INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2005-2008

STUDY GROUP 17

TD 2609 Rev.2

English only

Original: English

Question(s): 8/17 Geneva, 16-20 April, 2007

TEMPORARY DOCUMENT

Source: Q.8/17 Associate Rapporteur

Title: Revised draft Recommendation X.tsm-1 (Revision 3)

Attachment 1

Telebiometrics system mechanism - General biometric authentication protocol and system model profiles for telecommunication system

Summary

This draft Recommendation specifies biometric authentication protocols and profiles for telecommunication systems. It defines protocols for biometric authentication of unspecified end-users and service providers on open networks. In the open network, there are various biometric communication devices for the end-users. And there are various security policies for network services on the providers. This draft Recommendation defines nine telebiometrics authentication models for the environment. And it defines the negotiation protocol for the policies and device environments using the models, and it defines the requirements of biometric transportation data for the each model.

Keywords

<Optional>

Introduction

With the rapid and widespread diffusion of the Internet, various network services are in operation. In high value services such as Internet banking, Internet shopping, Internet trading, etc. , illegal trading by obtaining a PIN by means such as phishing are occurring. Therefore, we need high security authentication mechanism, 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 about; “What biometric device is in use at the end-user side?”, “What security level is it at?”, “How is it operated?”

Contact: Yoshiaki Isobe
Hitachi, Ltd. Systems Development Lab.
Japan
Tel: +81-44-959-0538
Fax: +81-44-959-0851
Email: yoshiaki.isobe.en@hitachi.com

Shin Yong Nyuo
Korea Information Security Agency
Korea
Tel: +82-2-405-5237
Fax: +82-2-405-5695
Email: ynshin@kisa.or.kr

- According to each biometric products, the accuracy (False Accept Rate) determined by the threshold parameter is different for different biometric products. So the service provider can NOT claim a uniform accuracy level.
- The accuracy of biometric verification may decline under aging of end-users, because biometrics uses feature of the human body.

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

CONTENTS

1. Scope

Figure 1 shows the scope of this Recommendation for a biometric security mechanism that authenticates a user via a non-face-to-face open network.

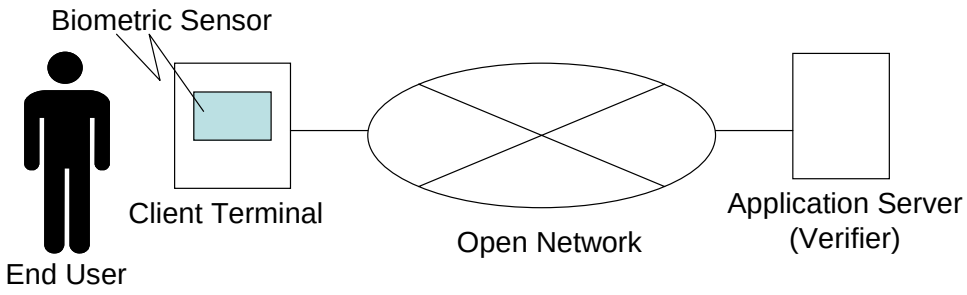


Figure 1 Scope of Recommendation

The meaning of the Open network:

+ Many unspecified verifiers connect to the network and use various types of biometric methods.
  - High value service provider
  - Efficient government service provider
  - Online shopping provider
  - Etc.
+ 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 above providers.

The verifier here is “open” in the following sense. The purpose of the use of biometric authentication is different for each verifier and the risk/value for the verifier is also different for each of them. So each verifier has a different authentication security policy.

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

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.

- ITU-T Recommendation X.509 (1993) ISO/IEC 9594-8: The Directory: Authentication Framework.
- ITU-T Recommendation X.680 (2002) ISO/IEC 8824-1:2002, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.
- IETF RFC2246: The TLS Protocol Ver.1.0, T. Dierks, C. Allen, Network Working Group (January 1999)

- IETF RFC4346: The Transport Layer Security (TLS) Protocol Version 1.1, T.Dierks, E.Rescorla, Network Working Group (April 2006)
- IETF RFC4366: Transport Layer Security (TLS) Extensions, S.Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright, Network Working Group (April 2006)
- ISO/IEC 19784-1 :2006, Biometric Application Programming Interface – Part1 : BioAPI Specification
- ISO/IEC 19785-1 :2006, Common Biometric Exchange Formats Framework – Part1: Data Element Specification
- ISO/IEC 19785-2 :2006, Common Biometric Exchange Formats Framework – Part2: Registration Authority Procedures
- ISO/IEC 19795-1 : 2006, Biometric Performance Testing and Reporting – Part1: Principles and Framework
- ISO/IEC 14888-1:1998 : Information technology -- Security techniques -- Digital signatures with appendix -- Part 1: General

NOTE:

The followings are on the process of international standardizing:

ITU-T draft Recommendation X.bip|ISO/IEC JTC1 SC 37 FCD 24708:BioAPI Interworking Protocol(BIP)

ISO/IEC JTC1 SC 27 CD 24761: Authentication Context for Biometrics (ACBio)

ITU-T draft Recommendation X.tai: Telebiometrics Authentication Infrastructure

3. Definitions

<To be add and delete definitions and to be check sentences>

3.1 Vocabulary definitions within ISO/IEC JTC1 SC 37

3.1.1 biometric (adjective)

pertaining to the field of biometrics.

3.1.2 biometrics (noun)

automated recognition of individuals based on their behavioural and biological characteristics.

3.1.3 biometric template

biometric sample or combination of biometric samples that is suitable for storage as a reference for future comparison.

3.1.4 biometric reference

biometric template that has been stored.

3.1.5 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.

3.1.6 False Reject Rate (FRR)

measure of the probability that a biometric comparison process will incorrectly fail to identify an individual.

3.1.7 comparison (match / matching)

one to one process of comparing a submitted biometric sample against a single biometric reference template and scoring the level of similarity.

3.1.8 threshold

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

3.2 Additional definitions

3.2.1 biometric authentication

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.

3.2.3 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 reference a template for matching

3.2.4 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 reference template

3.2.5 registration

the process in which a person shall prove their identity by presenting credentials to the biometric service provider before being allowed to enrol, and the assignment of an electronic identifier

3.2.6 risk management

coordinated activities to direct and control an organization with regard to risk

3.2.7 score

numerical representation of the degree of similarity between two matched templates

4. Abbreviations

<TBD>

TTP	Trusted Third Party
PKI	Public Key Infrastructure
CA	Certificate Authority
CRL	Certificate Revocation List
BCA	Biometric Certification Authority
TLS	Transport layer security
CBEFF	Common Biometric Exchange File Format

BioAPI	Biometric Application Program Interface
CC	Common Criteria
PP	Protection Profile
ST	Security Target
EAL	Evaluation Assurance Level

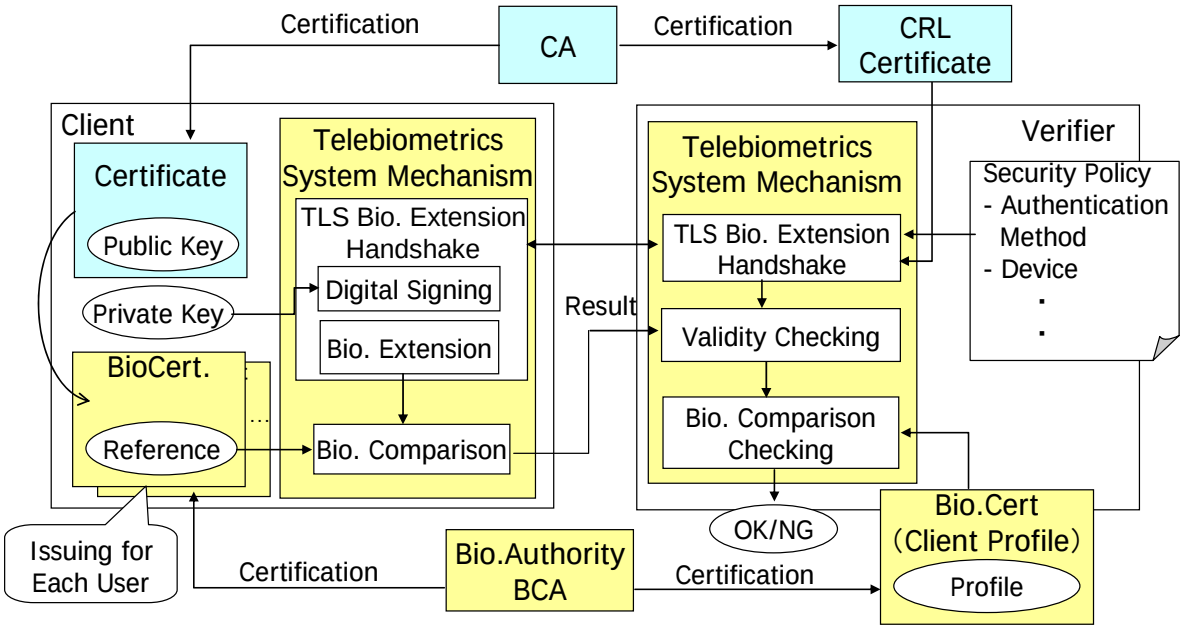
5. Conventions

<Describe any particular notation, style, presentation, etc. used within the Recommendation if any>

6. Prerequisites

This draft recommendation has the following prerequisites.

- A trusted third party (TTP) is required to authenticate a user's public key and to issue a certificate.
- A TTP is required to authenticate user registered biometric reference information and to digitally sign that information using the Biometric Certificate in draft Recommendation X.tai.
- A TTP is required to authenticate both the threshold of each biometric technology and the accuracy evaluation result using PDU report in ACBio. And BCA is required to grant the report by a digital signature using the Biometric Algorithm Certificate in draft Recommendation X.tai.
- 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 in ACBio.
- 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 color belongs to Biometric field, the blue one refers to PKI field.

Figure2 Outline system of this Recommendation

7. Authentication Models

This draft recommendation takes into account of the two perspectives below to divide 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 TTP for Client comparison
Server	Attached	Center	Reference management on TTP for Server comparison
TTP	Comparison Outsourcing by Client	Comparison Outsourcing by Server	Storage & comparison Outsourcing

(1)Local model

Figure 3 shows a local model.

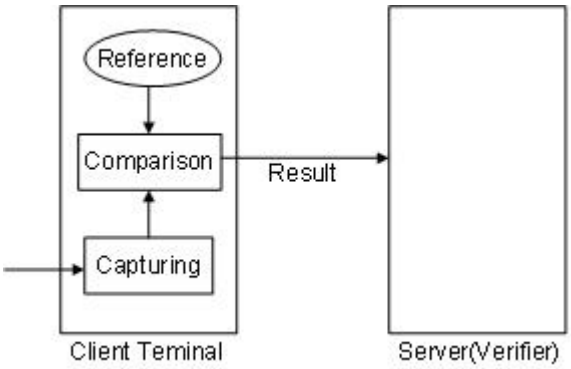


Figure 3 Local model

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 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.

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

(2)Download model

Figure 4 shows a download model.

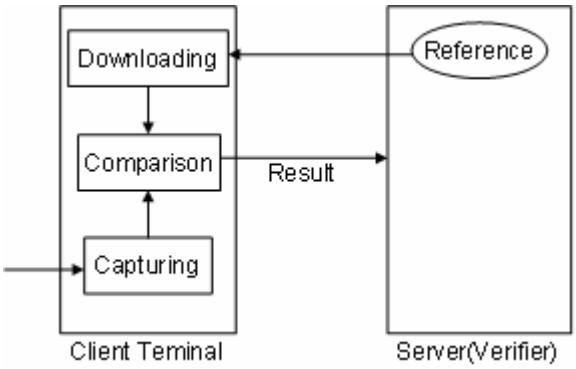


Figure 4 Download model

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 and sends the result to the verifier.

The download model has the same sequence as the local model, excepting the transfer template from the verifier. Template ID information is required, which is the comparison result.

For client terminal, we assume a temporary-use terminal (ex. 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..

(3)Attached model

Figure 5 shows an attached model.

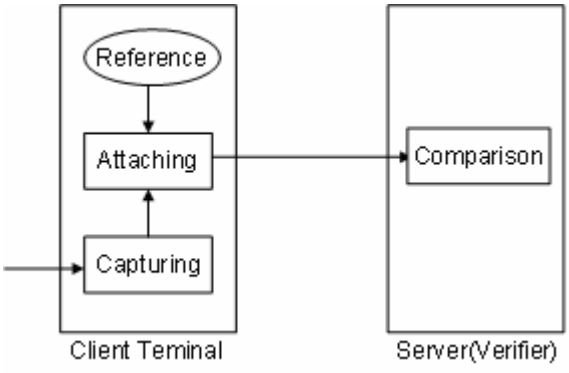


Figure 5 Attached model

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 finally done in 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 (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 needed for all client terminals.

(4)Center model

Figure 6 shows a center model.

