D3.10

Design of libraries for HR security and privacy services - V2

ABSTRACT

This deliverable provides the second version of the design of security and privacy services, in particular the components and the functional primitives regarding identity management, consent management, encryption and privacy. The content of this deliverable derived from the InteropEHRate deliverables D3.3 - Specification of remote and D2D IDM mechanisms for HRs Interoperability - V1 **[D3.3]**, D3.7 - Specification of consent management and decentralized authorization mechanisms for HR Exchange - V1 **[D3.7]** including the progress towards the second version of the aforementioned deliverables, D3.5 - Specification of data encryption mechanisms for mobile and web applications - V1 **[D3.5]** depicts the major features and principles of designing the security libraries addressing the aforementioned security and privacy related topics.

Delivery Date	20th April, 2021
Work Package	WP3
Task	T3.4
Dissemination Level	Public
Type of Deliverable	Report
Lead partner	UBIT

InteropEHRate project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826106.

This document has been produced in the context of the InteropEHRate Project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826106. All information provided in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose.

CONTRIBUTORS

	Name	Partner
Contributors	Sofianna Menesidou, Etnso Veliou, Thanassis Giannetsos	UBIT
Contributors	Chrysostomos Symvoulidis, Stella Dimopoulou	BYTE
Contributors	Thanos Kiourtis	UPRC
Reviewers	Marcel Klötgen	FRAU
Reviewers	Alessio Graziani	ENG

LOG TABLE

Version	Date	Change	Author	Partner
0.1	03-11-2020	First draft of ToC	Entso Veliou, Sofianna Menesidou	UBIT
0.2	05-11-2020	Introduction, Scope of the document, Intended audience, Structure of the document	Sofianna Menesidou	UBIT
0.3	11-11-2020	Updated ToC	Sofianna Menesidou	UBIT
0.4	20-11-2020	Mapping the security and privacy related requirements to implementation APIs	Sofianna Menesidou	UBIT
0.5	23-11-2020	Mapping the security and privacy related requirements to implementation APIs	Sofianna Menesidou	UBIT
0.6	25-11-2020	Design of HR security and privacy libraries in R2D	Sofianna Menesidou, Entso Veliou	UBIT
0.7	01-12-2020	Design of HR security and privacy libraries in D2D	Thanos Kiourtis	UPRC
0.8	02-12-2020	Design of HR security and privacy libraries in D2D	Sofianna Menesidou, Entso Veliou	UBIT

0.9	07-12-2020	Mapping the security and privacy related requirements to implementation APIs	Chrysostomos Symvoulidis	BYTE
1.0	09-12-2020	Design of HR security and privacy libraries in RDS Variant #1 (pseudo-identities)	Stella Dimopoulou	BYTE
1.1	15-12-2020	Design of HR security and privacy libraries in RDS Variant #2 (pseudonyms)	Sofianna Menesidou, Entso Veliou	UBIT
1.2	18-12-2020	Implementation	Sofianna Menesidou	UBIT
1.3	08-01-2020	Introduction, Updates with respect to previous version (if any)	Sofianna Menesidou	UBIT
1.4	12-01-2021	Conclusions	Sofianna Menesidou	UBIT
1.5	15-01-2021	Design of HR security and privacy libraries in RDS Variant #1 (pseudo-identities) updates	Stella Dimopoulou	BYTE
1.6	23-01-2021	Design of HR security and privacy libraries in R2D updated	Sofianna Menesidou	UBIT
1.7	25-01-2021	Design of HR security and privacy libraries in R2D update	Sofianna Menesidou	UBIT
1.8	05-02-2021	Appendix A	Sofianna Menesidou	UBIT
1.9	10-02-2021	Mapping the security and privacy related requirements to implementation APIs	Sofianna Menesidou	UBIT
2.0	19-02-2021	Appendix A	Sofianna Menesidou	UBIT
2.1	01-03-2021	Internal review.	Marcel Klötgen	FRAU
2.2	09-03-2021	Internal discussion/review updates	Alessio Graziani, Sofianna Menesidou, Etnso Veliou, Thanassis Giannetsos	ENG, UBIT
2.3	16-03-2021	Quality check	Argyro	UPRC

			Mavrogiorgou	
2.4	02-04-2021	Appendix B	Sofianna Menesidou	UBIT
2.5	23-04-2021	Additional technical review	Francesco Torelli	ENG
vFinal	21-04-2021	Final check and submission	Laura Pucci	ENG

ACRONYMS

Acronym	Term and definition
ABAC	Attribute-based Access Control
BLE	Bluetooth Low Energy
CA	Certificate Authority
CN	Central Node
D2D	Device to Device
DH	Diffie Hellman
HO	Health Organisation
HR	Health Record
HCP	Healthcare Professional
HSM	Hardware Security Storage Module
KDF	Key Derivation Function
MDS	Minimum Data Set
PI	Principal Investigator
PP	Pseudonym Provider
QSCD	Qualified Signature Creation Device
RDD	Research Definition Document
RDS	Research Data Sharing
RNG	Random Number Generator
RRC	Reference Research Center
R2D	Remote to Device
SAML	Security Assertion Markup Language
TPM	Trusted Platform Module

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the document

The security protocols and the corresponding libraries specify security schemes exploited by all the envisioned InteropEHRate protocols. They are intended to satisfy the security goals, and the necessary technical measures needed for enhanced "security and privacy-by-design", following the current standards as defined in the ENISA's Minimum Security Measures for Operators of Essentials Services **[ENISA 2020]** and the requirements of the healthcare domain **[D2.2]**.

The main goal of this document is to describe the second version of the design of the security libraries offered by the InteropEHRate Framework as a reference implementation of the HR security and privacy services. The current document outlines among others the most important design features addressing identity management, consent management, encryption and privacy based on state of the art crypto primitives. In addition, it includes the interfaces and the Java methods offered by the security libraries. At this stage of project implementation, the deliverable aims at depicting the major features and principles of designing the security libraries as well as the satisfied user requirements. A top-down approach was followed in order to identify the internal interactions for the D2D, R2D Access, R2D Backup, R2D Emergency and RDS protocols.

In more detail, this deliverable describes the security libraries for D2D (M-D2D-SM and T-D2D-SM), for R2D protocols (M-R2D-SM and T-R2D-SM) and for RDS (M-RDS-SM and T-RDS-SM) including the related external components. All security libraries are Java-based. In general, the security libraries invoked by the S-EHR App, the HCP Web App, the S-EHR Cloud, the reference Research Centre (RRC), the Central Node (CN) and other InteropEHRate libraries. Similar to other reports of the InteropEHRate project, this document presents the second version of the design of the libraries offered by the InteropEHRate Framework as a reference implementation of Health Record (HR) security and privacy services. Figure 1 depicts how this deliverable is related to the other deliverables of the project.

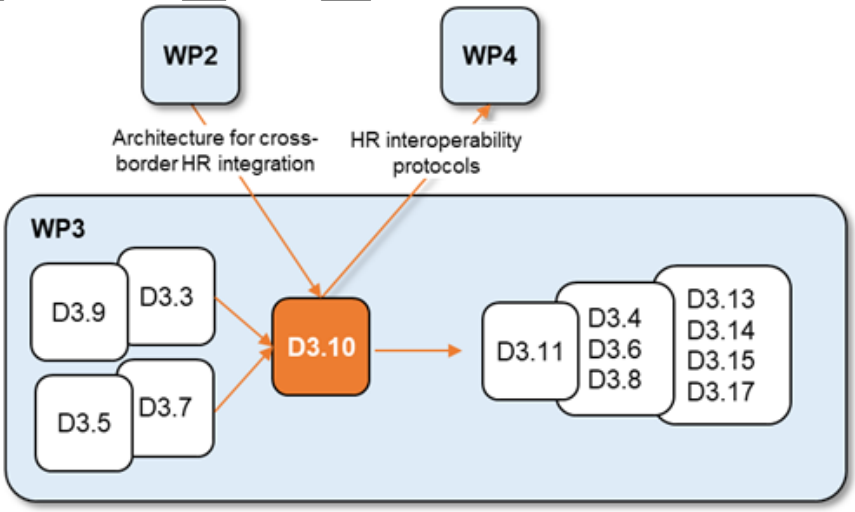

Figure 1 - Relation with other deliverables

1.2.Intended audience

This deliverable is intended primarily for a technical audience, interested in implementing the security functionalities of the InteropEHRate protocols. More specifically, the document is intended to security engineers, developers, architects, and all the InteropEHRate project participants and partners interested to have an overview of how InteropEHRate will support HR security and privacy services. These services will be described as libraries for mobile and application developers who desire to exploit and reuse the security functionalities offered by the InteropEHRate framework.

1.3.Structure of the document

The current document is organized in the following Sections:

Section 1 introduces the overall concept of the document, defining its scope, intended audience, relation to the other project tasks and reports as well as the main updates with respect to the previous version.

Section 2 describes the mapping between the user requirements, introduced in the Architecture of the InteropEHRate project, the security and privacy requirements (since it is not always straightforward what is implied by the user requirements), as well as the implementation APIs of the security libraries .

Section 3 focuses on the design of the security libraries of the D2D protocol, including how the identity management, encryption, consent management and authorization aspects are handled.

Section 4 focuses on the design of the security libraries of the R2D-family protocols, including how the identity management, encryption, consent management and authorization aspects are handled. This section includes the design of security and privacy aspects of the three different R2D protocols namely R2D Access, R2D Backup and R2D Emergency.

Section 5 focuses on the design of the security libraries of the RDS protocols, including how the identity management, encryption, consent management and authorization aspects are handled. The design of libraries includes both the two proposed variants.

Section 6 concludes the document, including future research, developments and summarises the updates of the security libraries.

Appendix A focuses on the implementation aspects, where the different security libraries and interfaces per component and per protocol including the provided APIs are summarised for better overview on the security libraries.

Appendix B describes the implementation aspects of the login interface, in the context of the R2D Access protocol, focusing on the Trusted Proxy Server implementation that will be part of the Healthcare EHR. The Trusted Proxy Server acts as a supplementary service to assist Healthcare EHR developers on an easy and straightforward integration with the eIDAS infrastructure.

1.4.Updates with respect to previous version (if any)

This document updates and supersedes the first version of the deliverable. Major updates are provided in this document such as security aspects regarding the encryption mechanisms and the RDS protocol, that were not included in the previous version. In addition, this version includes a detailed implementation

information for better overview on the implementation aspects. Also, the security properties that the protocols aim at have been defined, in order to be analysed in the final version of the deliverable. Last but not least the deliverable provides the updated user-requirements and how these are mapped to security and privacy aspects and hence to the final APIs.

