Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology

Technical Guideline for

Health Monitoring Station

Team 3
Chmielowska Irmina
Czyszczonik Jakub
Drzazga Bartosz
Jachniak Mateusz
Kwiatkowski Kamil
Tański Gabriel

Politechnika
Wrocławska
Wrocław 2020

Contents

1 Introduction

1.1 Documentation reference

Title:	Documentation for Health Monitoring Station
Sponsor:	PWr
CC Version:	3.1 (Revision 2)
Version Number:	1.0
Registration:	BSI-CC-PP-XXXX
Keywords:	health, monitoring, medicine, identification, verification

1.2 Overview

This Technical Guideline gives the common specifications, comprising the PKI used for Access Control as well as a mapping of the protocols to ASN.1-specifications.

The HMS has 2 separate interfaces, one for a terminal connection which enables users to communicate with the device, and the second interface used for connection with medical equipment.

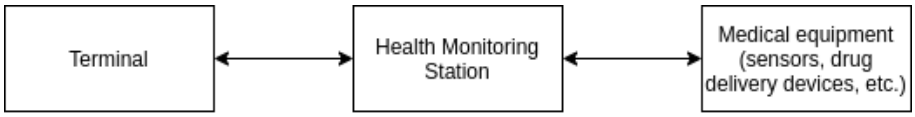

Figure 1.1: Both interfaces of the HSM

Some of the protocols can be executed on both interfaces (namely protocols used for authentication and encryption), but the rest of the protocols can be only executed on a correct interface. The specification of allowed protocols is given in the following chapters.

Each of the interfaces supports bidirectional data transmission. For communication between the HMS and a sensor, any device can initiate protocol execution. In the case of an interface connecting the HMS to a terminal, it is the terminal that initiates protocol execution.

1.3 Terminology

The key words ”MUST”, ”MUST NOT”, ”REQUIRED”, ”SHALL”, ”SHALL NOT”, ”SHOULD”, ”SHOULD NOT”, ”RECOMMENDED”, ”MAY”, and ”OPTIONAL” in this document are to be interpreted as described in RFC 2119 [8].

1.4 Abbreviations

The following abbreviations are commonly used throughout this specification:

- AES - Advanced Encryption Standard
- AKE - Authenticated Key Exchange
- CA - Certificate Authority / Authorities

- DACAs - Device Administrator Certificate Authorities
- DH - Diffie–Hellman
- ECDH - Elliptic-curve Diffie–Hellman
- ECDSA - Elliptic Curve Digital Signature Algorithm
- HMS - Health Monitoring Station
- MAC - Message authentication code
- MCA - Manufacturer Certificate Authority
- MFCAs - Medical Facility Certificate Authorities
- MSE - Mean square error
- PKI - Public Key Infrastructure
- SCAs - Supplier Certificate Authorities

2 Protocols

Protocols that are used in Health Monitoring Station are divided into two sections:

- Terminal connection - communication between the HMS and a device serving the user interface.
- Sensors connection - communication with medical equipment.

2.1 Terminal connection

Protocols that enable users to get, change or upload necessary data to the device:

1. User authentication - protocol used in every attempt to connect to the device
2. Treatment reconfiguration - includes actions like e.g. changing dosages, creating new tasks, changing priorities, selecting different mode.
3. Allowed medical staff management - protocol, that is used to manage doctors.
4. Software update - protocol which is used to get a new software to the HMS.
5. Audit - review of registered events.
6. Data verification - checking authenticity and integrity of data.
7. NTS - Network Time Security for the Network Time Protocol as described in RFC 8915 [6].

2.2 Sensors connection

Protocols used to enable a communication link with medical equipment (like sensors or drug delivery devices):

1. Sensor deployment - deployment of new sensors.
2. Software update - protocol which is used to get a new software to a sensor.
3. Audit - review of registered events.
4. Data verification - checking authenticity and integrity of data.
5. Sensor's health check - a liveness probe that allows the HMS to periodically check the status of a sensor (check if it is online, check system parameters).
6. NTS - Network Time Security for the Network Time Protocol as described in RFC 8915 [6].

2.3 Time protocol

Accurate time measurements are critical for safety of the patient and for audit of events. The HMS uses an internal quartz oscillator embedded in a real-time clock (RTC) circuit. The RTC is graded for usage in time-critical systems. The HMS MUST synchronize its time with trusted time servers over NTS protocol each time it is connected to the Internet.

The deployed sensors SHOULD have its own RTC circuit. Absence of an RTC can be justified only in cases where the device size (limited by medical conditions) makes it impossible to embed another circuit with its battery. In such cases the sensor MUST implement a software-based RTC.

3 Public Key Infrastructure (PKI)

Terminal Authentication requires the terminal to prove to the chip that it is entitled to access the device. Such a terminal is equipped with at least one Terminal Certificate, encoding the terminal's public key and access rights, and the corresponding private key. After the terminal has proven knowledge of this private key, the chip grants the terminal access to sensitive data as indicated in the Terminal Certificate. The PKI required for issuing and using Terminal Certificates consists of the following entities:

1. Manufacturer Certificate Authority (MCA)
2. Device Administrator Certificate Authorities (DACAs)
3. Supplier Certificate Authorities (SCAs)
4. Medical Facility Certificate Authorities (MFCAs)
5. Terminals

This PKI forms the basis of Extended Access Control. The DACAs, SCAs and MFCAs, due to their role in the chain, are also called Intermediary CAs (ICAs).

3.1 Manufacturer CA

The manufacturer of the device is in possession of a trust point that issues certificates for all other CAs - DACAs, SCAs, MFCAs.
In order to mitigate the problems posed by root key expiry (and make rollover easier), the Manufacturer CA actually possesses two root certificates with different public keys and expiry dates at any given time. Such certificates are called **MCA Link Certificates**. It does not make a significant difference in the PKI, and in the document the MCA (and its Link Certificate) is to be treated as a singular entity.
The MCA is a root CA in the certificate chain, and as such, does not issue Terminal Certificates directly. Rather, it issues certificates for other CAs, which in turn issue holder certificates for persons entitled to access some sensitive data or perform some predefined operations. The conditions under which a MCA grants access to sensitive data or functionalities of the device is out of the scope of this document and SHOULD be stated in a certificate policy. All Certificates MUST contain information, such as which data/functionality a certain certificate holder is entitled to access. To diminish the potential risk introduced by lost or stolen terminals, Certificates MUST contain a short validity period. The validity period is assigned by the issuing MCA at its own choice and this validity period may vary depending on the entity the certificate is issued to.

3.2 Device Administrator CA

The entity handling the HMS administrators (usually affiliated with the Manufacturer, although this is not a requirement) requires a CA status in order to be able to issue certificates to the HMS administrators in its employment. The Terminal Certificates issued by a DACA usually inherit both the access rights and the validity period from the DACA certificate, however, the DACA MAY choose to further restrict the access rights or the validity period depending on the terminal the certificate is issued for. The Terminal Cerificates issued by the DACA are intended to be used by the HMS administrators, and therefore the Certificate issued for DACAs properly reflect these intentions.

3.3 Supplier CA

The entities handling the distribution of the HMS devices to the end customers require being given a CA status in order to issue valid certificates for the end-user terminals. Similar to DACAs, the Terminal Certificates issued by a SCA inherit the access rights and the validity period from the CA certificates, and in some cases exemptions can be made, depending on the unique situation of the patient. As with other CAs, the SCA issues certificates only for one role - the user - and so the certificates given to the SCAs are adequately limited in scope.

3.4 Medical Facility CA

Medical Facilities can be granted CA status in order to be able to issue certificates authorized by the MCA. The Terminal certificates issued by the MFCA are intended to be used by authorized medical personnel, and as such, this is the scope that the MFCAs are limited to.

3.5 Card verifiable certificates

MCA Link Certificates, ICA certificates, and Terminal Certificates are to be validated by the HMS. Due to the computational restrictions of the device, the certificates MUST be in a card verifiable format:

- The certificate format and profile specified in chapter 6 SHALL be used.
- The signature algorithm, domain parameters, and key sizes to be used are determined by the MCA.
- MCA Link Certificates MAY include a public key that deviates from the current parameters, i.e. the MCA MAY switch to a new signature algorithm, new domain parameters, or key sizes.

3.6 Certificate Scheduling

Each certificate MUST contain a validity period. This validity period is identified by two dates, *the certificate effective date* and *the certificate expiration date*.

- **Certificate Effective Date:** The certificate effective date SHALL be the date of the certificate generation.
- **Certificate Expiration Date:** The certificate expiration date SHALL be the date *after* which the certificate expires. It may be arbitrarily chosen by the certificate issuer.

When generating certificates the issuer MUST carefully plan the roll-over of certificates, as sufficient time for propagation of certificates and set up of certificate chains MUST be provided. Obviously, a new certificate must be generated before the current certificate expires. The resulting maximum distribution time equals the certificate expiration date of the old certificate minus the certificate effective date of the new certificate. The validity period of the ICA-issued certificates MUST NOT extend beyond the validity period of the corresponding ICA certificate.

3.7 Certificate Validation

To validate a Terminal Certificate, the chip MUST be provided with a certificate chain starting at a trust-point stored on the chip. Due to the scheduling of MCA Link Certificates, at most two trust-points need to be stored on the chip. Those trust-points are more or less recent public keys of the chip's MCA. The initial trust-point(s) SHALL be stored securely in the chip's memory in the production or (pre-) personalization phase.

The chip MUST accept expired MCA Link Certificates but it MUST NOT accept expired ICA and Terminal Certificates. To determine whether a certificate is expired, the chip SHALL use its current date. This is achieved by the RTC embedded in the device and time synchronization protocol.

3.7.1 General Procedure

The certificate validation procedure consists of two steps:

1. **Certificate Verification**: The signature MUST be valid and unless the certificate is a MCA Link Certificate, the certificate MUST NOT be expired. If the verification fails, the procedure SHALL be aborted.
2. **Internal Status Update**: The public key and the attributes described in chapter 6 (including relevant certificate extensions) MUST be imported, new trust-points MUST be enabled, expired trust-points MUST be disabled for the verification of MFCA Certificates.

The operations of enabling and disabling a trust-point MUST be implemented as an atomic operation.
Enabling a trust-point: The new trust-point SHALL be added to the list of trust-points.
Disabling a trust-point: Expired trust-points MUST NOT be used for the verification of MFCA Certificates but MUST remain usable for the verification of MCA Link Certificates. Disabled trust-points MAY be deleted after the successful import of the successive Link Certificate.

3.8 Effective Authorization

Each certificate SHALL contain a Certificate Holder Authorization Template that identifies the terminal type and determines the relative authorization of the certificate holder assigned by the issuing certificate authority. To determine the effective authorization of a certificate holder, the chip MUST calculate a bitwise Boolean 'and' of the relative authorization contained in the Terminal Certificate, the referenced ICA Certificate (if applicable), and the referenced MCA Certificate.
The effective authorization SHALL be interpreted by the chip as follows:
The effective role is a MCA:

- This link certificate was issued by the MCA.
- The chip MUST update its internal trust-point, i.e. the public key and the effective authorization.
- The chip MUST NOT grant the MCA access to the device. (i.e. the effective authorization SHOULD be ignored).

The effective role is a ICA (optional):

- The certificate was issued by the MCA for an authorized ICA.
- The chip MUST NOT grant a ICA access to the device (i.e. the effective authorization SHOULD be ignored).

The effective role is a Terminal:

- The chip MUST grant the authenticated terminal access to sensitive data and/or functionality according to the effective authorization.

4 ASN.1 Specifications

4.1 Information on Supported Security Protocols

The ASN.1 data structure `SecurityInfos` SHALL be provided by the device to indicate supported security protocols. The data structure is specified as follows:

```
SecurityInfos    ::= SET OF SecurityInfo

SecurityInfo     ::= SEQUENCE {
    protocol         OBJECT IDENTIFIER,
    requiredData     ANY DEFINED BY protocol,
    optionalData     ANY DEFINED BY protocol OPTIONAL
}
```

The elements contained in a `SecurityInfo` data structure have the following meaning:

- The object identifier `protocol` identifies the supported protocol.
- The open type `requiredData` contains protocol specific mandatory data.
- The open type `optionalData` contains protocol specific optional data.

4.1.1 Supported Protocols

The ASN.1 specifications for the protocols provided in this specification are described in the following sections.

4.1.1.1 User authentication

To indicate support for User Authentication `SecurityInfos` may contain the following entries:

- At least one `UserAuthenticationPublicKeyInfo` MUST be present.
- At least one `UserAuthenticationInfo` MUST be present.

If more than one User Authentication Public Key is present the optional `keyId` MUST be used in all these data structures to indicate the local key identifier.
The implementation may support the following algorithms for the UA protocol:

- Static Key Agreement with ECDH
- Static Key Agreement with DH

The above algorithms may be combined with one (or more) of the following symmetric algorithms for secure messaging:

- AES 128
- AES 192
- AES 256

`UserAuthenticationInfo`: This data structure provides detailed information on an implementation of User Authentication.

- The object identifier `protocol` SHALL identify the algorithms to be used (i.e. key agreement, symmetric cipher and MAC).
- The integer `version` SHALL identify the version of the protocol. Currently, only version 1 is supported.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the device provides multiple public keys for User Authentication.

```
id-UA-DH                          OBJECT IDENTIFIER ::= {id-UA 1}
id-UA-DH-AES-CBC-CMAC-128         OBJECT IDENTIFIER ::= {id-UA-DH 1}
id-UA-DH-AES-CBC-CMAC-192         OBJECT IDENTIFIER ::= {id-UA-DH 2}
id-UA-DH-AES-CBC-CMAC-256         OBJECT IDENTIFIER ::= {id-UA-DH 3}
id-UA-ECDH                        OBJECT IDENTIFIER ::= {id-UA 2}
id-UA-ECDH-AES-CBC-CMAC-128       OBJECT IDENTIFIER ::= {id-UA-ECDH 1}
id-UA-ECDH-AES-CBC-CMAC-192       OBJECT IDENTIFIER ::= {id-UA-ECDH 2}
id-UA-ECDH-AES-CBC-CMAC-256       OBJECT IDENTIFIER ::= {id-UA-ECDH 3}

UserAuthenticationInfo  ::= SEQUENCE {
    protocol     OBJECT IDENTIFIER(
                 id-UA-DH-AES-CBC-CMAC-128 |
                 id-UA-DH-AES-CBC-CMAC-192 |
                 id-UA-DH-AES-CBC-CMAC-256 |
                 id-UA-ECDH-AES-CBC-CMAC-128 |
                 id-UA-ECDH-AES-CBC-CMAC-192 |
                 id-UA-ECDH-AES-CBC-CMAC-256),
    version      INTEGER, -- MUST be 1
    keyId        INTEGER OPTIONAL
}
```

`UserAuthenticationPublicKeyInfo`: This data structure provides a public key for User Authentication.

- The object identifier `protocol` SHALL identify the type of the public key (i.e. DH or ECDH).
- The sequence `userAuthenticationPublicKey` SHALL contain the public key in encoded form.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the device provides multiple public keys for User Authentication.

```
id-PK-DH    OBJECT IDENTIFIER ::= {id-PK 1}
id-PK-ECDH  OBJECT IDENTIFIER ::= {id-PK 2}

UserAuthenticationPublicKeyInfo ::= SEQUENCE {
    protocol                        OBJECT IDENTIFIER(id-PK-DH | id-PK-ECDH),
    userAuthenticationPublicKey     SubjectPublicKeyInfo,
    keyId                           INTEGER OPTIONAL
}
```

4.1.1.2 Treatment Reconfiguration

To indicate support for Treatment Reconfiguration `SecurityInfos` may contain the following entries:

- At least one `TreatmentReconfigurationHMAC` MUST be present.
- At least one `TreatmentReconfigurationInfo` MUST be present.

In order to ensure authenticity and integrity of reconfiguration, the implementation MUST support the following algorithm for the TR protocol:

- HMAC

which MAY use one of the following hash algorithms:

- SHA-256
- SHA-384
- SHA-512

`TreatmentReconfigurationInfo`:This data structure provides detailed information on an implementation of Reconfiguration Info

- The object identifier protocol SHALL identify the algorithm.
- The JSON object config SHALL consist of keys and values of new configuration data.
- The integer version SHALL identify the version of the protocol.
- The integer keyId MAY be used to indicate the local key identifier.

```
id-HMAC-SHA256          OBJECT IDENTIFIER ::= {id-HMAC-SHA256}
id-HMAC-SHA384          OBJECT IDENTIFIER ::= {id-HMAC-SHA384}
id-HMAC-SHA512          OBJECT IDENTIFIER ::= {id-HMAC-SHA512}

TreatmentReconfigurationInfo  ::= SEQUENCE {
    protocol     OBJECT IDENTIFIER(
                 id-HMAC-SHA256 |
                 id-HMAC-SHA384 |
                 id-HMAC-SHA512)
    config       ENUM,
    version      INTEGER,
    keyId        INTEGER OPTIONAL
}
```

`TreatmentReconfigurationHMAC`:This data structure provides detailed information about HMAC for Treatment Reconfiguration:

- The object identifier protocol SHALL identify the type of HMAC algorithm.
- The sequence TreatmentReconfigurationHMAC SHALL contain the version of the protocol. Currently, only version 1 is supported
- The integer keyId MAY be used to indicate the local key identifier.

```
id-HMAC-SHA256           OBJECT IDENTIFIER ::= {id-HMAC-SHA256}
id-HMAC-SHA384           OBJECT IDENTIFIER ::= {id-HMAC-SHA384}
id-HMAC-SHA512           OBJECT IDENTIFIER ::= {id-HMAC-SHA512}

TreatmentReconfigurationHMAC ::= SEQUENCE {
    protocol     OBJECT IDENTIFIER(
                 id-HMAC-SHA256 |
                 id-HMAC-SHA384 |
                 id-HMAC-SHA512),
    version      INTEGER,
    keyId        INTEGER OPTIONAL
}
```

4.1.1.3 Allowed medical staff management

To indicate support for Allowed medical staff management `SecurityInfos` may contain the following entries:

- At least one `MedicalStaffUpdate` SHALL be present.

`MedicalStaffUpdate` This data structure provides detailed information on medical staff updates and covers adding and removing medical staff members.

```
id-MSU        OBJECT IDENTIFIER ::= {id-MSU}

MedicalStaffUpdate  ::= SEQUENCE {
    id                  INTEGER
    name                STRING,
    publicKey           SubjectPublicKeyInfo
    action              ENUM
}
```

4.1.1.4 Software update

To indicate support for Software update `SecurityInfos` may contain the following entries:

- At least one `SoftwareUpdateInfo` SHALL be present.
- At least one `SoftwareUpdateSignature` MUST be present.

`SoftwareUpdateInfo` This data structure provides detailed information on software updates. The records are parts of software.

The integer `version` MUST identify the version of the update.

```
id-SU        OBJECT IDENTIFIER ::= {id-SU}

SoftwareUpdateInfo  ::= SEQUENCE {
    version               INTEGER,
    digestAlgorithms      DigestAlgorithmIdentifiers,
}
```

`SoftwareUpdateSignature` This data structure ensures integrity and authenticity of update files.

- The object identifier `type` SHALL identify the type of digital signature of update file.

```
id-SU-2        OBJECT IDENTIFIER ::= {id-SU-2}

SoftwareUpdateSignature  ::= SEQUENCE {
    encapContentInfo      EncapsulatedContentInfo,
}
```

4.1.1.5 Data Verification

To indicate support for Data Verification `SecurityInfos` may contain the following entries:

- At least one `DataVerificationInfo` MUST be present.
- At least one `DataVerificationPublicKeyInfo` MUST be present.

If more than one Data Verification Public Key is present the optional `keyId` MUST be used in all these data structures to indicate the local key identifier.

The implementation may support the following algorithms for the DV protocol:

- Static Key Agreement with ECDH
- Static Key Agreement with DH

`DataVerificationInfo`: This data structure provides detailed information on an implementation of Data Verification.

- The object identifier `protocol` SHALL identify the algorithms to be used (i.e. key agreement, symmetric cipher and MAC).
- The integer `version` SHALL identify the version of the protocol. Currently, only version 1 is supported.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the device provides multiple public keys for User Authentication.

```
id-DV-DH                          OBJECT IDENTIFIER ::= {id-DV 1}
id-DV-DH-AES-CBC-CMAC-128         OBJECT IDENTIFIER ::= {id-DV-DH 1}
id-DV-DH-AES-CBC-CMAC-192         OBJECT IDENTIFIER ::= {id-DV-DH 2}
id-DV-DH-AES-CBC-CMAC-256         OBJECT IDENTIFIER ::= {id-DV-DH 3}
id-DV-ECDH                        OBJECT IDENTIFIER ::= {id-DV 2}
id-DV-ECDH-AES-CBC-CMAC-128       OBJECT IDENTIFIER ::= {id-DV-ECDH 1}
id-DV-ECDH-AES-CBC-CMAC-192       OBJECT IDENTIFIER ::= {id-DV-ECDH 2}
id-DV-ECDH-AES-CBC-CMAC-256       OBJECT IDENTIFIER ::= {id-DV-ECDH 3}

DataVerificationInfo  ::= SEQUENCE {
    protocol     OBJECT IDENTIFIER(
                 id-DV-DH-AES-CBC-CMAC-128 |
                 id-DV-DH-AES-CBC-CMAC-192 |
                 id-DV-DH-AES-CBC-CMAC-256 |
                 id-DV-ECDH-AES-CBC-CMAC-128 |
                 id-DV-ECDH-AES-CBC-CMAC-192 |
                 id-DV-ECDH-AES-CBC-CMAC-256),
    version      INTEGER, -- MUST be 1
    keyId        INTEGER OPTIONAL
}
```

`DataVerificationPublicKeyInfo`: This data structure provides a public key for Data Verification.

- The object identifier `protocol` SHALL identify the type of the public key (i.e. DH or ECDH).
- The sequence `DataVerificationPublicKey` SHALL contain the public key in encoded form.
- The integer `keyId` MAY be used to indicate the local key identifier. It MUST be used if the device provides multiple public keys for User Authentication.

```
id-PK-DH    OBJECT IDENTIFIER ::= {id-PK 1}
id-PK-ECDH  OBJECT IDENTIFIER ::= {id-PK 2}

DataVerificationPublicKeyInfo ::= SEQUENCE {
    protocol                        OBJECT IDENTIFIER(
                                    id-PK-DH |
                                    id-PK-ECDH),
    DataVerificationPublicKey       SubjectPublicKeyInfo,
    keyId                           INTEGER OPTIONAL
}
```

4.1.1.6 Sensor deployment

To indicate support for Sensor deployment `SecurityInfos` may contain the following entries:

- At least one `SensorInfo` SHALL be present.

`SensorInfo` This data structure provides detailed information on sensor.

- The integer `version` MUST identify the version of protocol supported by sensor.

- The object `dataType` MUST specify dataType provided by sensor.
- The sequence SHALL contain the public key in encoded form.

```
id-SD        OBJECT IDENTIFIER ::= {id-SD}

SensorInfo  ::= SEQUENCE {
    version          INTEGER,
    dataType         OBJECT,
    publicKey        SubjectPublicKeyInfo
}
```

4.1.1.7 Additional specification

All object and structures used but not defined in this document are to be taken from [7].

5 ISO 7816 Mapping

In this chapter the protocols for User Authentication, Treatment Reconfiguration, Allowed Medical Staff Management, Software Update, Audit, Data Verification and Sensor Deployment are mapped to ISO 7816 [3] APDUs (Application Protocol Data Units).

5.1 User Authentication

The following sequence of commands SHALL be used with secure messaging to implement User Authentication:

1.MSE: Set AT
2.General Authenticate

The protocol specific data objects SHALL be exchanged with General Authenticate commands as shown below, command chaining MUST NOT be used. At least one of the steps MUST be executed:

Step	Description	Protocol Command Data	Protocol Response Data
1.	Restricted Identification (CONDITIONAL)	0xA0 1st Sector Public Key	0x81 1st Sector-specific Identifier
2.	Restricted Identification (CONDITIONAL)	0xA2 2nd Sector Public Key	0x83 2nd Sector-specific Identifier

5.1.1 Public Key

The Sector Public Key SHALL be encoded as public key data object without tag 0x7F49(i.e. 0x7F49 is replaced by 0xA0/0xA2, respectively), the domain parameters MUST be included.

5.1.2 Sector-specific Identifier

The sector-specific identifier SHALL be encoded as octet string.

5.2 Treatment Reconfiguration

The following command SHALL be used with secure messaging to implement Treatment Reconfiguration in version1 with 3DES Secure Messaging:

1.MSE: Set KAT

The following sequence of commands SHALL be used with secure messaging

- to implement Treatment Reconfiguration in version 1 with AES and
- to implement Treatment Reconfiguration in version 2.

Additionally, this sequence MAY be used to implement Treatment Reconfiguration version 1 with 3DES.

1.MSE: Set AT

2.General Authenticate

The protocol specific data objects SHALL be exchanged with a General Authenticate command as shown below:

Step	Description	Protocol Command Data	Protocol Response Data
1.	Treatment Reconfiguration	0x80 Ephemeral Public Key	0x81 Nonce 0x82 Authentication Token

5.2.1 Ephemeral Public Key

The ephemeral public keys SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

5.2.2 Nonce

The nonce SHALL be encoded as octet string of size 8 octets.

5.2.3 Authentication Token

The authentication token SHALL be encoded as octet string.

5.3 Allowed Medical Staff Management

The following application specific command SHALL be used with secure messaging to implement the verification function:

1.Verify

The following authenticated data MUST have been sent to the terminal as a part of User Authentication:

- For Age Verification the terminal MUST have sent the required date of birth.
- For Name Verification the terminal MUST have sent the required name of the Medical Staff.
- For Document Validity Verification the terminal MUST have sent the current date.

5.4 Software Update

The following command SHALL be used with secure messaging to implement Software Update in version1 with 3DES Secure Messaging:

1.MSE: Set KAT

The following sequence of commands SHALL be used with secure messaging

- to implement Software Update in version 1 with AES and
- to implement Software Update in version 2.

Additionally, this sequence MAY be used to implement Software Update version 1 with 3DES.

1.MSE: Set AT
2.General Authenticate

The protocol specific data objects SHALL be exchanged with a General Authenticate command as shown below:

Step	Description	Protocol Command Data	Protocol Response Data
1.	Software Update	0x80 Ephemeral Public Key	0x81 Nonce 0x82 Authentication Token

5.4.1 Ephemeral Public Key

The ephemeral public keys SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

5.4.2 Nonce

The nonce SHALL be encoded as octet string of size 8 octets.

5.4.3 Authentication Token

The authentication token SHALL be encoded as octet string.

5.5 Data Verification

The following application specific command SHALL be used with secure messaging to implement the verification function:

1.Verify

The following authenticated data MUST have been sent to the terminal as a part of User Authentication:

- For Age Verification the terminal MUST have sent the required date of birth.
- For Name Verification the terminal MUST have sent the required name of the doctor.
- For Document Validity Verification the terminal MUST have sent the current date.

5.6 Sensor Deployment

The following command SHALL be used with secure messaging to implement Sensor Deployment in version1 with 3DES Secure Messaging:

1.MSE: Set KAT

The following sequence of commands SHALL be used with secure messaging

- to implement Sensor Deployment in version 1 with AES and
- to implement Sensor Deployment in version 2.

Additionally, this sequence MAY be used to implement Sensor Deployment version 1 with 3DES.

1.MSE: Set AT
2.General Authenticate

The protocol specific data objects SHALL be exchanged with a General Authenticate command as shown below:

Step	Description	Protocol Command Data	Protocol Response Data
1.	Sensor Deployment	0x80 Ephemeral Public Key	0x81 Nonce 0x82 Authentication Token

5.6.1 Ephemeral Public Key

The ephemeral public keys SHALL be encoded as elliptic curve point (ECDH) or unsigned integer (DH).

5.6.2 Nonce

The nonce SHALL be encoded as octet string of size 8 octets.

5.6.3 Authentication Token

The authentication token SHALL be encoded as octet string.

6 CV Certificates

6.1 Certificate Profile

Self-descriptive card verifiable (CV) certificates according to ISO 7816 [3] SHALL be used. Details on the encoding of the data objects used in the certificate profile can be found in chapter DER Encoding [7].

6.1.1 Certificate Profile Identifier

The version of the profile is indicated by the Certificate Profile Identifier.

6.1.2 Certification Authority Reference

The Certification Authority Reference is used to identify the public key to be used to verify the signature of the certification authority. The Certification Authority Reference MUST be equal to the Certificate Holder Reference in the corresponding certificate of the certification authority.

6.1.3 Public Key

Details on the encoding of public keys can be found in chapter DER Encoding [7].

Data Object
Certificate
Certificate Body
Certificate Profile Identifier
Certification Authority Reference
Public Key
Certificate Holder Reference
Certificate Holder Authorization Template
Certificate Effective Date
Certificate Expiration Date
Certificate Extensions
Certificate Extensions

6.1.4 Certificate Holder Reference

The Certificate Holder Reference is used to identify the public key contained in the certificate.

6.1.5 Certificate Holder Authorization Template

The role and authorization of the certificate holder SHALL be encoded in the Certificate Holder Authorization Template. This template is a sequence that consists of the following data objects:

- An object identifier that specifies the terminal type and the format of the template.
- A discretionary data object that encodes the relative authorization, i.e. the role and authorization of the certificate holder relative to the certification authority.

6.1.6 Certificate Effective/Expiration Date

Indicates the validity period of the certificate. The Certificate Effective Date MUST be the date of the certificate generation.

6.1.7 Signature

The signature on the certificate SHALL be created over the encoded certificate body (i.e. including tag and length). The Certification Authority Reference SHALL identify the public key to be used to verify the signature.

6.2 Certificate Requests

Certificate requests are reduced CV certificates that may carry an additional signature. Details on the encoding of the data objects used in the certificate request profile can be found in chapter DER Encoding [7].

6.2.1 Certificate Profile Identifier

The version of the profile is identified by the Certificate Profile Identifier.

6.2.2 Certification Authority Reference

The Certification Authority Reference SHOULD be used to inform the certification authority about the private key that is expected by the applicant to be used to sign the certificate. If the Certification Authority Reference contained in the request is different than the Certification Authority Reference contained in the issued certificate, the corresponding certificate of the certification authority SHOULD also be provided to the applicant in response.

6.2.3 Public Key

Details on the encoding of public keys can be found in chapter DER Encoding [7].

6.2.4 Certificate Holder Reference

The Certificate Holder Reference is used to identify the public key contained in the request and the resulting certificate.

7 DER Encoding (Normative)

The Distinguished Encoding Rules (DER) according to X.690 [4] SHALL be used to encode both ASN.1 data structures and (application specific) data objects. The encoding results in a Tag-Length-Value (TLV) structure as follows:
Tag: The tag is encoded in one or two octets and indicates the content.
Length: The length is encoded as unsigned integer in one, two, or three octets resulting in a maximum length of 65535 octets. The minimum number of octets SHALL be used.
Value: The value is encoded in zero or more octets.

7.1 ANS.1

The encoding of data structures defined in ASN.1 syntax is described in X.690

7.2 Data Objects

7.2.1 Encoding of Values

The basic value types used in this specification are the following: (unsigned) integers, character strings, octet strings, object identifiers, and sequences.

7.2.2 Unsigned Integers

All integers used in this specification are unsigned integers. An unsigned integer SHALL be converted to an octet string using the binary representation of the integer in big-endian format. The minimum number of octets SHALL be used, i.e. leading octets of value 0x00 MUST NOT be used.
Note: In contrast the ASN.1 type INTEGER is always a signed integer.

7.2.3 Dates

A date is encoded in 6 digits $d_1, \ldots, d_6$ in the format YYMMDD using timezone GMT. It is converted to an octet string $o_1, \ldots, o_6$ by encoding each digit d_j to an octet o_j as unpacked BCDs ($1 \leqslant j \leqslant 6$). The year YY is encoded in two digits and to be interpreted as 20YY, i.e. the year is in the range of 2000 to 2099.

7.2.4 Character Strings

A character string $c_1 \cdots c_n$ is a concatenation of n characters c_j with $1 \leqslant j \leqslant n$. It SHALL be converted to an octet string $o_1 \cdots o_n$ by converting each character o_j to an octet o_j using the ISO/IEC8859-1 [1] character set. The character codes 0x00-0x1F and 0x7F-0x9F are unassigned and MUST NOT be used. The conversion of an octet to an unassigned character SHALL result in an error.

7.2.5 Octet string

An octet string $o_1 \cdots o_n$ is a concatenation of n octets o_j with $1 \leqslant j \leqslant n$. Every octet o_j consists of 8 bits.

7.2.6 Object Identifiers

An object identifier i_1,$i_2, \ldots, i_n$ is encoded as an ordered list of n unsigned integers i_j with $1 \leqslant j \leqslant n$. It SHALL be converted to an octet string $o_1 \cdots o_{n-1}$ using the following procedure:

1. The first two integers i_1 and i_2 are packed into a single integer i that is then converted to the octet string o_1. The value i is calculated as follows: $i = i_1 \cdot 40 + i_2$
2. The remaining integers i_j are directly converted to octet strings o_{j-1} with $3 \leqslant j \leqslant n$.

More details on the encoding can be found in [4].

Note: The unsigned integers are encoded as octet strings using the big-endian format as described in section 7.2.2 , however only bits 1-7 of each octet are used. Bit 8 (the leftmost bit) set to one is used to indicate that this octet is not the last octet in the string.

7.2.7 Sequences

A sequence $D_1 \cdots D_n$ is an ordered list of n data objects D_j with $1 \leqslant j \leqslant n$. The sequence SHALL be converted to a concatenated list of octet strings $O_1 \cdots O_n$ by DER encoding each data object D_j to an octet string O_j.

7.3 Public Key Data Objects

A public key data object contains a sequence of an object identifier and several context specific data objects:

- The object identifier is application specific and refers not only to the public key format (i.e. the context specific data objects) but also to its usage.
- The context specific data objects are defined by the object identifier and contain the public key value and the domain parameters.

The format of public keys data objects used in this specification is described below.

7.3.1 Diffie Hellman Public Keys

The data objects contained in a DH public key are shown in table below. The order of the data objects is fixed.

Data Object	Abbrev.	Tag	Type
Object Identifier		0x06	Object Identifier
Prime modulus	p	0x81	Unsigned Integer
Order of the subgroup	q	0x82	Unsigned Integer
Generator	g	0x83	Unsigned Integer
Public value	y	0x84	Unsigned Integer

7.3.2 Elliptic Curve Public Keys

The data objects contained in an EC public key are shown in the table below. The order of the data objects is fixed.

Data Object	Abbrev.	Tag	Type
Object Identifier		0x06	Object Identifier
Prime modulus	p	0x81	Unsigned Integer
First coefficient	a	0x82	Unsigned Integer
Second coefficient	b	0x83	Unsigned Integer
Base point	G	0x84	Elliptic Curve Point
Order of the base point	r	0x85	Unsigned Integer
Public point	Y	0x86	Elliptic Curve Point
Cofactor	f	0x87	Unsigned Integer

- CVCA Link Certificates MAY contain domain parameters.
- DV and Terminal Certificates MUST NOT contain domain parameters. The domain parameters of DV and terminal public keys SHALL be inherited from the respective CVCA public key.
- Certificate Requests MUST always contain domain parameters.

7.3.3 Ephemeral Public Keys

For ephemeral public keys the format and the domain parameters are already known. Therefore, only the plain public key value, i.e. and the public value y for Diffie-Hellman public keys is used to convey the ephemeral public key in a context specific data object.

8 Secure Messaging (Normative)

Secure Messaging provides a secure channel (i.e. encrypted and authenticated) between the HMS and authorised user. The provided security level however depends on the mechanism used to set up Secure Messaging. A session is started when secure messaging is established. The session only ends with the release of secure messaging, e.g. by sending a command without secure messaging. Within a session the secure messaging keys may be changed.

8.1 Cryptographic Algorithms

Secure Messaging is based on AES[9] in encrypt-then-authenticate mode, i.e. data is encrypted first and afterwards the formatted encrypted data is input to the authentication calculation. The session keys SHALL be derived from a secure AKE (such as NAXOS[10]) scheme executed between the HMS and the second party. Authentication thereof SHALL be done using ECDSA[11].

8.1.1 ECDSA key generation

The device shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm (ECDSA key pair generation with given elliptic curve domain parameters) and specified cryptographic key sizes of minimum 256 bits that meet the following: ECDSA key pair generation for ECC domain parameters Curve P-256, Curve P-384 or Curve P-521 as specified in [FIPS.186-4][11].

8.1.2 ECC signature generation/verification

The device shall perform signature generation and verification in accordance with ECDSA algorithm with specified cryptographic key sizes of minimum 256 bits that meet the [FIPS.186-4][11].

8.1.3 AES key generation

The device shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm (AES key generation) and specified cryptographic key sizes of minimum 128 bits that meet the following: Advanced Encryption Standard (AES) as specified in [FIPS.197][9]

8.1.4 AES Encryption

For message encryption AES SHALL be used in CBC-mode according to ISO 10116 [12] with key K_{Enc} and $IV = (K_{Enc}, SCC)$.

8.1.5 AES Authentication

For message authentication AES SHALL be used in CMAC-mode [5] with K_{mac} with a MAC length of 8 bytes.

8.1.6 HMAC

Hash-based message authentication code is a specific type of MAC involving a cryptographic hash function and a secret cryptographic key. We will use SHA functions with size: 256, 384, 512 bytes. [2]

Bibliography

[1] 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1. https://www.iso.org/standard/28245.html.

[2] HMAC: Keyed-Hashing for Message Authentication. https://tools.ietf.org/html/rfc2104.

[3] Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange. https://www.iso.org/standard/54550.html.

[4] ITU-T. Information Technology – ASN.1 encoding rules: Specification of Basic Encoding Rules(BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), X.690, 2002. https://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf.

[5] Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication. https://csrc.nist.gov/publications/detail/sp/800-38b/final.

[6] RFC 8915: Network Time Security for the Network Time Protocol. https://tools.ietf.org/html/rfc8915.

[7] Technical Guideline TR-03110-3 Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3 – Common Specifications. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI˙TR-03110˙Part-3˙V2-1.pdf.

[8] RFC 2119: Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Harvard University. https://tools.ietf.org/html/rfc2119, March 1997.

[9] Advanced Encryption Standard (AES). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, November 2001.

[10] Stronger Security of Authenticated Key Exchange. https://www.researchgate.net/publication/221106180˙Stronger˙Security˙of˙Authenticated˙Key˙Exchange, November 2007.

[11] Digital Signature Standard (DSS). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf, July 2013.

[12] ISO/IEC 10116:2017 - Information technology — Security techniques — Modes of operation for an n-bit block cipher. https://www.iso.org/obp/ui/#iso:std:iso-iec:10116:ed-4:v1:en, 2017.