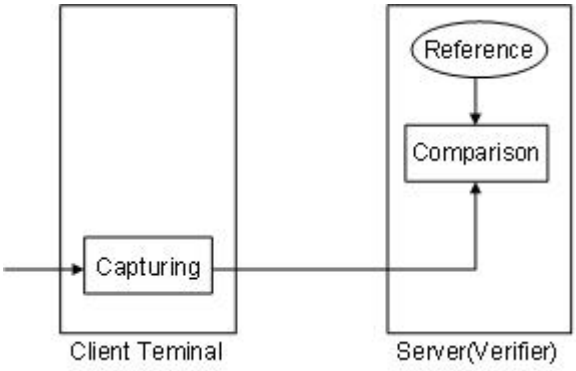


Figure 6 Center model

The verifier requests the sample data which is needed 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 center model, we assume the user-side terminal cannot guarantee sufficient resources (processing power, memory etc.).

The user registers their 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 7 shows a reference management on TTP for local model.

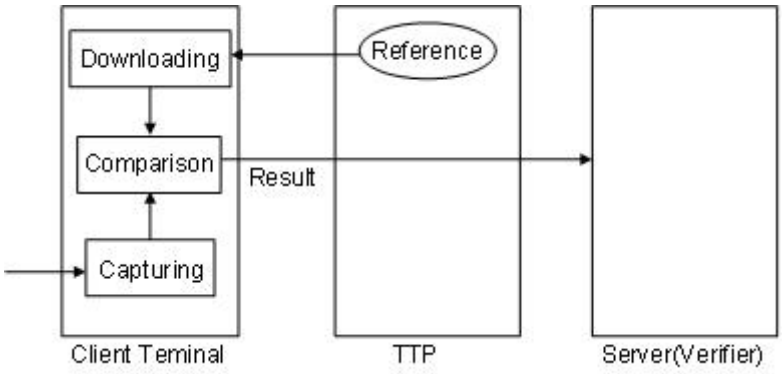


Figure 7 Reference Management on TTP for Local model

The verifier requests verification and the client terminal acquire sample data and request 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 wants to use multiple services on a temporary-use terminal. (Ex, a common credit card authorization terminal)

Template ID information is required which is the comparison result.

A user registers their biometric reference template in advance with a trusted TTP. This structure is same 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 TTP process.

Like the client comparison model, this model uses the RFC3739 qualified certificate.

(6) Reference Management on TTP for Center model

Figure 8 shows a reference management on TTP for center model.

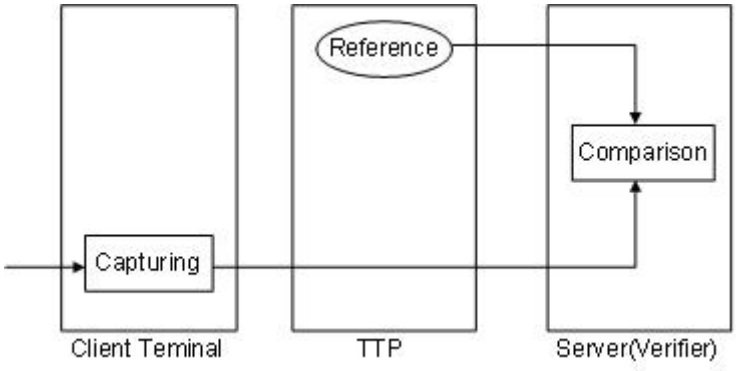


Figure 8 Reference Management on TTP for Center model

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 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 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, a TTP is required to receive permission in advance to present user data and verifier data.

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

Like the server comparison model, this uses the RFC3739 qualified certificate.

(7)Comparison Outsourcing by Client model

Figure 9 shows a comparison outsourcing by client model.

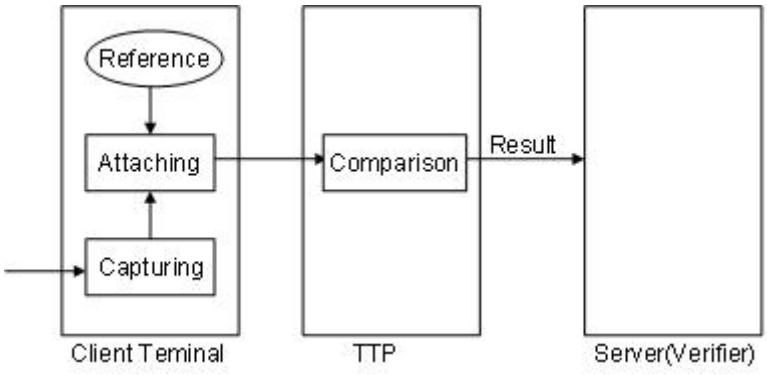


Figure 9 Comparison Outsourcing by Client model

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

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

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 10 shows a comparison outsourcing by server model.

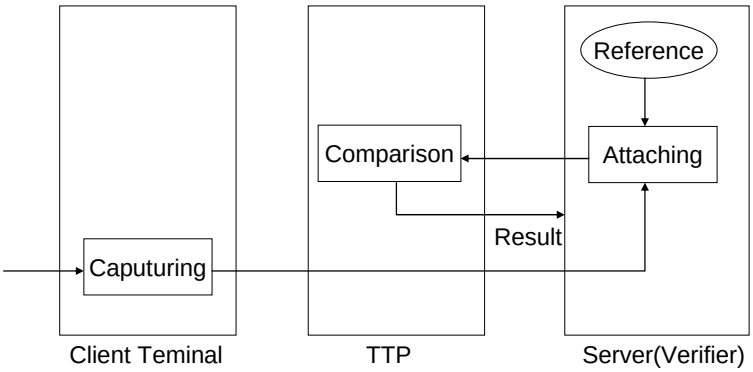


Figure 10 Comparison Outsourcing by Server model

The verifier requests sample data from the client terminal, and verification with user's template to TTP, and then TTP achieve 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 & Comparison Outsourcing model

Figure 11 shows the storage and comparison outsourcing model.

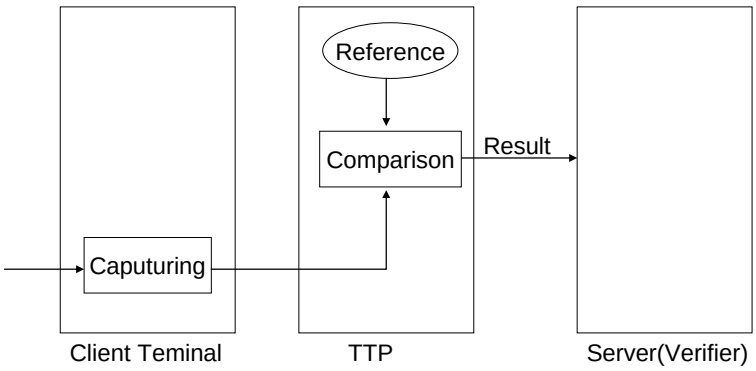


Figure 11 Storage & Comparison Outsourcing model

For this model, we assume that client resources (memory, disk etc.) are insufficient and the server-side has no comparison ability, so 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 verification 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, and 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 gets the captured sample data from the client terminal. Based on this, the verifier requests verification from a TTP and the result is transferred to verifier.

A user registers their biometric reference template in advance with a TTP and outsources comparison processing to a 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

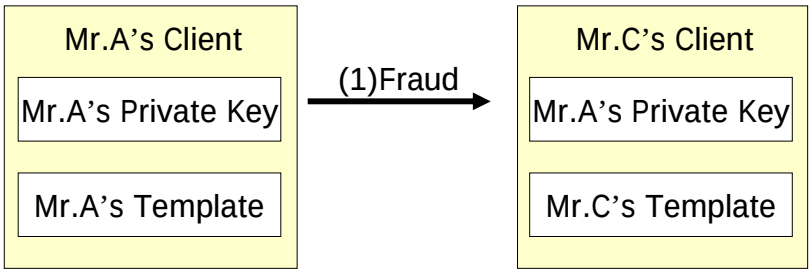
Table 2 shows security threats allowing illegal use by an unauthorized user and possible counter measures for each model.

Table 2 Security threats for each model

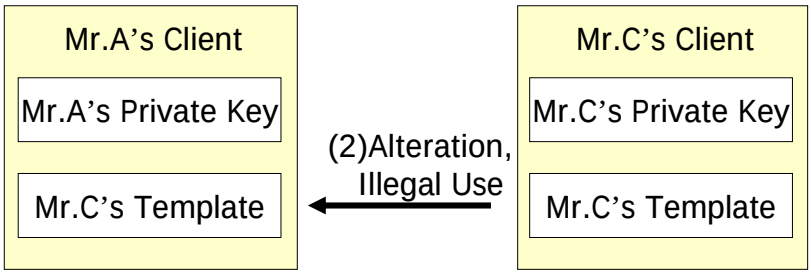
Model	Threats allowing illegal use by an unauthorized user	Possible counter measures
(1)Local	- He/she may use an illegal biometric reference template data - He/she may use an illegal comparison program - He/she may replace to an illegal capture data such as stolen or altered data - He/she may replace an illegal result data	- Reference with signature - Secure client - Client authentication
(2)Download	- He/she may use an illegal comparison program - He/she may replace to an illegal capture data such as stolen or altered data - He/she may replace an illegal result data	- Secure client - Client authentication
(3)Attached	- He/she may use an illegal biometric reference template data - He/she may replace to an illegal capture data such as stolen or altered data	- Reference with signature - Secure client - Client authentication
(4)Center	- He/she may replace to an illegal capture data such as stolen or altered data	- Secure client - Client authentication
(5) Reference Management on TTP for Local	- He/she may use an illegal comparison program - He/she may replace to an illegal capture data such as stolen or altered data - He/she may replace an illegal result data	- Secure client - Client authentication
(6) Reference Management on TTP for Center	- He/she may replace to an illegal capture data such as stolen or altered data	- Secure client - Client authentication
(7)Comparison outsourcing by C.	- He/she may use an illegal biometric reference template data - He/she may use an illegal comparison program - He/she may replace to an illegal capture data such as stolen or altered data	- Reference with signature - Secure client - Client authentication
(8)Comparison outsourcing by S.	- He/she may replace to an illegal capture data such as stolen or altered data	- Secure client - Client authentication
(9)-1 Storage & Comparison outsourcing by C.	- He/she may replace to an illegal capture data such as stolen or altered data - He/she may use an illegal party	- Secure client - Client authentication - Validity check to the TTP of the client
(9)-2 Storage & Comparison outsourcing by S.	- He/she may replace to an 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

And, Figure 12 shows specific threats for cooperation of PKI and a biometric authentication such as the environment of this ITU-T Recommendation X.tsm. For a model that incorporates PKI, there is

a threat of a private key leaking out and that somebody uses it illegally such as case (a). Even without private key leakage such as case (b), the relation between PKI information and biometric template information may not be guaranteed. So, the biometric template information should be required the identifying information of PKI Certificate and validity information for itself. This Recommendation assumes to use Biometric Certificate (BC) of ITU-T Recommendation X.tai as the biometric template.



(a) Case of leakage of private key



(b) Case of illegal use by template alteration

Fig.12 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

Counter measures are listed below.

- Authenticate a server using the PKI
- Authenticate a TTP using the PKI
- Encrypt the session key exchange by TLS protocol.

9. General Requirement

(1) Requirement for Authentication policies in the Verifier

The followings are conditions for a security policy which must meet to offset threats to server-side biometrics.

- A statement of the acceptable biometric technologies (modality, device, algorithm) with ordering by 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 (ISO15408 authentication protection profile (PP), evaluation assurance level (EAL) for PP, and FIPS140-2 certification levels)
- A statement of the acceptable biometric system models (9 models – see Section 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 using TTP models

In addition to the security conditions 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, center model, storage outsourcing by server model)
- Profiles of biometric technology (device, algorithm) which should be approved in advance.

(2) Authentication parameter in Client

There are 2 type parameters for the authentication parameter in the client. They are the profile of client terminal and the privacy settings by client user. The profile of client terminal should be certified by TTP, regarding its performance and security assurance level. This certificate or report is provided BAC in ITU-T Recommendation of X.tai or BPU report in ISO/IEC CD 24761.

- Regarding a profile of client terminal:
  - Profiles (quality) for collectable biometric sample using the client-side
  - Profiles (authentication accuracy) for suitable biometric mode using the client-side
  - Type of biometric reference template held by a client user
  - Client security guarantee (EAL, functional level)
  - 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, which is 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 for a negotiation of the policies in the verifier and the functions and acceptable system model in the client. And This Recommendation provides a transfer protocol for a biometric data for each negotiated system model (see Figure 13).

However, the transfer protocol for the biometric data has variable implementations. This section describes the handshake protocol, and the next section describes the requirements for biometric

transportation data for each system model. The transfer protocol is described in Annex 1 and Annex 2.

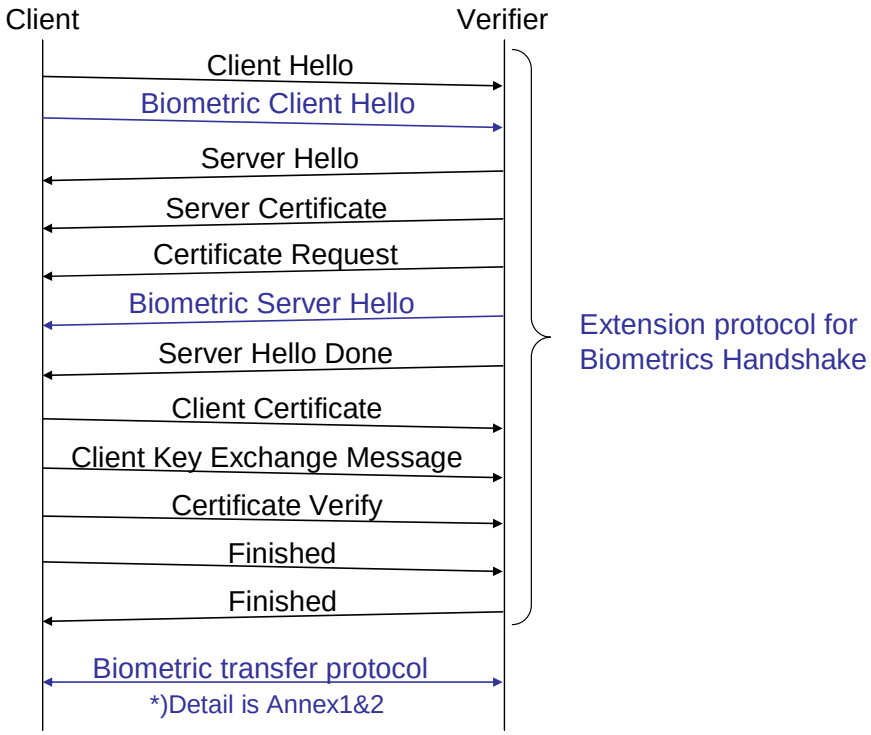


Fig. 13 TLS Extension protocol for Biometrics Handshake

Followings are described the TLS expansion protocol for biometrics handshake.

10.1 Biometrics Handshake Protocol

10.1.1 General Extension Mechanism by RFC4366

This draft Recommendation applies the RFC4366: TLS Extensions for this Biometrics Handshake Protocol. RFC4366 extended the message contents of Client Hello and Server Hello for TLS Extension protocol. These extended messages are described followings:

(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 ::= 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  Extension          -- Same as TLS Extension
}

Extension ::= SEQUENCE {
     Extension-type          ExtensionType,      -- Maintained by IANA
     extension-data          INTEGER(0..2)       -- 0:Normal TLS Extension,
                                                 -- 1:Annex 1,
                                                 -- 2:Annex 2
}
```

Editor Note:

The Extension Type is maintained by IANA. A reservation of the type needs IETF consensus.

(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 ::= 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  Extension          -- Same as TLS Extension
}
```

10.1.2 Extensions for Biometric Handshake Protocol

This draft Recommendation recommends the use of two extended handshake messages, “Biometric Client Hello” and “Biometric Server Hello”. These messages are described in Section 10.1.3 and Section 10.1.4. The extended handshake message structures are 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),
     biometric-client-hello  (31),
     biometric-server-hello  (32)
}(0..255)

Handshake ::= SEQUENCE{
     msg-type  HandshakeType,
     length    UINT24,
     body CHOICE{
          hello-request-body          HelloRequest,
```

```
          client-hello-body            ClientHello,
          server-hello-body            ServerHello,
          certificate-body             Certificate,
          server-key-exchange-body     ServerKeyExchange,
          certificate-request-body     CertificateRequest,
          server-hello-done-body       ServerHelloDone,
          certificate-verify-body      CertificateVerify,
          client-key-exchange-body     ClientKeyExchange,
          finished                     Finished,
          biometric-client-hello-body      BiometricClientHello,
          biometric-server-hello-body      BiometricServerHello
     }
}
```

10.1.3 Biometric Client Hello

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

```
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).

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

Biometric Type and BSP-BFP-Schema are defined in standard specifications of ISO/IEC JTC1 SC37. This Recommendation complies with these reference standards for the followings.

```
BiometricType ::= BIT STRING{
     no-value                     (0), -- no selection --
     multiple-biometric-types     (1),
     face-image                   (2),
     voice                        (3),
     finger-minutiae              (4),
     iris                         (5),
     retina                       (6),
     hand-geometry                (7),
     signature-dynamics           (8),
     keystroke-dynamics           (9),
     lip-movement                 (10),
     thermal-face-image           (11),
     thermal-hand-image           (12),
     gait                         (13),
     body-odour                   (14),
     dna                          (15),
     ear-shape                    (16),
     finer-geometry               (17),
     palm-print                   (18),
     vein-pattern                 (19),
     foot-print                   (20),
```

```
     finger-pattern              (21),
     finger-image                (22),
     signature-or-sign           (23)
} (SIZE (0..24))

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

This Recommendation describes network authentication models in clause 7. Each model is identified by 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))
```

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

If the Biometric Client Hello does not have permissive network authentication models or biometric types, the server sends an alert message with the followings description to the client (see clause 10.3).

```
AlertDescription ::= INTEGER{
     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, request score level of result in biometric comparison (request FMR), request number of trial for biometric comparison process (request trial number), and request sample quality for biometric data.

```
BiometricAuthenticationRequest ::=    SEQUENCE {
     biometricMethod    BiometricMethod,
     requestFMR         BioAPI-FMR,    -- (32bit integer value:requestFMR/231-1)
     requestTrialNumber INTEGER(1..15),
     requestQuality     INTEGER(0..100),
     requestTemplateData     XtsmTemplate OPTIONAL
```

```
                                    -- for download model (no value available)
}
```

10.2 Extension alert protocol

The following describes extended alert protocol of TLS for this draft Recommendation. This draft Recommendation defines alert description number from 115 to 117 in order to avoid a conflict with the TLS Extension definition.

unsupported-extension (110)

This alert is the message of TLS extension (RFC4366). This alert may be returned if the verifier received this extended biometrics handshake protocol unsupported. This message is always fatal.

unacceptable-model (115)

This alert may be returned if the verifier received the unacceptable models only, which 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.

11. Requirements of Biometric transportation stage for each the model

The following section describes requirements of exchange data contents for all the models

11.1 Local model

Figure 3 outlines a local model. Table 3 lists three 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 process	The verifier needs to know "how the biometric process operates in the client" It divides some functions: capturing, pre-processing, and comparing. In BioAPI, the services and functions define the BSPs (or BFPs). There are various types of BSPs that provide all functions or only one function, or pre-processing and comparison functions. If the client expects all types of functions to be provided, it uses various BSPs together. Therefore, this item should be 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 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 least include the reference template ID information. The verifier can then check the revocation template.
3	Sample data quality and Comparison score (similarity) for client process	The verifier needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
4	Security code of client process	The verifier needs the assurance of integrity of client process.

Followings are an example for this message:

```
BDforLocalModel ::= SEQUENCE {
     biometricClientProcess  BiometricClientProcess,
     digitalSignature        SignedData,
     aCforBioOnClient        ACBioInformation   OPTIONAL
                                   -- see ISO/IEC JTC1 SC27 CD24761
}

BiometricClientProcess ::= SEQUENCE {
     bFPSchemaForClientProcess    SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID                   TemplateID,
     sampleQuality                INTEGER(0..100),
     score                        BioAPI-FMR
}

BSP-BFP-Schema ::= CHOICE {
     bSPSchema      BioAPI-BSP-SCHEMA,
     bFPSchema      BioAPI-BFP-SCHEMA       -- import from ISO/IEC 19784:BioAPI
}

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

TemplateInfo ::= SEQUENCE {
     biometricType      BiometricType,
```

```
     creater             UTF8String,
     createdBFPSchema    BSP-BFP-Schema,
     templateID          CertificateIDInformation
                    -- such as CertificateSerialNumber (no value available)
}
```

11.2 Download model

Figure 4 outlines a download model.

It has 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 contents are same as the local model.

```
BDforDownloadModel ::= SEQUENCE {
     biometricClientProcess  BiometricClientProcess,
     digitalSignature        SignedData,
     aCforBioOnClient        ACBioContentInformation OPTIONAL
                                        -- see ISO/IEC JTC1 SC27 CD24761
}
```

Table 4 lists the items required for the download data from the server for the download model. The verifier needs the template ID (=Certificate ID) 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 needs 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,    -- (32bit integer value:requestFMR/231-1)
     requestTrialNumber INTEGER(1..15),
     requestQuality     INTEGER(0..100),
     requestTemplateData     XtsmTemplate OPTIONAL
                             -- for download model (no value available)
}
```

11.3 Attached model

Figure 5 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 needs the reference template for the comparison process in the server.
2	Sample data	The verifier needs sampling data and sampling device information. If the sampling data complies with the BioAPI standard then the data (BIR) has BSP information. The verifier can then check the security level and the quality of sampling data from the sensor device for the comparison process.
3	Security code of client process	The verifier needs the assurance of integrity of client process.

Following is an example for this message:

```
BDforAttachedModel ::= SEQUENCE {
     templateData       XtsmTemplate,
     sampleData         SampleData,             -- BIR: BioAPI defined format --
     digitalSignature   SignedData,
     aCforBioOnClient   ACBioContentInformation OPTIONAL
                                                -- see ISO/IEC JTC1 SC27 CD24761
}
```

11.4 Center model

Figure 6 outlines a center model.

Table 6 Items required for center model

#	Item	Details
1	Sample data	The verifier needs sampling data and sampling device information. If the sampling data complies with the BioAPI standard then the data (BIR) has BSP information. The verifier can then check the security level and the quality of sampling data from the sensor device for the comparison process.
2	Security code of client process	The verifier needs the assurance of integrity of client process.

Following is an example for this message:

```
BDforCenterModel ::= SEQUENCE {
     sampleData          SampleData,         -- BIR: BioAPI defined format --
     digitalSignature    SignedData,
     aCforBioOnClient    ACBioContentInformation OPTIONAL
                                             -- see ISO/IEC JTC1 SC27 CD24761
}
```

11.5 Reference management on TTP for local model

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

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

Following is an example for this message:

```
BDforRefOnTTPforLocalModel ::= SEQUENCE {
     thirdPartyInfo              UTF8String,
     biometricClientProcess      BiometricClientProcess,
     aCforBioOnTTP               ACBioInformation,
     digitalSignaturebyClient    SignedData,
     aCforBioOnClient            ACBioContentInformation OPTIONAL
                                             -- see ISO/IEC JTC1 SC27 CD24761
}
```

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

Table 7 Items required for download data from TTP

#	Item	Details
1	Reference template	The client needs the reference template in TTP for the comparison process on the client terminal. (TTP to Client)
2	Security code of TTP process	The client & the verifier need an assurance of integrity of TTP process.

Following is an example for this message:

```
BiometricTTPProcess ::= SEQUENCE {
     templateData        XtsmTemplate,
     digitalSignature    SignedData,
     aCforBioOnTTP       ACBioContentInformation OPTIONAL
```

```
                                                    -- see ISO/IEC JTC1 SC27 CD24761
}
```

And alert messages should be defined for illegal case on the transmission between TTP such as no response of TTP and no reference template in TTP.

11.6 Reference management on TTP for center model

Figure 8 outlines reference management on TTP for the center model.

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

Following is an example for this message:

```
BDforRefOnTTPforCenterModel ::= SEQUENCE {
     thirdPartyInfo     UTF8String,
     sampleData         SampleData,        -- BIR: BioAPI defined format --
     digitalSignature   SignedData,
     aCforBioOnClient   ACBioContentInformation OPTIONAL
                                                    -- see ISO/IEC JTC1 SC27 CD24761
}
```

Table 8 and 9 lists 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 needs template ID (=Certificate ID) to retrieve a reference DB. (Server to TTP) It should maintain secure transportation between TTP to the server. (The editor will revise the flow-chart for the TLS handshake in biometric authentication cases)

Following is an example for this message:

```
TTPRequestRefOnTTPforCenterModel ::= SEQUENCE {
     templateID         TemplateID
}
```

Table 9 Items required for download data from TTP

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

Following is an example for this message:

```
TTPResponseRefOnTTPforCenterModel ::= SEQUENCE {
     templateData       XtsmTemplate,
     digitalSignature   SignedData,
     aCforBioOnTTP ACBioContentInformation OPTIONAL
```

```
                                                    -- see ISO/IEC JTC1 SC27 CD24761
}
```

And alert messages should be defined for illegal case on the transmission between TTP such as no response of TTP and no reference template in TTP.

11.7 Comparison outsourcing by client model

Figure 9 outlines comparison outsourcing by the client model.

(Client to TTP)

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

#	Items	Details
1	Reference template	TTP needs a reference template in the client terminal for the comparison process in TTP.
2	Sample data	TTP needs sampling data on the client terminal for the comparison process in TTP. The sampling data should comply with the BioAPI standard (BIR). The BIR format has BSP information.

Following is an example for this message:

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

(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 needs the sample data quality for an input data of comparison process and the results of comparison to enable risk management of the verifier’s service.
3	Security code of TTP process	The client & the verifier need an assurance of integrity of TTP process.

Following is an example for this message:

```
TTPResponsCObyClientModel ::= SEQUENCE {
     bFPSchema onTTPProcess   SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID               TemplateID,
     sampleQuality            INTEGER(0..100),
     score                    BioAPI-FMR,
     digitalSignature         SignedData,
     aCforBioTTP              ACBioContentInformation OPTIONAL
                                   -- see ISO/IEC JTC1 SC27 CD24761
```

}

And an alert message should be defined for illegal case on the transmission between TTP such as no response of TTP.

(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" to enable risk management of the verifier's service.
2	Information on BSP(s) for client & TTP process	The verifier needs to know "How the biometric process operates in the client". It divides some functions: capturing, pre-processing, and comparing. In BioAPI, the services and the functions define the BSPs. There are various types of BSPs, providing all functions or only one function, or pre-processing and comparison functions. If the client expects all types of functions to be provided, it uses various BSPs together. Therefore, this item should be able to define various BSPs and operating locations. The verifier can then check whether the security level and performance of BSPs are sufficiently high for the verifier's service.
3	Reference template information for client process	The verifier needs to know "Whose reference template it is using". However, this model is of the privacy protection type for the reference template. Therefore, this item should at least include the reference template ID information. The verifier can then check the revocation template.
4	Sample data quality and Comparison score (similarity) for TTP process	The verifier needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
5	Security code of all process	The verifier needs an assurance of integrity of client and TTP process.

Following is an example for 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                INTEGER(0..100),
     score                        BioAPI-FMR,
     digitalSignaturebyClient     SignedData,
     digitalSignaturebyTTP        SignedData,
     aCforBioOnClient             ACBioContentInformation OPTIONAL,
     aCforBioOnTTP                ACBioContentInformation OPTIONAL
                                                  -- see ISO/IEC JTC1 SC27 CD24761
}
```

11.8 Comparison outsourcing by server model

Figure 10 outlines comparison outsourcing by the server model.

(Client to Server)

Table 13 Items required for comparison outsourcing by server model

#	Item	Details
1	Sample data	The verifier needs the sample data and capturing device information. If the sample data complies with the BioAPI standard then the data (BIR) has BSP information. The verifier can then check the security level and the quality of sampling data from the sensor device for the comparison process on TTP.
2	Security code of client process	The verifier needs an assurance of integrity of client process.

Following is an example for this message:

```
BDforCObyServerModel ::= SEQUENCE {
     sampleData SampleData,       -- BIR: BioAPI defined format --
     digitalSignature   SignedData,
     aCforBiometrics    ACBioContentInformation OPTIONAL
                                              -- see ISO/IEC JTC1 SC27 CD24761
}
```

(Server to TTP)

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

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

Following is an example for this message:

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

(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 has to obtain comparison results from 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 needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
3	Security code of TTP process	The verifier needs an assurance of integrity of TTP process.

Following is an example for this message:

```
TTPResponsebyServer ::= SEQUENCE {
     bFPSchema                SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID               TemplateID,
     sampleQuality            INTEGER(0..100),
     score                    BioAPI-FMR,
     digitalSignaturebyTTP    SignedData,
     aCforBioOnTTP            ACBioContentInformation OPTIONAL
                                        -- see ISO/IEC JTC1 SC27 CD24761
}
```

And alert messages should be defined for illegal case on the transmission between TTP such as no response of TTP or no reference template in TTP.

11.9 Storage and comparison outsourcing model

Figure 11 outlines the storage and comparison outsourcing model.

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

(1) Outsourcing by Client

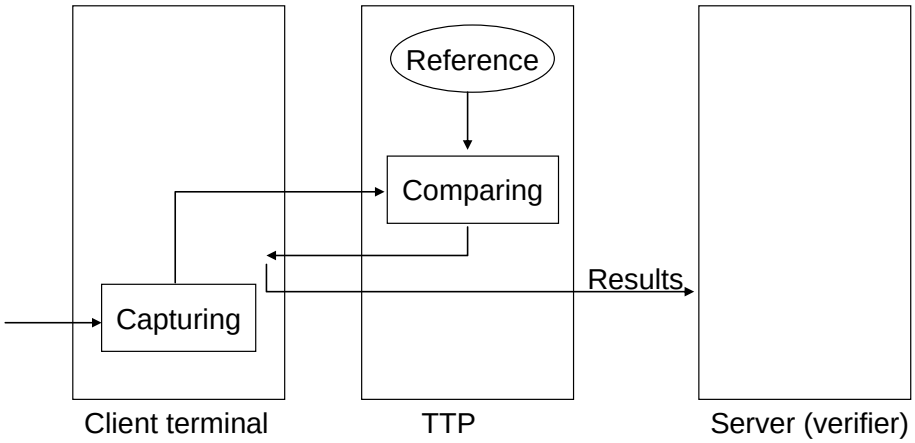


Fig. 13 Storage and comparison outsourcing by client model

(Client to TTP)

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

#	Item	Details
1	Sample data	TTP needs sampling data and capturing device information for comparison process in TTP. The BIR format has the capturing BSP information.

Following is an example for this message:

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

(TTP to Client)

Table 17 Items required for storage and 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 the client. TTP has to 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 needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
3	Security code of TTP process	The client & the verifier need an assurance of integrity of TTP process.

Following is an example for this message:

```
BDforSCObyCModel2 ::= SEQUENCE {
     bFPSchemaForTTPProcess  SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID              TemplateID,
     sampleQuality           INTEGER(0..100),
     score                   BioAPI-FMR,
     digitalSignatureByTTP   SignedData,
     aCforBioOnTTP           ACBioContentInformation OPTIONAL
                                  -- see ISO/IEC JTC1 SC27 CD24761
}
```

And alert messages should be defined for illegal case on the transmission between TTP such as no response of TTP and no reference template in TTP.

(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 process	The verifier needs to know how the biometric process operates in the client. It divides some functions: capturing, pre-processing, and comparing. In BioAPI, the services and the functions define BSPs. There are various types of BSPs, providing all functions or only one function, or pre-processing and comparison functions. If the client expects all types of function to be provided, it uses various BSPs together. Therefore, this item should be able to define various BSPs and operating locations. The verifier can then check whether the security level and performance of all BSPs are sufficiently high for the verifier's service.
2	Reference template information for client process	The verifier needs to know "whose reference template it is using". However, this model is of the privacy protection type for the reference template. Therefore, this item should at least include the reference template ID information. The verifier can then check the revocation template.
3	Sample data quality and Comparison score (similarity) for TTP process	The verifier needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
4	Security code of all process	The verifier needs an assurance of integrity of client and TTP process.

Following is an example for this message:

```
BDforSCObyCModel3 ::= SEQUENCE {
     bFPSchema ForClientProcess   SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     thirdPartyInfo               UTF8String,
     bFPSchemaForTTPProcess       SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID                   TemplateID,
     sampleQuality                INTEGER(0..100),
     score                        BioAPI-FMR,
     digitalSignatureByClient     SignedData,
     digitalSignatureByTTP        SignedData,
     aCforBioOnClient             ACBioContentInformation OPTIONAL,
     aCforBioOnTTP                ACBioContentInformation OPTIONAL
                                                  -- see ISO/IEC JTC1 SC27 CD24761
}
```

(2) Outsourcing by server

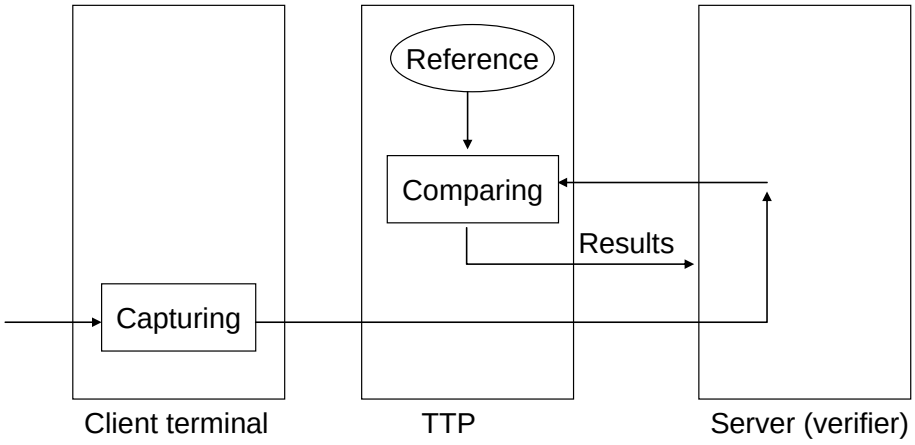


Fig. 14 Storage and comparison outsourcing by server model

(Client to Server)

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

#	Item	Details
1	Sample data	The verifier needs sampling data and capturing device information for comparison process in TTP. The BIR format has the capturing BSP information. The verifier can then check information from the sensor device for the security level and the quality of sampling data for the comparison process.
2	Security code of client process	The verifier needs an assurance of integrity of client process.

Following is an example for this message:

```
BDforSCObySModel ::= SEQUENCE {
     sampleData                    SampleData,    -- BIR: BioAPI defined format --
     digitalSignatureByClient      SignedData,
     aCforBioOnClient              ACBioContentInformation OPTIONAL
                                   -- see ISO/IEC JTC1 SC27 CD24761
}
```

(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 needs template ID (=Certificate ID) to retrieve a reference DB. (Server to TTP) It should maintain secure transportation between TTP to the server.
2	Sample data	TTP needs sampling data and capturing device information for comparison process in TTP. The BIR format has BSP information.

Following is an example for this message:

```
TTPRequestSCObyServerModel ::= SEQUENCE {
```

```
    templateID          TemplateID,
    sampleData          SampleData          -- BIR: BioAPI defined format --
}
```

(TTP to Server)

Table 21 Items required for storage and 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 has to 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 needs the sample data quality for an input data of comparison process and the comparison results to enable risk management of the verifier's service.
3	Security code of TTP process	The verifier needs an assurance of integrity of TTP process.

Following is an example for this message:

```
TTPResponseSCObyServer ::= SEQUENCE {
    bFPSchemaforTTPProcess  SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
    templateID              TemplateID,
    sampleQuality           INTEGER(0..100),
    score                   BioAPI-FMR,
    digitalSignatureByTTP   SignedData,
    aCforBioOnTTP           ACBioContentInformation OPTIONAL
                                                -- see ISO/IEC JTC1 SC27 CD24761
}
```

And alert messages should be defined for illegal case on the transmission between TTP such as no response of TTP and no reference template in TTP.

12. Bibliography

[1] ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.

[2] ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002, Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.

[3] ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.

[4] IETF RFC3739: Internet X.509 Public Key Infrastructure Qualified Certificates Profile, S. Santesson, M. Nystrom, T. Polk, Network Working Group

[5] ISO 19092-1 :2006 : Financial services – Biometrics – Part1: Security framework

[6] The Common Criteria Project Sponsoring Organizations:" Common Criteria for Information Technology Security Evaluation Part1:Introduction and general model Version 2.1"," Common Criteria for Information Technology Security Evaluation Part2:Security functional requirements Version 2.1", "Common Criteria for Information Technology Security Evaluation Part3:Security assurance requirements Version 2.1", Aug.1999

Annex 1 Telebiometrics System Mechanism definitions by TLS extension

This annex describes the all protocol for Telebiometrics system mechanism using the TLS Extension (RFC4366).

This draft Recommendation applies the RFC4366: “TLS Extensions for this Biometrics Handshake Protocol and Biometrics Transfer Protocol”. The Biometrics Handshake Protocol is defined in Section 10. This annex further extended the handshake protocol.

A1.1 Extensions for Biometric Transfer Protocol

This annex recommends the use of five extended handshake messages, “Biometric Verify”, “Biometric Retry Request”, “Biometric Finished”, “Biometric TTP Request” and “Biometric TTP Response”. These extended message flows shows Figure A1.1-3. Figure A1 shows a simple network mechanism for a client and server. Figure A2 shows outsourcing to TTP by a client mechanism. Figure A3 shows outsourcing to TTP by a server mechanism.

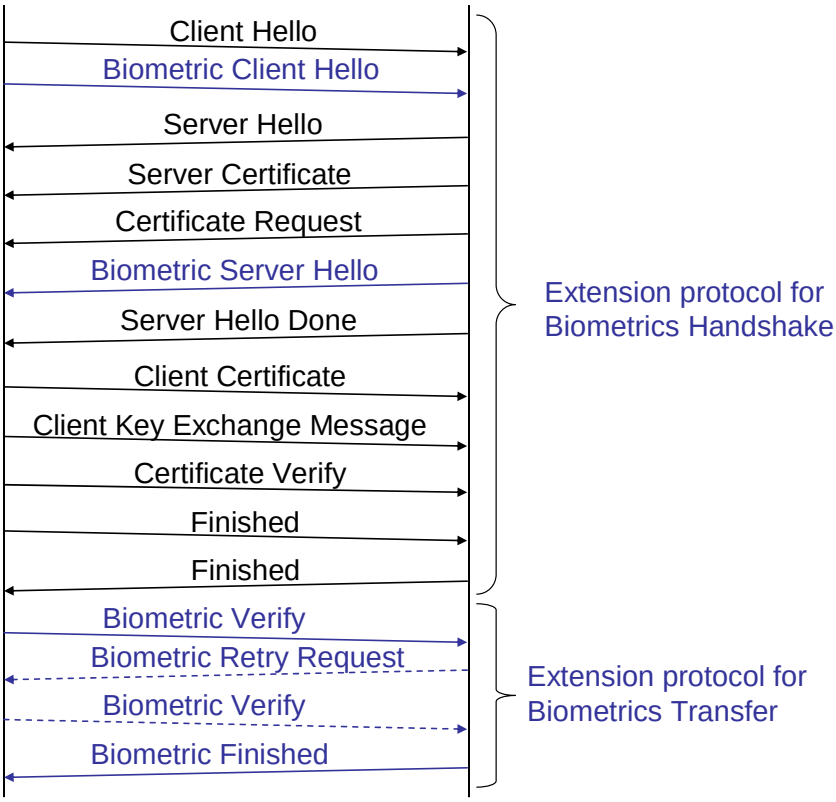


Figure A1.1 Telebiometrics system mechanism by TLS Extensions

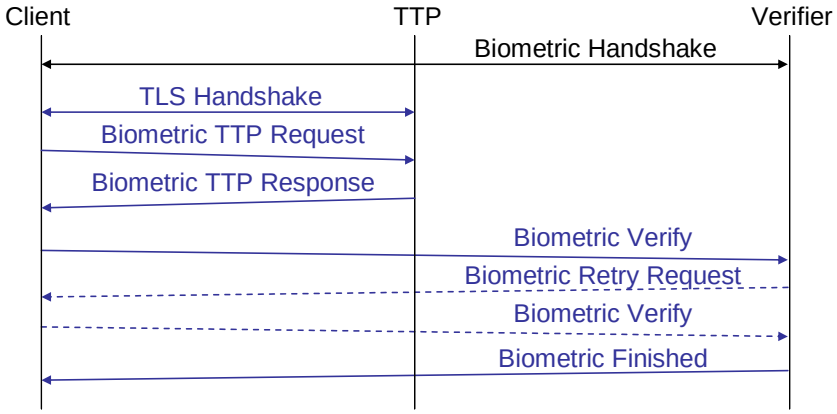


Figure A1.2 Biometrics Extension Protocol for outsourcing to TTP by Client

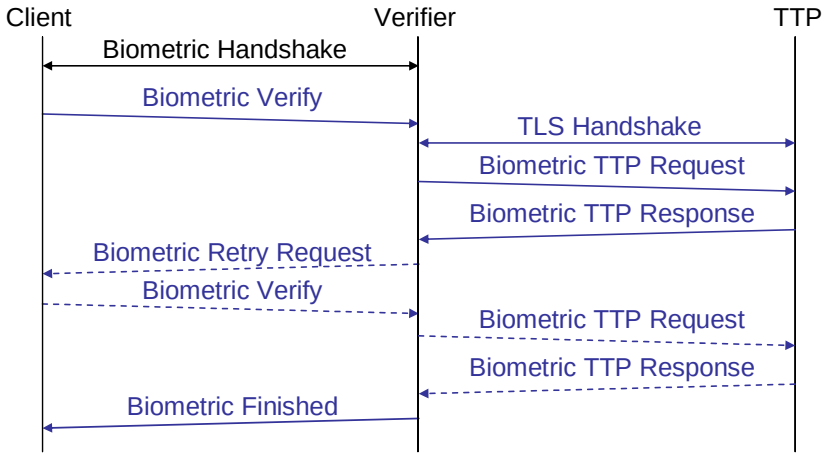


Figure A1.3 Biometrics Extension Protocol for outsourcing to TTP by Verifier

These messages are described in Sections A1.2, A1.3, A1.4, A1.5, and A1.6. 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     (31),
     biometrics-server-hello     (32),
     biometrics-verify           (33),
     biometrics-retry-request    (34),
     biometrics-finished         (35),
     biometrics-ttp-request      (36),
     biometrics-ttp-response     (37)
```

```
}(0..255)

Handshake ::= SEQUENCE{
     msg-type  HandshakeType,
     length    UINT24,
     body ::= CHOICE{
          hello-request-body               HelloRequest,
          client-hello-body                ClientHello,
          server-hello-body                ServerHello,
          certificate-body                 Certificate,
          server-key-exchange-body         ServerKeyExchange,
          certificate-request-body         CertificateRequest,
          server-hello-done-body           ServerHelloDone,
          certificate-verify-body          CertificateVerify,
          client-key-exchange-body         ClientKeyExchange,
          finished                         Finished,
          biometrics-client-hello-body     BiometricsClientHello,
          biometrics-server-hello-body     BiometricsServerHello,
          biometrics-verify-body           BiometricsVerify,
          biometrics-retry-request-body    BiometricsRetryRequest,
          biometrics-finished-body         BiometricsFinished,
          biometrics-ttp-request-body      BiometricsTTPRequest,
          biometrics-ttp-response-body     BiometricsTTPResponse
     }
}
```

A1.2 Biometrics Verify

Data contents corresponding to the selected system model is transmitted to the verifier as follows.

```
BiometricsVerify ::= SEQUENCE    {
     networkAuthenticationModel  NetworkAuthenticationModel,
     biometricData CHOICE  {
          local-model                                  BDforLocalModel,
          download-model                               BDforDownloadModel,
          attached-model                               BDforAttachedModel,
          center-model                                 BDforCenterModel,
          ref-onttp-for-local-model                    BDforRefOnTTPforLocalModel,
          ref-onttp-for-center-model                   BDforRefOnTTPforCenterModel,
          comparison-outsourcing-by-client-model       BDforCObyClientModel,
          comparison-outsourcing-by-server-model       BDforCObyServerModel,
          storage-comparison-outsourcing-by-client-model BDforSCObyClientModel,
          storage-comparison-outsourcing-by-server-model BDforSCObyServerModel
    },
     digitalSignature   SignedDatabyClient
}

SignedDatabyClient ::= CHOICE {
     digital-signature  SignedData,        --import from X9.84-CMS
     aCBioOnClient      ACBioContentInformation
                                           --import from ISO/IEC SC27 CD24761
}
```

The followings present Biometric Data of “Biometric Verify” for the local model.

```
BDforLocalModel ::= SEQUENCE {
     biometricClientProcess BiometricClientProcess
}
```

```
BiometricClientProcess ::= SEQUENCE {
     bFPSchema          SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID         TemplateID,
     sampleQuality      INTEGER(0..100),
     score              BioAPI-FMR
}

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

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

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

The following presents Biometric Data of “Biometric Verify” for the download model. The contents are the same as those in 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 center model is given as follow.

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

The following presents Biometric Data of “Biometric Verify” for the reference management on TTP for client comparison model. The client can omit Biometrics TTP Response with user’s template for her/his privacy protection.

```
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 given as follows.

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

The Biometric Data of "Biometric Verify" for the comparison outsourcing by client model is given as follow.

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

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

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

The Biometric Data of "Biometric Verify" for the storage and comparison outsourcing by client model are given 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 the storage and comparison outsourcing by server model is given as follows.

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

A1.3 Biometrics Retry Request

In case 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 approved biometric system are selected from a list by the client and are sent to the client.

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

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

And if the “Biometric Client Verify” data is based on an illegal template, a revoked template, an expired template or has an unknown BCA’s digital signature or an unacceptable biometric comparison score, the server sends to the client alert message with the followings description (see clause A1.7).

```
AlertDescription ::= INTEGER{
     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 response using “Biometric Verify” again.

However, the data content in the previous ”Biometric Verify” and the followed “Biometric Verify” for Biometric Retry Request are different new one. And the content contains information that guarantees correspondence between a user template and a user certificate. When a template is sent to a server, a format that guarantees correspondence (refer to Section 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.

A1.4 Finished Biometrics

A biometric-finished message is given as follows.

```
BiometricFinished ::= SEQUENCE {
     result    BiometricAuthenticationResult
}

BiometricAuthenticationResult ::= BOOLEAN
```

A1.5 Biometrics TTP request

This annex defines an outsourcing type for using TTP of the network authentication models. The definition is as following:

```
BiometricTTPOutsourcingType ::= BIT STRING{
     storage-type                (1),
     comparison-type             (2),
     storage-comparison-type     (3)
}
```

Table A1.1 lists the relationship between TTP's authentication models and outsourcing type.

Table A1.1 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 center	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

Biometric TTP Request message is given as follows.

```
BiometricsTTPRequest ::= SEQUENCE {
     outsourcing-type   BiometricTTPOutsourcingType,
     request-body CHOICE     {
          storage-type       BDforStorageOutsourcing,
          comparison-type    BDforComparisonOutsourcing,
          storage-comparison-type      BDforSCoutsourcing
     }
}

BDforStorageOutsourcing ::= SEQUENCE {
     templateID     TemplateID
}

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

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

A1.6 Biometrics TTP response

Biometric TTP response message is given as follows.

```
BiometricsTTPResponse ::= SEQUENCE {
     outsourcing-type   BiometricTTPOutsourcingType,
     request-body CHOICE     {
          storage-type            RBDforStorageOutsourcing,
          comparison-type         RBDforComparisonOutsourcing,
          storage-comparison-type RBDforComparisonOutsourcing
     },
     digital-signature  SignedDatabyTTP
}
```

```
RBDforStorageOutsourcing ::= SEQUENCE {
     templateData    XtsmTemplate
}

RBDforComparisonOutsourcing ::= SEQUENCE {
     bFPSchema           SEQUENCE SIZE(1..MAX) OF BSP-BFP-Schema,
     templateID          TemplateID,
     sampleQuality       INTEGER(0..100),
     score               BioAPI-FMR
}

SignedDatabyTTP ::= CHOICE {
     digital-signature   SignedData,         --import from X9.84-CMS
     aCBioOnTTP          ACBioContentInformation
                                             --import from ISO/IEC SC27 CD24761
}
```

A1.7 Extension alert protocol

The following describes extended alert protocol of TLS for this draft Recommendation, which defines alert description number from 115 to 122 in order to avoid a conflict with the TLS Extension definition.

unsupported-extension (110)

This alert is a message of TLS Extension (RFC4366) and may be returned if the verifier received an unsupported extended biometrics handshake protocol. This message is always fatal.

unacceptable-model (115)

This alert may be returned if the verifier received unacceptable models only, which 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.

unsuported-biometrics (117)

In regards to the server comparison models, the server received 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 the biometric comparison. This message is always fatal. The falsification means that certificate has been detected to have a non-decodable syntax, incorrect identifier, etc.

template-revoked (119)

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

template-expired (120)

The verifier found the biometric template used for the biometric comparison was expired. This message is always fatal.

unknown-bra (121)

This alert may be returned if the verifier could not find a receivable certificate of BRA, or the verifier received a biometric template published by the BRA that is different from the one 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 the 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 to the server the alert response. This message is always fatal alert.

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 to the server the alert response. This message is always fatal alert.

Annex 2 Implementation example of biometric transfer protocol using BIP

This annex describes an implementation example of the biometric transfer protocol using BIP (BioAPI Interworking Protocol).

If an authentication system is configured with BIP, communicators should pre-define its roles, i.e., master or slave. The master requests BioAPI commands and the slave returns responses to these requests. In this Recommendation, the verifier is the master on BIP, because the verifier always leads the client for the biometric process.

There are three tasks and solutions to applying BIP to TSM as follows:

(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.

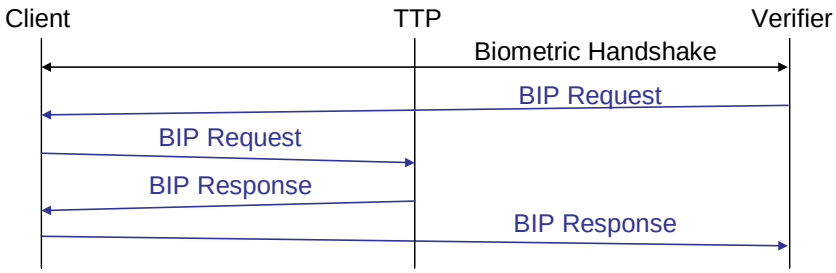


Figure A2.1 Implementation sample by BIP for TTP outsourcing by client

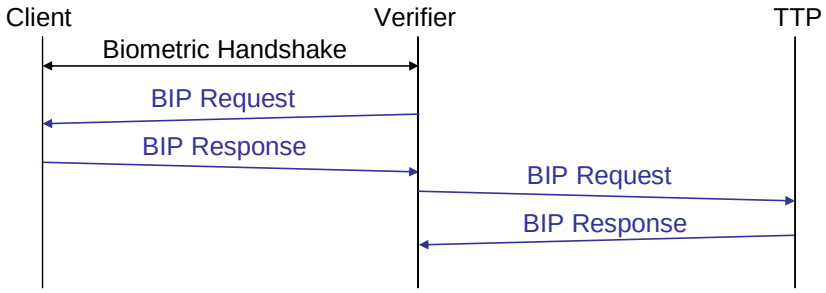


Figure A2.2 Implementation sample by BIP for TTP outsourcing by Verifier

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

Accordingly, they apply multiple BIP commands.

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

It should have validity information such as MAC or digital signatures and it should have the authority to certify their validity. If it has validity information such as ACBio for BioAPI security amendment in the future, this Recommendation suggests applying the security amendment to the validity information.

The following describes the implementations and tasks for each TSM model using BIP. These implementations only describe biometric functions; however, a real implementation system needs preparation functions such as load and attach, and terminate functions such as dettach, for BIP operation.

A2.1 Local model

Figure A2.1 outlines an implementation sample for the local model using a BIP message.

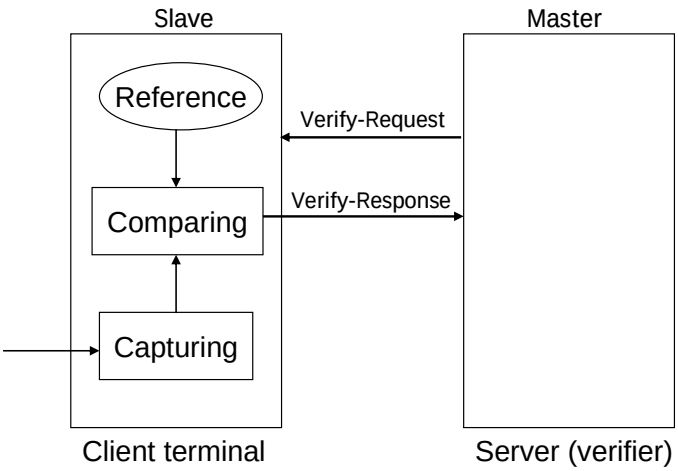


Figure A2.3 Local model using BIP

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 execute a comparison process with the addressed reference template using the PKI certificate by means of the TLS handshake process in advance. However, the BIP's Verify-Request message cannot address the BIR that correlates to the PKI certificate. The message can only be selected 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 draft Recommendation X.tsm. Template ID information and the profile for the client's biometric process should be added. The profile may be useful for ACBio on ISO/IEC JTC1 SC27.

A2.2 Download model

Figure A2.2 outlines an implementation sample for the download model using a BIP message.

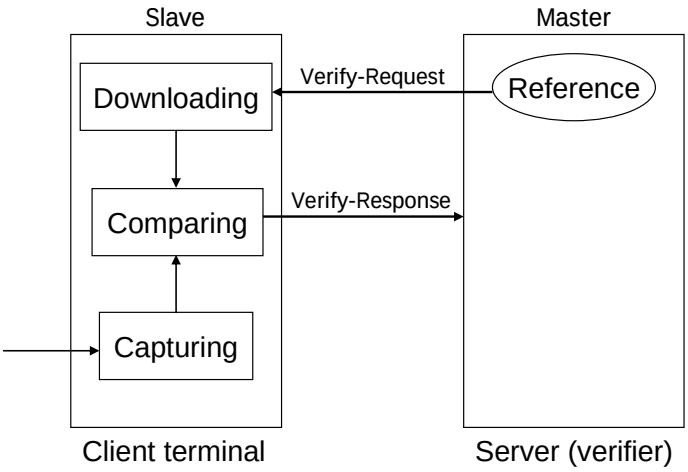


Figure A2.4 Download model

Here, the verifier sets the reference template on the server to the Verify-Request message and sends 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.

A2.3 Attached model

Figure A2.3 outlines an implementation sample for the attached model using a BIP message.

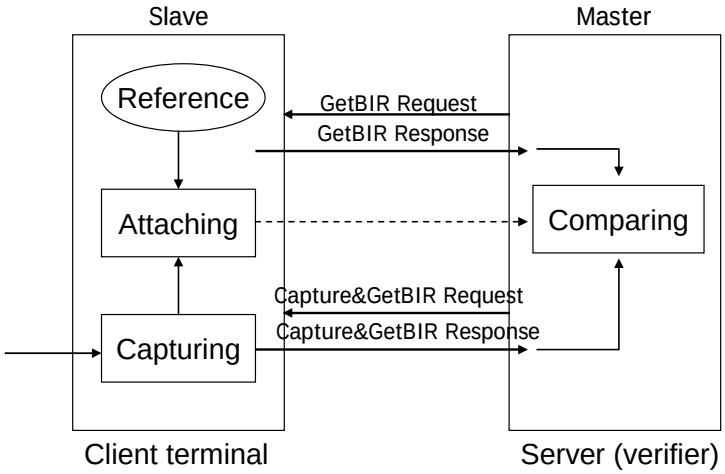


Figure A2.5 Attached model

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

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

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

However, the verifier could not 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.

A2.4 Center model

Figure A2.4 outlines an implementation sample for the center model using a BIP message.

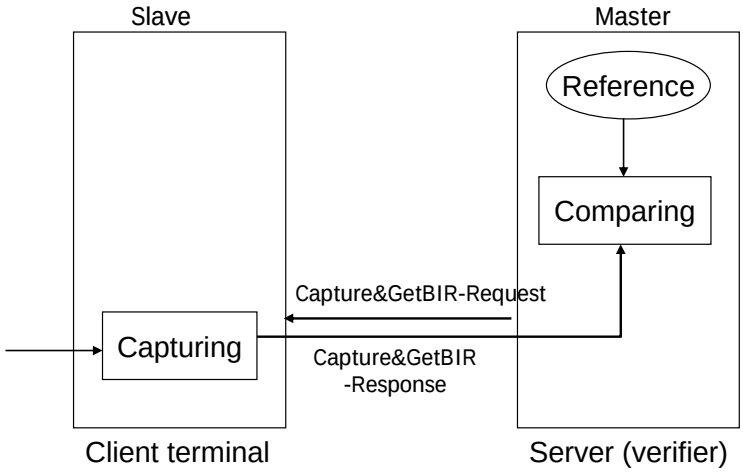


Figure A2.6 Center model

As explained by the figure, the verifier can acquire sampling data using the capture command and the GetBIRFromHandle command.

A2.5 Comparison outsourcing by client model

Figure A2.5 outlines an implementation sample for comparison outsourcing by the client model using a BIP message.

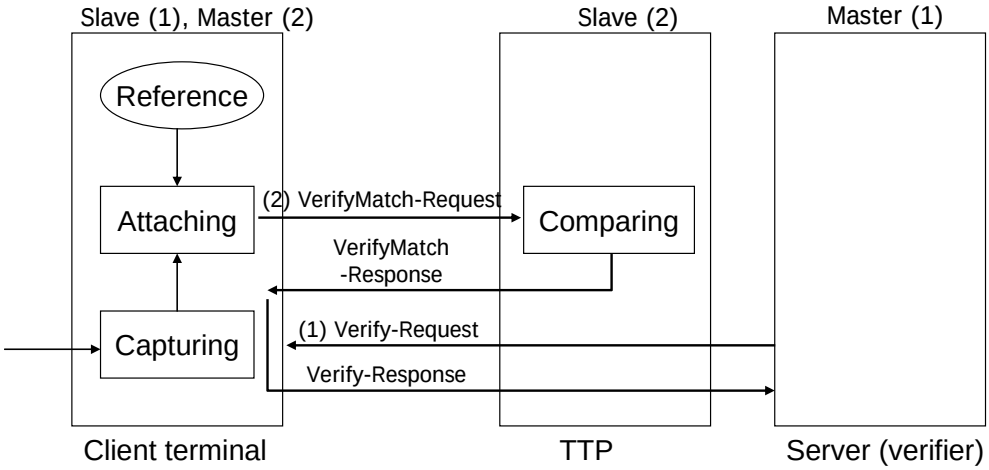


Figure 2.7 Comparison outsourcing by client model

This model uses BIP hierarchically. First, the verifier sends biometric verification request to the client terminal using a Verify-Request message. Second, the client terminal operates by capturing

sample data from the client. Third, the client terminal sends a comparison process request to the TTP using a VerifyMatch-Request message with the sampling data and the reference template. Fourth, the TTP is the slave of the client, which operates the comparison process 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.

A2.6 Reference management on TTP for local model

Figure A2.6 outlines an implementation sample for the model of the reference template managing on TTP for local comparison using a BIP message.

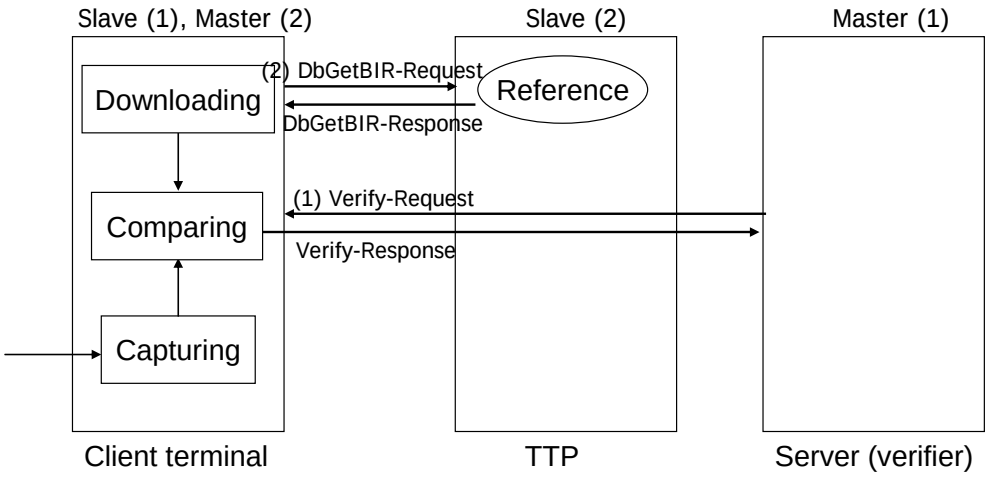


Figure A2.8 Reference management on TTP for local model

This model also uses BIP hierarchically. First, the verifier sends biometric verification request to the client terminal using a Verify-Request message. Second, the client terminal operates by capturing sample data from the client. Third, the client terminal sends a dbGetBIR request to TTP to obtain the addressed reference template on TTP. Finally, the client terminal operates a comparison

process using the sample data and the template, and returns 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 might require one of following selections of PKI certificate information as a key value such as 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
        }
```

A2.7 Reference management on TTP for center model

Figure A2.7 outlines an implementation sample for the model of the reference template managing on TTP for the server comparison using a BIP message.

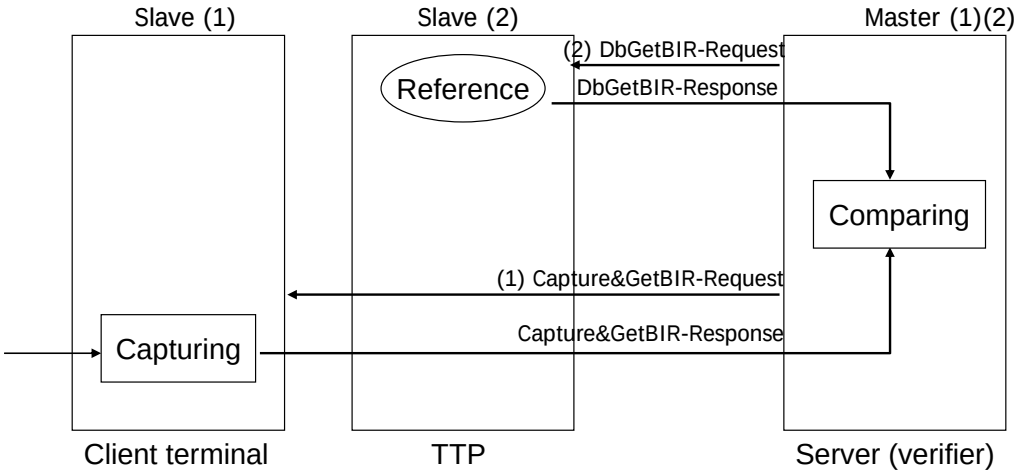


Figure A2.9 Reference management on TTP for center model

As explained by the figure, the verifier first requests the Capture command and the GetBIRFromHandle command to acquire the client biometric sample. Second, the verifier requests the dbGetBIR command to obtain the addressed reference template on TTP.

The retrieval for the reference template from TTP is the same as for reference management on TTP for the client model in Subsection A2.6. Data is sampled from the client terminal in the same way as in the center model in Subsection A2.4.

A2.8 Comparison outsourcing by server model

Figure A2.8 outlines an implementation sample for the model of comparison outsourcing by the server using a BIP message.

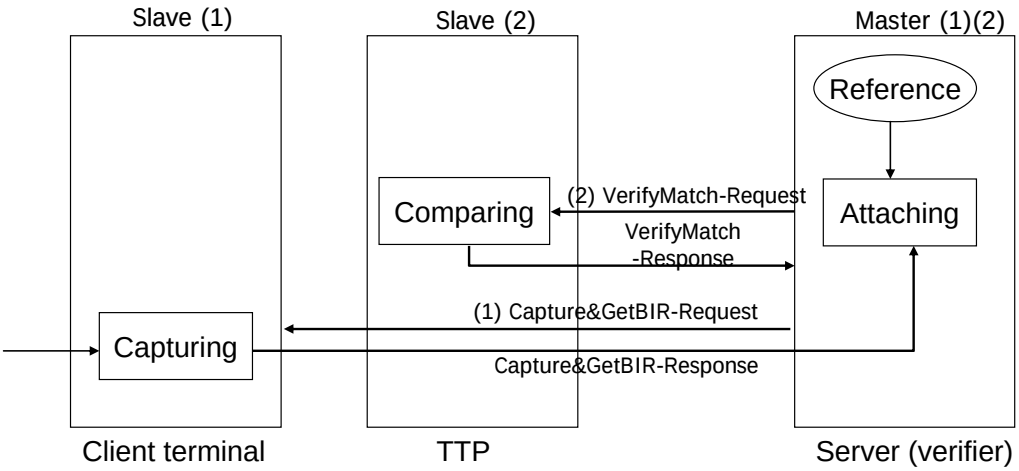


Figure A2.10 Comparison outsourcing by server model

As explained by the figure, the verifier first sends the Capture-Request and the GetBIRFromHandle-Request to acquire the client biometric sample. Second, 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 for the center model in Subsection A2.4. Comparison outsourcing to TTP is the same as for comparison outsourcing by the client model in Subsection A2.5.

A2.9 Storage and comparison outsourcing model

(1) Outsourcing by client model

Figure A2.9 shows an implementation sample for the model of the comparison and reference template managing outsourcing by the client using a BIP message.

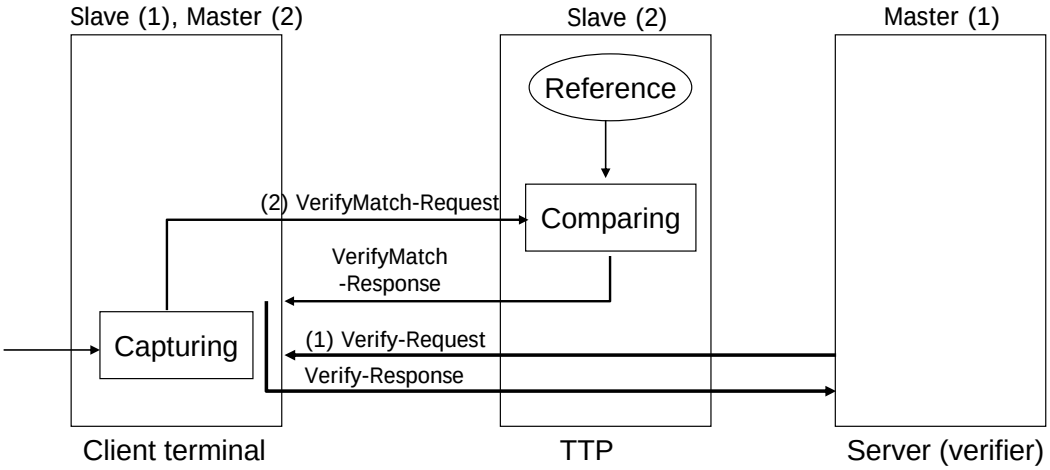


Figure A2.11 Storage and comparison outsourcing by client model

This model also uses BIP hierarchically. The verifier first sends biometric verification request to the client terminal using a Verify-Request message. Second, the client terminal operates by capturing the sample data from the client. Third, the client terminal sends a comparison process request to the TTP using a VerifyMatch-Request message with the sampling data. Fourth, 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. Fifth, 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 clients 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 Subsection A2.5.

(2) Outsourcing by server model

Figure A2.10 outlines an implementation sample for the model of the comparison and reference template managing outsourcing by the server using a BIP message.

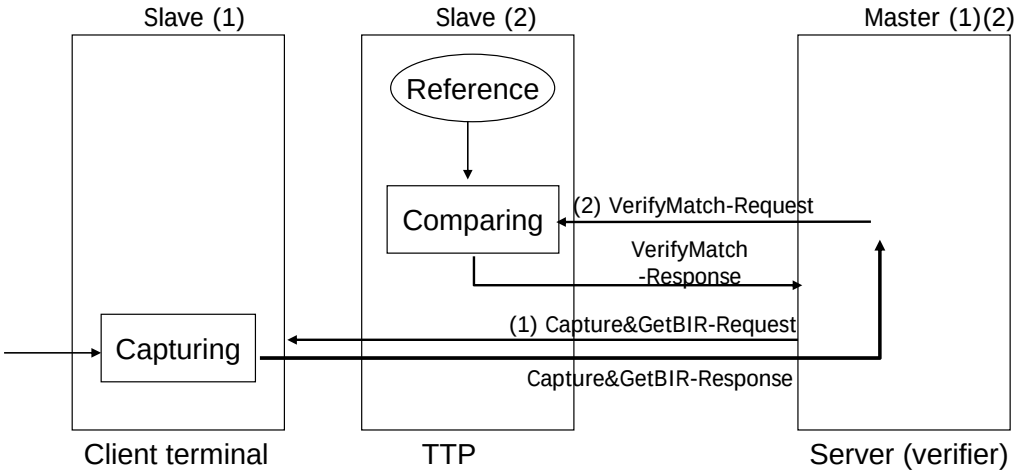


Figure A2.12 Storage and comparison outsourcing by server model

As explained by the figure, the verifier first requests the capture command and the GetBIRFromHandle command to acquire the client biometric sample. Second, 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.

Annex 3 ASN.1 definitions for protocol of TSM based on Annex 1

```
TSM DEFINITIONS AUTOMATIC TAGS ::=
BEGIN


IMPORTS
        BioAPI-BFP-SCHEMA,BioAPI-BSP-SCHEMA,BioAPI-FMR,BioAPI-BIR-BIOMETRIC-TYPE
        FROM BIP
--        SampleData
--        FROM BioAPI-BIR
        BiometricDeviceCertificate
        FROM TAI

AlertLevel,ProtocolVersion,Random,SessionID,CipherSuite,CompressionMethod,UINT24
,
        HelloRequest,ServerKeyExchange,ClientKeyExchange,CertificateRequest,
        CertificateVerify,Finished,ServerHelloDone,CertificateURL,
     CertificateStatus,HelloRequest,ClientHello,ServerHello,Certificate,
     ServerKeyExchange,CertificateRequest,ServerHelloDone,CertificateVerify,
     ClientKeyExchange,Finished,EXTENSION,TLS-ExtensionValues
        FROM TLS
        SignedData
        FROM X9-84-CMS
        ACBioContentInformation
        FROM ACBio
        Name
        FROM InformationFramework
                {joint-iso-itu-t ds(5) module(1)
                informationFramework(1) 5}
        CertificateSerialNumber
        FROM AuthenticationFramework
                {joint-iso-itu-t ds(5) module(1)
                 authenticationFramework(7) 5};

SampleData ::= OCTET STRING

ACBioInformation   ::= OCTET STRING

BiometricCertificate ::= BiometricDeviceCertificate

BiometricType ::= BioAPI-BIR-BIOMETRIC-TYPE

HandshakeType ::=  ENUMERATED {
     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),
     ...,
     certificate-url (21),
```

```
     certificate-status (22),
     biometrics-client-hello (100),
     biometrics-server-hello (101),
     biometrics-verify (102),
     biometrics-retry-request (103),
     biometrics-finished (104),
     biometrics-ttp-request (105),
     biometrics-ttp-response (106)
     }

Handshake ::= CHOICE {
     hello-request           [0]  HelloRequest,
     client-hello            [1]  ClientHello,
     server-hello            [2]  ServerHello,
     certificate             [11] Certificate,
     server-key-exchange          [12] ServerKeyExchange,
     certificate-request          [13] CertificateRequest,
     server-hello-done       [14] ServerHelloDone,
     certificate-verify      [15] CertificateVerify,
     client-key-exchange          [16] ClientKeyExchange,
     finished           [17] Finished,
     ...,
     [[
     certificate-url              [21] CertificateURL,
     certificate-status      [22] CertificateStatus
     ]],
     [[
        biometrics-client-hello       [100]     BiometricsClientHello,
        biometrics-server-hello       [101]     BiometricsServerHello,
        biometrics-verify           [102]  BiometricsVerify,
        biometrics-retry-request [103]     BiometricsRetryRequest,
        biometrics-finished         [104]  BiometricsFinished,
        biometrics-ttp-request        [105]     BiometricsTTPRequest,
        biometrics-ttp-response       [106]     BiometricsTTPResponse
        ]]
        }

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

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

BSP-BFP-SchemaType ::= ENUMERATED {
     bsp(0),
     bfp(1)
     }

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-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),
    ...
        }


BiometricsServerHello ::=       BiometricAuthenticationRequest

Quality ::= INTEGER(0..100)

BiometricAuthenticationRequest ::=      SEQUENCE {
        biometricMethod     BiometricMethod,
        requestFMR              BioAPI-FMR,
                    -- (32bit 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

BiometricsVerify ::= SEQUENCE   {
        biometricData   CHOICE  {
                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]
                                        BDforSCObyClientModel,
                storage-comparison-outsourcing-by-server-model [10]
                                        BDforSCObyServerModel,
          ...
                },
        digitalSignature        SignedDatabyClient
        }

SignedDatabyClient ::= CHOICE {
        digital-signature       [0] SignedData,
                    --import from X9.84-CMS
        aCBioOnClient           [1] ACBioContentInformation
                                --import from ISO/IEC SC27 CD24761
```

```
        }

BDforLocalModel ::= BiometricClientProcess

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

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

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

CertificateIDInformation        ::=     CertificateSerialNumber

BDforDownloadModel ::= BDforLocalModel

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

BDforCenterModel ::= SampleData     -- BIR: BioAPI defined format --

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

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

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

BDforCObyServerModel ::= SampleData          -- BIR: BioAPI defined format --

BDforSCObyClientModel ::= SEQUENCE {
        bFPSchemaforTTPProcess          BSP-BFP-Schemas,
        thirdPartyInfo                  UTF8String,
        biometric-ttp-Process           BiometricsTTPResponse
        }
```

```
BDforSCObyServerModel ::= SampleData          -- BIR: BioAPI defined format --

BiometricsRetryRequest ::= BiometricAuthenticationRequest


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

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
BiometricsHandshake
        unacceptable-biometrics (116),     -- Extension item for
BiometricsHandshake
        unsupported-biometrics  (117),     -- Extension item for
BiometricsHandshake
        bad-template            (118),     -- Extension item for
BiometricsVerify
        template-revoked        (119),     -- Extension item for
BiometricsVerify
        template-expired        (120),     -- Extension item for
BiometricsVerify
        unknown-bca             (121),     -- Extension item for
BiometricsVerify
        unacceptable-fmr        (122),     -- Extension item for
BiometricsVerify
        no-response-ttp         (123),     -- Extension item for Biometric TTP
Request
        no-template-in-ttp       (124)     -- Extension item for Biometric TTP
Request
        }

BiometricsFinished ::= BiometricAuthenticationResult

BiometricAuthenticationResult ::= BOOLEAN
```

```
BiometricTTPOutsourcingType ::= ENUMERATED {
        storage-type                    (1),
        comparison-type                 (2),
        storage-comparison-type         (3),
     ...
        }

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

BDforStorageOutsourcing ::= TemplateID

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

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

BiometricsTTPResponse ::= SEQUENCE {
     request-body CHOICE {
                storage-type            [1] RBDforStorageOutsourcing,
                comparison-type         [2] RBDforComparisonOutsourcing,
                storage-comparison-type [3] RBDforSCOutsourcing
                },
        digital-signature       SignedDatabyTTP
        }

RBDforStorageOutsourcing ::= XtsmTemplate

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

RBDforSCOutsourcing ::= RBDforComparisonOutsourcing

SignedDatabyTTP ::= CHOICE {
        digital-signature       [0] SignedData,
                  --import from X9.84-CMS
        aCBioOnTTP              [1] ACBioContentInformation
                                --import from ISO/IEC SC27 CD24761
        }

END
```

Annex 4 Template Registration and Updating Process for X.tsm

A4.1 Registration process

This registration process is carried out without the telecommunication environment. This process should be carried out the face to face environment. Therefore everyone is able to trust the relation of the user's certificate and created biometric reference.

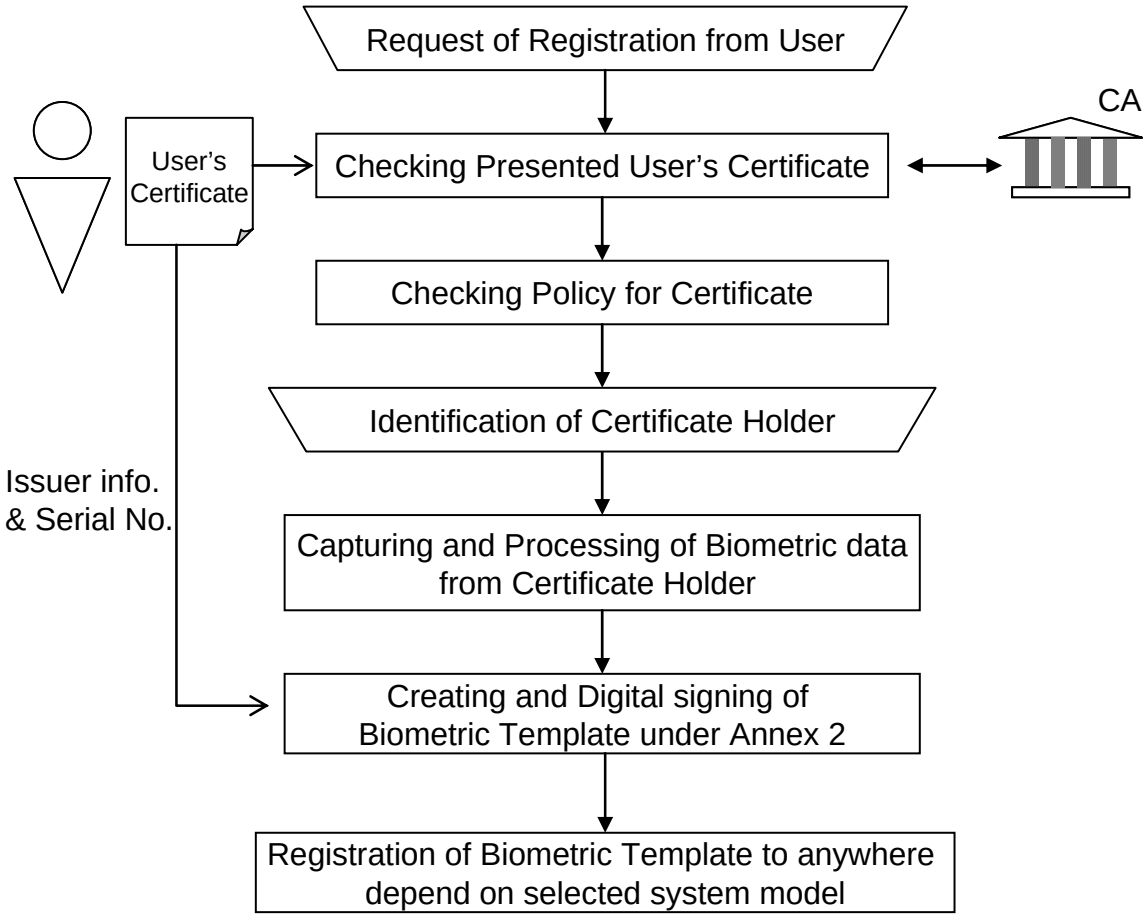


Fig. A2.1 Registration process and procedure for X.tsm

(1)Request of Registration from User

User requests a registration process to operator of a telebiometric template issuer. The issuer should be certified from CA, and the operator should be managed his operation from the issuer by certified manner for security and privacy.

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

(2)Checking Presented User's Certificate

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

(3)Checking Policy for Certificate

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

(4)Identification of Certificate Holder

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

(5)Capturing and Processing of Biometric sample from Certificate Holder

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

(6)Creating and Digital signing of Biometric Template under Table 4

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

(7)Registration of Biometric Reference to anywhere depend on selected system model

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

Note:

The "Identification of Certificate Holder" procedures should refer X.tpp-1:"A Guideline of Technical and Managerial Countermeasures for Biometric Data Security". And transportation procedures on the "Registration of Biometric Template" should refer X.tpp-1 too.

A4.2 Updating process or Revocation process

The biometric reference has accuracy decline under aging of end-users. So It should update before going off the reference. This is a primary process for biometric reference updating. (In the future, this process may be revised for advanced updating process.)

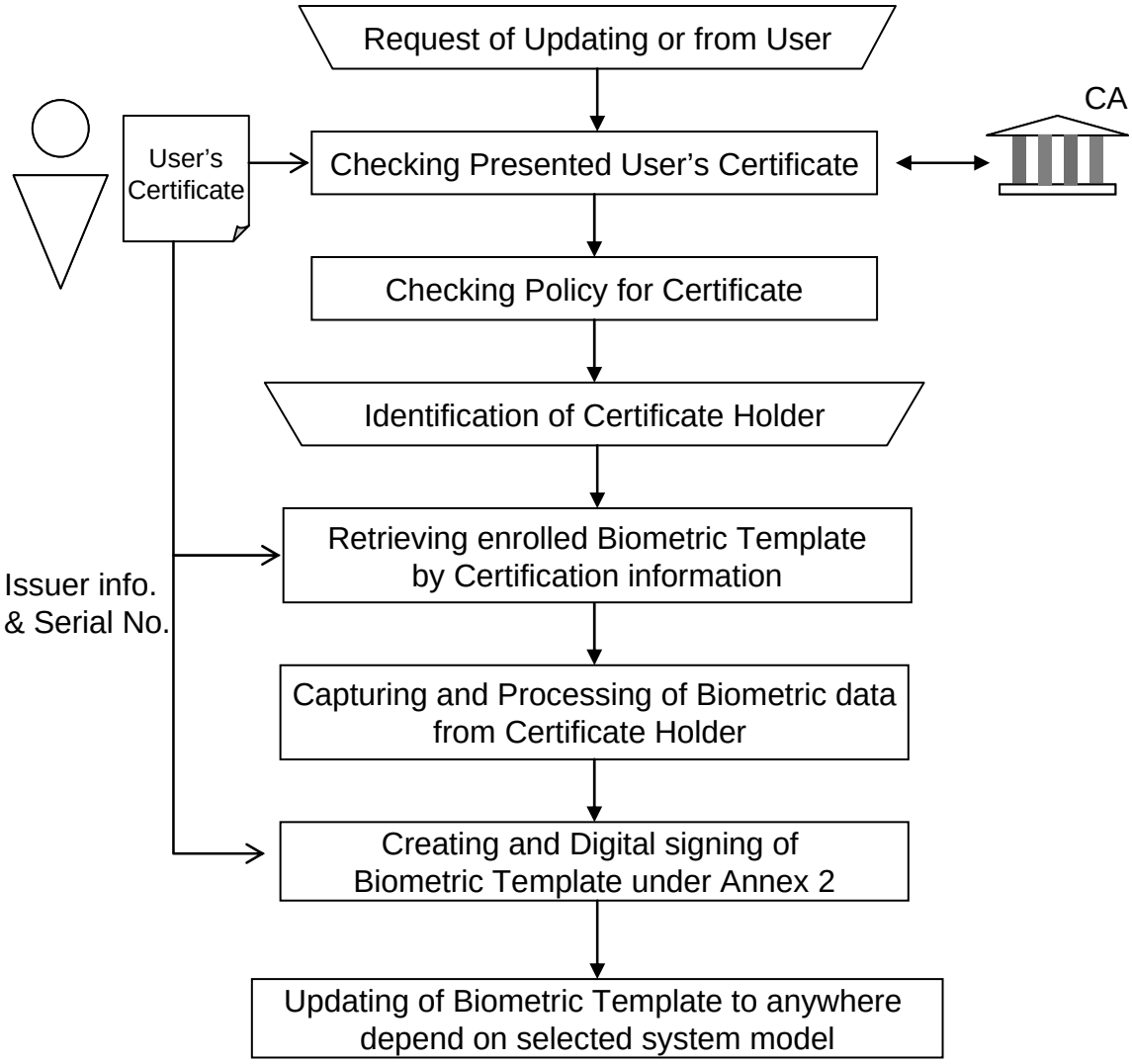


Fig.A2.2 Updating process and procedure for X.tsm

(1)Request of Updating from User

User requests a updating process to operator of a telebiometric template issuer. The issuer should be certified from CA, and the operator should be managed his operation from the issuer by certified manner for security and privacy.

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

(2)Checking Presented User's Certificate

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

(3)Checking Policy for Certificate

The operator checks policy for certificate, i.e., cypher algorithm, key length, etc.\

(4)Identification of Certificate Holder

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

(5)Retrieving enrolled Biometric Reference by Certification information depend on selected system model

If the user selected the TTP storage model, then the operator retrieves registered an old biometric reference, and deletes the registered biometric reference.

(6)Capturing and Processing of Biometric sample from Certificate Holder

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

(7)Creating and Digital signing of Biometric Template under Table4

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

(8)Updating of Biometric Reference to anywhere depend on selected system model

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

Note:

The "Identification of Certificate Holder" procedures should refer X.tpp-1. And transportation procedures on the "Updating of Biometric Template" should refer X.tpp-1 too.

Appendix 1 ASN.1 definitions for TLS and TLS extension

```
TLS DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

IMPORTS   DistinguishedName
     FROM InformationFramework
          {joint-iso-itu-t ds(5) module(1) informationFramework(1) 5}
     Certificate
     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

SequenceNumber::=  UINT64

ConnectionEnd      ::=  ENUMERATED {
     server,
     client
     }

BulkCipherAlgorithm     ::=  ENUMERATED {
     null,
     rc4,
     rc2,
     des,
     triple-des,
     des40,
     ides,
     ...
     }

CipherType         ::=  ENUMERATED {
     stream,
     block
     }

IsExportable  ::=  BOOLEAN

MACAlgorithm  ::=  ENUMERATED {
     null,
     md5,
     sha,
     ...
     }

CompressionMethod  ::=  ENUMERATED {
     null,
```

```
     ...
     }

SecurityParameters ::=  SEQUENCE {
     entity                  ConnectionEnd,
     bulk-cipher-algorithm   BulkCipherAlgorithm,
     cipher-type             CipherType,
     key-size                UINT8,
     key-material-length     UINT8,
     is-exportable           IsExportable,
     mac-algorithm           MACAlgorithm,
     hash-size               UINT8,
     compression-algorithm   CompressionMethod,
     master-secret           Opaque(SIZE(48)),
     client-random           Opaque(SIZE(32)),
     server-random           Opaque(SIZE(32)),
     ...
     }

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

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
          }
     }

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),
     ...
     }

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),
     ...
     }

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

HandshakeType ::=  ENUMERATED {
     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),
     certificate-url (21),
     certificate-status (22),
     ...
     }

Handshake ::= CHOICE {
     hello-request       [0]  HelloRequest,
     client-hello        [1]  ClientHello,
     server-hello        [2]  ServerHello,
     certificate         [11] Certificate,
     server-key-exchange     [12] ServerKeyExchange,
     certificate-request     [13] CertificateRequest,
     server-hello-done   [14] ServerHelloDone,
     certificate-verify  [15] CertificateVerify,
     client-key-exchange     [16] ClientKeyExchange,
     finished            [20] Finished,
     ...,
        [[
         certificate-url         [21] CertificateURL,
         certificate-status [22] CertificateStatus
        ]]
     }

HelloRequest  ::= NULL

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

SessionID ::= UINT32

CipherSuite    ::= ENUMERATED {
     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-ede-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-ede-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-ede-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-ede-cbc-sha (19),
     tls-dhe-rsa-export-with-des40-cbc-sha (20),
     tls-dh-anon-export-rc4-40-md5 (21),
     tls-dh-anon-with-rc4-128-md5(22),
     tls-dh-anon-export-with-des40-cbc-sha (23),
     tls-dh-anon-with-des-cbc-sha (24),
     tls-dh-anon-with-3des-ede-cbc-sha (25),
     ...
```

```
     }

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

ApplicationData     ::=  Opaque

ExtensionType ::=  INTEGER(0..66535)

NameType  ::=  ENUMERATED {
     host-name(0),
     ...
     }

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 ::= {
     CertificateURL IDENTIFIED BY 2
     }

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))

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

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

ListOfTrustedAuthority  ::=  SEQUENCE OF TrustedAuthority

IdentifierType::=  ENUMERATED  {
     pre-agreed (0),
     key-sha1-hash (1),
     x509-name (2),
     cert-sha1-hash (3),
     ...
     }

TrustedAuthority ::= CHOICE {
     pre-agreed     [0]  NULL,
     key-sha1-hash  [1]  SHA1Hash,
     x509-name      [2]  DistinguishedName,
     cert-sha1-hash[3]   SHA1Hash,
     ...
     }

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

TruncatedHMAC ::= Opaque(SIZE(10))

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

CertificateStatusType   ::=  ENUMERATED {
     ocsp (0),
     ...
     }

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})
     }

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

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

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

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

ServerCertificate   ::=  SEQUENCE OF Certificate


KeyExchangeAlgorithm    ::=  ENUMERATED {
     rsa (0),
     diffie-hellman (1),
     ...
     }

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)
     }

ServerKeyExchange  ::=  CHOICE {
     rsa       [0]  SEQUENCE {
          params         ServerRSAParams,
          signed-params  Signature
                 },
     diffie-hellman[1]  SEQUENCE {
          params         ServerDHParams,
          signed-params    Signature
          },
        ...
     }

SignatureAlgorithm ::=  ENUMERATED {
     anonymous (0),
     rsa (1),
     dsa (2),
     ...
     }

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))
                 },
     ...
     }

ClientCertificateTypes  ::=  SEQUENCE OF ClientCertificateType

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

DistinguishedNames ::=  SEQUENCE OF DistinguishedName

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

ServerHelloDone         ::=  NULL

ClientCertificate  ::=  ServerCertificate

ClientKeyExchange  ::=  CHOICE {
     rsa                EncryptedPreMasterSecret,
     diffie-hellman     ClientDiffieHellmanPublic,
     ...
     }
```

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

EncryptedPreMasterSecret     ::=  ENCRYPTED{PreMasterSecret}

PublicValueEncoding      ::=  ENUMERATED {
     implicit,
     explicit
     }

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

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

CertificateVerify  ::=  SEQUENCE {
     signature Signature
     }

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

CertificateStatus  ::=  CHOICE {
     ocsp [0]  OCSPResponse,
     ...
     }

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

END
```