2. MAPPING THE USER REQUIREMENTS TO THE SECURITY AND PRIVACY REQUIREMENTS

The main goal of this section is to highlight the security and privacy requirements necessary to implement the InteropEHRate framework considering also the level of maturity of the solutions within the countries. This section describes the mapping between the HR security and privacy requirements and the identified user requirements. The user requirements have already been identified in the Architecture of the InteropEHRate project. These can be summarized as user and data privacy, confidentiality and access control, integrity and authenticity, availability, traceability and non-repudiation which, as explained below, are achieved through several state-of-the-art technical measures (see Table 4 in **[D3.1]**). Table 1 lists all the security and privacy related user requirements. In general, the user requirements identified to fulfil the use cases are related to (a) assure the security of the service (like for example identification, authentication or patient consent and encryption) and (b) access the information from/to another country (in a cross-border context).

The security and privacy implementations targeting the Citizen as a main actor are the M-D2D-SM, M-R2D-SM, M-RDS-SM components, the security implementations targeting the HCP as a main actor are the T-D2D-SM, T-R2D-SM components and the security implementation targeting and the Researcher as a main actor is the T-RDS-SM component. The next sections will provide more details on the implementation aspects. The following table (Table 1) summarises all these requirements, with detailed descriptions according to the users, the scenario, the component, the security requirements that are mapped and the detailed APIs offered by the libraries. In general, twenty eight user requirements have been extracted from all user requirements that are directly or indirectly related to the security and privacy aspects. Six of them are related to all the three scenarios, six to the Medical Visit scenario, eight to the Research scenario and eight to the Emergency scenario.

#	User Requirement	SW Application	Scenario	Security Requirements and APIs
1	Enabling of Citizen identification from S-EHR	S-EHR Mobile App	All	**Citizen Identification**. It ensures citizen identification. fetchCertificate, sendSEHRCertificate, signPayload, verifySignature, createKeyStore, loadKeyStore
		The S-EHR stores and sends to the HCP App the identification data of the citizen (the identification data allows the HCP to confirm the identity of the citizen by comparing them with the ID card of the Citizen).		

2	Enabling of HCP identification from HCP app	HCP App	All	**HCP Identification**. It ensures HCP identification. fetchCertificate, sendHCPCertificate, signPayload, verifySignature
		The HCP app sends to the S-EHR a description of the identity of the HCP.		
3	Enabling of healthcare organization identification from HCP app	HCP App	All	**HO Identification**. It ensures HO identification. fetchCertificate, verifySignature
		The HCP app sends to the S-EHR a description of the identity of the HCP.		
4	D2D authorization to download and upload S-EHR data from HCP App	S-EHR Mobile & HCP App	Medical Visit	**Citizen Consent.** It ensures lawful processing of personal data. **Confidentiality**. It ensures secure and confidential communication and processing of data. signPayload, verifySignature, aliceInitKeyPair, aliceKeyAgreement, alicePubKeyEnc, aliceKeyAgreementFin, generateSymmetricKey, encrypt, decrypt, bobInitKeyPair,bobKeyAgreement, bobKeyAgreementFin, bobPubKeyEnc, generateSymmetric, encrypt, decrypt
		Download and upload of health data on S-EHR from an authorized HCP App is possible only if the Citizen's consent is valid and includes the specific operation performed by the HCP. If consent is not valid, a new consent request should be triggered in the S-EHR.		
5	Consent to store Citizen's data	HCP App	Medical Visit	**Citizen Consent.** It ensures lawful processing of personal data. signPayload, verifySignature
		Citizen's data can be stored by authorized HCP App and only until the Citizen's consent expires.		
6	Non repudiable data provenance tracking	S-EHR Mobile & HCP App	All	**Non-repudiation**. It ensures non-repudiation of the exchanged data. **Provenance tracking**. It ensures where each piece of data comes from and whether it is still up-to-date. signPayload, verifySignature
		The author and data origin of any health data is verified (i.e. non repudiable), tracked, visible to any authorized user and legally valid.		

7	Integrity of medical information	S-EHR Mobile App & HCP App	All	**Integrity**. It ensures integrity of the stored exchanged data. encrypt, decrypt, signPayload, verifySignature
		Users are guaranteed that the managed health data (stored or transferred) hasn't been modified maliciously or accidentally.		
8	R2D import of (portion of) Patient Summary from national health care system on S-EHR (with security)	S-EHR App	Medical Visit	**Citizen Identification**. It ensures citizen identification. login, logout, isAuthenticated
		Citizen health data (portion of Patient Summary) can be imported from the Citizen national health care system on Citizen S-EHR.		
9	Enabling of Citizen identification from S-EHR (with CA)	S-EHR Mobile App	Medical Visit	**Citizen Identification**. It ensures citizen identification. fetchCertificate, sendHCPCertificate, signPayload, verifySignature, createKeyStore, loadKeyStore
		The S-EHR asks the Citizen and stores on the device a qualified certificate that identifies the Citizen. The certificate is released by a CEF eID trusted certification authority.		
10	Enabling of HCP identification from HCP app (with CA)	HCP App	Medical Visit	**HCP Identification**. It ensures HCP identification. fetchCertificate, sendHCPCertificate, signPayload, verifySignature
		The S-EHR asks the HCP and stores on the HCP app a certificate that identifies the HCP. The certificate is released by a CEF eID trusted certification authority.		
11	Enabling of healthcare organization identification from HCP app (with CA)	HCP App	Medical Visit	**HO Identification**. It ensures HO identification. fetchCertificate, verifySignature
		The healthcare organization obtains a qualified certificate (release by a CEF eID trusted certification authority) that is stored on HCP app		
12	Digital signature by Reference Research Centre of Citizen's consent	S-EHR App	Research	**Citizen Consent.** It ensures lawful processing of personal data. getCertificate, signPayload
		The Citizen receives on his/her S-EHR a digital copy of the consent (to participate in a research he/she has been invited to) signed at the Reference Research Centre. The digital copy is digitally signed by the Research Centre using a legal binding mechanism.		

13	Citizen's digital signature of consent to share health data for a given study	S-EHR App	Research	**Citizen Consent.** It ensures lawful processing of personal data. getCertificate, retrievePseudonym, retrievePseudoIdentity, signPayload, login, logout, isAuthenticated
		The Citizen can give his/her consent to participate in a research he/she has been invited to and digitally sign it (in a legally binding way) directly on his/her S-EHR.		
14	Citizen's digital revocation of consent to share health data for a given study	S-EHR App	Research	**Citizen Consent.** It ensures lawful processing of personal data. signPayload
		A Citizen may revoke, directly from his/her S-EHR, a consent previously released to participate in a medical research. The revocation of the consent using the S-EHR must be legally binding for the Reference Research Centre also in case the S-EHR does not support the digital signature of the consent by the citizen directly on the S-EHR.		
15	Automatic anonymization and sharing of citizen's health data for research.	S-EHR App + InteropEHRate Research Services	Research	**Privacy**. It ensures the privacy of the data owner. **Citizen Consent.** It ensures lawful processing of personal data. setPseudo, anonymizeData, pseudonymizeData retrievePseudoIdentity, retrievePseudonym
		After a citizen accepts an invitation to a research study and the study is started according to the specified protocol, the S-EHR automatically queries its content for the data required by the study, once or periodically, depending on the study, and automatically sends the matching data to the IEHR Research Network. The S-EHR anonymises the data before sharing it, if required by the protocol.		
16	Activation of automatic backup of S-EHR content on selected S-EHR Cloud	S-EHR Mobile App & S-EHR Cloud	Emergency	**Confidentiality**. It ensures secure and confidential communication and processing of data. encrypt
		Citizens can activate by means of explicit consent the automatic backup, of all the health records stored on their S-EHR, on their preferred S-EHR Cloud service (selected by the list of certified S-EHR Cloud services provided by the S-EHR).		
17	Sharing of health data with qualified HCPs for emergency by means of S-EHR Cloud	S-EHR Mobile App & S-EHR Cloud	Emergency	**Citizen Consent.** It ensures lawful processing of personal data. signPayload, verifySignature
		Citizens can consent HCPs of Healthcare organisations to access, only for emergency reasons, to their health data stored on the S-EHR cloud. Giving the consent activates the automatic backup of the health data from the S-EHR to the preferred S-EHR Cloud (selected by the list of certified S-EHR Cloud services provided by the S-EHR). The consent authorises the HCP to access health data using an emergency token or the identification data of the citizen.		

18	Citizen's consent to be part of InteropEHRate Open Research Network	S-EHR Mobile App	Research	**Citizen Consent.** It ensures lawful processing of personal data. signPayload
		Using their S-EHR, and signing a digital consent, citizens can become part of the InteropEHRate Open Research Network. From this moment the S-EHR will receive the details of new research studies and will be authorised to match the health data of the citizen with the enrolment criteria of the study (without sending any health data to any party).		
19	Citizen's consent to share health data for a research protocol	S-EHR App & InteropEHRate Research Services	Research	**Citizen Consent.** It ensures lawful processing of personal data. signPayload, verifySignature
		Using the S-EHR a Citizen may give an electronic consent to participate in a specific research protocol, so accepting the condition described within the formal published specification of that research protocol. The electronic consent will be successively signed on paper by the citizen.		
20	Pseudoidentity restricted to single research protocol	S-EHR Mobile App	Research	**Privacy**. It ensures the privacy of the data owner. setPseudo, anonymizeData, pseudonymizeData retrievePseudoIdentity, retrievePseudonym
		When a citizen gives a digital consent to participate in a research protocol, a specific pseudo-id for that patient will be generated, to be used only for the pseudonymization of data shared within that specific research protocol.		
21	Citizen's access to emergency token	S-EHR App	Emergency	**Availability**. It ensures availability. generateSymmetricKey, signPayload
		Citizens may use their S-EHR to access and exchange with other applications an image containing their "emergency token". The emergency token allows a qualified HCP (authorised by his/her organization) to identify the Citizen and access in case of emergency to his/her health data stored on the S-EHR Cloud, also if the Citizen is unable to provide his/her identity or if the S-EHR is not available. To this end, the Citizen will have to print, preferably on a medal or bracelet, and wear the emergency token provided by the S-EHR.		
22	Encryption of S-EHR content exchanged with S-EHR Cloud.	S-EHR App	Emergency	**Confidentiality**. It ensures secure and confidential communication and processing of data. generateSymmetricKey, encrypt
		Every data sent by a S-EHR to the S-EHR Cloud is encrypted by the S-EHR, before of the transmission, with a private key unknown to the S-EHR Cloud provider, so that the S-EHR Cloud provider cannot decrypt any stored data, but only the Citizen and the HCP can.		

23	Encryption of health data written by HCP on S-EHR Cloud.	HCP App	Emergency	**Confidentiality**. It ensures secure and confidential communication and processing of data. verifySignature, encrypt
		Every data sent by an HCP App to the S-EHR Cloud is encrypted by the HCP App, before of the transmission, with a private key unknown to the S-EHR Cloud provider, so that the S-EHR Cloud provider cannot decrypt any stored data, but only the Citizen and the HCP can.		
24	Legal identification and authentication of qualified HCPs	HCP App + S-EHR Cloud	Emergency	**HCP Identification and authentication**. It ensures HCP identification and authentication. getCertificate, login, logout, isAuthenticated
		Only HCP belonging to a recognised Healthcare organisation can access emergency data. Each Healthcare organisation is identified by a digital identity issued by a legal national or local authority recognised by the S-EHR Cloud and that assures the Healthcare role of the organisation.		
25	Legal identification and authentication of qualified Healthcare organisations	HCP App + S-EHR Cloud	Emergency	**HO Identification and authentication**. It ensures HO identification and authentication. getCertificate, login, logout, isAuthenticated
		Only HCPs having a digital identity issued by a legal national or local authority recognised by the S-EHR Cloud can access (for emergency reasons) the health data of the Citizen.		
26	Authorisation to the healthcare team for emergency	HCP App + S-EHR Cloud	Emergency	**HCP Identification.** It ensures HCP identification and authentication. **Access Control**. It ensures secure and confidential communication and processing of data. getCertificate, login, logout, isAuthenticated
		When a qualified HCP gains access to the health data for emergency reasons, also the rest of the healthcare team that treats that patient for that specific emergency encounter automatically gains access to the same health data.		
27	Identification and authorisation of organisations and researchers accessing to IRS	InteropEHRate Research Services	Research	**HO Identification**. It ensures HO identification. **Access Control**. It ensures secure and confidential communication and processing of data. getCertificate, login, logout, isAuthenticated
		Only authorised researchers belonging to organisations belonging to the InteropEHRate Research Network may access specific functionalities, they are authorised for, on the Open Research Network.		

28	Encryption of health data written on S-EHR App	S-EHR App	All	**Confidentiality**. It ensures secure and confidential communication and processing of data. loadKeyStore, encrypt, decrypt
		The personal data of the Citizen is stored on the mobile device by the S-EHR App in an encrypted format, decryptable only by the citizen, to avoid any unauthorized access on the data.		

Table 1 - Security and Privacy related User Requirements

3. DESIGN OF HR SECURITY AND PRIVACY LIBRARIES IN D2D

This section emphasizes on the calls of the security and privacy libraries focused on Device to Device (D2D) and describes the way they operate, their outputs and implementation details. The D2D protocol defines the set of operations and the exchanged messages that allow the exchange of health data between a S-EHR App and a near HCP App without the usage of internet. The security libraries of the D2D protocol provide the necessary security functionalities for identity management, consent management, authentication and encryption for data storage and in transit. Before digging into details of the protocols, first there is a need to flesh out the exact security properties for the D2D protocol. A detailed security analysis will be provided in the last version of this deliverable.

- **P-1 Confidentiality.** The term 'confidentiality' means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information **[**NIST 2003**]**.

- **P-2 Integrity.** The term 'integrity' means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity **[**NIST 2003**]**.

- **P-3 Authentication.** Security measures designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information **[**NIST 2003**]**.

- **P-4 Freshness.** A freshness value must have the property that it can be guaranteed not to have been used before. There are three common types of freshness values used: timestamps, nonces and counters **[**BOYD 2003**]**.

- **P-5 Immutability.** The impossibility to tamper with the protocol's participant actions **[**SCHIEDERMEIER 2019**]**. Therefore, the exchange of messages must be immutable in such a way that the parties always receive the correct messages and hence the parties are not compromised.

- **P-6 Protocol Correctness.** The protocol logic is sound. Each assertion about an action or sequence of actions holds in any run of the protocol, under attack, in which the given actions occur is provable **[**DURGIN 2002**]**.

The main novelty regarding security is the instantiation of appropriate models leveraging two supported variants for secure and authenticated Identity Management. The first variant is linked to the ID-Card of the citizen and a QR code, generated by the hospital, which provides stronger (physical) security properties and demonstrates high feasibility and applicability features, as a possible enabler to be put immediately into practice after the end of the project. This variant, however, assumes the user authentication through physical presence and the use of identifiable documents (ID card) which might hinder its scalability.

Compounding this issue, the second variant proposes to leverage citizen's Qualified Digital Signatures. A qualified electronic signature is an advanced electronic signature with a qualified digital certificate that has been created by a qualified signature creation device (QSCD). This variant overcomes the (aforementioned) scalability issues, however, it is based on the use of trusted computing technologies where a decentralized

"root-of-trust" (e.g., Hardware Security Storage Module (HSM), Trusted Platform Module (TPM), etc.) needs to be attached to the user's end device. While the integration of such advanced trusted computing technologies provides confidence in a system, especially if the system's behaviour isn't fully secure or might become insecure, thus requiring verifiable evidence on the correct execution of the security protocols by the system (provided by the "root-of-trust" crypto signing operations), it adds additional deployment costs. Therefore, the goal is the adoption of such solutions when the smart-phone technology will be mature enough for supporting qualified digital signatures through appropriate hardware- or software-based roots-of-trust.

Apart from the usage of digital signatures for identification, such primitives were also leveraged for signing the citizen's consent. In addition, as it pertains to confidentiality, state-of-the-art key agreement protocols were leveraged based on the use of the Diffie Hellman (DH) scheme and strong Pseudo-Random Number Generators (RNG), exhibiting high entropy, thus, enabling the provision of strong security levels, tailored to Bluetooth as the underlying network mechanism. In general, key agreement protocols are "fairer" than key transport and can result in higher quality random keys than key transport, while by basing key agreement on the Diffie Hellman protocol, forward secrecy can be achieved **[BOYD 2003]**. This is also in alignment with the currently proposed Bluetooth Low Energy (BLE) standard.

More specifically, this section provides the design of the security libraries that can be used by any S-EHR app, HCP App and D2D libraries. In addition, the description of the Public Java Components contained in each library is defined, including a description of the offered interfaces of those components, while the description of the interactions of the internal components is also described.

3.1. D2D Implementation

This section includes the components and the public interfaces for the security libraries M-D2D-SM and T-D2D-SM. The component names are based on the security functionalities offered. These components are also offering specific interfaces that are analysed in this section. The objective of the libraries is to allow the usage of D2D without the need for developers to know all the technical details of the underlying D2D protocol and technologies. The M-D2D-SM and T-D2D-SM libraries act as a proxy for a health care system compliant to D2D protocol specifications.

3.1.1. Components

S-EHR-side M-D2D-SM Components: The M-D2D-SM library incorporates a set of components (Figure 2) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), including the: (a) Identity Management, where the exchange the exchange certificates is taking place, (b) Consent Management, that includes the signature, the validation and the timestamp of the consent, (c) Encrypted Communication, where the Diffie Helman key agreement is used to establish an AES256 symmetric key for the actual encryption and (d) Encrypted Storage, where the data are stored in encrypted form using the AES256 algorithm too.

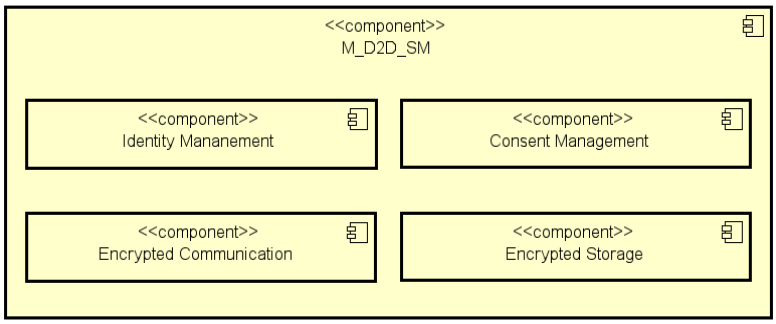

Figure 2 - M-D2D-SM Public Java Components

HCP-side T-D2D-SM Components: The T-D2D-SM library incorporates a set of components (Figure 3) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), with the same aforementioned functionalities in the context of M-D2D-SM library.

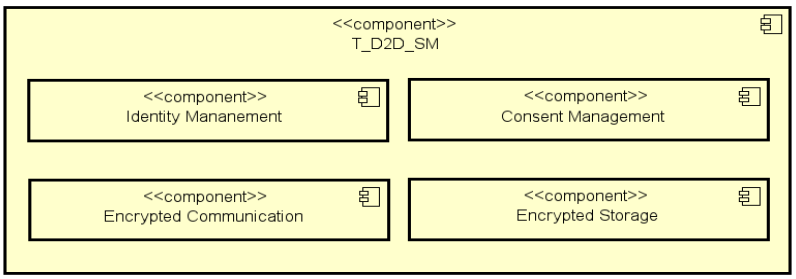

Figure 3 - T-D2D-SM Public Java Components

3.1.2. Public Interfaces

MD2DI-Security: MD2DI-Security is the name of the interface that is offered by the Mobile D2D Security Management component (M-D2D-SM), containing the operations for letting the S-EHR app and M-D2D-E interact with the M-D2D-SM library and finally perform necessary security functionalities by invoking these operations (Figure 4).

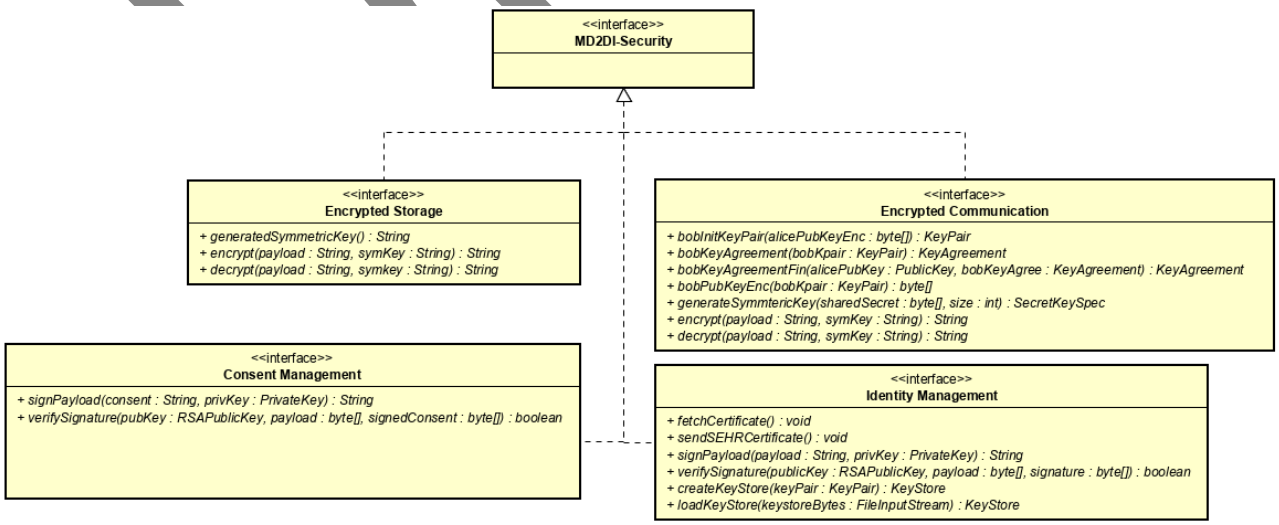

Figure 4 - MD2DI-Security Public Java Components Interfaces

Identity Management / Operation fetchCertificate

Name	fetchCertificate
Description	This call generates an X.509 certificate signed by the citizen's CA upon a CSR request. This certificate is received and stored to the Android keystore of the mobile device. When generating or importing a key into the Android keystore the key will be used if the user has been authenticated first in the device. Once keys are in the keystore, they can be used for cryptographic operations with the key material remaining non-exportable. The Android keystore allows access to certificates and keys from PKCS12 files. This operation is invoked by the S-EHR App.
Arguments	• String name: Full name of the owner • String organisation: Organisation he/she belongs (optional) • String country: Nationality (optional) • String uid: Unique identifier
Return Value	• void

Exceptions	• In the first version of the library implementation, certificates will be generated and self-signed, and the following exceptions will be emitted:
	  ○ NoSuchProviderException, in cases the KeyPairGenerator not able to identity the provider for key pair generation (e.g.: "BC")
	  ○ NoSuchAlgorithmException, in cases the KeyPairGenerator not able to identity the algorithm for key pair generation (e.g. RSA)
	• In the final version with the actual CA utilisation this will be replaced by a general Exception:
	  ○ Exception, in case of signal error or an unknown error

Preconditions	• Internet Connection • Public/Private key generation and storage in Android keystore

Identity Management / Operation sendSHERCertificate

Name	sendEHRCertificate
Description	After the bluetooth pairing establishment, the first message should be the transfer of S-EHR public key (certificate). Such a message is necessary for the HCP App to be able to validate the S-EHR signature for identification purposes.

	This operation is invoked by D2D library to transfer the HCP public key.
Arguments	• void
Return Value	• void
Exceptions	• Exception
Preconditions	• Bluetooth Pairing has taken place

Identity Management / Operation signPayload

Name	signPayload
Description	The signature algorithm used is the RSA with SHA-256. The call will use the private key stored in the keystore to initialize the signing operation. This operation is invoked by S-EHR App.
Arguments	• String payload • PrivateKey privateKey
Return Value	• String - Returns the signature of the payload of the Consent
Exceptions	• IOException, • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException
Preconditions	• HCP has successfully fetched his/her credentials

Identity Management / Operation verifySignature

Name	verifySignature
Description	This verifies that the HCP has signed the consent with their digital signature and the citizen verifies and acknowledges the data transfer. This operation is invoked by S-EHR App.
Arguments	• RSAPublicKey publicKey • byte[] payload • byte[] signature
Return Value	• boolean - true if the signature was verified, false if not.
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• HCP has requested the consent

Identity Management / Operation createKeyStore

Name	createKeyStore
Description	Creates a .jks file. A Java Keystore is a container for authorization certificates or public key certificates, and is often used by Java-based applications for encryption, authentication, and serving over HTTPS.
Arguments	• KeyPair keyPair
Return Value	• KeyStore keystore
Exceptions	• N/A
Preconditions	• A keypair is required

Identity Management / Operation loadKeyStore

Name	loadKeyStore
Description	Loads the created keystore in the memory of the application
Arguments	• FileInputStream keystoreBytes
Return Value	• Keystore keystore
Exceptions	• KeyStoreException • CertificateException • NoSuchAlgorithmException • IOException
Preconditions	• Keystore must have been created first

Consent Management / Operation signPayload

Name	signPayload
Description	The signature algorithm used is the RSA with SHA-256. The call will use the private key stored in the keystore to initialize the signing operation. This operation is invoked by the HCP App.
Arguments	• Payload payload • Privkey privkey
Return Value	• String - Returns the signature of the payload in as String form
Exceptions	• IOException, • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException

Preconditions	• HCP has successfully fetched his/her credentials

Consent Management / Operation verifySignature

Name	verifySignature
Description	Verifies the signature of the payload
Arguments	• RSAPublicKey publicKey • byte[] payload • byte[] signature
Return Value	• boolean
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• Payload should be signed first

Encrypted Communication / Operation bobInitKeyPair

Name	bobInitKeyPair
Description	S-EHR App creates his own DH key pair.
Arguments	• byte[] alicePubKeyEnc
Return Value	• KeyPair bobKpair
Exceptions	• Exception

Preconditions	• Successful generation of public key

Encrypted Communication / Operation bobKeyAgreement

Name	bobKeyAgreement
Description	S-EHR App creates and initializes a DH KeyAgreement object.
Arguments	• KeyPair bobKpair
Return Value	• KeyAgreement bobKeyAgree
Exceptions	• Exception
Preconditions	• Successful generation of Diffie Hellman key agreement object

Encrypted Communication / Operation bobKeyAgreementFin

Name	bobKeyAgreementFin
Description	S-EHR App generates the (same) shared secret.
Arguments	• PublicKey alicePubKey, • KeyAgreement bobKeyAgree
Return Value	• KeyAgreement keyagreement
Exceptions	• Exception

Preconditions	• Successful generation of Diffie Hellman key agreement

Encrypted Communication / Operation bobPubKeyEnc

Name	bobPubKeyEnc
Description	S-EHR App encodes his public key, and sends it over to the HCP App.
Arguments	• KeyPair bobKpair
Return Value	• byte[] bobPubKeyEnc
Exceptions	• Exception
Preconditions	• Key pair successfully created

Encrypted Communication / Operation generateSymmetricKey

Name	generateSymmetricKey
Description	At this stage, both S-EHR App and HCP App have completed the DH key agreement protocol. Both generate the (same) shared secret. AES session key that will be used for encrypted communication between the S-EHR App and HCP App.
Arguments	• byte[] sharedSecret, • int size
Return Value	• SecretKeySpec symKey
Exceptions	• NoSuchAlgorithmException

Preconditions	• The same secret is generated

Encrypted Communication / Operation encrypt

Name	encrypt
Description	Transmitting an encrypted message from S-EHR App to HCP App. S-EHR Appencrypts, using AES in CBC mode.
Arguments	• String payload, • String symKey
Return Value	• String cipher
Exceptions	• Exception
Preconditions	• Symmetric key agreement established

Encrypted Communication / Operation decrypt

Name	decrypt
Description	S-EHR App decrypts, using AES in CBC mode
Arguments	• String payload, • String symKey
Return Value	• String plaintext
Exceptions	• Exception

Preconditions	• Symmetric key agreement established

Encrypted Storage / Operation generateSymmetricKey

Name	generateSymmetricKey
Description	At this stage, both S-EHR App and HCP App have completed the DH key agreement protocol. Both generate the (same) shared secret. AES session key that will be used for encrypted communication between the S-EHR App and HCP App.
Arguments	• void
Return Value	• String symKey
Exceptions	• NoSuchAlgorithmException
Preconditions	• N/A

Encrypted Storage / Operation encrypt

Name	encrypt
Description	Encryption of files locally on the storage available to the application with AES
Arguments	• String payload, • String symKey
Return Value	• String cipher
Exceptions	• Exception

Preconditions	• Symmetric key agreement generated

Encrypted Storage / Operation decrypt

Name	decrypt
Description	Decryption of files locally on the storage available to the application with AES
Arguments	• String payload, • String symKey
Return Value	• String plaintext
Exceptions	• Exception
Preconditions	• Symmetric key agreement generated

TD2DI-Security: MT2DI-Security is the name of the interface that is offered by the Terminal D2D Security Management component, containing the operations for letting the HCP app and T-D2D-E interact with the T-D2D-SM library and finally perform necessary security functionalities, by invoking these operations (Figure 5).

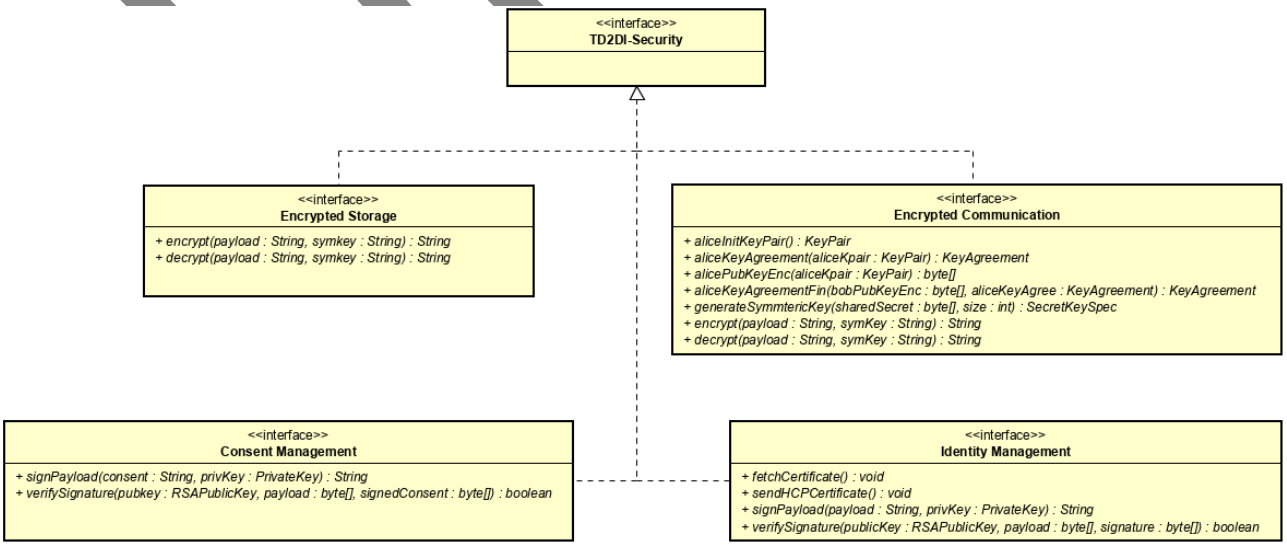

Figure 5 - TD2DI-Security Public Java Components Interfaces

Identity Management / Operation fetchCertificate

Name	fetchCertificate
Description	This call generates an X.509 certificate from the HCP's CA upon a CSR request. This certificate is received by and stored to the keystore of the HCP's device. In Java 9, the default keystore type will be changed to PKCS12, while in earlier versions this was the Java Key Store (JKS). PKCS12 is a file format to store certificates and private keys. The KeyStore API in Java also allows to access certificates and keys from PKCS12 files. This operation is invoked by HCP App.
Arguments	• String name: Full name of the owner • String organisation: Organisation he/se belongs (optional) • String country: Nationality (optional) • String uid: Unique identifier
Return Value	• void
Exceptions	• NoSuchKeyException • KeyStoreException • CertificateException • NoSuchAlgorithmException • IOException
Preconditions	• Internet Connection • Public/Private key generation and storage in keystore

Identity Management / Operation sendHCPCertificate

Name	sendHCPCertificate
Description	After the bluetooth pairing establishment, the first message should be the transfer of HCP public key (certificate). Such a message is necessary for the S-EHR App to be able to validate the HCP signature for identification purposes. This operation is invoked by D2D library to transfer the HCP public key.
Arguments	• void

Return Value	• void
Exceptions	• Exception
Preconditions	• Bluetooth Pairing has taken place

Identity Management / Operation signPayload

Name	signPayload
Description	The signature algorithm used is the RSA with SHA-256. The call will use the private key stored in the keystore to initialize the signing operation. This operation is invoked by the HCP App.
Arguments	• String Payload • Privkey privkey
Return Value	• String - Returns the signature of the payload in as String form
Exceptions	• IOException • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException
Preconditions	• HCP has successfully fetched his/her credentials

Identity Management / Operation verifySignature

Name	verifySignature
Description	This verifies that the HCP has signed the consent with their digital signature and the citizen verifies and acknowledges the data transfer. This operation is invoked by the S-EHR App.

Arguments	• RSAPublicKey publicKey • byte[] payload • byte[] signature
Return Value	• boolean - true if the signature was verified, false if not.
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• HCP has requested the consent

Consent Management / Operation signPayload

Name	signPayload
Description	The signature algorithm used is the RSA with SHA-256. The call will use the private key stored in the keystore to initialize the signing operation. This operation is invoked by the HCP App.
Arguments	• String Payload • Privkey privkey
Return Value	• String - Returns the signature of the payload in as String form
Exceptions	• IOException • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException
Preconditions	• HCP has successfully fetched his/her credentials

Consent Management / Operation verifySignature

Name	verifySignature
Description	Verifies the signature of the payload
Arguments	• RSAPublicKey publicKey • byte[] payload • byte[] signature
Return Value	• boolean
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• Payload should be signed first

Encrypted Communication / Operation aliceInitKeyPair

Name	aliceInitKeyPair
Description	HCP App creates his own DH key pair.
Arguments	• void
Return Value	• KeyPair aliceKpair
Exceptions	• Exception
Preconditions	• S-EHR App has generated and sent its Public key

Encrypted Communication / Operation aliceKeyAgreement

Name	aliceKeyAgreement
Description	Alice gets the DH parameters associated with Bob's public key. He must use the same parameters when he generates his own keypair.
Arguments	• KeyPair aliceKpair
Return Value	• KeyAgreement aliceKeyAgree
Exceptions	• Exception
Preconditions	• Successful Diffie Hellman key pair generation

Encrypted Communication / Operation alicePubKeyEnc

Name	alicePubKeyEnc
Description	Alice encodes her public key, and sends it over to Bob.
Arguments	• KeyPair aliceKpair
Return Value	• byte[] alicePubKeyEnc
Exceptions	• Exception
Preconditions	• Key pair successfully created

Encrypted Communication / Operation aliceKeyAgreementFin

Name	aliceKeyAgreementFin
Description	HCP App generates the (same) shared secret.
Arguments	• byte[] bobPubKeyEnc, • KeyAgreement aliceKeyAgree
Return Value	• KeyAgreement keyagreement
Exceptions	• Exception
Preconditions	• Successful generation of Diffie Hellman key agreement

Encrypted Communication / Operation generateSymmetricKey

Name	generateSymmetricKey
Description	At this stage, both S-EHR App and HCP App have completed the DH key agreement protocol. Both generate the (same) shared secret. AES session key that will be used for encrypted communication between the S-EHR App and HCP App.
Arguments	• byte[] sharedSecret, • int size
Return Value	• SecretKeySpec symKey
Exceptions	• Exception
Preconditions	• The same secret is generated

Encrypted Communication / Operation encrypt

Name	**encrypt**
Description	Transmitting an encrypted message from HCP App to S-EHR App. HCP App encrypts, using AES in CBC mode.
Arguments	• String payload • String symkey
Return Value	• String cipher
Exceptions	• Exception
Preconditions	• Symmetric key agreement established

Encrypted Communication / Operation decrypt

Name	**decrypt**
Description	HCP App App decrypts, using AES in CBC mode
Arguments	• String payload • String symkey
Return Value	• String plaintext
Exceptions	• Exception
Preconditions	• Symmetric key agreement established

Encrypted Storage / Operation encrypt

Name	**encrypt**
Description	Encryption of files locally on the storage available to the application
Arguments	• String payload • String symkey
Return Value	• String cipher
Exceptions	• Exception
Preconditions	• Symmetric key agreement generated

Encrypted Storage / Operation decrypt

Name	**decrypt**
Description	Decryption of files locally on the storage available to the application
Arguments	• String payload • String symkey
Return Value	• String plaintext
Exceptions	• Exception
Preconditions	• Symmetric key agreement generated

4. DESIGN OF HR SECURITY AND PRIVACY LIBRARIES IN R2D

This section emphasizes on the calls of the security and privacy libraries focused on Remote to Device (R2D) protocols and describes the way they operate, their outputs and implementation details. The R2D protocols namely R2D Access, R2D Backup and R2D Emergency define the set of operations and the exchanged messages that allow the exchange of health data over the Internet. The security libraries of R2D protocols provide the necessary security functionalities for identity management, consent management, authentication and encryption for data storage and in transit. Before digging into details of the protocols there is a need to flesh out the exact security properties for the R2D protocols. A detailed security analysis will be provided in the last version of this deliverable.

- **P-1 Confidentiality.** The term 'confidentiality' means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information **[**NIST 2003**]**.

- **P-2 Integrity.** The term 'integrity' means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity **[**NIST 2003**]**.

- **P-3 Authentication.** Security measures designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information **[**NIST 2003**]**.

- **P-4 Authorization and Access Control.** Access control or authorization, on the other hand, is the decision to permit or deny a subject access to system object **[**NIST 2014**]**.

- **P-5 Freshness.** A freshness value must have the property that it can be guaranteed not to have been used before. There are three common types of freshness value used: timestamps, nonces and counters **[**BOYD 2003**]**.

- **P-6 Immutability.** The impossibility to tamper with the protocol's participant actions **[**SCHIEDERMEIER 2019**]**. Therefore, the exchange of messages must be immutable such that the parties always receive the correct messages and hence the parties are not compromised.

- **P-7 Protocol Correctness.** The protocol logic is sound. Each assertion about an action or sequence of actions holds in any run of the protocol, under attack, in which the given actions occur is provable **[**DURGIN 2002**]**.

The main novelty of security and privacy is that the R2D Access leverages an eIDAS-based architecture for cross-border identification/authentication of the citizen supporting the trust services and electronic identification, as defined by the current eIDAS framework. In addition, all established communication sessions are protected with the most suitable and robust encryption technologies needed to secure different types of information, while still allowing for (future) advanced knowledge discovery through the provision of enhanced data search services and advanced security and privacy-preserving primitives for authentication, authorization and data integrity verification. More specifically, all exchanged and stored information in the S-EHR Cloud leveraging the R2D Backup and R2D Emergency protocols are symmetrically

encrypted with AES-256, using encryption keys generated from a strong KDF that demonstrates high entropy and randomness **[NIST ENTR]**. The main advantage of such a mechanism is the efficiency and effectiveness provided, through the use of appropriate lightweight cryptographic primitives.

More specifically, this section provides the design of the security libraries that can be used by any S-EHR app, HCP App and R2D libraries. In addition, the description of the Public Java Components contained in each library is defined, including a description of the offered interfaces of those components, while the description of the interactions of the internal components is also provided. The following sections describe the security and privacy models in the context of R2D. More precisely, the S-EHR-side R2D security management (i.e. M-R2D-SM) and the HCP-side R2D security management (i.e. T-D2D-SM) libraries contain all the crypto operations needed from the side of the S-EHR and HCP respectively.

4.1. R2D Implementation

This section includes the components and the public interfaces for the security libraries M-R2D-SM and the T-R2D-SM. As with the D2D, the component names are based on the security functionalities offered. These components are also offering specific interfaces that are analysed in this section. The objective of the libraries is to allow the usage of R2D without the need for developers to know all the technical details of the underlying R2D concrete protocols and technologies. The M-R2D-SM and T-R2D-SM libraries act as a proxy for a health care system compliant to R2D protocol specifications.

4.1.1. Components

S-EHR-side M-R2D-SM Components: The M-R2D-SM library incorporates a set of components (Figure 6) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), including the: (a) Identity Management, where the identification and authentication over eIDAS or Keycloak is supported, (b) Consent Management, that includes the signature, the validation and the timestamp of the consent, (c) Encrypted Communication, where the Diffie Helman key agreement is used to establish an AES256 symmetric key for the actual encryption and (d) Encrypted Storage, where the data are stored in encrypted form using the AES256 algorithm. Encrypted Communication, Encrypted Storage and Consent Management offer exactly the same APIs with the M-D2D-SM and T-D2D-SM and for that reason we will omit to analyse them in new tables.

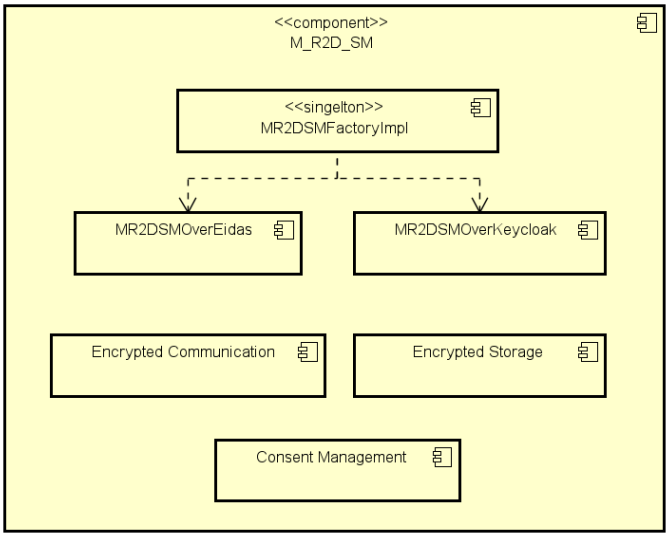

Figure 6 - M-R2D-SM Public Java Components

HCP-side T-R2D-SM Components: The T-R2D-SM library incorporates a set of components (Figure 7) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), including the: (a) Identity Management and authorization, where the exchange the exchange certificates is taking place, (b) Consent Management, that includes the signature, the validation and the timestamp of the consent, (c) Encrypted Communication, where the Diffie Helman key agreement is used to establish an AES256 symmetric key for the actual encryption and (d) Encrypted Storage, where the data are stored in encrypted form using the AES256 algorithm too. Again we omit to analyse in new tables the same APIs.

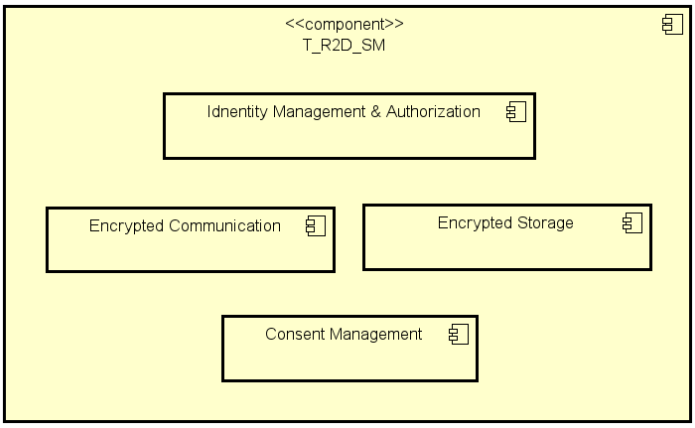

Figure 7 - T-R2D-SM Public Java Components

Trusted Proxy Server Component: The P-R2D-SM library incorporates a set of components offering different functionalities and capabilities to the R2D Access Server. More details regarding the implementation aspects of this component are described in APPENDIX B, with a detailed sequence diagram and API calls.

4.1.2. Public Interfaces

MR2DI-Security: MR2DI-Security is the name of the interface that is offered by the Mobile R2D Security Management component, containing the operations for letting the S-EHR app and M-R2D-E interact with the M-R2D-SM library and finally perform necessary security functionalities, by invoking these operations (Figure 8).

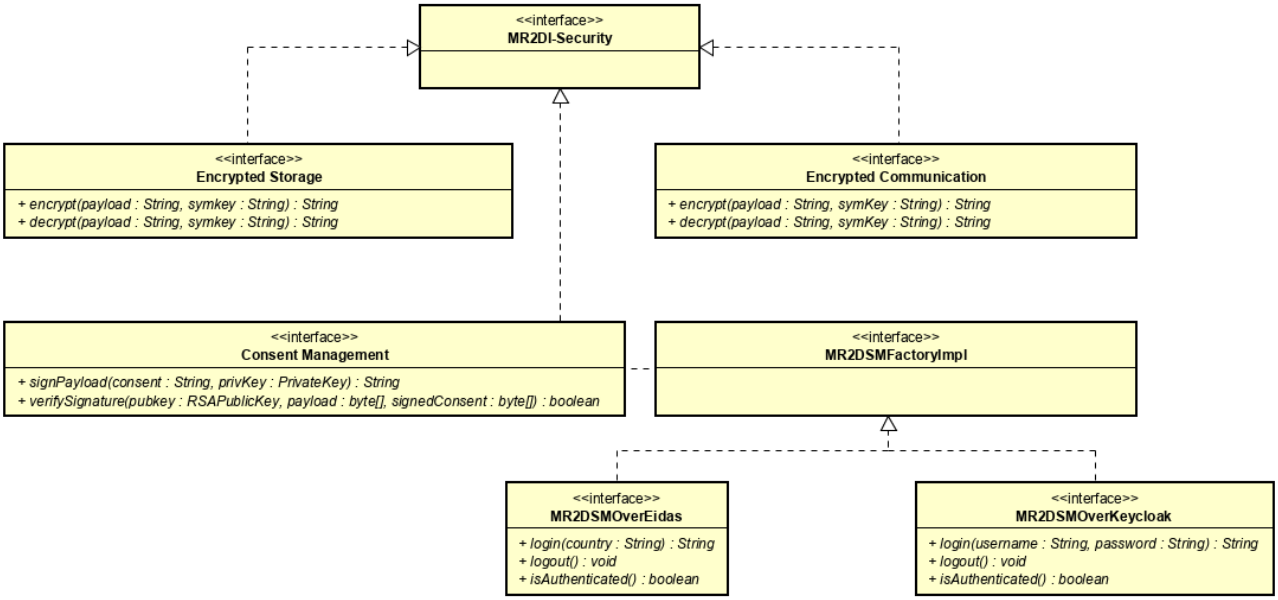

Figure 8 - MR2DI-Security Public Java Components Interfaces

R2DAccess / Operation login

Name	**login**
Description	The patient is using a secure login service provided by eIDAS provider or a trusted third party for getting an authenticated session to the InteropEHRate services. Besides secure authentication, the goal is to also provide enhanced protection of the user login credentials against malicious S-EHR App instantiations.
Arguments	• String countryA - parameter needed for redirecting to the respective eIDAS Node of the provided country, where the user through a secure login service (provided by eIDAS or a trusted third party) will get an authenticated session to the InteropEHRate services. This allows InteropEHRate to enhance protection against malicious S-EHR apps.
Return Value	• HTTP redirect with the requested records
Exceptions	• Exception

Preconditions	• eIDAS registration

R2DAccess / Operation logout

Name	**Logout**
Description	The patient will log out
Arguments	• void
Return Value	• void
Exceptions	• Exception
Preconditions	• An already active session

R2DAccess / Operation isAuthenticated

Name	**isAuthenticated**
Description	A boolean value that checks if the patient is still in an active session
Arguments	• void
Return Value	• boolean
Exceptions	• Exception
Preconditions	• Patient has authenticated through eIDAS successfully

R2DBackup / Operation encrypt

Name	**encrypt**
Description	Storage encryption of the Health Records of the patient
Arguments	• String payload • String symkey
Return Value	• String
Exceptions	• Exception
Preconditions	• Symmetric key agreement generated

TR2DI-Security: MT2DI-Security is the name of the interface that is offered by the Terminal R2D Security Management component, containing the operations for letting the HCP app and T-R2D-E interact with the T-R2D-SM library and finally perform necessary security functionalities, by invoking these operations (Figure 9).

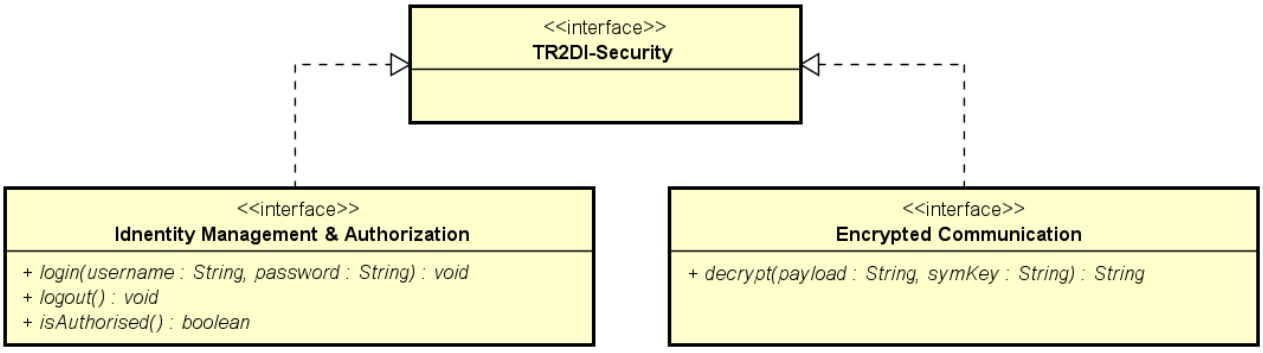

Figure 9 - TR2DI-Security Public Java Components Interfaces

R2DEmergency / Operation login

Name	**login**

Description	Healthcare professional login action as he is a part of valid healthcare organisation
Arguments	• String username • String password
Return Value	• void
Exceptions	• Exception
Preconditions	• HCP has registered to the service

R2DEmergency / Operation logout

Name	**logout**
Description	The HCP will log out from our services
Arguments	• void
Return Value	• void
Exceptions	• Exception
Preconditions	• An already active session

R2DEmergency /Operation isAuthorised

Name	**isAuthorised**
Description	A boolean value that checks if the HCP has authorization to perform the specific action

Arguments	• void
Return Value	• boolean
Exceptions	• Exception
Preconditions	• HCP has authenticated through the service successfully

R2DEmergency /Operation decrypt

Name	**decrypt**
Description	Storage decryption of the Health Records of the patient
Arguments	• String payload • String symKey
Return Value	• String
Exceptions	• Exception
Preconditions	• HCP has generated his/her key pair

5. DESIGN OF HR SECURITY AND PRIVACY LIBRARIES IN RDS

This section emphasizes on the calls of the security and privacy libraries focused on Research Data Sharing (RDS) and describes the way they operate, their outputs and implementation details. The RDS protocol addresses the general problem of collecting health data for cross-border medical research in order to enable secure and privacy-preserving cross-border data collection **[D4.8]**. The security libraries of the RDS protocol provide the necessary security functionalities for identity management, consent management, authentication, encryption and privacy for data storage and in transit, while two variants are supported for privacy (anonymisation) a) with pseudo-identities and b) with pseudonyms. Before digging into details of the protocols there is a need first to flesh out the exact security properties for the RDS protocols. A detailed security analysis will be provided in the last version of this deliverable.

- **P-1 Confidentiality.** The term 'confidentiality' means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information **[NIST 2003]**.

- **P-2 Integrity.** The term 'integrity' means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity **[NIST 2003]**.

- **P-3 Authentication.** Security measures designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information **[NIST 2003]**.

- **P-4 Freshness.** A freshness value must have the property that it can be guaranteed not to have been used before. There are three common types of freshness value used: timestamps, nonces and counters **[BOYD 2003]**.

- **P-5 Immutability.** The impossibility to tamper with the protocol's participant actions **[SCHIEDERMEIER 2019]**. Therefore, the exchange of messages must be immutable such that the parties always receive the correct messages and hence the parties are not compromised.

- **P-6 Protocol Correctness.** The protocol logic is sound. Each assertion about an action or sequence of actions holds in any run of the protocol, under attack, in which the given actions occur is provable **[DURGIN 2002]**.

- **P-7 Privacy.** Freedom from intrusion into the private life of an individual when that intrusion results from undue or illegal gathering and use of data about that individual **[ISO/IEC 2382:2015]**.

- **P-9 Unlikability.** All data processing is operated in such a way that the privacy-relevant data are unlinkable to any other set of privacy-relevant data outside of the domain, or at least that the implementation of such linking would require disproportionate efforts for the entity to establishing such linkage **[Zwingelberg 2011]**.

- **P-8 Anonymity.** Patients should not be identifiable. Observers should not be able to infer private information and whether a user performed or will perform a specific action. Moreover, no observer

should be able to link an action to the user or infer if two (or more) actions were performed by the same user (S-EHR App). Anonymity is conditional in the sense that it can be revoked when users deliberately disrupt the operation of the system or contaminate the data collection process **[SSPEAR 2014]**.

- **P-10 Non Repudiation and Accountability.** Actions should be non-repudiable and all system entities (i.e., users and infrastructure components) should be held accountable for their actions **[SSPEAR 2014]**. In addition, in the case of emergency, de-anonymization should be possible.

One of the novelties regarding security and privacy is the federated trust relationship among the eIDAS Node and the Pseudonym Provider (PP) established by means of Security Assertion Markup Language (SAML). The RDS protocol leverages an eIDAS-based architecture for cross-border identification/authentication of the citizen and the usage of pseudonyms for privacy preservation. The German BSI agency introduced several security mechanisms regarding the use of identity tokens for authentication purposes **[Bundesamt 2015]**. In such situations, a token for electronic Identification, Authentication and trust Services (e.g. eIDAS token) connects to a service provider. This idea was also adopted in our case with the PP. Pseudonymisation has an important role in GDPR as a security measure (art. 32 GDPR), as well as in the context of data protection by design (art. 25 GDPR) **[ENISA 2018]**. In terms of privacy preservation, two variants are used, one standardized with the state-of-the-art crypto primitives for enriching privacy and one with the currently adopted mechanisms by the end-users. These variants are addressed in the text below. As already stated in **[ENISA 2021]**, there is no fit-for-all pseudonymisation technique and a detailed analysis of the case is necessary. The usage of the second variant does not enhance the applicability of the InteropEHRate framework but allows to perform a detailed investigation of new privacy-preserving enablers that can extend the state of the art and be potentially considered as a new standard. Pseudonymisation can go beyond hiding real identities and data minimisation into supporting the unlinkability **[ENISA 2021]** making high entropy pseudonyms necessary. Currently three different pseudonymisation policies have been considered **[ENISA 2021]**: a) deterministic pseudonymisation, b) document randomised pseudonymisation and c) full randomised pseudonymisation. The first variant matches the document randomised pseudonymisation policy, while the second variant applies to the full randomised pseudonymisation policy. With regard to pseudonymisation policies, fully randomised pseudonymisation offers the best protection level (e.g. un-linkability).

In the first variant, we use pseudo-identities in order to replace all direct and indirect identifiers of a citizen with an alphanumeric sequence. In this way, if necessary, the citizen can be re-identified only by the Principal Investigator (PI) of the study. Such cases may be for example an emergency or an identified health issue of the citizen, identified during the research. The abovementioned sequence consists of three parts; a prefix, which is declared inside the Research Definition Document (RDD) by the organization who conducts the research, an incrementally increasing number, and a suffix. The prefix will be the same for a specific study, and it will change per study. The increasing number, that follows the prefix, will be used in order to distinguish the citizens in a study, whereas the suffix will be a random value, which will add a factor of randomness in the pseudo-id.

In addition, every citizen will have a different pseudo-id for a specific study, and by extension each citizen will have a different pseudo-id for a different study. Even though the pseudo-ids will be unique, no one,

except for the PI of the study, will be able to re-identify a citizen. After the pseudo-id is created by the PI at the Reference Research Center (RRC), it will be sent and stored locally at the citizen's phone (S-EHR Mobile Application). Afterwards, the S-EHR app will anonymize the citizen's data. The anonymized data is then sent along with the pseudo-id to the RRC. The anonymized data accompanied with the pseudo-id is also known as pseudonymized data. In contrast with the pseudonyms below, the pseudo-ids have a lower degree of randomness, and so they are less secure when it comes to unauthorized identification. Furthermore, they are human-readable which is mostly preferred by the hospitals and the organizations, in general, as opposed to the pseudonyms.

In the second variant, we use pseudonyms by leveraging a trusted PP. A PP is a trusted organisation responsible for the pseudonym management of the short term anonymous credentials (according to the IEEE 1609.2 specification), to be provided to the S-EHR App, and use for the anonymous communication of the citizen's health data to a (Research) Reference Centre (RRC). Such a scheme provides higher user privacy levels even in the complex scenario where users move around different domains (i.e., countries or member states) and they need to acquire pseudonyms without revealing personal information regarding their country of origin. Furthermore, this specification also copes with important aspects of the pseudonym lifecycle like pseudonym resolution (when there is a need for linking - anonymized - data back to users in case of a health emergency), protection from misuse by authorities, and even pseudonym change while demonstrating high levels of scalability and efficiency. The exchanged information is symmetrically encrypted following the current AES-256 crypto standard.

More specifically, this section provides the design of the security libraries that can be used by any S-EHR app, RRC, CN and RDS libraries. In addition, the description of the Public Java Components contained in each library is defined, including a description of the offered and required interfaces of those components, while the description of the interactions of the internal components is also described.

5.1. RDS Implementation

This section includes the components and the public interfaces for the security libraries M-RDS-SM and the T-RDS-SM. The component names are based on the security functionalities offered. These components are also offering specific interfaces that are analysed in this section. The objective of the libraries is to allow the usage of RDS without the need for developers to know all the technical details of the underlying RDS protocol and technologies. The M-RDS-SM and T-RDS-SM libraries act as a proxy for a health care system compliant to RDS protocol specifications.

5.1.1. Components

S-EHR-side M-RDS-SM Components: The M-RDS-SM library incorporates a set of components (Figure 10) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), including the: (a) Identity Management, where the exchange the exchange certificates is taking place and eIDAS-based authentication of the 2nd variant, (b) Consent Management, that includes the signature, the validation and the timestamp of the consent, (c) Encrypted Communication, where the Diffie Helman key agreement is used to establish an AES256 symmetric key for the actual encryption, (d) Encrypted Storage, where the data are stored in encrypted form using the AES256 algorithm too and (e) Privacy and Anonymisation, where the data to be shared anonymised. Again the same APIs are omitted to be analysed in new tables .

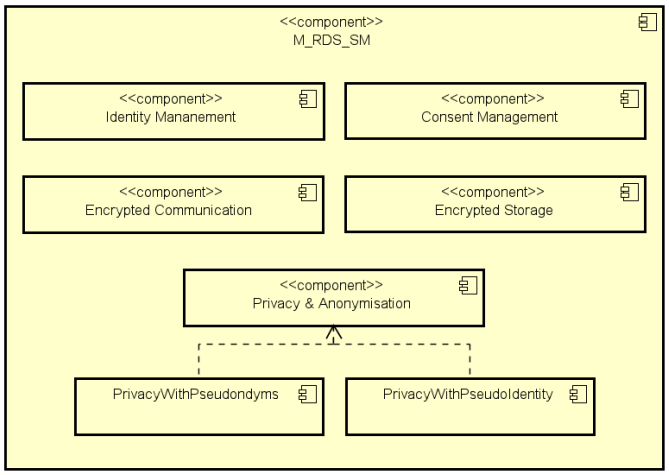

Figure 10 - M-RDS-SM Public Java Components

RRC and CN-side T-D2D-SM Components: The T-RDS-SM library incorporates a set of components (Figure 11) offering different functionalities and capabilities to the developer. These components can be offered publicly (i.e. Public components), including the: (a) Identity Management, where the exchange the exchange certificates is taking place, (b) Consent Management, that includes the signature, the validation and the timestamp of the consent, and (c) Encrypted Communication, where the Diffie Helman key agreement is used to establish an AES256 symmetric key for the actual decryption. Again the same APIs are omitted to be analysed in new tables .

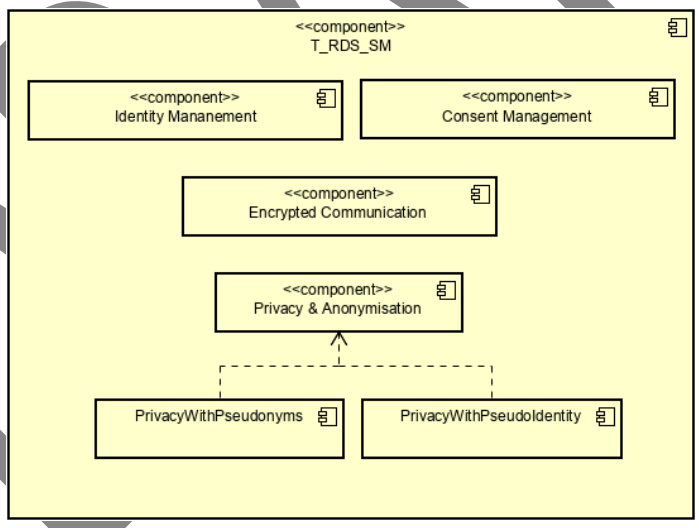

Figure 11 - T-RDS-SM Public Java Components

5.1.2. Public Interfaces

MRDSI-Security: MRDSI-Security is the name of the interface that is offered by the Mobile RDS Security Management component, containing the operations for letting the S-EHR app and M-RDS interact with the M-RDS-SM library and finally perform necessary security functionalities, by invoking these operations.

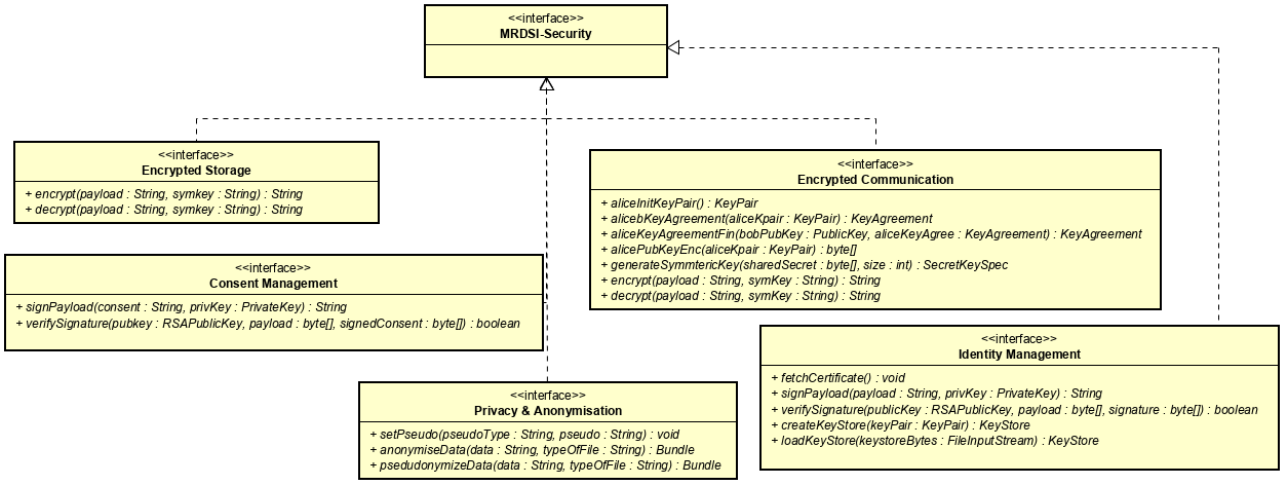

Figure 12 - MRDSI-Security Public Java Components Interfaces

Privacy & Anonymization /Operation setPseudo

Name	**setPseudo**
Description	Set a pseudo-identity (Variant 1) or a pseudonym (Variant 2), used to pseudonymize a citizen's health data.
Arguments	• String pseudoType, indicates whether the library should use a pseudo-identity or a pseudonym • String pseudo, sets the pseudo-identity/pseudonym which will substitute a citizen's personal information
Return Value	• void
Exceptions	• N/A
Preconditions	• N/A

Privacy & Anonymization /Operation anonymiseData

Name	**anonymiseData**
Description	Anonymize structured data.
Arguments	• String data, a FHIR bundle containing FHIR resources, attributes, and values, that need to be anonymized • String typeOfFile, the type of the data file
Return Value	• Bundle, the same FHIR bundle as in the input, except that identifying data provided in the input are deleted.
Exceptions	• FHIRParsingException, in case the input is not parseable as a FHIR bundle
Preconditions	• The pseudo-id/pseudonym must be set.

Privacy & Anonymization /Operation pseudonymizeData

Name	**pseudonymizeData**
Description	Pseudonymize structured data.
Arguments	• String data, a FHIR bundle containing FHIR resources, attributes, and values, that need to be anonymized • String typeOfFile, the type of the data file
Return Value	• Bundle, the same FHIR bundle as in the input, except that identifying data provided in the input are deleted.
Exceptions	• FHIRParsingException, in case the input is not parseable as a FHIR bundle
Preconditions	• The pseudo-id/pseudonym must be set.

TRDSI-Security: TRDSI-Security is the name of the interface that is offered by the Research RDS Security Management component, containing the operations for letting the research center and S-RDS interact with

the R-RDS-SM library and finally perform necessary security functionalities, by invoking these operations. In the final version we will also explore how to reverse the Pseudo-id and Pseudonyms and an updated API will be provided included reversePsedoIdentity and reversePseudonym operations.

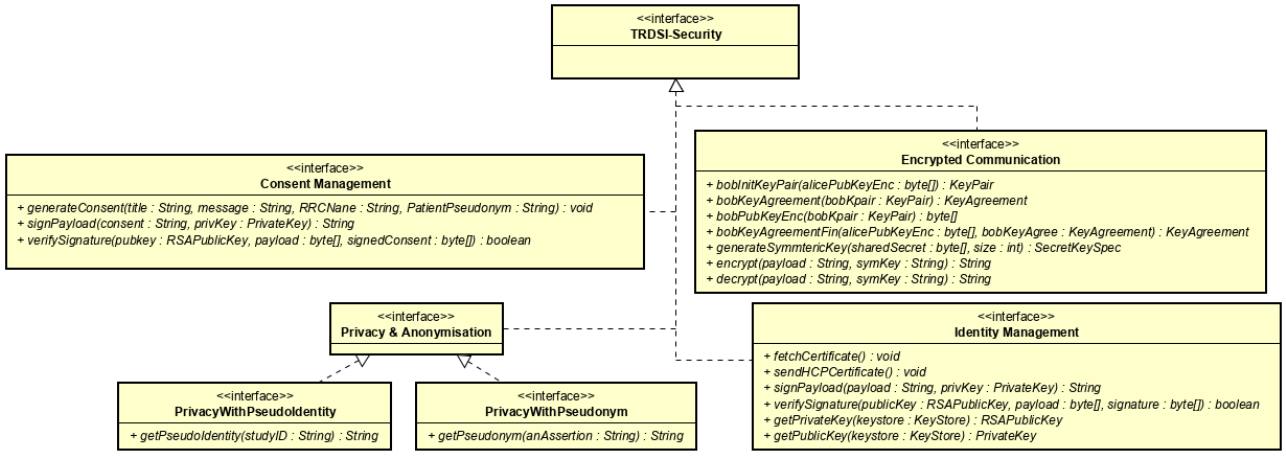

Figure 13 - TRDSI-Security Public Java Components Interfaces

Privacy & Anonymization /Operation getPseudoIdentity (Variant 1 - for pseudo-identity-based studies)

Name	**getPseudoIdentity**
Description	Allows a S-EHR App to receive a pseudo-identity which has been generated at the RRC.
Caller	The RDS-Logic library running on the S-EHR App, through the intermediary of the RDSI-Client library.
Arguments	• String studyID, the ID of the study for which the pseudo-id shall be generated
Return Value	• String, a string containing the pseudo-identity generated.
Exceptions	• Exception, invalid content (study ID);
Preconditions	• The S-EHR App must have checked that the Citizen's data fulfils the enrolment criteria for the study, and that the Citizen consents to participating in the study.

Privacy & Anonymization /Operation getPseudonym (Variant 2- for pseudonym-based studies)

Name	**getPseudonym**
Description	Allows a S-EHR App to receive a pseudonym from a trusted third party that acts as a PP. This trusted third party could also be the RRC or any other entity.
Caller	The RDS-Logic library running on the S-EHR App, through the intermediary of the RDSI-Client library.
Arguments	• String anAssertion, Anonymous assertion token
Return Value	• String, a string containing the pseudonym generated.
Exceptions	• Exception, invalid content (study ID);
Preconditions	• The S-EHR App must have checked that the Citizen's data fulfils the enrolment criteria for the study, and that the Citizen consents to participating in the study. • S-EHR App already authenticated by an eIDAS node

6. CONCLUSIONS AND NEXT STEPS

The objective of this report was to deliver the second version of the design of the security and privacy services libraries offered by the InteropEHRate Framework as a reference implementation. In the same notion as the other reports of the InteropEHRate project, this document presents the second draft of the intended content of the security and privacy libraries and their further functionality purposes. The deliverable serves as a detailed guide for the design of all the security and privacy aspects until the second year of the project.

The deliverable among others, highlights a) the mapping between the user requirements and the implemented interfaces, b) the different security components along with the offered functionalities.

Last but not least, the third version of this report will include the final design of libraries for HR security and privacy services. This final report will be updated with the final modifications and improvements on all the security libraries, including on how to address the consent revocation, ABAC-based HCP authorization and also updates regarding the D2D and R2D Access that are currently in an improvement and refactoring phase. Last but not least in the final release of the deliverable we will also provide a detailed security analysis of all the protocols.

REFERENCES

- **[BOYD 2003]** Boyd, Colin and Mathuria, Anish. Protocols for Authentication and Key Establishment. 2003. Springer-Verlag Berlin Heidelberg. 10.1007/978-3-662-09527-0
- **[Bundesamt 2015]** Bundesamt fur Sicherheit in der Informationstechnik (BSI). Advanced security mechanisms for machine readable travel documents and eIDAS token, part 2 – protocols for electronic identification, authentication and trust services (eIDAS), technical guideline TR-03110-2, v2.20, February 2015.
- **[D2.2]** InteropEHRate Consortium, D2.2-User Requirements for cross-border HR integration - V1, 2020. www.interopehrate.eu/resources/#dels
- **[D3.1]** InteropEHRate Consortium, D3.1: Specification of S-EHR mobile privacy and security conformance levels - V1, 2020. www.interopehrate.eu/resources/#dels
- **[D3.3]** InteropEHRate Consortium, D3.3: Specification of remote and D2D IDM mechanisms for HRs Interoperability - V1, 2019. www.interopehrate.eu/resources/#dels
- **[D3.5]** InteropEHRate Consortium, D3.5: Specification of data encryption mechanisms for mobile and web applications - V1, 2020. www.interopehrate.eu/resources/#dels
- **[D3.7]** InteropEHRate Consortium, D3.7: Specification of consent management and decentralized authorization mechanisms for HR Exchange - V1, 2019. www.interopehrate.eu/resources/#dels
- **[D4.2]** InteropEHRate Consortium, D4.2: Specification of remote and D2D protocol and APIs for HR exchange - V2, 2020. www.interopehrate.eu/resources/#dels
- **[D4.8]** InteropEHRate Consortium, D4.8: Specification of protocol and APIs for research health data sharing - V1, 2021. www.interopehrate.eu/resources/#dels
- **[DURGIN 2002]** N. Durgin, J. Mitchell and D. Pavlovic, "A compositional logic for protocol correctness," in Proceedings 14th IEEE Computer Security Foundations Workshop, 2001., Cape Breton, Novia Scotia, Canada, 2001, doi: 10.1109/CSFW.2001.930150
- **[eIDAS 2016]** eIDAS Technical Sub-group, eIDAS SAML Attribute Profile V1.1, 2016
- **[ENISA 2018]** ENISA, "Recommendations on shaping technology according to GDPR provisions - An overview on data pseudonymisation", 2018.
- **[ENISA 2020]** ENISA. "Minimum Security Measures for Operators of Essentials Services". 2020.
- **[ENISA 2021]** ENISA. "Pseudonymisation for Personal Data Protection". 2021.
- **[Gisdakis 2013]** S. Gisdakis, M. Laganà, T. Giannetsos and P. Papadimitratos, "SEROSA: SERvice oriented security architecture for Vehicular Communications," *2013 IEEE Vehicular Networking Conference*, Boston, MA, USA, 2013, pp. 111-118, doi: 10.1109/VNC.2013.6737597.
- **[ISO/IEC 2382:2015]** ISO/IEC 2382:2015 Information technology — Vocabulary — