ::=
    {joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1)}

serverKeyExchangeHANDSHAKE ::= {
    ServerKeyExchange IDENTIFIED BY server-key-exchange
}

ServerKeyExchange ::= CHOICE {
    rsa SEQUENCE {
        params          ServerRSAParams,
        signed-params    Signature
    },
    diffie-hellman SEQUENCE {
        params          ServerDHParams,
        signed-params    Signature
    },
    ...
}

ServerDHParams ::= SEQUENCE {
    dh-p    INTEGER(1..65535),
    dh-g    INTEGER(1..65535),
    dh-Ys   INTEGER(1..65535)
}

ServerRSAParams ::= SEQUENCE {
    rsa-modulus INTEGER(1..65535),
    rsa-exponent INTEGER(1..65535)
}

Signature ::= CHOICE {
    anonymous NULL,
    rsa SEQUENCE {
        md5-hash    Opaque(SIZE(16)),
        sha-hash    Opaque(SIZE(20))
    },
    dsa SEQUENCE {
        sha-hash    Opaque(SIZE(20))
    },
    ...
}

certificateRequest    HANDSHAKE ::= {
    CertificateRequest IDENTIFIED BY certificate-request
}

CertificateRequest ::= SEQUENCE {
    certificate-types      ClientCertificateTypes,
    certificate-authorities DistinguishedNames
}

```

```

ClientCertificateTypes ::= SEQUENCE OF ClientCertificateType

ClientCertificateType ::= ENUMERATED {
    rsa-sign          (1),
    dss-sign          (2),
    rsa-fixed-dh      (3),
    dss-fixed-dn      (4),
    ...
}

DistinguishedNames ::= SEQUENCE OF DistinguishedName

serverHelloDone      HANDSHAKE ::= {
    ServerHelloDone   IDENTIFIED BY server-hello-done
}

ServerHelloDone ::= NULL

clientKeyExchangeHANDSHAKE ::= {
    ClientKeyExchange IDENTIFIED BY client-key-exchange
}

ClientKeyExchange ::= Opaque(SIZE(0..65535))

PreMasterSecret ::= SEQUENCE {
    client-version   ProtocolVersion,
    random           Opaque(SIZE(46))
}

EncryptedPreMasterSecret ::= ENCRYPTED{PreMasterSecret}

ClientDiffieHellmanPublic ::= CHOICE {
    implicit   NULL,
    explicit   Opaque(SIZE(1..65535))
}

ENCRYPTED{ToBeEnciphered} ::= OCTET STRING(SIZE(0..255))
(CONSTRAINED BY {ToBeEnciphered})

certificateVerifyHANDSHAKE ::= {
    CertificateVerify IDENTIFIED BY certificate-verify
}

CertificateVerify ::= SEQUENCE {
    signature   Signature
}

finished      HANDSHAKE ::= {
    Finished   IDENTIFIED BY finished
}

Finished ::= SEQUENCE {
    verify-data Opaque(SIZE(12))
}

XtsmTemplate ::= BiometricCertificate -- Import from TAI

SignedDatabyClient ::= CHOICE {
    digital-signature [0] SignedData,
    --import from X9.84-CMS
    aCBioOnClient     [1] SignedDataACBio
    --import from ISO/IEC 24761
}

BDforLocalModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess,
    digitalSignature       SignedData,
    aCforBioOnClient       ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BiometricClientProcess ::= SEQUENCE {
    bFPSchema      BSP-BFP-Schemas,
    templateID     TemplateID,
    sampleQuality   Quality,
    score          BioAPI-FMR
}

```

```

TemplateID ::= SEQUENCE {
    certificateIssuer      Name, -- see Rec. ITU-T X.509
    serialNumber           CertificateSerialNumber, -- see Rec. ITU-T X.509
    templateInfo           TemplateInfo
}

TemplateInfo ::= SEQUENCE {
    biometricType          BiometricType,
    creator                UTF8String,
    createdBFPSchema       BSP-BFP-Schema,
    templateID             CertificateIDInformation
    -- such as CertificateSerialNumber (no value available)
}

CertificateIDInformation ::= CertificateSerialNumber

BDforDownloadModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess,
    digitalSignature        SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BDforAttachedModel ::= SEQUENCE {
    templateData           XtсмTemplate,
    sampleData             SampleData, -- BIR: BioAPI defined format --
    digitalSignature        SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BDforCenterModel ::= SEQUENCE {
    sampleData             SampleData, -- BIR: BioAPI defined format --
    digitalSignature        SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BDforRefOnTTPforLocalModel ::= SEQUENCE {
    thirdPartyInfo         UTF8String,
    biometricClientProcess BiometricClientProcess,
    aCforBioOnTTP          ACBioContentInformation,
    digitalSignaturebyClient SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BiometricTTPProcess ::= SEQUENCE {
    templateData           XtсмTemplate,
    digitalSignature        SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

BDforRefOnTTPforCenterModel ::= SEQUENCE {
    thirdPartyInfo         UTF8String,
    sampleData             SampleData, -- BIR: BioAPI defined format --
    digitalSignature        SignedData,
    aCforBioOnClient        ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

TTPRequestRefOnTTPforCenterModel ::= SEQUENCE {
    templateID             TemplateID
}

TTPResponseRefOnTTPforCenterModel ::= SEQUENCE {
    templateData           XtсмTemplate,
    digitalSignature        SignedData,
    aCforBioOnTTP          ACBioContentInformation OPTIONAL
    -- see ISO/IEC 24761
}

```

```

TTPRequestCobyClientModel ::= SEQUENCE {
    templateData      XtsmTemplate,
    sampleData        SampleData -- BIR: BioAPI defined format --
}

TTPResponseCobyClientModel ::= SEQUENCE {
    bFPSchemaOnTTPProcess BSP-BFP-Schemas,
    templateID           TemplateID,
    sampleQuality        Quality,
    score                BioAPI-FMR,
    digitalSignature     SignedData,
    aCforBioTTP          ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

BDforCobyClientModel ::= SEQUENCE {
    bFPSchemaforClientProcess BSP-BFP-Schemas,
    thirdPartyInfo           UTF8String,
    bFPSchemaforTTPProcess   BSP-BFP-Schemas,
    templateID               TemplateID,
    sampleQuality             Quality,
    score                    BioAPI-FMR,
    digitalSignaturebyClient SignedData,
    digitalSignaturebyTTP    SignedData,
    aCforBioOnClient         ACBioContentInformation OPTIONAL,
    aCforBioOnTTP            ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

BDforCobyServerModel ::= SEQUENCE {
    sampleData      SampleData, -- BIR: BioAPI defined format --
    digitalSignature SignedData,
    aCforBiometrics ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

TTPRequestCobyServerModel ::= SEQUENCE {
    templateData      XtsmTemplate,
    sampleData        SampleData -- BIR: BioAPI defined format --
}

TTPResponsebyServer ::= SEQUENCE {
    bFPSchema      BSP-BFP-Schemas,
    templateID     TemplateID,
    sampleQuality   Quality,
    score          BioAPI-FMR,
    digitalSignature SignedData,
    aCforBioOnTTP  ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

TTPRequestSCobyClientModel ::= SEQUENCE {
    sampleData      SampleData -- BIR: BioAPI defined format --
}

BDforSCobyCModel2 ::= SEQUENCE {
    bFPSchemaForTTPProcess BSP-BFP-Schemas,
    templateID             TemplateID,
    sampleQuality           Quality,
    score                  BioAPI-FMR,
    digitalSignatureByTTP   SignedData,
    aCforBioOnTTP          ACBioContentInformation OPTIONAL
                        -- see ISO/IEC 24761
}

BDforSCobyCModel3 ::= SEQUENCE {
    bFPSchemaForClientProcess BSP-BFP-Schemas,
    thirdPartyInfo           UTF8String,
    bFPSchemaForTTPProcess   BSP-BFP-Schemas,
    templateID               TemplateID,
    sampleQuality             Quality,
    score                    BioAPI-FMR,
    digitalSignatureByClient SignedData,
    digitalSignatureByTTP    SignedData,

```

```

        aCforBioOnClient      ACBioContentInformation OPTIONAL,
        aCforBioOnTTP         ACBioContentInformation OPTIONAL
                                -- see ISO/IEC 24761
    }

    BDforSCObysModel ::= SEQUENCE {
        sampleData              SampleData, -- BIR: BioAPI defined format --
        digitalSignatureByClient SignedData,
        aCforBioOnClient        ACBioContentInformation OPTIONAL
                                -- see ISO/IEC 24761
    }

    TTPRequestSCObysServerModel ::= SEQUENCE {
        templateID              TemplateID,
        sampleData              SampleData -- BIR: BioAPI defined format --
    }

    TTPResponseSCObysServer ::= SEQUENCE {
        bFPSchemaforTTPProcess  BSP-BFP-Schemas,
        templateID              TemplateID,
        sampleQuality            Quality,
        score                    BioAPI-FMR,
        digitalSignatureByTTP    SignedData,
        aCforBioOnTTP            ACBioContentInformation OPTIONAL
                                -- see ISO/IEC 24761
    }

    END

```

Appendix I

Telebiometrics system mechanism definitions by TLS extension

(This appendix does not form an integral part of this Recommendation)

This appendix describes all protocols for telebiometrics system mechanisms using the TLS Extension [IETF RFC 4366].

This appendix applies the [IETF RFC 4366]: "TLS Extensions" for this biometrics handshake protocol and biometrics transfer protocol. The biometrics handshake protocol is defined in clause 10. This appendix further extends the handshake protocol.

I.1 Extensions for biometric transfer protocol

This appendix recommends the use of five extended handshake messages, namely, "Biometric Verify," "Biometric Retry Request," "Biometric Finished," "Biometric TTP Request," and "Biometric TTP Response". These extended message flows are shown in Figures I.1 to I.3. Figure I.1 illustrates a simple network mechanism for a client and a server. Figure I.2 illustrates outsourcing to TTP by a client mechanism. Figure I.3 illustrates outsourcing to TTP by a server mechanism.

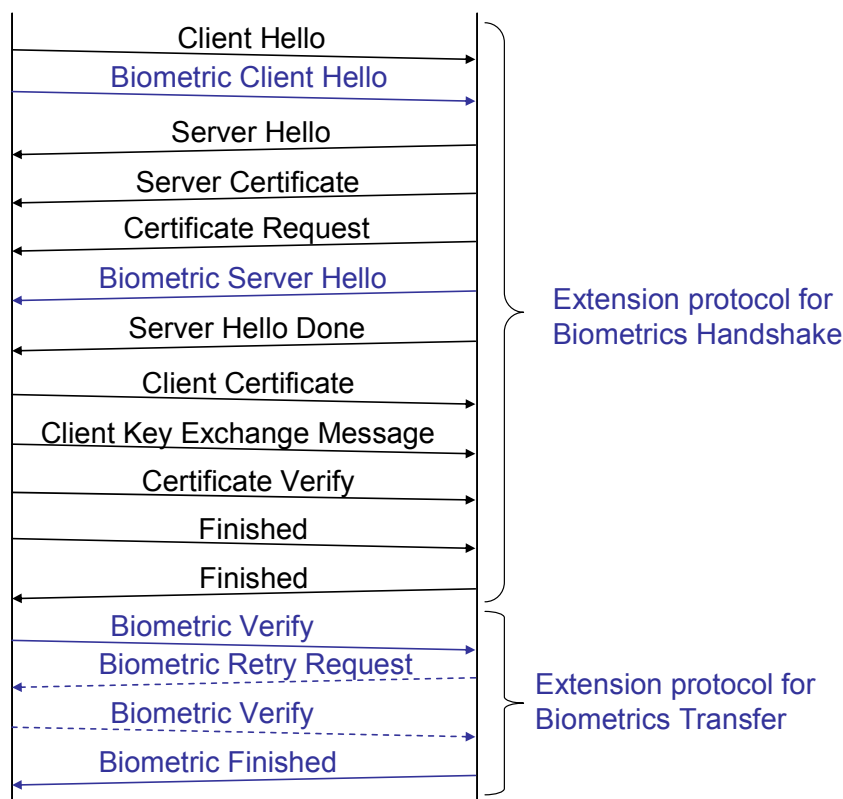


Figure I.1 – Telebiometrics system mechanism by TLS extension

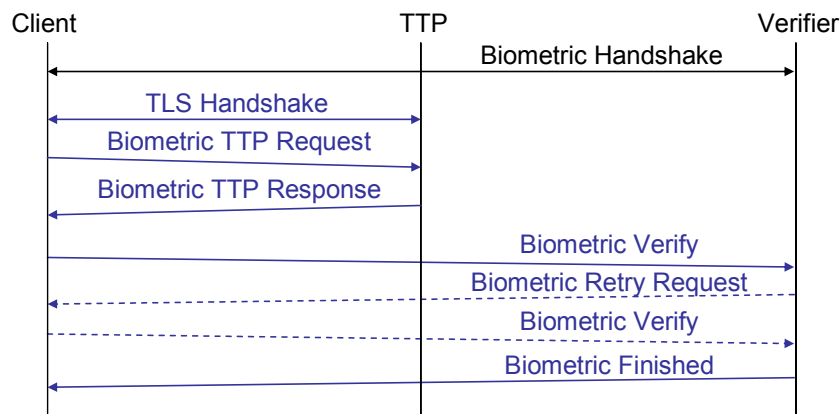


Figure I.2 – Biometrics extension protocol for outsourcing to TTP by client

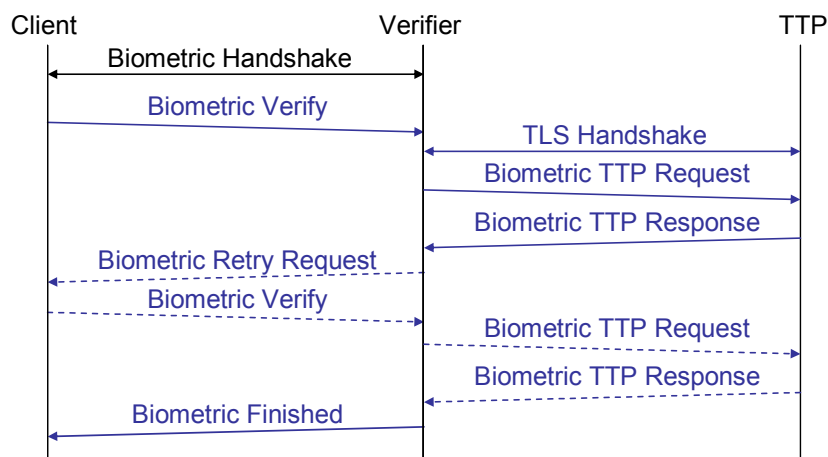


Figure I.3 – Biometrics extension protocol for outsourcing to TTP by verifier

These messages are described in clauses I.2, I.3, I.4, I.5 and I.6. The new structure of the handshake message is as follows:

```

HandshakeType ::= INTEGER{
    hello-request                (0),
    client-hello                 (1),
    server-hello                 (2),
    certificate                   (11),
    server-key-exchange           (12),
    certificate-request           (13),
    server-hello-done            (14),
    certificate-verify            (15),
    client-key-exchange           (16),
    finished                     (20),
    biometrics-client-hello       (101),
    biometrics-server-hello       (102),
    biometrics-verify             (103),
    biometrics-retry-request      (104),
    biometrics-finished           (105),
    biometrics-ttp-request        (106),
    biometrics-ttp-response       (107)
} (0..255)

Handshake ::= SEQUENCE {
    type HANDSHAKE.&id({Handshakes}),
    value HANDSHAKE.&Type({Handshakes}{@type})
}

Handshakes HANDSHAKE ::= {
    helloRequest |
  
```

```

clientHello |
serverHello |
certificateList |
serverKeyExchange |
certificateRequest |
serverHelloDone |
certificateVerify |
clientKeyExchange |
finished |
certificateURL |
certificateStatus |
biometricClientHello |
biometricServerHello |
biometricVerify |
biometricRetryRequest |
biometricFinished |
biometricTTPRequest |
biometricTTPResponse,
...
}

```

I.2 Biometrics Verify

The data content corresponding to the selected system model is transmitted to the verifier as follows:

```

biometricVerify      HANDSHAKE ::= {
    BiometricVerify  IDENTIFIED BY biometric-verify
}

BiometricVerify ::= SEQUENCE {
    biometricData    CHOICE {
        no-value                      [0] NULL,
        local-model                 [1] BDforLocalModel,
        download-model              [2] BDforDownloadModel,
        attached-model              [3] BDforAttachedModel,
        center-model                [4] BDforCenterModel,
        ref-onttp-for-local-model    [5] BDforRefOnTTPforLocalModel,
        ref-onttp-for-center-model   [6] BDforRefOnTTPforCenterModel,
        comparison-outsourcing-by-client-model [7] BDforCObyClientModel,
        comparison-outsourcing-by-server-model [8] BDforCObyServerModel,
        storage-comparison-outsourcing-by-client-model
                                     [9] BDforSCbyClientModel,
        storage-comparison-outsourcing-by-server-model
                                     [10] BDforSCbyServerModel,
        ...
    },
    digitalSignature    SignedDatabyClient
}

SignedDatabyClient ::= CHOICE {
    digital-signature    [0] SignedData,
                        --import from X9.84-CMS
    aCBioOnClient        [1] AuthenticationContextForBiometrics
                        --import from ISO/IEC 24761
}

```

The following presents the Biometric Data of "Biometric Verify" for the local model:

```

BDforLocalModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess
}

BiometricClientProcess ::= SEQUENCE {
    bFPSchema        BSP-BFP-Schemas,
    templateID       TemplateID,
    sampleQuality     Quality,
    score             BioAPI-FMR
}

```

```

TemplateID ::= SEQUENCE {
    certificateIssuerName, Name -- see Rec. ITU-T X.509
    serialNumber      CertificateSerialNumber, -- see Rec. ITU-T X.509
    templateinfo      TemplateInfo
}

TemplateInfo ::= SEQUENCE {
    biometricType      BiometricType,
    creator            UTF8String,
    createdBFPSchema   BSP-BFP-Schema,
    templateID         CertificateIDInformation
                        -- such as CertificateSerialNumber (no value available)
}

CertificateIDInformation ::= CertificateSerialNumber

```

The following presents Biometric Data of "Biometric Verify" for the download model. The content is the same as for the local model:

```

BDforDownloadModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess
}

```

The following presents Biometric Data of "Biometric Verify" for the attached model:

```

BDforAttachedModel ::= SEQUENCE {
    templateData XtsmTemplate,
    sampleData SampleData      -- BIR: BioAPI defined format --
}

XtsmTemplate ::= BiometricCertificate      -- Import from TAI

```

The Biometric Data of "Biometric Verify" for the centre model is detailed as follows:

```

BDforCenterModel ::= SEQUENCE {
    sampleData SampleData      -- BIR: BioAPI defined format --
}

```

The following presents Biometric Data of "Biometric Verify" for reference management on TTP for the client comparison model. The client may omit Biometrics TTP Response with the user's template for the protection of his/her privacy:

```

BDforRefOnTTPforLocalModel ::= SEQUENCE {
    thirdPartyInfo      UTF8String,
    biometric-ttp-Process BiometricsTTPResponse OPTIONAL,
    biometricClientProcess BiometricClientProcess
}

```

The Biometric Data of "Biometric Verify" for the reference management on TTP for server comparison model is detailed as follows:

```

BDforRefOnTTPforCenterModel ::= SEQUENCE {
    thirdPartyInfo UTF8String,
    sampleData SampleData      -- BIR: BioAPI defined format --
}

```

The Biometric Data of "Biometric Verify" for comparison outsourcing by the client model is detailed as follows:

```

BDforCObyClientModel ::= SEQUENCE {
    bFPSchemaForClientProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    thirdPartyInfo            UTF8String,
    biometric-ttp-Process      BiometricsTTPResponse
}

```

The Biometric Data of "Biometric Verify" for comparison outsourcing by the server model is detailed as follows:

```

BDforCObyServerModel ::= SEQUENCE {
    sampleData SampleData      -- BIR: BioAPI defined format --
}

```

The Biometric Data of "Biometric Verify" for storage and comparison outsourcing by the client model are detailed as follows:

```
BDforSCobyClientModel ::= SEQUENCE {
    bFPSchema          ForClientProcess SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    thirdPartyInfo      UTF8String,
    biometric-ttp-Process BiometricsTTPResponse
}
```

The Biometric Data of "Biometric Verify" for storage and comparison outsourcing by the server model are detailed as follows:

```
BDforSCobyServerModel ::= SEQUENCE {
    sampleData SampleData -- BIR: BioAPI defined format --
}
```

I.3 Biometrics Retry Request

When the biometric result fails to authenticate a user, according to the server's security policy (user authentication policy), the same or another supported biometric model and an approved biometric system are selected from a list by the client, and are sent to the client:

```
biometricRetryRequest HANDSHAKE ::= {
    BiometricRetryRequest IDENTIFIED BY biometric-retry-request
}

BiometricsRetryRequest ::= SEQUENCE{
    retryRequest BiometricAuthenticationRequest
}
```

When the server selects the same model and the same approved biometric system, the request trial number should be decremented by one.

When the "Biometric Client Verify" data is based on an illegal template, a revoked template, an expired template, or where it contains an unknown BCA digital signature or unacceptable biometric comparison score, the server sends an alert message (with one of the following descriptions) to the client (refer to clause I.7):

```
AlertDescription ::= ENUMERATED{
    ...,
    unacceptable-model          (115),-- Extension item for BiometricsHandshake
    unacceptable-biometrics     (116),-- Extension item for BiometricsHandshake
    unsupported-biometrics       (117),-- Extension item for BiometricsHandshake
    bad-template                 (118),-- Extension item
    template-revoked             (119),-- Extension item
    template-expired             (120),-- Extension item
    unknown-bca                  (121),-- Extension item
    unacceptable-fmr             (122),-- Extension item
    ...
}
```

The client should respond using again "Biometric Verify".

However, the data content in the previous "Biometric Verify" and the following "Biometric Verify" for Biometric Retry Request differ. The content contains information that guarantees a correspondence between a user template and a user certificate. When a template is sent to a server, a format that guarantees the correspondence, (refer to clause 8), with the template, is required. When a template is not sent to a server, a processing code that guarantees correspondence with the template is required. The profile for each model determines the data content in detail.

I.4 Finished Biometrics

A biometric-finished message is given as follows:

```
biometricFinishedHANDSHAKE ::= {
    BiometricFinishedIDENTIFIED BY biometric-finished
}
```

```

biometricFinishedHANDSHAKE ::= {
    BiometricFinishedIDENTIFIED BY biometric-finished
}

BiometricFinished ::= SEQUENCE {
    result      BiometricAuthenticationResult
}

BiometricAuthenticationResult ::= BOOLEAN

```

I.5 Biometrics TTP Request

This appendix defines an outsourcing type for the use of TTP of the network authentication models. The definition is as follows:

```

biometricTTPRequest HANDSHAKE ::= {
    BiometricTTPRequest IDENTIFIED BY biometric-ttp-request
}

BiometricTTPRequest ::= CHOICE {
    storage-type          [1] BDforStorageOutsourcing,
    comparison-type       [2] BDforComparisonOutsourcing,
    storage-comparison-type [3] BDforComparisonOutsourcing
}

```

Table I.1 lists the relationship between the TTP authentication models and the outsourcing type.

Table I.1 – Relationship between TTP authentication model and outsourcing type

	Network authentication model	Outsourcing type	Outsourcer
1	Reference management on TTP for local	storage-type	client
2	Reference management on TTP for centre	storage-type	verifier
3	Comparison outsourcing by client	comparison-type	client
4	Comparison outsourcing by verifier	comparison-type	verifier
5	Storage & comparison outsourcing by client	storage-comparison-type	client
6	Storage & comparison outsourcing by verifier	storage-comparison-type	verifier

The Biometric TTP request message is supplied as follows:

```

biometricTTPRequest HANDSHAKE ::= {
    BiometricTTPRequest IDENTIFIED BY biometric-ttp-request
}

BiometricTTPRequest ::= CHOICE {
    storage-type          [1] BDforStorageOutsourcing,
    comparison-type       [2] BDforComparisonOutsourcing,
    storage-comparison-type [3] BDforComparisonOutsourcing
}

BDforStorageOutsourcing ::= SEQUENCE {
    templateID TemplateID
}

BDforComparisonOutsourcing ::= SEQUENCE {
    templateData XtsmTemplate,
    samplaData SampleData
}
--BIR: BioAPI defined format

BDforSCoutsourcing ::= SEQUENCE {
    templateID TemplateID,
    samplaData SampleData
}
--BIR: BioAPI defined format

```

I.6 Biometrics TTP response

The Biometric TTP response message is given as follows:

```
biometricTTPResponse  HANDSHAKE  ::= {
    BiometricTTPResponse  IDENTIFIED BY biometric-ttp-response
}

BiometricTTPResponse ::= SEQUENCE {
    request-body CHOICE {
        storage-type           [1] RBDforStorageOutsourcing,
        comparison-type        [2] RBDforComparisonOutsourcing,
        storage-comparison-type [3] RBDforComparisonOutsourcing
    },
    digital-signatureSignedDatabyTTP
}

RBDforStorageOutsourcing ::= SEQUENCE {
    templateData      XtsmTemplate
}

RBDforComparisonOutsourcing ::= SEQUENCE {
    bFPSchema      BSP-BFP-Schemas,
    templateID      TemplateID,
    sampleQuality    Quality,
    score            BioAPI-FMR
}

SignedDatabyTTP ::= CHOICE {
    digital-signatureSignedData,      --import from X9.84-CMS
    acBioOnTTP      ACBioContentInformation
                        --import from ISO/IEC 24761
}
```

I.7 Extension alert protocol

The following describes the extended alert protocol of TLS for this Recommendation, which defines alert description numbers from 115 to 122 in order to avoid any conflict with the TLS extension definition.

- unsupported-extension (110)

This alert is a message of TLS Extension [IETF RFC 4366] and may be returned if the verifier receives an unsupported extended biometrics handshake protocol. This message is always fatal.

- unacceptable-model (115)

This alert may be returned if the verifier receives only unacceptable models that do not conform to the verification policy of the verifier. This message is always fatal.

- unacceptable-biometrics (116)

This alert may be returned if the verifier receives unacceptable biometrics, modalities, biometric algorithms, or biometric devices that do not conform to the verification policy of the verifier. This message is always fatal.

- unsupported-biometrics (117)

With regard to the server comparison models, this alert may be returned when the server receives unsupported biometric algorithms only – which are not supported by the verifier. This message is always fatal.

- bad-template (118)

The verifier detected the falsification of the template for biometric comparison. This message is always fatal. Falsification means that the certificate has been detected as having a non-decodable syntax, or incorrect identifier, etc.

- template-revoked (119)

The verifier finds that the biometric template used for biometric comparison was revoked. This message is always fatal.

- template-expired (120)

The verifier finds that the biometric template used for the biometric comparison has expired. This message is always fatal.

- unknown-bca (121)

This alert may be returned if the verifier cannot find a receivable certificate of BCA, or if the verifier receives a biometric template published by the BCA that differs from that which the verifier trusts. This message is always fatal.

- unacceptable-fmr (122)

This alert may be returned if the biometric comparison score is under the threshold value for verification. If this message is a warning alert, the biometric comparison process is usually executed repeatedly. If this message is a fatal alert, the verifier does not permit further iteration of the verification process.

- no-response-ttp (123)

This alert may be returned if an outsourcer, (client or server), is NOT able to get the response from the outsourcing TTP. If the outsourcer is client side, then the client forwards the alert response to the server. This message is always fatal.

- no-template-in-ttp (124)

This alert may be returned if the outsourcing TTP is NOT able to find the requested template in TTP. If the outsourcer is client side, then the client forwards this alert response to the server. This message is always fatal.

```
Alert ::= SEQUENCE {
    level          AlertLevel,
    description    AlertDescription
}

AlertLevel ::= ENUMERATED {
    warning      (1),
    fatal        (2)
}

AlertDescription ::= ENUMERATED {
    close-notify           (0),
    unexpected-message     (10),
    bad-record-mac         (20),
    decryption-failed      (21),
    record-overflow        (22),
    decompression-failure  (30),
    handshake-failure      (40),
    bad-certificate        (42),
    unsupported-certificate (43),
    certificate-revoked     (44),
    certificate-expired     (45),
    certificate-unknown     (46),
    illegal-parameter      (47),
    unknown-ca             (48),
    access-denied          (49),
    decode-error           (50),
    decrypt-error          (51),
    export-restriction     (60),
    protocol-version       (70),
    insufficient-security  (71),
    internal-error         (80),
    user-canceled          (90),
```

```

no-renegotiation          (100),
...,
unacceptable-model        (115),-- Extension item for BiometricHandshake
unacceptable-biometrics   (116),-- Extension item for BiometricHandshake
unsupported-biometrics     (117),-- Extension item for BiometricHandshake
bad-template              (118),-- Extension item for BiometricVerify
template-revoked          (119),-- Extension item for BiometricVerify
template-expired          (120),-- Extension item for BiometricVerify
unknown-bca               (121),-- Extension item for BiometricVerify
unacceptable-fmr          (122),-- Extension item for BiometricVerify
no-response-ttp           (123),-- Extension item for TTP
no-template-in-ttp        (124) -- Extension item for TTP
}

```

Appendix II

Implementation example of the biometric transfer protocol using BIP

(This appendix does not form an integral part of this Recommendation)

This appendix describes an implementation example of the biometric transfer protocol using [b-ITU-T X.1083].

If an authentication system is configured with BIP, communicators should predefine its role, i.e., master or slave. The master requests BioAPI commands, whereas the slave returns responses to these requests. In this Recommendation, the verifier is the master on BIP because the verifier always leads the client in a biometric process.

There are three tasks and solutions involved in applying BIP to TSM, which are listed as per the following:

- 1) BIP cannot define transportation between the client and dynamically changing TTPs.

The client operates BIP implementation hierarchically. However, the verifier, first, always operates BIP as the master for TSM.

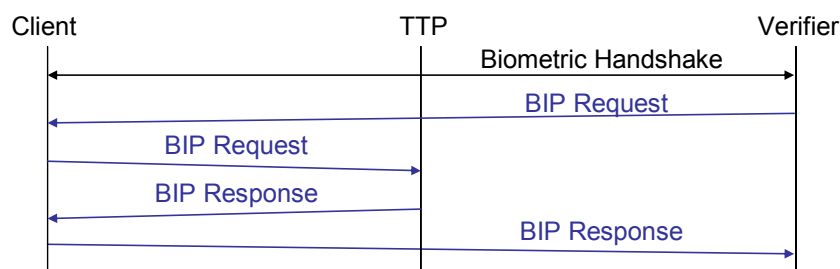


Figure II.1 – Implementation sample by BIP for TTP outsourcing by client

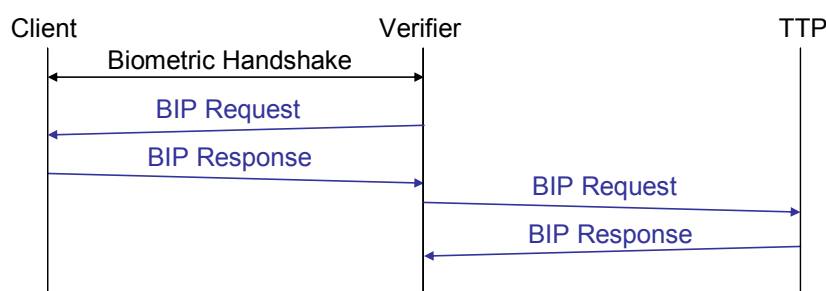


Figure II.2 – Implementation sample by BIP for TTP outsourcing by verifier

- 2) BIP cannot define some TSM models with only one BIP command.

Accordingly, they apply multiple BIP commands.

- 3) BIP does not have the information on the validity for remote biometric processes.

The BIP should have validity information, such as MAC or digital signatures, and the authority to certify the validity thereof. In case it gets any validity information, such as ACBio for BioAPI security amendments, in the future, this Recommendation suggests applying the security amendments to the validity information.

The following describes the implementations and tasks for each TSM model using BIP. These implementations only describe biometric functions. However, real implementation systems require preparation functions, such as load and attach, and terminate functions, such as detach, for successful BIP operation.

II.1 Local model

Figure II.3 outlines an implementation sample for the local model using a BIP message.

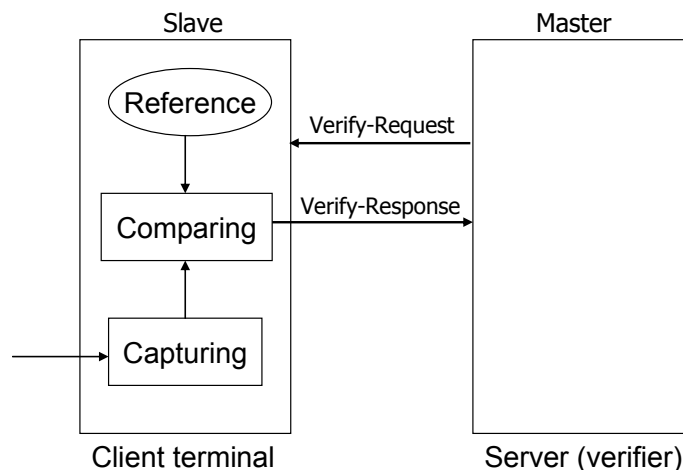


Figure II.3 – Local model using BIP

The following is the content of a Verify-Request message:

```
Verify-RequestParams ::= SEQUENCE {
    originalBSPHandle      BioAPI-HANDLE,
    maxFMRRequested       BioAPI-FMR,
    referenceTemplate      BioAPI-INPUT-BIR,
    subtype                BioAPI-BIR-SUBTYPE,
    timeout                SignedInt,
    no-adaptedBIR          BOOLEAN,
    no-fmrAchieved         BOOLEAN,
    no-payload             BOOLEAN,
    no-auditData           BOOLEAN
}
```

The client terminal should, in advance, execute a comparison process with the addressed reference template using the PKI certificate, by means of the TLS handshake process. However, the BIP's Verify-Request message cannot address the BIR that correlates to the PKI certificate. The message can only be selected as follows:

```
BioAPI_INPUT_BIR_FORM Form;
union {
    BioAPI_DBBIR_ID *BIRinDb;
    BioAPI_BIR_HANDLE *BIRinBSP;
    BioAPI_BIR *BIR;
} InputBIR;
} BioAPI_INPUT_BIR;
```

The following is the content of the Verify-Response message:

```
Verify-ResponseParams ::= SEQUENCE {
    adaptedBIR             BioAPI-BIR-HANDLE OPTIONAL,
    result                 BOOLEAN,
    fmrAchieved            BioAPI-FMR OPTIONAL,
    payload                BioAPI-DATA OPTIONAL,
    auditData              BioAPI-BIR-HANDLE OPTIONAL
}
```

There is no biometric-process information such as the BFP, process-result information, or reference-template information. These are required by this Recommendation. Template ID information and the profile for the client's biometric process should be added. The profile may be useful for ACBio [ISO/IEC 24761].

II.2 Download model

Figure II.4 outlines an implementation sample for the download model using a BIP message.

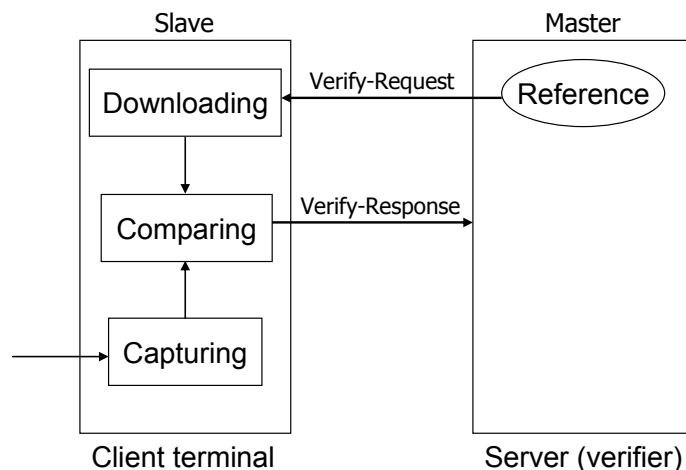


Figure II.4 – Download model

Here, the verifier sets the reference template on the server to the Verify-Request message, sending it to the client terminal as a slave.

However, the Verify-Response message should have template ID information and the profile for the client's biometric process. It is the same as for the local model.

II.3 Attached model

Figure II.5 outlines an implementation sample for the attached model using a BIP message.

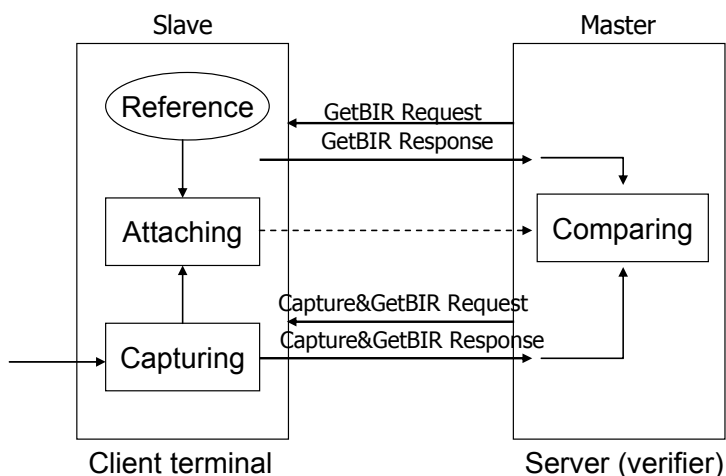


Figure II.5 – Attached model

As explained by Figure II.5, the verifier can obtain sampling data using the capture command and the GetBIRFromHandle command, and he or she may also obtain the reference template using the GetBIRFromHandle command.

```

GetBIRFromHandle-RequestParams ::= SEQUENCE {
    originalBSPHandle    BioAPI-HANDLE,
    birHandle            BioAPI-BIR-HANDLE
}

GetBIRFromHandle-ResponseParams ::= SEQUENCE {
    bir                  BioAPI-BIR
}

```

However, the verifier cannot obtain the addressed reference template using the PKI certificate on the TLS handshake in advance. With the DbGetBIR command, on the other hand, the verifier may retrieve the template using the certificate information, if the client terminal implements the database for the template.

```

DbGetBIR-RequestParams ::= SEQUENCE {
    originalBSPHandle    BioAPI-HANDLE,
    dbHandle             BioAPI-DB-HANDLE,
    keyValue             BioAPI-UUID
}

```

In conclusion, BIP should support the GetBIRFromHandle command that includes the PKI certificate information.

II.4 Centre model

Figure II.6 outlines an implementation sample for the centre model using a BIP message.

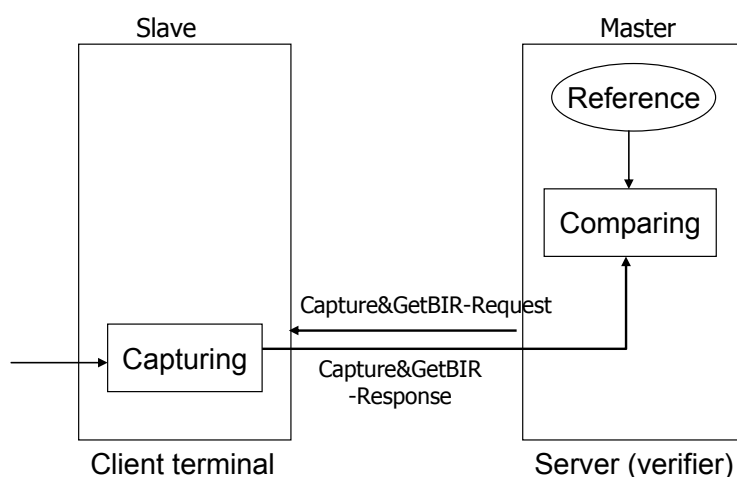


Figure II.6 – Centre model

As explained in Figure II.6, the verifier can acquire sampling data using the capture command and the GetBIRFromHandle command.

II.5 Comparison outsourcing by client model

Figure II.7 outlines an implementation sample for comparison outsourcing by the client model using a BIP message.

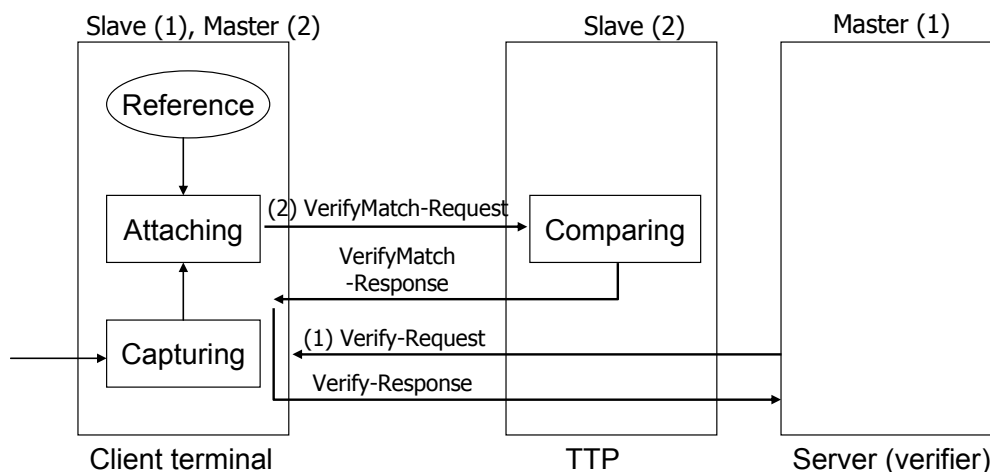


Figure II.7 – Comparison outsourcing by client model

This model uses BIP hierarchically. Firstly, the verifier sends a biometric verification request to the client terminal using a Verify-Request message. Secondly, the client terminal is operated by capturing sample data from the client. Thirdly, the client terminal sends a comparison process request to the TTP, using a VerifyMatch-Request message with the sampling data and the reference template. Fourthly, the TTP is the slave of the client, which operates the comparison process, and then returns a VerifyMatch-Response message to the client. Finally, the client terminal returns a Verify-Response message to the verifier, (server), with the comparison results.

```

VerifyMatch-RequestParams ::= SEQUENCE {
    originalBSPHandle      BioAPI-HANDLE,
    maxFMRRequested        BioAPI-FMR,
    processedBIR            BioAPI-INPUT-BIR,
    referenceTemplate       BioAPI-INPUT-BIR,
    no-adaptedBIR           BOOLEAN,
    no-fmrAchieved          BOOLEAN,
    no-payload              BOOLEAN
}

VerifyMatch-ResponseParams ::= SEQUENCE {
    adaptedBIR             BioAPI-BIR-HANDLE OPTIONAL,
    result                  BOOLEAN,
    fmrAchieved             BioAPI-FMR OPTIONAL,
    payload                 BioAPI-DATA OPTIONAL
}
  
```

The client terminal in this Verify-Response should return the TTP information to the verifier.

II.6 Reference management on TTP for local model

Figure II.8 outlines an implementation sample for the model of the reference template managing the TTP for local comparison using a BIP message.

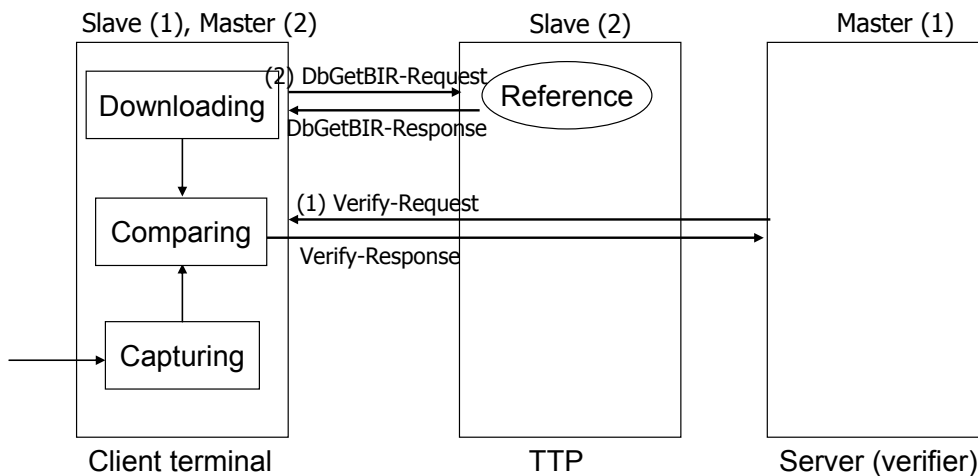


Figure II.8 – Reference management on TTP for local model

This model also uses BIP hierarchically. Firstly, the verifier sends a biometric verification request to the client terminal using a Verify-Request message. Secondly, the client terminal is operated by capturing sample data from the client. Thirdly, the client terminal sends a dbGetBIR request to TTP in order to obtain the addressed reference template on TTP. Finally, the client terminal operates a comparison process using the sample data and the template, returning the results to the verifier using Verify-Response.

If BIP supports the dbGetBIR command, the client terminal may then retrieve the addressed reference template using the PKI certificate on the TLS handshake in advance. This retrieval may require one of the following selections of PKI certificate information, such as a key value, for example, UUID.

```

DbGetBIR-RequestParams ::= SEQUENCE {
    originalBSPHandle BioAPI-HANDLE,
    dbHandle          BioAPI-DB-HANDLE,
    keyValue          BioAPI-UUID
}

DbGetBIR-ResponseParams ::= SEQUENCE {
    retrievedBIR      BioAPI-BIR-HANDLE,
    markerHandle      BioAPI-DB-MARKER-HANDLE
}
  
```

II.7 Reference management on TTP for centre model

Figure II.9 outlines an implementation sample for the model of the reference template managing the TTP for the server comparison using a BIP message.

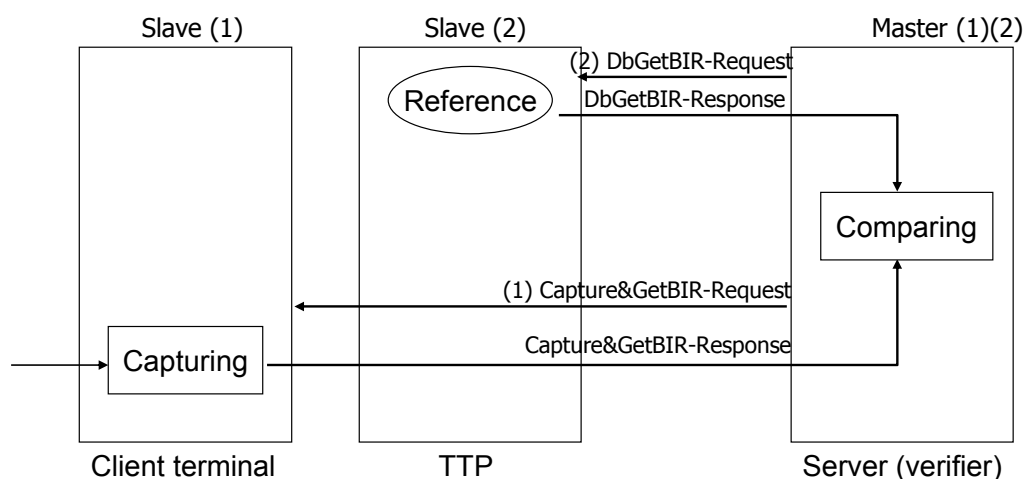


Figure II.9 – Reference management on TTP for centre model

As explained in Figure II.9, the verifier first requests the Capture command and the GetBIRFromHandle command in order to acquire the client biometric sample. Secondly, the verifier requests the dbGetBIR command to obtain the addressed reference template on TTP.

The retrieval of the reference template from TTP is the same as for the reference management on TTP for the client model, as outlined in clause II.6. Data is sampled from the client terminal in the same way as in the centre model, as detailed in clause II.4.

II.8 Comparison outsourcing by server model

Figure II.10 outlines an implementation sample for the model of comparison outsourcing by the server using a BIP message.

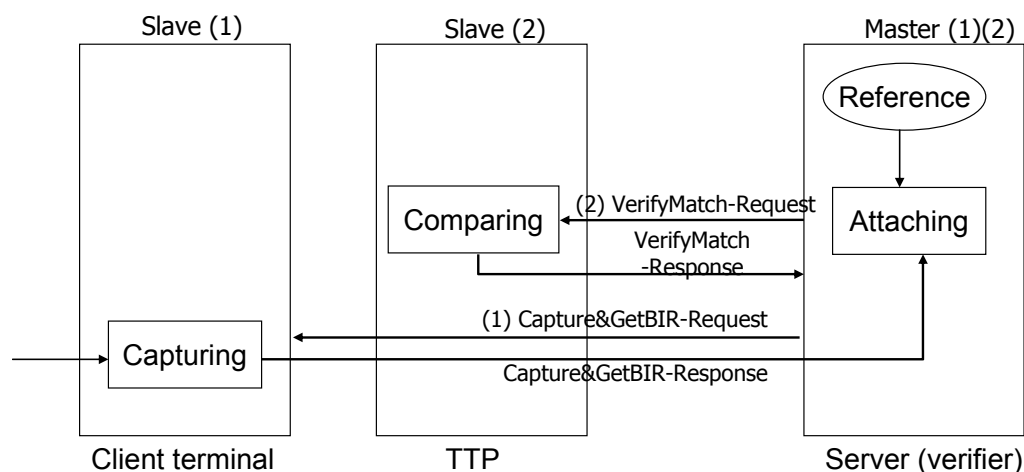


Figure II.10 – Comparison outsourcing by server model

As explained in Figure II.10, the verifier first sends the Capture-Request and the GetBIRFromHandle-Request so as to acquire the client biometric sample. Secondly, the verifier sends a comparison process request to TTP, using the VerifyMatch-Request with the sample, and the addressed reference template on the server.

Data is sampled from the client in the same way as it is for the centre model in clause II.4. Comparison outsourcing to TTP is the same as for comparison outsourcing by the client model, as described in clause II.5.

II.9 Storage and comparison outsourcing model

1) Outsourcing by client model

Figure II.11 illustrates an implementation sample for the model of comparison and reference template managing outsourcing by the client that uses a BIP message.

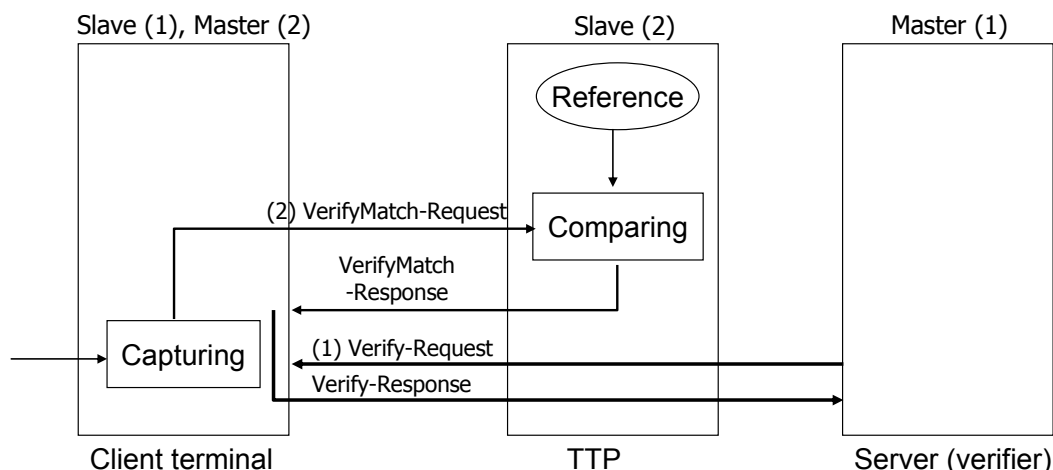


Figure II.11 – Storage and comparison outsourcing by client model

This model also uses BIP hierarchically. The verifier first sends a biometric verification request to the client terminal using a Verify-Request message. Secondly, the client terminal is operated by capturing the sample data from the client. Thirdly, the client terminal sends a comparison process request to the TTP using a VerifyMatch-Request message with the sampling data. Fourthly, TTP becomes the slave of the client terminal, and operates the comparison process with the received sampling data, and the addressed reference template on TTP. Fifthly, TTP returns the comparison results to the client terminal, using a VerifyMatch-Response message. Finally, the client terminal returns the comparison results to the verifier, using a Verify-Response message.

The client in the VerifyMatch-Request should request the addressed reference template on TTP using the ID information on the PKI certificate. The verifier (and client terminal) in the VerifyMatch-Response needs the compared reference template information. The client terminal in this Verify-Response should return the TTP information to the verifier. This is the same as for comparison outsourcing by the client model in clause II.5.

2) Outsourcing by server model

Figure II.12 outlines an implementation sample for the model of comparison and reference template managing outsourcing by the server using a BIP message.

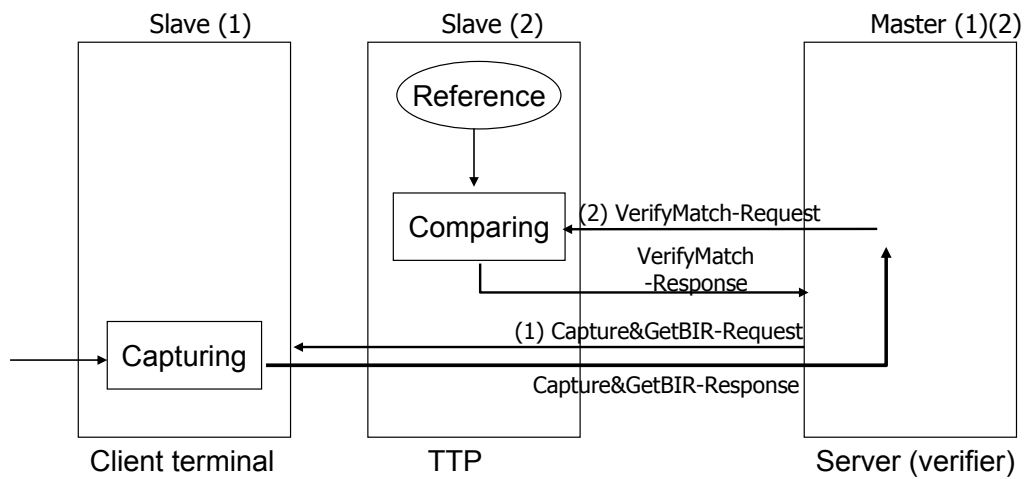


Figure II.12 – Storage and comparison outsourcing by server model

As illustrated in Figure II.12, the verifier first requests the capture command and the GetBIRFromHandle command so as to acquire the client biometric sample. Secondly, the verifier sends a comparison process request to TTP, using the VerifyMatch-Request with the sample.

The verifier in the VerifyMatch-Request should also request the addressed reference template on TTP, using the ID information on the PKI certificate.

Appendix III

Template registration and updating process for this Recommendation

(This appendix does not form an integral part of this Recommendation)

III.1 Registration process

This registration process is conducted without a telecommunications environment. This process should be carried out in a face-to-face environment. Therefore, everyone is able to trust the relationship between the user's certificate and the biometric reference that is formed.

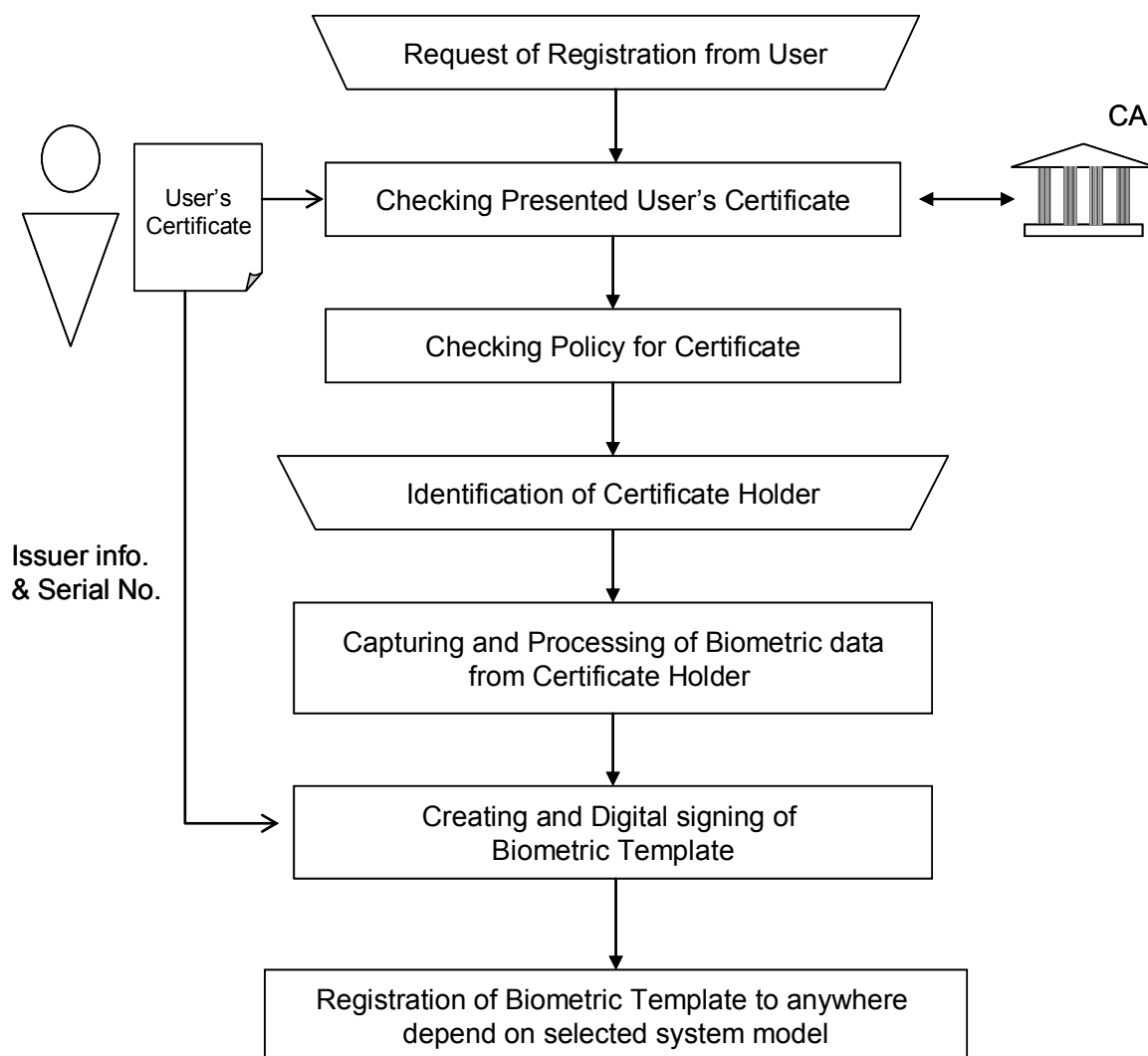


Figure III.1 – Registration process and procedure for this Recommendation

1) Request of registration from user

User requests a registration process to the operator of a telebiometric template issuer. The issuer should be certified from CA, and the operator should manage his or her operation from the issuer, in a certified manner, for the purposes of security and privacy.

User presents a registration form with his Certificate to the operator.

2) Checking presented user's certificate

The operator checks the validity and integrity of the presented user's certificate using CA's public

key and CA's CRL information.

3) Checking policy for certificate

The operator checks the policy for the certificate, i.e., cipher algorithm, key length, etc.

4) Identification of certificate holder

The operator identifies the certificate holder, (user), using public ID items, (i.e., drivers license, passport, national ID card, etc.), with portrait photography.

5) Capturing and processing of the biometric sample from the certificate holder

The operator captures a biometric sample from the user, and creates a biometric template.

6) Creating and digital signing of the biometric template

The operator collects the certificate information, (CA information and serial number), and digital-signs the information with the biometric template using the issuer's private key.

7) Registration of biometric reference to anywhere depending on the selected system model

The operator stores the digital-signed biometric reference to anywhere, depending on the selected system model on the registration form. If the user selects the client storage model and the application server storage model, then the operator stores it on storage media, and presents it to the user. If the user selects the TTP storage model, then the operator stores it to the TTP database.

NOTE – The "Identification of Certificate Holder" procedures are described by [b-ITU-T X.1086]. The transportation procedures on the "Registration of Biometric Template" are also described by [b-ITU-T X.1086].

III.2 Updating or revocation process

The biometric reference has an accuracy decline feature with the aging of end-users. It, therefore, must be updated prior to going off the reference. This is a primary process for biometric reference updating. (In the future, this process may be revised in light of advances in the updating process.)

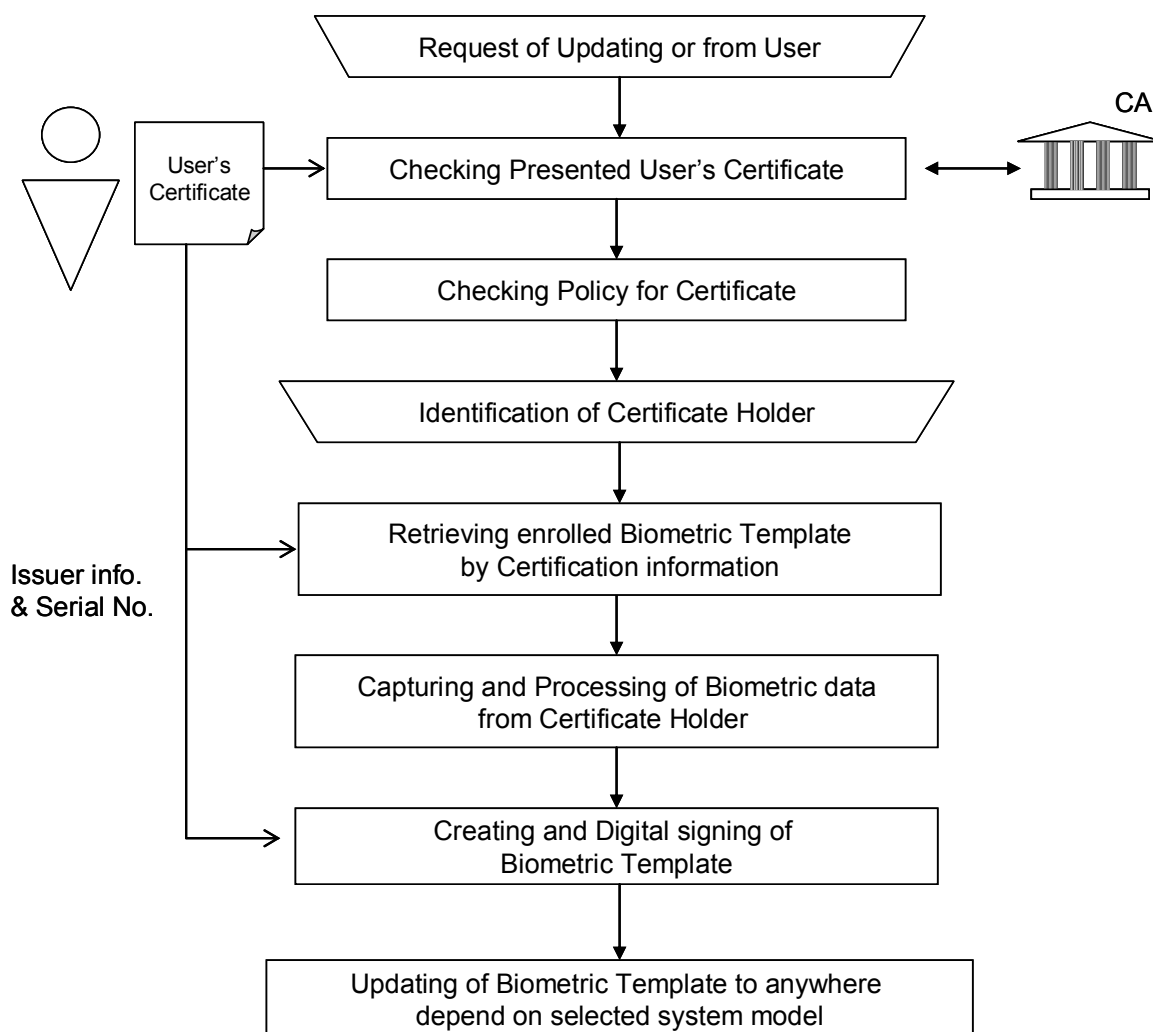


Figure III.2 – Updating process and procedure for this Recommendation

1) Request of updating from user

User requests an updating process to the operator of a telebiometric template issuer. The issuer should be certified from CA, and the operator should manage his or her operations from the issuer, in a certified manner, for the purposes of security and privacy.

User presents an updating form with his Certificate to the operator.

2) Checking presented user's certificate

The operator checks the validity and integrity of the presented user's certificate, using CA's public key and CA's CRL information.

3) Checking policy for certificate

The operator checks the policy for the certificate, i.e., cipher algorithm, key length, etc.

4) Identification of certificate holder

The operator identifies the certificate holder, (user), using public ID items, (i.e., drivers license, passport, national ID card, etc.), with portrait photography.

5) Retrieving enrolled biometric reference by certification information depending on the selected system model

If the user selects the TTP storage model, then the operator retrieves the registered old biometric

reference, and deletes the registered biometric reference.

6) Capturing and processing of the biometric sample from certificate holder

The operator captures the biometric sample from the user, and creates a biometric template.

7) Creating and digital-signing of the biometric template

The operator collects the certificate information, (CA information and serial number), and digital-signs the information with the biometric template using an issuer's private key.

8) Updating of biometric reference to anywhere depending on the selected system model

The operator stores the digital-signed biometric reference to anywhere, depending on the selected system model on the registration form. If the user selects the client storage model and the application server storage model, then the operator stores it on storage media, and presents it to the user. If the user selects the TTP storage model, then the operator stores it to the TTP database.

NOTE – The "Identification of Certificate Holder" procedures are described by [b-ITU-T X.1086]. The transportation procedures on the "Updating of Biometric Template" are also described by [b-ITU-T X.1086].

Appendix IV

ASN.1 definitions for the protocol of TSM based on Appendix I

(This appendix does not form an integral part of this Recommendation)

```
TSM{itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0)
    tls-extended-protocol(1) version1(1)}

DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS BioAPI-BFP-SCHEMA, BioAPI-BSP-SCHEMA, BioAPI-FMR, BioAPI-BIR,
        BioAPI-BIR-BIOMETRIC-TYPE
        FROM BIP {joint-iso-itu-t(2) bip(41) modules(0) bip(0) version1(1)}
BiometricCertificate
    FROM TAI {itu-t(0) recommendation(0) x(24) tai(1089) modules(0)
        framework(0) version1(1)}
SignedData
    FROM X9-84-CMS {iso(1) identified-organization(3) tc68(133) country(16)
        x9(840) x9Standards(9) x9-84(84) module(0) cms(2) rev(1)}
SignedDataACBio
    FROM AuthenticationContextForBiometrics
        {iso(1) standard(0) acbio(24761) module(1) acbio(2) version1(1)}
DistinguishedName, Name
    FROM InformationFramework
        {joint-iso-itu-t ds(5) module(1) informationFramework(1) 5}
Certificate, CertificateSerialNumber
    FROM AuthenticationFramework
        {joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 5};

UINT8 ::= INTEGER(0..255)

UINT16 ::= INTEGER(0..65535)

UINT24 ::= INTEGER(0..16777215)

UINT32 ::= INTEGER(0..4294967295)

UINT64 ::= INTEGER(0..18446744073709551615)

Opaque ::= OCTET STRING

BiometricType ::= BioAPI-BIR-BIOMETRIC-TYPE

SampleData ::= BioAPI-BIR

HandshakeType ::= INTEGER {
    hello-request          (0),
    client-hello           (1),
    server-hello           (2),
    certificate-list        (11),
    server-key-exchange     (12),
    certificate-request     (13),
    server-hello-done       (14),
    certificate-verify      (15),
    client-key-exchange     (16),
    finished                (20),
    certificate-url         (21),
    certificate-status      (22),
    biometric-client-hello  (100)
    -- used only if TLS extension type is 2 --,
    biometric-server-hello  (101)
    -- used only if TLS extension type is 2 --,
    biometric-verify        (102),
    biometric-retry-request (103),
    biometric-finished      (104),
    biometric-ttp-request   (105),
    biometric-ttp-response  (106)
} (0..255)
```

```

HANDSHAKE ::= CLASS {
    &Type,
    &id HandshakeType    UNIQUE
}
WITH SYNTAX {
    &Type IDENTIFIED BY &id
}

Handshake ::= SEQUENCE {
    type HANDSHAKE.&id({Handshakes}),
    value HANDSHAKE.&Type({Handshakes}{@type})
}

Handshakes HANDSHAKE ::= {
    helloRequest |
    clientHello |
    serverHello |
    certificateList |
    serverKeyExchange |
    certificateRequest |
    serverHelloDone |
    certificateVerify |
    clientKeyExchange |
    finished |
    certificateURL |
    certificateStatus |
    biometricClientHello |
    biometricServerHello |
    biometricVerify |
    biometricRetryRequest |
    biometricFinished |
    biometricTTPRequest |
    biometricTTPResponse,
    ...
}

helloRequest    HANDSHAKE ::= {
    HelloRequest    IDENTIFIED BY hello-request
}

HelloRequest    ::= NULL

clientHello HANDSHAKE ::= {
    ClientHello    IDENTIFIED BY client-hello
}

ClientHello ::= SEQUENCE {
    client-version    ProtocolVersion,
    random            ClientRandom,
    session-id        SessionID,
    cipher-suites      CipherSuites,
    compression-methods CompressionMethods,
    ...,
    ...,
    client-hello-extension-list TLS-ExtensionValues
}

ProtocolVersion ::= SEQUENCE {
    major    UINT8,
    minor    UINT8
}

ClientRandom ::= SEQUENCE {
    gmt-unix-time    UINT32,
    random-bytes      Opaque(SIZE(25))
}

SessionID ::=    UINT32

CipherSuites    ::=    SEQUENCE(SIZE(1..32767)) OF CipherSuite

CipherSuite ::=    ENUMERATED {
    tls-null-with-null-null                (0),
    tls-rsa-with-null-md5                  (1),
    tls-rsa-with-null-sha                    (2),

```

```

tls-rsa-export-with-rc4-40-md5          (3),
tls-rsa-with-rc4-128-md5                (4),
tls-rsa-with-rc4-128-sha                (5),
tls-rsa-export-with-rc2-cbc-40-md5      (6),
tls-rsa-with-idea-cbc-sha                (7),
tls-rsa-export-with-des40-cbc-sha        (8),
tls-rsa-with-des-cbc-sha                 (9),
tls-rsa-with-3des-edc-cbc-sha            (10),
tls-dh-dss-export-with-des40-cbc-sha    (11),
tls-dh-dss-with-des-cbc-sha             (12),
tls-dh-dss-with-3des-edc-cbc-sha        (13),
tls-dh-rsa-export-with-des40-cbc-sha     (14),
tls-dh-rsa-with-des-cbc-sha             (15),
tls-dh-rsa-with-3des-edc-cbc-sha        (16),
tls-dhe-dss-export-with-des40-cbc-sha    (17),
tls-dhe-dss-with-des-cbc-sha            (18),
tls-dhe-dss-with-3des-edc-cbc-sha        (19),
tls-dhe-rsa-export-with-des40-cbc-sha     (20),
tls-dhe-rsa-with-des-cbc-sha            (21),
tls-dhe-rsa-with-3des-edc-cbc-sha        (22),
tls-dh-anon-export-rc4-40-md5           (23),
tls-dh-anon-with-rc4-128-md5            (24),
tls-dh-anon-export-with-des40-cbc-sha    (25),
tls-dh-anon-with-des-cbc-sha            (26),
tls-dh-anon-with-3des-edc-cbc-sha        (27),
-- numbers 28 and 29 are reserved to prevent confusion with SSLv3
tls-krb5-with-des-cbc-sha                (30),
tls-krb5-with-3des-edc-cbc-sha           (31),
tls-krb5-with-rc4-128-sha                (32),
tls-krb5-with-idea-cbc-sha               (33),
tls-krb5-with-des-cbc-md5                (34),
tls-krb5-with-3des-edc-cbc-md5           (35),
tls-krb5-with-rc4-128-md5                (36),
tls-krb5-with-idea-cbc-md5               (37),
tls-krb5-export-with-des-cbc-40-sha       (38),
tls-krb5-export-with-rc2-cbc-40-sha       (39),
tls-krb5-export-with-rc4-40-sha           (40),
tls-krb5-export-with-des-cbc-40-md5       (41),
tls-krb5-export-with-rc2-cbc-40-md5       (42),
tls-krb5-export-with-rc4-40-md5           (43),
tls-psk-with-null-sha                    (44),
tls-dhe-psk-with-null-sha                (45),
tls-rsa-psk-with-null-sha                (46),
tls-rsa-with-aes-128-cbc-sha              (47),
tls-dh-dss-with-aes-128-cbc-sha           (48),
tls-dh-rsa-with-aes-128-cbc-sha           (49),
tls-dhe-dss-with-aes-128-cbc-sha         (50),
tls-dhe-rsa-with-aes-128-cbc-sha         (51),
tls-dh-anon-with-aes-128-cnc-sha         (52),
tls-rsa-with-aes-256-cbc-sha              (53),
tls-dh-dss-with-aes-256-cbc-sha           (54),
tls-dh-rsa-with-aes-256-cbc-sha           (55),
tls-dhe-dss-with-aes-256-cbc-sha         (56),
tls-dhe-rsa-with-aes-256-cbc-sha         (57),
tls-dh-anon-with-aes-256-cbc-sha         (58),
-- numbers 59 to 64 are not allocated --
tls-rsa-with-camellia-128-cbc-sha         (65),
tls-dh-dss-with-camellia-128-cbc-sha      (66),
tls-dh-rsa-with-camellia-128-cbc-sha      (67),
tls-dhe-dss-with-camellia-128-cbc-sha     (68),
tls-dhe-rsa-with-camellia-128-cbc-sha     (69),
tls-dh-anon-with-camellia-128-cbc-sha     (70),
-- numbers 71 to 131 are reserved or used by some implementations --
tls-rsa-with-camellia-256-cbc-sha         (132),
tls-dh-dss-with-camellia-256-cbc-sha      (133),
tls-dh-rsa-with-camellia-256-cbc-sha      (134),
tls-dhe-dss-with-camellia-256-cbc-sha     (135),
tls-dhe-rsa-with-camellia-256-cbc-sha     (136),
tls-dh-anon-with-camellia-256-cbc-sha     (137),
tls-psk-with-rc4-128-sha                  (138),
tls-psk-with-3des-edc-cbc-sha             (139),

```

```

tls-psk-with-aes-128-cbc-sha      (140),
tls-psk-with-aes-256-cbc-sha     (141),
tls-dhe-psk-with-rc4-128-sha     (142),
tls-dhe-psk-with-3des-edc-cbc-sha (143),
tls-dhe-psk-with-aes-128-cbc-sha (144),
tls-dhe-psk-with-aes-256-cbc-sha (145),
tls-rsa-psk-with-rc4-128-sha     (146),
tls-rsa-psk-with-3des-edc-cbc-sha (147),
tls-rsa-psk-with-aes-128-cbc-sha (148),
tls-rsa-psk-with-aes-256-cbc-sha (149),
tls-rsa-with-seed-cbc-sha        (150),
tls-dh-dss-with-seed-cbc-sha     (151),
tls-dh-rsa-with-seed-cbc-sha     (152),
tls-dhe-dss-with-seed-cbc-sha    (153),
tls-dhe-rsa-with-seed-cbc-sha    (154),
tls-dh-anon-with-seed-cbc-sha    (155),
-- unallocated numbers --
tls-ecdh-ecdsa-with-null-sha     (49153),
tls-ecdh-ecdsa-with-rc4-128-sha  (49154),
tls-ecdh-ecdsa-with-3des-edc-cbc-sha (49155),
tls-ecdh-ecdsa-with-aes-128-cbc-sha (49156),
tls-ecdh-ecdsa-with-aes-256-cbc-sha (49157),
tls-ecdh-ecdsa-with-null-sha     (49158),
tls-ecdh-ecdsa-with-rc4-128-sha  (49159),
tls-ecdh-ecdsa-with-3des-edc-cbc-sha (49160),
tls-ecdh-ecdsa-with-aes-128-cbc-sha (49161),
tls-ecdh-ecdsa-with-aes-256-cbc-sha (49162),
tls-ecdh-rsa-with-null-sha       (49163),
tls-ecdh-rsa-with-rc4-128-sha    (49164),
tls-ecdh-rsa-with-3des-edc-cbc-sha (49165),
tls-ecdh-rsa-with-aes-128-cbc-sha (49166),
tls-ecdh-rsa-with-aes-256-cbc-sha (49167),
tls-ecdh-rsa-with-null-sha       (49168),
tls-ecdh-rsa-with-rc4-128-sha    (49169),
tls-ecdh-rsa-with-3des-edc-cbc-sha (49170),
tls-ecdh-rsa-with-aes-128-cbc-sha (49171),
tls-ecdh-rsa-with-aes-256-cbc-sha (49172),
tls-ecdh-anon-with-null-sha      (49173),
tls-ecdh-anon-with-rc4-128-sha   (49174),
tls-ecdh-anon-with-3des-edc-cbc-sha (49175),
tls-ecdh-anon-with-aes-128-cbc-sha (49176),
tls-ecdh-anon-with-aes-256-cbc-sha (49177),
...
}

CompressionMethods ::= SEQUENCE(SIZE(1..255)) OF CompressionMethod

CompressionMethod ::= ENUMERATED {
    null,
    ...
}

ContentType ::= ENUMERATED {
    change-cipher-spec (20),
    alert (21),
    handshake (22),
    application-data (23),
    ...
}

TLSPlainText ::= SEQUENCE {
    type ContentType,
    version ProtocolVersion,
    fragment Opaque(SIZE(0..65535))
}

TLSCompressed ::= SEQUENCE {
    type ContentType,
    version ProtocolVersion,
    fragment Opaque(SIZE(0..65535))
}

```

```

TLSCipherText ::= SEQUENCE {
    type      ContentType,
    version    ProtocolVersion,
    fragment   CHOICE {
        stream    GenericStreamCipher,
        block    GenericBlockCipher
    }
}

TLSStreamCipherText ::= SEQUENCE {
    type      ContentType,
    version    ProtocolVersion,
    fragment   GenericStreamCipher
}

TLSBlockCipherText ::= SEQUENCE {
    type      ContentType,
    version    ProtocolVersion,
    fragment   GenericBlockCipher
}

GenericStreamCipher ::= SEQUENCE {
    content    Opaque(SIZE(0..65535)),
    mAC        HASH{Opaque}
}

HASH{ToBeHashed} ::= Opaque(SIZE(0..255))
                    (CONSTRAINED BY {ToBeHashed})

GenericBlockCipher ::= SEQUENCE {
    content    Opaque(SIZE(0..65535)),
    mAC        HASH{Opaque},
    padding    Opaque(SIZE(0..255))
                (CONSTRAINED BY {-- each octet contains the number of
                -- padding octets minus 1 to obtain
                -- a length multiple of block length
                GenericBlockCipher})
}

ChangeCipherSpec ::= ENUMERATED {
    change-cipher-spec(1),
    ...
}

serverHello HANDSHAKE ::= {
    ServerHello IDENTIFIED BY server-hello
}

ServerHello ::= SEQUENCE {
    server-version      ProtocolVersion,
    random              ServerRandom,
    session-id          SessionID,
    cipher-suite         CipherSuite,
    compression-method  CompressionMethod,
    ...,
    ...,
    server-hello-extension-list  TLS-ExtensionValues
}

ServerRandom ::= SEQUENCE {
    gmt-unix-time      UINT32,
    random-bytes       Opaque(SIZE(57))
}

certificateList HANDSHAKE ::= {
    CertificateList IDENTIFIED BY certificate-list
}

CertificateList ::= SEQUENCE {
    certificates        Certificates
}

```

```

Certificates          ::= SEQUENCE OF X509Certificate
X509Certificate ::= OCTET STRING (CONTAINING Certificate ENCODED BY der)
der OBJECT IDENTIFIER ::=
    {joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1)}
serverKeyExchangeHANDSHAKE ::= {
    ServerKeyExchangeIDENTIFIED BY server-key-exchange
}
ServerKeyExchange ::= CHOICE {
    rsa [0] SEQUENCE {
        params      ServerRSAParams,
        signed-params Signature
    },
    diffie-hellman [1] SEQUENCE {
        params      ServerDHParams,
        signed-params Signature
    },
    ...
}
ServerDHParams ::= SEQUENCE {
    dh-p      INTEGER (1..65535),
    dh-g      INTEGER (1..65535),
    dh-Ys     INTEGER (1..65535)
}
ServerRSAParams ::= SEQUENCE {
    rsa-modulus INTEGER (1..65535),
    rsa-exponent INTEGER (1..65535)
}
Signature ::= CHOICE {
    anonymous [0] NULL,
    rsa [1] SEQUENCE {
        md5-hash Opaque (SIZE (16)),
        sha-hash Opaque (SIZE (20))
    },
    dsa [2] SEQUENCE {
        sha-hash Opaque (SIZE (20))
    },
    ...
}
certificateRequest HANDSHAKE ::= {
    CertificateRequest IDENTIFIED BY certificate-request
}
CertificateRequest ::= SEQUENCE {
    certificate-types ClientCertificateTypes,
    certificate-authorities DistinguishedNames
}
ClientCertificateTypes ::= SEQUENCE OF ClientCertificateType
ClientCertificateType ::= ENUMERATED {
    rsa-sign (1),
    dss-sign (2),
    rsa-fixed-dh (3),
    dss-fixed-dn (4),
    ...
}
DistinguishedNames ::= SEQUENCE OF DistinguishedName
serverHelloDone HANDSHAKE ::= {
    ServerHelloDone IDENTIFIED BY server-hello-done
}
ServerHelloDone ::= NULL
clientKeyExchangeHANDSHAKE ::= {
    ClientKeyExchangeIDENTIFIED BY client-key-exchange
}

```

```

ClientKeyExchange ::= Opaque(SIZE(0..65535))

PreMasterSecret ::= SEQUENCE {
    client-version ProtocolVersion,
    random          Opaque(SIZE(46))
}

EncryptedPreMasterSecret ::= ENCRYPTED{PreMasterSecret}

ClientDiffieHellmanPublic ::= CHOICE {
    implicit    NULL,
    explicit    Opaque(SIZE(1..65535))
}

ENCRYPTED{ToBeEnciphered} ::= OCTET STRING(SIZE(0..255))
(CONSTRAINED BY {ToBeEnciphered})

certificateVerifyHANDSHAKE ::= {
    CertificateVerifyIDENTIFIED BY certificate-verify
}

CertificateVerify ::= SEQUENCE {
    signature Signature
}

finished HANDSHAKE ::= {
    Finished IDENTIFIED BY finished
}

Finished ::= SEQUENCE {
    verify-data Opaque(SIZE(12))
}

certificateURL HANDSHAKE ::= {
    CertificateURL IDENTIFIED BY certificate-url
}

CertificateURL ::= SEQUENCE {
    type CertChainType,
    url-and-hash-list URLAndOptionalHashList
}

CertChainType ::= ENUMERATED {
    individual-certs (0),
    pkipath (1),
    ...
}

URLAndOptionalHashList ::= SEQUENCE OF URLAndOptionalHash

URLAndOptionalHash ::= SEQUENCE {
    url Opaque(SIZE(1..65535)),
    hash SHA1Hash OPTIONAL
}

SHA1Hash ::= Opaque(SIZE(20))

certificateStatusHANDSHAKE ::= {
    CertificateStatusIDENTIFIED BY certificate-status
}

CertificateStatus ::= CHOICE {
    ocspl OCSPLResponse,
    ...
}

OCSPLResponse ::= Opaque(SIZE(1..16777215))

EXTENSION ::= CLASS {
    &id ExtensionType UNIQUE,
    &Type
}
WITH SYNTAX {
    &Type IDENTIFIED BY &id
}

```

```

ExtensionType      ::=  INTEGER(0..65535)

server-name        EXTENSION ::= {
    ServerNameList IDENTIFIED BY 0
}

ServerNameList     ::=  SEQUENCE {
    server-name-list ListOfServerName
}

ListOfServerName   ::=  SEQUENCE OF ServerName

ServerName         ::=  CHOICE {
    host-name       [0]   HostName,
    ...
}

HostName           ::=  Opaque(SIZE(1..65535))

max-fragment-length EXTENSION ::= {
    MaxFragmentLength IDENTIFIED BY 1
}

MaxFragmentLength ::=  INTEGER(512 | 1024 | 2048 | 4096,...)

client-certificate-url EXTENSION ::= {
    ClientCertificateURL IDENTIFIED BY 2
}

ClientCertificateURL ::= CertificateURL

trusted-ca-keys     EXTENSION ::= {
    TrustedAuthorities IDENTIFIED BY 3
}

TrustedAuthorities ::=  SEQUENCE {
    trusted-authorities-list ListOfTrustedAuthority
}

ListOfTrustedAuthority ::= SEQUENCE OF TrustedAuthority

TrustedAuthority    ::=  CHOICE {
    pre-agreed       [0]   NULL,
    key-shal-hash    [1]   SHA1Hash,
    x509-name         [2]   DistinguishedName,
    cert-shal-hash    [3]   SHA1Hash,
    ...
}

truncated-hmac      EXTENSION ::= {
    TruncatedHMAC     IDENTIFIED BY 4
}

TruncatedHMAC       ::=  Opaque(SIZE(10))

status-request      EXTENSION ::= {
    CertificateStatusRequest IDENTIFIED BY 5
}

CertificateStatusRequest ::= CHOICE {
    ocsp              [0]   OCSPStatusRequest,
    ...
}

OCSPStatusRequest   ::=  SEQUENCE {
    responder-id-list ResponderIDList,
    request-extensions Extensions
}

ResponderIDList     ::=  SEQUENCE OF ResponderID

ResponderID         ::=  Opaque(SIZE(1..65535))

Extensions          ::=  Opaque(SIZE(0..65535))

TLS-Extensions      EXTENSION ::= {
    server-name |
    max-fragment-length |

```

```

    client-certificate-url |
    trusted-ca-keys |
    truncated-hmac |
    status-request,
    ...
}

TLS-ExtensionValues ::= SEQUENCE OF TLS-ExtensionValue

TLS-ExtensionValue ::= SEQUENCE {
    extension-type EXTENSION.&id({TLS-Extensions}),
    extension-data EXTENSION.&Type({TLS-Extensions}){@extension-type}
}

biometricClientHello HANDSHAKE ::= {
    BiometricClientHello IDENTIFIED BY biometric-client-hello
}

BiometricClientHello ::= SEQUENCE(SIZE(1..MAX)) OF BiometricMethod

BiometricMethod ::= SEQUENCE {
    biometricType BiometricType,
    biometricFunctionProvider BSP-BFP-Schema,
    networkAuthenticationModel NetworkAuthenticationModel,
    thirdPartyInfo UTF8String
}

BSP-BFP-Schema ::= CHOICE {
    bSPSchema [0] BioAPI-BSP-SCHEMA,
    bFPSchema [1] BioAPI-BFP-SCHEMA
}

BSP-BFP-Schemas ::= SEQUENCE(SIZE(1..MAX)) OF BSP-BFP-Schema

NetworkAuthenticationModel ::= ENUMERATED {
    no-value (0), -- no selection --
    local-model (1),
    download-model (2),
    attached-model (3),
    center-model (4),
    ref-onhttp-for-local-model (5),
    ref-onhttp-for-center-model (6),
    comparison-outsourcing-by-client-model (7),
    comparison-outsourcing-by-server-model (8),
    storage-comparison-outsourcing-by-client-model (9),
    storage-comparison-outsourcing-by-server-model (10),
    ...
}

biometricServerHello HANDSHAKE ::= {
    BiometricServerHello IDENTIFIED BY biometric-server-hello
}

BiometricServerHello ::= BiometricAuthenticationRequest

Quality ::= INTEGER(0..100)

BiometricAuthenticationRequest ::= SEQUENCE {
    biometricMethod BiometricMethod,
    requestFMR BioAPI-FMR,
    -- (32-bit integer value:requestFMR/231-1)
    requestTrialNumber INTEGER(1..15),
    requestQuality Quality,
    requestTemplateData XtsmTemplate OPTIONAL
    -- for download model (no value available)
}

XtsmTemplate ::= BiometricCertificate -- Import from TAI

biometricVerify HANDSHAKE ::= {
    BiometricVerify IDENTIFIED BY biometric-verify
}

BiometricVerify ::= SEQUENCE {
    biometricData CHOICE {
        no-value [0] NULL,

```

```

        local-model [1] BDforLocalModel,
        download-model [2] BDforDownloadModel,
        attached-model [3] BDforAttachedModel,
        center-model [4] BDforCenterModel,
        ref-onttp-for-local-model [5] BDforRefOnTTPforLocalModel,
        ref-onttp-for-center-model [6] BDforRefOnTTPforCenterModel,
        comparison-outsourcing-by-client-model [7] BDforCObyClientModel,
        comparison-outsourcing-by-server-model [8] BDforCObyServerModel,
        storage-comparison-outsourcing-by-client-model [9]
            BDforSCbyClientModel,
        storage-comparison-outsourcing-by-server-model [10]
            BDforSCbyServerModel,
        ...
    },
    digitalSignature SignedDatabyClient
}

SignedDatabyClient ::= CHOICE {
    digital-signature [0] SignedData,
        --import from X9.84-CMS
    aCBioOnClient [1] SignedDataACBio
        --import from ISO/IEC 24761
}

BDforLocalModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess
}

BiometricClientProcess ::= SEQUENCE {
    bFPSchema BSP-BFP-Schemas,
    templateID TemplateID,
    sampleQuality Quality,
    score BioAPI-FMR
}

TemplateID ::= SEQUENCE {
    certificateIssuer Name, -- see Rec. ITU-T X.509
    serialNumber CertificateSerialNumber, -- see Rec. ITU-T X.509
    templateInfo TemplateInfo
}

TemplateInfo ::= SEQUENCE {
    biometricType BiometricType,
    creator UTF8String,
    createdBFPSchema BSP-BFP-Schema,
    templateID CertificateIDInformation
        -- such as CertificateSerialNumber (no value available)
}

CertificateIDInformation ::= CertificateSerialNumber

BDforDownloadModel ::= SEQUENCE {
    biometricClientProcess BiometricClientProcess
}

BDforAttachedModel ::= SEQUENCE {
    templateData XtsmTemplate,
    sampleData SampleData -- BIR: BioAPI defined format --
}

BDforCenterModel ::= SEQUENCE {
    sampleData SampleData -- BIR: BioAPI defined format --
}

BDforRefOnTTPforLocalModel ::= SEQUENCE {
    thirdPartyInfo UTF8String,
    biometric-ttp-process BiometricTTPResponse OPTIONAL,
    biometricClientProcess BiometricClientProcess
}

BDforRefOnTTPforCenterModel ::= SEQUENCE {
    thirdPartyInfo UTF8String,
    sampleData SampleData -- BIR: BioAPI defined format --
}

```

```

BDforCObyClientModel ::= SEQUENCE {
    bFPSchemaForClientProcess    BSP-BFP-Schemas,
    thirdPartyInfo               UTF8String,
    biometric-ttp-Process        BiometricTTPResponse
}

BDforCObyServerModel ::= SEQUENCE {
    sampleData SampleData        -- BIR: BioAPI defined format --
}

BDforSCobyClientModel ::= SEQUENCE {
    bFPSchemaForClientProcess    BSP-BFP-Schemas,
    thirdPartyInfo               UTF8String,
    biometric-ttp-Process        BiometricTTPResponse
}

BDforSCobyServerModel ::= SEQUENCE {
    sampleData SampleData        -- BIR: BioAPI defined format --
}

biometricRetryRequest HANDSHAKE ::= {
    BiometricRetryRequest IDENTIFIED BY biometric-retry-request
}

BiometricRetryRequest ::= SEQUENCE {
    retryRequest    BiometricAuthenticationRequest
}

Alert ::= SEQUENCE {
    level           AlertLevel,
    description     AlertDescription
}

AlertLevel ::= ENUMERATED {
    warning    (1),
    fatal      (2)
}

AlertDescription ::= ENUMERATED {
    close-notify                (0),
    unexpected-message          (10),
    bad-record-mac              (20),
    decryption-failed           (21),
    record-overflow              (22),
    decompression-failure       (30),
    handshake-failure           (40),
    bad-certificate             (42),
    unsupported-certificate      (43),
    certificate-revoked          (44),
    certificate-expired          (45),
    certificate-unknown          (46),
    illegal-parameter           (47),
    unknown-ca                  (48),
    access-denied               (49),
    decode-error                (50),
    decrypt-error               (51),
    export-restriction           (60),
    protocol-version            (70),
    insufficient-security       (71),
    internal-error              (80),
    user-canceled               (90),
    no-renegotiation            (100),
    ...,
    unacceptable-model          (115), -- Extension item for BiometricHandshake
    unacceptable-biometrics     (116), -- Extension item for BiometricHandshake
    unsupported-biometrics      (117), -- Extension item for BiometricHandshake
    bad-template                (118), -- Extension item for BiometricVerify
    template-revoked            (119), -- Extension item for BiometricVerify
    template-expired            (120), -- Extension item for BiometricVerify
    unknown-bca                 (121), -- Extension item for BiometricVerify
    unacceptable-fmr            (122), -- Extension item for BiometricVerify
    no-response-ttp             (123), -- Extension item for TTP
    no-template-in-ttp          (124) -- Extension item for TTP
}

```

```

biometricFinishedHANDSHAKE ::= {
    BiometricFinishedIDENTIFIED BY biometric-finished
}

BiometricFinished ::= SEQUENCE {
    result      BiometricAuthenticationResult
}

BiometricAuthenticationResult ::= BOOLEAN

biometricTTPRequest    HANDSHAKE ::= {
    BiometricTTPRequest    IDENTIFIED BY biometric-ttp-request
}

BiometricTTPRequest ::= CHOICE {
    storage-type           [1] BDforStorageOutsourcing,
    comparison-type        [2] BDforComparisonOutsourcing,
    storage-comparison-type [3] BDforComparisonOutsourcing
}

BDforStorageOutsourcing ::= SEQUENCE {
    templateID TemplateID
}

BDforComparisonOutsourcing ::= SEQUENCE {
    templateData    XtsmTemplate,
    sampleData      SampleData      --BIR: BioAPI defined format
}

BDforSCOutsourcing ::= SEQUENCE {
    templateID      TemplateID,
    sampleData      SampleData      --BIR: BioAPI defined format
}

biometricTTPResponse    HANDSHAKE ::= {
    BiometricTTPResponse    IDENTIFIED BY biometric-ttp-response
}

BiometricTTPResponse ::= SEQUENCE {
    request-body CHOICE {
        storage-type           [1] RBDforStorageOutsourcing,
        comparison-type        [2] RBDforComparisonOutsourcing,
        storage-comparison-type [3] RBDforComparisonOutsourcing
    },
    digital-signatureSignedDatabyTTP
}

RBDforStorageOutsourcing ::= SEQUENCE {
    templateData    XtsmTemplate
}

RBDforComparisonOutsourcing ::= SEQUENCE {
    bFPSchema      BSP-BFP-Schemas,
    templateID      TemplateID,
    sampleQuality    Quality,
    score           BioAPI-FMR
}

SignedDatabyTTP ::= CHOICE {
    digital-signature [0] SignedData,
    --import from X9.84-CMS
    aCBioOnTTP        [1] SignedDataACBio
    --import from ISO/IEC 24761
}

ApplicationData ::= Opaque

END

```

Appendix V

ECN modules for Appendix IV

(This appendix does not form an integral part of this Recommendation)

This appendix contains ECN modules (see [b-ITU-T X.692]) to encode and decode the PDUs contained in Annex C of [b-ITU-T X.692]. Using these ECN modules (encoding definition module and encoding link module), the encodings of the PDUs will be compatible with TLS encodings described in [b-IETF RFC 2246], [IETF RFC 4346] and [IETF RFC 4366].

V.1 EDM module

```
TSM-ENCODING{itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0)
               tls-extended-protocol-encoding(3) version1(1)}

ENCODING-DEFINITIONS ::=
    BEGIN

EXPORTS
    TSM-encodings,
    Directory-encodings,
    HelloRequest-encodings, ClientHello-encodings, ServerHello-encodings,
    CertificateList-encodings, ServerKeyExchange-encodings,
    CertificateRequest-encodings,
    ServerHelloDone-encodings, CertificateVerify-encodings,
    ClientKeyExchange-encodings,
    Finished-encodings, CertificateURL-encodings,
    CertificateStatus-encodings,
    BiometricClientHello-encodings,
    BiometricServerHello-encodings,
    BiometricVerify-encodings, BiometricRetryRequest-encodings,
    BiometricFinished-encodings, BiometricTTPRequest-encodings,
    BiometricTTPResponse-encodings;

IMPORTS
    #Opaque, #Handshake, #HelloRequest, #ClientHello,
    #ProtocolVersion, #ClientRandom, #CipherSuites,
    #CompressionMethods, #TLSPlainText,
    #TLSCompressed, #TLSCipherText, #TLSStreamCipherText, #TLSBlockCipherText,
    #GenericStreamCipher, #GenericBlockCipher, #ChangeCipherSpec,
    #ServerHello, #ServerRandom,
    #CertificateList, #Certificates, #ServerKeyExchange,
    #ServerDHParams, #ServerRSAParams, #Signature,
    #CertificateRequest, #ClientCertificateTypes,
    #DistinguishedNames, #ServerHelloDone, #ClientKeyExchange,
    #PreMasterSecret,
    #CertificateVerify,
    #Finished, #CertificateURL, #URLAndOptionalHashList,
    #URLAndOptionalHash, #SHA1Hash, #CertificateStatus,
    #ServerNameList, #ListOfServerName, #ServerName,
    #MaxFragmentLength, #ClientCertificateURL, #TrustedAuthorities,
    #ListOfTrustedAuthority, #TrustedAuthority, #TruncatedHMAC,
    #CertificateStatusRequest, #OCSPStatusRequest,
    #ResponderIDList, #TLS-ExtensionValues,
    #TLS-ExtensionValue, #BiometricClientHello, #BiometricMethod,
    #BSP-BFP-Schema, #BSP-BFP-Schemas,
    #BiometricServerHello, #Quality,
    #BiometricAuthenticationRequest, #XtsmTemplate, #BiometricVerify,
    #SignedDataByClient, #BDforLocalModel, #BiometricClientProcess,
    #TemplateID, #TemplateInfo, #CertificateIDInformation,
    #BDforDownloadModel, #BDforAttachedModel, #BDforCenterModel,
    #BDforRefOnTTPforLocalModel, #BDforRefOnTTPforCenterModel,
    #BDforCObyClientModel,
    #BDforCObyServerModel, #BDforSCobyClientModel, #BDforSCobyServerModel,
    #BiometricRetryRequest, #Alert,
    #BiometricFinished, #BiometricAuthenticationResult,
    #BiometricTTPRequest, #BDforStorageOutsourcing,
```

```

#BDforComparisonOutsourcing,
#BiometricTTPResponse, #RBDforStorageOutsourcing,
#RBDforComparisonOutsourcing, #SignedDatabyTTP,
#ApplicationData, #SampleData
    FROM TSM{itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0)
        tls-extended-protocol(1) version1(1)}
#BiometricCertificate
    FROM TAI{itu-t(0) recommendation(0) x(24) tai(1089)
        modules(0) framework(0) version1(1)}
#BioAPI-BFP-SCHEMA, #BioAPI-BSP-SCHEMA, #BioAPI-FMR, #BioAPI-BIR
    FROM BIP{joint-iso-itu-t bip(41) modules(0) bip(0) version1(1)}
#DistinguishedName, #Name
    FROM InformationFramework{joint-iso-itu-t ds(5) module(1)
        informationFramework(1) 5}
#Certificate, #CertificateSerialNumber
    FROM AuthenticationFramework{joint-iso-itu-t ds(5) module(1)
        authenticationFramework(7) 5}
#ID
    FROM UsefulDefinitions{joint-iso-itu-t ds(5) module(1)
        usefulDefinitions(0) 5}
#SignedData
    FROM CryptographicMessageSyntax2004{iso(1) member-body(2) us(840)
        rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)
        cms-2004(24)}
#SignedDataACBio
    FROM AuthenticationContextForBiometrics{iso(1) standard(0)
        acbio(24761) module(1) acbio(2) version1(1)}
#BiometricType
    FROM CBEFF-DATA-ELEMENTS{iso standard 19785 modules(0)
        types-for-cbeff-data-elements(1)};

TSM-encodings    #ENCODINGS ::= {
    outer-encoding |
    iD-encoding |
    tLSPlainText-encoding |
    tLSCompressed-encoding |
    tLSCipherText-encoding |
    tLSStreamCipherText-encoding |
    tLSBlockCipherText-encoding |
    changeCipherSpec-encoding |
    sampleData-encoding |
    alert-encoding |
    handshake-encoding |
    preMasterSecret-encoding |
    applicationData-encoding
}

Directory-encodings    #ENCODINGS ::= {
    BER }

outer-encoding #OUTER ::= {
    ADDED BITS DECODING next-value
}

iD-encoding    #ID    ::= {
    ENCODE WITH Directory-encodings
}

integer-1-encoding    #INT ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 1
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

```

```

integer-2-encoding    #INT ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 2
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

integer-3-encoding    #INT ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 3
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

integer-4-encoding    #INT ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 4
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

enumerated-1-encoding #ENUMERATED ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 1
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

enumerated-2-encoding #ENUMERATED ::= {
    ENCODING {
        ENCODING-SPACE
        SIZE 2
        MULTIPLE OF octet
        ENCODING positive-int
    }
}

boolean-encoding #BOOLEAN ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    TRUE-PATTERN octets:'01'H
    FALSE-PATTERN octets:'00'H
}

opaque-1-encoding    #OCTETS ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #OpaqueWithLength
            ENCODED BY opaque-1WithLength-encoding
    }
}

#OpaqueWithLength{<#Element>} ::= #CONCATENATION {
    length    #INT,
    element    #Element
}

opaque-1WithLength-encoding{<#Element>}
#OpaqueWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length    integer-1-encoding,
        element    countedOctetString-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

countedOctetString-encoding{<REFERENCE:length>}      #OCTETS ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE
        SIZE variable-with-determinant
        MULTIPLE OF octet
        DETERMINED BY field-to-be-set
        USING length
    }
}

opaque-2-encoding      #OCTETS ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #OpaqueWithLength
        ENCODED BY opaque-2WithLength-encoding
    }
}

opaque-2WithLength-encoding{<#Element>}
#OpaqueWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-2-encoding,
        element countedOctetString-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

opaque-3-encoding#OCTETS ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #OpaqueWithLength
        ENCODED BY opaque-3WithLength-encoding
    }
}

opaque-3WithLength-encoding{<#Element>}
#OpaqueWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-3-encoding,
        element      countedOctetString-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

#SequenceOfWithLength{<#Element>} ::= #CONCATENATION {
    length      #INT,
    value #Element
}

sequenceOfWithLength-encoding{<REFERENCE:length>}      #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE
        SIZE variable-with-determinant
        MULTIPLE OF octet
        DETERMINED BY field-to-be-set
        USING length
    }
}

#StructureWithLength{<#Element>} ::= #CONCATENATION {
    length      #INT,
    value #Element
}

structureWithLength-encoding{<REFERENCE:length>}
#CONCATENATION ::= {
    ENCODING-SPACE
    SIZE variable-with-determinant
    MULTIPLE OF octet
    DETERMINED BY field-to-be-set
    USING length
}

```

```

structure-encoding      #CONCATENATION ::= {
    ENCODING-SPACE
    SIZE self-delimiting-values
}

#Presence ::= #OPTIONAL

presence-encoding{<REFERENCE:flag>}      #OPTIONAL ::= {
    PRESENCE
    DETERMINED BY field-to-be-set
    USING flag
}

tLSPlainText-encoding  #TLSPlainText      ::= {
    ENCODE STRUCTURE {
        type enumerated-1-encoding,
        version USE-SET,
        fragment opaque-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

tLSCompressed-encoding #TLSCompressed      ::= {
    ENCODE STRUCTURE {
        type enumerated-1-encoding,
        version USE-SET,
        fragment opaque-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

tLSCipherText-encoding #TLSCipherText      ::= {
    ENCODE STRUCTURE {
        type enumerated-1-encoding,
        version USE-SET,
        fragment {
            ENCODE STRUCTURE {
                stream genericStreamCipher-encoding,
                block genericBlockCipher-encoding
            }
            STRUCTURED WITH cipherType-encoding
        }
    }
    WITH PER-BASIC-UNALIGNED
}

tLSStreamCipherText-encoding #TLSStreamCipherText ::= {
    ENCODE STRUCTURE {
        type enumerated-1-encoding,
        version USE-SET,
        fragment genericStreamCipher-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

tLSBlockCipherText-encoding #TLSBlockCipherText ::= {
    ENCODE STRUCTURE {
        type enumerated-1-encoding,
        version USE-SET,
        fragment genericBlockCipher-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

genericStreamCipher-encoding #GenericStreamCipher ::= {
    ENCODE STRUCTURE {
        content opaque-2-encoding,
        mAC opaque-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

genericBlockCipher-encoding      #GenericBlockCipher      ::= {
    ENCODE STRUCTURE {
        content opaque-2-encoding,
        mAC      opaque-2-encoding,
        padding  opaque-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

cipherType-encoding      #ALTERNATIVES ::=
    NON-ECN-BEGIN
        extern cipherType;
        return(cipherType);
    NON-ECN-END

changeCipherSpec-encoding #ChangeCipherSpec ::= enumerated-1-encoding

sampleData-encoding      #SampleData ::= {
    ENCODE STRUCTURE {
        patronFormatOwnerUSE-SET,
        patronFormatType USE-SET,
        formattedBIR      formattedBIR-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

formattedBIR-encoding      #OCTETS ::= opaque-2-encoding

alert-encoding      #Alert ::= {
    ENCODE STRUCTURE {
        level      enumerated-1-encoding,
        description enumerated-1-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

handshake-encoding      #OPEN-TYPE ::= {
    REPLACE STRUCTURE
        WITH #OpaqueWithLength
        ENCODED BY handshakeWithLength-encoding
}

handshakeWithLength-encoding{<#Element>}
#OpaqueWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-3-encoding,
        element      handshakeValue-encoding{<length>}}
    WITH PER-BASIC-UNALIGNED
}

handshakeValue-encoding{<REFERENCE:length>} #OPEN-TYPE ::= {
    ENCODING-SPACE
        SIZE variable-with-determinant
        MULTIPLE OF octet
        DETERMINED BY field-to-be-set
        USING length
    ENCODED WITH Handshake-encodings
}

preMasterSecret-encoding      #PreMasterSecret ::= {
    ENCODE STRUCTURE {
        client-version      protocolVersion-encoding,
        random      USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

protocolVersion-encoding      #ProtocolVersion ::= {
    ENCODE STRUCTURE {
        major USE-SET,
        minor USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

biometricCertificate #BiometricCertificate ::= {
    ENCODE WITH Directory-encodings
}

Handshake-encodings #ENCODINGS ::= {
    helloRequest-encoding |
    clientHello-encoding |
    serverHello-encoding |
    certificateList-encoding |
    serverKeyExchange-encoding |
    certificateRequest-encoding |
    serverHelloDone-encoding |
    certificateVerify-encoding |
    clientKeyExchange-encoding |
    finished-encoding |
    certificateURL-encoding |
    certificateStatus-encoding |
    biometricClientHello-encoding |
    biometricServerHello-encoding |
    biometricVerify-encoding |
    biometricRetryRequest-encoding |
    biometricFinished-encoding |
    biometricTTPRequest-encoding |
    biometricTTPResponse-encoding
}

HelloRequest-encodings #ENCODINGS ::= {
    helloRequest-encoding
}

helloRequest-encoding #HelloRequest ::=
    {ENCODE WITH PER-BASIC-UNALIGNED}

ClientHello-encodings #ENCODINGS ::= {
    clientHello-encoding
}

clientHello-encoding #ClientHello ::= {
    ENCODE STRUCTURE {
        client-version                protocolVersion-encoding,
        random                        clientRandom-encoding,
        session-id                    USE-SET,
        cipher-suites                 cipherSuites-encoding,
        compression-methods           compressionMethods-encoding,
        client-hello-extension-list   tLS-Extensions-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

compressionMethods-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY compressionMethodsWithLength-encoding
        }
    }

compressionMethodsWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length integer-1-encoding,
        value compressionMethodsValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

cipherSuites-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY cipherSuitesWithLength-encoding
        }
    }
}

```

```

cipherSuitesWithLength-encoding{<#Element>}
    #SequenceOfWithLength{<#Element>} ::= {
        ENCODE STRUCTURE {
            length      integer-2-encoding,
            value cipherSuitesValue-encoding{<length>}
        }
        WITH PER-BASIC-UNALIGNED
    }

cipherSuitesValue-encoding{<REFERENCE:length>} #CipherSuites ::= {
    ENCODE STRUCTURE {
        enumerated-2-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

ServerNameList-encodings #ENCODINGS ::= {
    serverNameList-encoding
}

MaxFragmentLength-encodings #ENCODINGS ::= {
    maxFragmentLength-encoding
}

ClientCertificateURL-encodings #ENCODINGS ::= {
    clientCertificateURL-encoding
}

TrustedAuthorities-encodings #ENCODINGS ::= {
    trustedAuthorities-encoding
}

TruncatedHMAC-encodings #ENCODINGS ::= {
    truncatedHMAC-encoding
}

CertificateStatusRequest-encodings #ENCODINGS ::= {
    certificateStatusRequest-encoding
}

serverNameList-encoding #ServerNameList ::= {
    ENCODE STRUCTURE {
        server-name-list listOfServerName-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

listOfServerName-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY listOfServerNameWithLength-encoding
    }
}

listOfServerNameWithLength-encoding{<#Element>}
    #SequenceOfWithLength{<#Element>} ::= {
        ENCODE STRUCTURE {
            length      integer-2-encoding,
            value listOfServerNameValue-encoding{<length>}
        }
        WITH PER-BASIC-UNALIGNED
    }

listOfServerNameValue-encoding{<REFERENCE:length>} #ListOfServerName ::= {
    ENCODE STRUCTURE {
        serverName-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

serverName-encoding      #ServerName ::= {
    ENCODE STRUCTURE {
        host-name [serverNameTag-encoding] opaque-2-encoding
        STRUCTURED WITH serverNameChoice-encoding
    }
}

serverNameTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "ServerName" AT {0..7} AS tag:any
}

serverNameChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "ServerName"
}

maxFragmentLength-encoding      #MaxFragmentLength ::= {
    USE #INT-1
    MAPPING VALUES {
        512 TO 1,
        1024 TO 2,
        2048 TO 3,
        4096 TO 4
    }
    WITH PER-BASIC-UNALIGNED
}

#INT-1 ::= #INT(0..255)

clientCertificateURL-encoding      #ClientCertificateURL ::=
    certificateURL-encoding

trustedAuthorities-encoding      #TrustedAuthorities      ::= {
    ENCODE STRUCTURE {
        trusted-authorities-list listOfTrustedAuthority-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

listOfTrustedAuthority-encoding      #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #SequenceOfWithLength
        ENCODED BY listOfTrustedAuthorityWithLength-encoding
    }
}

listOfTrustedAuthorityWithLength-encoding{<#Element>}
    #SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-2-encoding,
        value listOfTrustedAuthorityValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

listOfTrustedAuthorityValue-encoding{<REFERENCE:length>}
    #ListOfTrustedAuthority ::= {
    ENCODE STRUCTURE {
        trustedAuthority-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

trustedAuthority-encoding      #TrustedAuthority ::= {
    ENCODE STRUCTURE {
        pre-agreed [trustedAuthorityTag-encoding] USE-SET,
        key-shal-hash [trustedAuthorityTag-encoding] USE-SET,
        x509-name [trustedAuthorityTag-encoding]
    }
}

```

```

        distinguishedName-encoding,
        cert-shal-hash [trustedAuthorityTag-encoding] USE-SET
        STRUCTURED WITH trustedAuthorityChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

trustedAuthorityTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "TrustedAuthority" AT {0..7} AS tag:any
}

trustedAuthorityChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "TrustedAuthority"
}

truncatedHMAC-encoding #TruncatedHMAC ::= {
    ENCODE WITH PER-BASIC-UNALIGNED
}

certificateStatusRequest-encoding #CertificateStatusRequest ::= {
    ENCODE STRUCTURE {
        oosp [certificateStatusRequestTag-encoding]
        oCSPStatusRequest-encoding
        STRUCTURED WITH certificateStatusRequestChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

certificateStatusRequestTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "CertificateStatusRequest" AT {0..7} AS tag:any
}

certificateStatusRequestChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "CertificateStatusRequest"
}

oCSPStatusRequest-encoding #OCSPStatusRequest ::= {
    ENCODE STRUCTURE {
        responder-id-list responderIDList-encoding,
        request-extensions opaque-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

responderIDList-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #SequenceOfWithLength
        ENCODED BY responderIDListWithLength-encoding
    }
}

responderIDListWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE{
        length integer-2-encoding,
        value responderIDListValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

responderIDListValue-encoding{<REFERENCE:length>} #ResponderIDList ::= {
    ENCODE STRUCTURE {
        opaque-2-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
}

TLS-Extensions-encodings #ENCODINGS ::= {
    ServerNameList-encodings |
    MaxFragmentLength-encodings |
    ClientCertificateURL-encodings |
    TrustedAuthorities-encodings |
    TruncatedHMAC-encodings |
    CertificateStatusRequest-encodings
}

tLS-Extensions-encoding #TLS-ExtensionValues ::= {
    ENCODE STRUCTURE {
        tLS-ExtensionValue-encoding
        STRUCTURED WITH sequenceOfTLS-Extension-encoding
    }
}

tLS-ExtensionValue-encoding #TLS-ExtensionValue ::= {
    ENCODE STRUCTURE {
        extension-type integer-1-encoding,
        extension-data extension-data-encoding
        STRUCTURED WITH structure-encoding
    }
}

extension-data-encoding #OPEN-TYPE ::= {
    ENCODE WITH TLS-Extensions-encodings
    COMPLETED BY PER-BASIC-UNALIGNED
}

sequenceOfTLS-Extension-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE
        SIZE variable-with-determinant
        MULTIPLE OF octet
        DETERMINED BY container
        USING OUTER
    }
}

compressionMethodsValue-encoding{<REFERENCE:length>} #CompressionMethods ::= {
    ENCODE STRUCTURE {
        enumerated-1-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

clientRandom-encoding #ClientRandom ::= {
    ENCODE STRUCTURE {
        gmt-unix-time USE-SET,
        random-bytes USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

ServerHello-encodings #ENCODINGS ::= {
    serverHello-encoding
}

serverHello-encoding #ServerHello ::= {
    ENCODE STRUCTURE {
        server-version protocolVersion-encoding,
        random serverRandom-encoding,
        session-id USE-SET,
        cipher-suite enumerated-2-encoding,

```

```

        compression-method          enumerated-1-encoding,
        server-hello-extension-list  tLS-Extensions-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

serverRandom-encoding      #ServerRandom ::= {
    ENCODE STRUCTURE {
        gmt-unix-time      USE-SET,
        random-bytes       USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

CertificateList-encodings  #ENCODINGS ::= {
    certificateList-encoding
}

certificateList-encoding   #CertificateList ::= {
    ENCODE STRUCTURE {
        certificates        certificates-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

certificates-encoding     #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY certificateListWithLength-encoding
    }
}

certificateListWithLength-encoding{<#Element>}
    #SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length             integer-3-encoding,
        value certificateListValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

certificateListValue-encoding{<REFERENCE:length>}    #Certificates ::= {
    ENCODE STRUCTURE {
        x509Certificate-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
}

x509Certificate-encoding   #OCTETS ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #OpaqueWithLength
            ENCODED BY x509CertificateWithLength
    }
}

x509CertificateWithLength{<#Element>}    #OpaqueWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length             integer-3-encoding,
        element countedX509Certificate-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

countedX509Certificate-encoding{<REFERENCE:length>}    #OCTETS ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE
            SIZE variable-with-determinant
            MULTIPLE OF octet
            DETERMINED BY field-to-be-set
            USING length
        }
    CONTENTS-ENCODING {Directory-encodings}
}

```

```

certificate-encoding #Certificate ::= {ENCODE WITH Directory-encodings}

ServerKeyExchange-encodings #ENCODINGS ::= {
    serverKeyExchange-encoding
}

serverKeyExchange-encoding #ServerKeyExchange ::= {
    ENCODE STRUCTURE {
        rsa [serverKeyExchangeTag-encoding] {
            ENCODE STRUCTURE {
                params          serverRSAParams-encoding,
                signed-params    signature-encoding
            }
            WITH PER-BASIC-UNALIGNED
        },
        diffie-hellman [serverKeyExchangeTag-encoding] {
            ENCODE STRUCTURE {
                params          serverDHParams-encoding,
                signed-params    signature-encoding
            }
            WITH PER-BASIC-UNALIGNED
        }
        STRUCTURED WITH serverKeyExchangeChoice-encoding
    }
}

serverKeyExchangeTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "ServerKeyExchange" AT {0..7} AS tag:any
}

serverKeyExchangeChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "ServerKeyExchange"
}

serverDHParams-encoding #ServerDHParams ::= {
    ENCODE STRUCTURE {
        dh-p integer-2-encoding,
        dh-g integer-2-encoding,
        dh-Ys integer-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

signature-encoding #Signature ::= {
    ENCODE STRUCTURE {
        anonymous [signatureTag-encoding] USE-SET,
        rsa       [signatureTag-encoding] USE-SET,
        dsa       [signatureTag-encoding] USE-SET
        STRUCTURED WITH signatureChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

signatureTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "Signature" AT {0..7} AS tag:any
}

signatureChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "Signature"
}

```

```

serverRSAParams-encoding      #ServerRSAParams ::= {
    ENCODE STRUCTURE {
        rsa-modulus integer-2-encoding,
        rsa-exponent   integer-2-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

CertificateRequest-encodings #ENCODINGS ::= {
    certificateRequest-encoding
}

certificateRequest-encoding #CertificateRequest ::= {
    ENCODE STRUCTURE {
        certificate-types clientCertificateTypes-encoding,
        certificate-authorities distinguishedNames-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

clientCertificateTypes-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #SequenceOfWithLength
        ENCODED BY clientCertificateTypesWithLength-encoding
    }
}

clientCertificateTypesWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-1-encoding,
        value clientCertificateTypesValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

clientCertificateTypesValue-encoding{<REFERENCE:length>}
#ClientCertificateTypes ::= {
    ENCODE STRUCTURE {
        enumerated-1-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

distinguishedNames-encoding #SEQUENCE-OF      ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #SequenceOfWithLength
        ENCODED BY distinguishedNamesWithLength-encoding
    }
}

distinguishedNamesWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length      integer-2-encoding,
        value distinguishedNamesValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

distinguishedNamesValue-encoding{<REFERENCE:length>} #DistinguishedNames ::= {
    ENCODE STRUCTURE {
        distinguishedName-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

distinguishedName-encoding #DistinguishedName ::= {
    ENCODE WITH Directory-encodings
}

```

```

ServerHelloDone-encodings    #ENCODINGS ::= {
    serverHelloDone-encoding
}

serverHelloDone-encoding #ServerHelloDone ::=
    {ENCODE WITH PER-BASIC-UNALIGNED}

CertificateVerify-encodings #ENCODINGS ::= {
    certificateVerify-encoding
}

certificateVerify-encoding #CertificateVerify ::= {
    ENCODE STRUCTURE {
        signature    signature-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

ClientKeyExchange-encodings #ENCODINGS ::= {
    clientKeyExchange-encoding
}

clientKeyExchange-encoding #ClientKeyExchange ::= opaque-2-encoding

Finished-encodings          #ENCODINGS ::= {
    finished-encoding
}

finished-encoding #Finished ::= {ENCODE WITH PER-BASIC-UNALIGNED}

CertificateURL-encodings    #ENCODINGS ::= {
    certificateURL-encoding
}

certificateURL-encoding      #CertificateURL ::= {
    ENCODE STRUCTURE {
        type          enumerated-1-encoding,
        url-and-hash-listURLAndOptionalHashList-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

uRLAndOptionalHashList-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY uRLAndOptionalHashListWithLength-encoding
        }
    }
}

uRLAndOptionalHashListWithLength-encoding{<#Element>}
    #SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length        integer-2-encoding,
        value uRLAndOptionalHashListValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

uRLAndOptionalHashListValue-encoding{<REFERENCE:length>}
    #URLAndOptionalHashList ::= {
    ENCODE STRUCTURE {
        uRLAndOptionalHash-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

uRLAndOptionalHash-encoding #URLAndOptionalHash ::= {
    USE #URLAndOptionalHashWithFlag
    MAPPING FIELDS WITH {
    ENCODE STRUCTURE {
        flag    boolean-encoding,
        url     opaque-2-encoding,
        hash    USE-SET
    }
    }
}

```

```

        OPTIONAL-ENCODING presence-encoding{<flag>}
        STRUCTURED WITH structure-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

#URLAndOptionalHashWithFlag ::= #CONCATENATION {
    flag #BOOLEAN,
    url #Opaque,
    hash #SHA1Hash OPTIONAL-ENCODING #Presence
}

CertificateStatus-encodings #ENCODINGS ::= {
    certificateStatus-encoding
}

certificateStatus-encoding #CertificateStatus ::=
    {ENCODE WITH PER-BASIC-UNALIGNED}

BiometricClientHello-encodings #ENCODINGS ::= {
    biometricClientHello-encoding
}

biometricClientHello-encoding #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #SequenceOfWithLength
        ENCODED BY biometricClientHelloWithLength-encoding
    }
}

biometricClientHelloWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length integer-2-encoding,
        value biometricClientHelloValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

biometricClientHelloValue-encoding{<REFERENCE:length>}
#BiometricClientHello ::= {
    ENCODE STRUCTURE {
        biometricMethod-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
}

biometricMethod-encoding #BiometricMethod ::= {
    ENCODE STRUCTURE {
        biometricType USE-SET,
        biometricFunctionProvider bSP-BFP-Schema-encoding,
        networkAuthenticationModel USE-SET,
        thirdPartyInfo UTF8String-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

UTF8String-encoding #UTF8String ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
        WITH #UTF8StringWithLength
        ENCODED BY UTF8StringWithLength-encoding
    }
}

#UTF8StringWithLength{<#Element>} ::= #CONCATENATION {
    length #INT(0..65535),
    element #Element
}

UTF8StringWithLength-encoding{<#Element>}
#UTF8StringWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {

```

```

        length USE-SET,
        element countedString-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

countedString-encoding{<REFERENCE:length>} #UTF8String ::= {
    TRANSFORMS {{CHAR-TO-BITS AS iso10646 SIZE 2 MULTIPLE OF octet}}
    REPETITION-ENCODING {
        REPETITION-SPACE
        SIZE variable-with-determinant
        MULTIPLE OF octet
        DETERMINED BY field-to-be-set
        USING length
    }
}

bSP-BFP-Schema-encoding #BSP-BFP-Schema ::= {
    ENCODE STRUCTURE {
        bSPSchema [bSP-BFP-SchemaTag-encoding]
        bioAPI-BSP-SCHEMA-encoding,
        bFPSchema [bSP-BFP-SchemaTag-encoding]
        bioAPI-BFP-SCHEMA-encoding
        STRUCTURED WITH bSP-BFP-SchemaChoice-encoding
    }
}

bSP-BFP-SchemaTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "BSP-BFP-Schema" AT {0..7} AS tag:any
}

bSP-BFP-SchemaChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
    DETERMINED BY handle
    HANDLE "BSP-BFP-Schema"
}

bioAPI-BSP-SCHEMA-encoding #BioAPI-BSP-SCHEMA ::= {
    ENCODE WITH Directory-encodings
}

bioAPI-BFP-SCHEMA-encoding #BioAPI-BFP-SCHEMA ::= {
    ENCODE WITH Directory-encodings
}

BiometricServerHello-encodings #ENCODINGS ::= {
    biometricServerHello-encoding
}

biometricServerHello-encoding #BiometricServerHello ::=
    biometricAuthenticationRequest-encoding

biometricAuthenticationRequest-encoding #BiometricAuthenticationRequest ::= {
    USE #BiometricAuthenticationRequestWithFlag
    MAPPING FIELDS WITH {
        ENCODE STRUCTURE {
            biometricMethod biometricMethod-encoding,
            requestFMR USE-SET,
            requestTrialNumber integer-1-encoding,
            requestQuality quality-encoding,
            flag boolean-encoding,
            requestTemplateData xtsmTemplate-encoding
            OPTIONAL-ENCODING presence-encoding{<flag>}
            STRUCTURED WITH structure-encoding
        }
        WITH PER-BASIC-UNALIGNED
    }
}

```

```

quality-encoding #Quality ::= integer-1-encoding

#BiometricAuthenticationRequestWithFlag ::= #CONCATENATION {
    biometricMethod      #BiometricMethod,
    requestFMR           #BioAPI-FMR,
    requestTrialNumber   #INT(1..15),
    requestQuality       #Quality,
    flag                 #BOOLEAN,
    requestTemplateData   #XtsmTemplate OPTIONAL-ENCODING #Presence
}

xtsmTemplate-encoding #XtsmTemplate ::= {
    ENCODE WITH Directory-encodings
}

BiometricVerify-encodings #ENCODINGS ::= {
    biometricVerify-encoding
}

biometricVerify-encoding #BiometricVerify ::= {
    ENCODE STRUCTURE {
        biometricData {
            ENCODE STRUCTURE {
                no-value
                [networkAuthenticationModelTag-encoding]
                null-value-encoding,
                local-model
                [networkAuthenticationModelTag-encoding]
                bDforLocalModel-encoding,
                download-model
                [networkAuthenticationModelTag-encoding]
                bDforDownloadModel-encoding,
                attached-model
                [networkAuthenticationModelTag-encoding]
                bDforAttachedModel-encoding,
                center-model
                [networkAuthenticationModelTag-encoding]
                bDforCenterModel-encoding,
                ref-onttp-for-local-model
                [networkAuthenticationModelTag-encoding]
                bDforRefOnTTPforLocalModel-encoding,
                ref-onttp-for-center-model
                [networkAuthenticationModelTag-encoding]
                bDforRefOnTTPforCenterModel-encoding,
                comparison-outsourcing-by-client-model
                [networkAuthenticationModelTag-encoding]
                bDforCObyClientModel-encoding,
                comparison-outsourcing-by-server-model
                [networkAuthenticationModelTag-encoding]
                bDforCObyServerModel-encoding,
                storage-comparison-outsourcing-by-client-model
                [networkAuthenticationModelTag-encoding]
                bDforSCObyClientModel-encoding,
                storage-comparison-outsourcing-by-server-model
                [networkAuthenticationModelTag-encoding]
                bDforSCObyServerModel-encoding
            }
            STRUCTURED WITH
            networkAuthenticationModelChoice-encoding
        }
        WITH PER-BASIC-UNALIGNED
    },
    digitalSignature signedDataByClient-encoding
}
WITH PER-BASIC-UNALIGNED
}

networkAuthenticationModelTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "NetworkAuthenticationModel" AT {0..7} AS tag:any
}

```

```

networkAuthenticationModelChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
        DETERMINED BY handle
        HANDLE "NetworkAuthenticationModel"
    }

null-value-encoding    #NULL ::= {ENCODE WITH PER-BASIC-UNALIGNED}

signedDatabyClient-encoding    #SignedDatabyClient ::= {
    ENCODE STRUCTURE {
        digital-signature    [signedDatabyClientTag-encoding]
                             signedData-encoding,
        aCBioOnClient        [signedDatabyClientTag-encoding]
                             signedDataACBio-encoding
        STRUCTURED WITH signedDatabyClientChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

signedDatabyClientTag-encoding #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "SignedDatabyClient" AT {0..7} AS tag:any
}

signedDatabyClientChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
        DETERMINED BY handle
        HANDLE "SignedDatabyClient"
    }

signedData-encoding #SignedData ::= {
    ENCODE WITH Directory-encodings
}

signedDataACBio-encoding #SignedDataACBio ::= {
    ENCODE WITH Directory-encodings
}

bDforLocalModel-encoding    #BDforLocalModel ::= {
    ENCODE STRUCTURE {
        biometricClientProcess biometricClientProcess-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

biometricClientProcess-encoding    #BiometricClientProcess ::= {
    ENCODE STRUCTURE {
        bFPSchema    bSP-BFP-Schemas-encoding,
        templateID    templateID-encoding,
        sampleQuality    quality-encoding,
        score            USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

bSP-BFP-Schemas-encoding    #SEQUENCE-OF ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE
            WITH #SequenceOfWithLength
            ENCODED BY bSP-BFP-SchemasWithLength-encoding
        }
    }

bSP-BFP-SchemasWithLength-encoding{<#Element>}
#SequenceOfWithLength{<#Element>} ::= {
    ENCODE STRUCTURE {
        length    integer-4-encoding,
        value bSP-BFP-SchemasValue-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

```

```

bSP-BFP-SchemasValue-encoding{<REFERENCE:length>} #BSP-BFP-Schemas ::= {
    ENCODE STRUCTURE {
        bSP-BFP-Schema-encoding
        STRUCTURED WITH sequenceOfWithLength-encoding{<length>}
    }
    WITH PER-BASIC-UNALIGNED
}

templateID-encoding #TemplateID ::= {
    ENCODE STRUCTURE {
        certificateIssuername-encoding,
        serialNumber          certificateSerialNumber-encoding,
        templateInfo          templateInfo-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

name-encoding      #Name ::= {
    ENCODE WITH Directory-encodings
}

certificateSerialNumber-encoding #CertificateSerialNumber ::=
    integer-4-encoding

templateInfo-encoding #TemplateInfo ::= {
    ENCODE STRUCTURE {
        biometricType          USE-SET,
        creator                uTF8String-encoding,
        createdBFPSSchema bSP-BFP-Schema-encoding,
        templateID            certificateIDInformation-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

certificateIDInformation-encoding #CertificateIDInformation ::=
    certificateSerialNumber-encoding

bDforDownloadModel-encoding #BDforDownloadModel ::= {
    ENCODE STRUCTURE {
        biometricClientProcess biometricClientProcess-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforAttachedModel-encoding      #BDforAttachedModel ::= {
    ENCODE STRUCTURE {
        templateData          xtSMTemplate-encoding,
        sampleData            sampleData-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

sampleData #SampleData ::= opaque-2-encoding

bDforCenterModel-encoding #BDforCenterModel ::= {
    ENCODE STRUCTURE {
        sampleData sampleData-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforRefOnTTPforLocalModel-encoding #BDforRefOnTTPforLocalModel ::= {
    USE #BDforRefOnTTPforLocalModelWithFlag
    MAPPING FIELDS WITH {
        ENCODE STRUCTURE {
            thirdPartyInfo          uTF8String-encoding,
            flag                    boolean-encoding,
            biometric-ttp-process biometricTTPResponse-encoding
            OPTIONAL-ENCODING presence-encoding{<flag>},

```

```

        biometricClientProcess biometricClientProcess-encoding
        STRUCTURED WITH structure-encoding
    }
}

#BDforRefOnTTPforLocalModelWithFlag ::= #CONCATENATION {
    thirdPartyInfo      UTF8String,
    flag                #BOOLEAN,
    biometric-ttp-process #BiometricTTPResponse
        OPTIONAL-ENCODING #Presence,
    biometricClientProcess #BiometricClientProcess
}

bDforRefOnTTPforCenterModel-encoding #BDforRefOnTTPforCenterModel ::= {
    ENCODE STRUCTURE {
        thirdPartyInfo  UTF8String-encoding,
        sampleData      sampleData-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforCObyClientModel-encoding      #BDforCObyClientModel ::= {
    ENCODE STRUCTURE {
        bFPSchemaForClientProcess  bSP-BFP-Schemas-encoding,
        thirdPartyInfo              UTF8String-encoding,
        biometric-ttp-Process        biometricTTPResponse-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforCObyServerModel-encoding      #BDforCObyServerModel ::= {
    ENCODE STRUCTURE {
        sampleData sampleData-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforSCobyClientModel-encoding      #BDforSCobyClientModel ::= {
    ENCODE STRUCTURE {
        bFPSchemaForClientProcess  bSP-BFP-Schemas-encoding,
        thirdPartyInfo              UTF8String-encoding,
        biometric-ttp-Process        biometricTTPResponse-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

bDforSCobyServerModel-encoding      #BDforSCobyServerModel ::= {
    ENCODE STRUCTURE {
        sampleData sampleData-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

BiometricRetryRequest-encodings      #ENCODINGS ::= {
    biometricRetryRequest-encoding
}

biometricRetryRequest-encoding      #BiometricRetryRequest ::= {
    ENCODE STRUCTURE {
        retryRequest      biometricAuthenticationRequest-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

BiometricFinished-encodings      #ENCODINGS ::= {
    biometricFinished-encoding
}

biometricFinished-encoding      #BiometricFinished ::= {
    ENCODE STRUCTURE {
        result      biometricAuthenticationResult-encoding
    }
}

```

```

        WITH PER-BASIC-UNALIGNED
    }

    biometricAuthenticationResult-encoding #BiometricAuthenticationResult ::=
        boolean-encoding

    BiometricTTPRequest-encodings #ENCODINGS ::= {
        biometricTTPRequest-encoding
    }

    biometricTTPRequest-encoding #BiometricTTPRequest ::= {
        ENCODE STRUCTURE {
            storage-type
                [biometricTTPOutsourcingTypeTag-encoding]
                bDforStorageOutsourcing-encoding,
            comparison-type
                [biometricTTPOutsourcingTypeTag-encoding]
                bDforComparisonOutsourcing-encoding,
            storage-comparison-type
                [biometricTTPOutsourcingTypeTag-encoding]
                bDforComparisonOutsourcing-encoding
            STRUCTURED WITH
                biometricTTPOutsourcingTypeChoice-encoding
        }
        WITH PER-BASIC-UNALIGNED
    }

    biometricTTPOutsourcingTypeTag-encoding #TAG ::= {
        ENCODING-SPACE
        SIZE 1
        MULTIPLE OF octet
        EXHIBITS HANDLE "BiometricTTPOutsourcingType" AT {0..7}
        AS tag:any
    }

    biometricTTPOutsourcingTypeChoice-encoding #ALTERNATIVES ::= {
        ALTERNATIVE
            DETERMINED BY handle
            HANDLE "BiometricTTPOutsourcingType"
    }

    bDforStorageOutsourcing-encoding #BDforStorageOutsourcing ::= {
        ENCODE STRUCTURE {
            templateID templateID-encoding
        }
        WITH PER-BASIC-UNALIGNED
    }

    bDforComparisonOutsourcing-encoding #BDforComparisonOutsourcing ::= {
        ENCODE STRUCTURE {
            templateData xtsmTemplate-encoding,
            sampleData sampleData-encoding
        }
        WITH PER-BASIC-UNALIGNED
    }

    BiometricTTPResponse-encodings #ENCODINGS ::= {
        biometricTTPResponse-encoding
    }

    biometricTTPResponse-encoding #BiometricTTPResponse ::= {
        ENCODE STRUCTURE {
            request-body {
                ENCODE STRUCTURE {
                    storage-type
                        [biometricTTPOutsourcingTypeTag-encoding]
                        rBDforStorageOutsourcing-encoding,
                    comparison-type
                        [biometricTTPOutsourcingTypeTag-encoding]
                        rBDforComparisonOutsourcing-encoding,
                    storage-comparison-type
                        [biometricTTPOutsourcingTypeTag-encoding]
                        rBDforComparisonOutsourcing-encoding
                }
                STRUCTURED WITH
            }
        }
    }

```

```

        biometricTTPOutsourcingTypeChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
},
digital-signaturesignedDatabyTTP-encoding
}
WITH PER-BASIC-UNALIGNED
}

signedDatabyTTP-encoding      #SignedDatabyTTP ::= {
    ENCODE STRUCTURE {
        digital-signature      [signedDatabyTTPTag-encoding]
        signedData-encoding,
        aCBioOnTTP              [signedDatabyTTPTag-encoding]
        signedDataACBio-encoding
        STRUCTURED WITH signedDatabyTTPChoice-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

signedDatabyTTPTag-encoding  #TAG ::= {
    ENCODING-SPACE
    SIZE 1
    MULTIPLE OF octet
    EXHIBITS HANDLE "SignedDatabyTTP" AT {0..7} AS tag:any
}

signedDatabyTTPChoice-encoding #ALTERNATIVES ::= {
    ALTERNATIVE
        DETERMINED BY handle
        HANDLE "SignedDatabyTTP"
}

rBDforStorageOutsourcing-encoding #rBDforStorageOutsourcing ::= {
    ENCODE STRUCTURE {
        templateData      xtsmTemplate-encoding
    }
    WITH PER-BASIC-UNALIGNED
}

rBDforComparisonOutsourcing-encoding #rBDforComparisonOutsourcing ::= {
    ENCODE STRUCTURE {
        bFPSchema      bSP-BFP-Schemas-encoding,
        templateID      templateID-encoding,
        sampleQuality    quality-encoding,
        score            USE-SET
    }
    WITH PER-BASIC-UNALIGNED
}

applicationData-encoding      #ApplicationData ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE
        SIZE variable-with-determinant
        DETERMINED BY container
        USING OUTER
    }
}

END

```

V.2 ELM module

```

TSM-ELM{itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0)
        tls-extended-protocol-link(4) version1(1)}

LINK-DEFINITIONS ::=
    BEGIN

```

```

IMPORTS TSM-encodings,
    HelloRequest-encodings, ClientHello-encodings, ServerHello-encodings,
    CertificateList-encodings, ServerKeyExchange-encodings,
    CertificateRequest-encodings,
    ServerHelloDone-encodings, CertificateVerify-encodings,
    ClientKeyExchange-encodings,
    Finished-encodings, CertificateURL-encodings, CertificateStatus-encodings,
    BiometricClientHello-encodings, BiometricServerHello-encodings,
    BiometricVerify-encodings, BiometricRetryRequest-encodings,
    BiometricFinished-encodings, BiometricTTPRequest-encodings,
    BiometricTTPResponse-encodings, Directory-encodings
    FROM TSM-ENCODING{itu-t(0) recommendation(0) x(24) tsm-1(1084)
        modules(0) tls-extended-protocol-encoding(3) version1(1)}
    #TLSPlainText, #TLSCompressed, #TLSCipherText, #TLSStreamCipherText,
    #TLSBlockCipherText, #ChangeCipherSpec, #ContentType, #ApplicationData,
    #Alert, #Handshake, #BiometricClientHello, #BiometricServerHello,
    #BiometricVerify, #BiometricRetryRequest, #BiometricFinished,
    #BiometricTTPRequest, #BiometricTTPResponse, #HelloRequest, #ClientHello,
    #ServerHello, #CertificateList, #ServerKeyExchange,
    #CertificateRequest, #ServerHelloDone, #CertificateVerify,
    #ClientKeyExchange, #Finished, #CertificateURL, #CertificateStatus, #Opaque
    FROM TSM{itu-t(0) recommendation(0) x(24) tsm-1(1084) modules(0)
        tls-extended-protocol(1) version1(1)}
#ID
    FROM UsefulDefinitions{joint-iso-itu-t ds(5) module(1)
        usefulDefinitions(0) 5};

ENCODE    #TLSPlainText, #TLSCompressed, #TLSCipherText, #TLSStreamCipherText,
    #TLSBlockCipherText, #ChangeCipherSpec, #ContentType,
    #ApplicationData, #Alert, #Handshake
    WITH TSM-encodings
    COMPLETED BY PER-BASIC-UNALIGNED

ENCODE    #BiometricClientHello
    WITH BiometricClientHello-encodings

ENCODE    #BiometricServerHello
    WITH BiometricServerHello-encodings

ENCODE    #BiometricVerify
    WITH BiometricVerify-encodings

ENCODE    #BiometricRetryRequest
    WITH BiometricRetryRequest-encodings

ENCODE    #BiometricFinished
    WITH BiometricFinished-encodings

ENCODE    #BiometricTTPRequest
    WITH BiometricTTPRequest-encodings

ENCODE    #BiometricTTPResponse
    WITH BiometricTTPResponse-encodings

ENCODE    #HelloRequest
    WITH HelloRequest-encodings

ENCODE    #ClientHello
    WITH ClientHello-encodings

ENCODE    #ServerHello
    WITH ServerHello-encodings

ENCODE    #CertificateList
    WITH CertificateList-encodings

ENCODE    #ServerKeyExchange
    WITH ServerKeyExchange-encodings

ENCODE    #CertificateRequest
    WITH CertificateRequest-encodings

ENCODE    #ServerHelloDone
    WITH ServerHelloDone-encodings

```

```
ENCODE      #CertificateVerify
            WITH CertificateVerify-encodings

ENCODE      #ClientKeyExchange
            WITH ClientKeyExchange-encodings

ENCODE      #Finished
            WITH Finished-encodings

ENCODE      #CertificateURL
            WITH CertificateURL-encodings

ENCODE      #CertificateStatus
            WITH CertificateStatus-encodings

ENCODE      #Opaque
            WITH PER-BASIC-UNALIGNED

ENCODE      #ID
            WITH Directory-encodings

END
```

Bibliography

- [b-ITU-T X.680] Recommendation ITU-T X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- [b-ITU-T X.681] Recommendation ITU-T X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- [b-ITU-T X.682] Recommendation ITU-T X.682 (2002) | ISO/IEC 8824-3:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- [b-ITU-T X.683] Recommendation ITU-T X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.
- [b-ITU-T X.692] Recommendation ITU-T X.692 (2002) | ISO/IEC 8825-3:2002, *Information technology – ASN.1 encoding rules: Specification of Encoding Control Notation (ECN)*.
- [b-ITU-T X.1083] Recommendation ITU-T X.1083 (2007) | ISO/IEC 24708:2008, *Information technology – Biometrics – BioAPI interworking protocol*.
- [b-ITU-T X.1086] Recommendation ITU-T X.1086 (2008), *Telebiometrics protection procedures – Part 1: A guideline to technical and managerial countermeasures for biometric data security*.
- [b-IETF RFC 2246] IETF RFC 2246 (1999), *The TLS Protocol Version 1.0*.
- [b-IETF RFC 3739] IETF RFC 3739 (2004), *Internet X.509 Public Key Infrastructure: Qualified Certificates Profile*.
- [b-ISO 19092] ISO 19092:2008, *Financial services – Biometrics – Security framework*.
- [b-SC37SD2V8] ISO/IEC JTC 1/SC 37 N 2263 (2007), *ISO/IEC JTC 1/SC 37 Standing Document 2 – Harmonized Biometric Vocabulary Version 8*.
- [b-CCPart1] The Common Criteria Recognition Agreement, *Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model Version 3.1 Revision 1*, September 2006.
<http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf>
- [b-CCPart2] The Common Criteria Recognition Agreement, *Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements Version 3.1 Revision 2*, September 2006.
<http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R2.pdf>
- [b-CCPart3] The Common Criteria Recognition Agreement, *Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1 Revision 2*, September 2006.
<http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R2.pdf>

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems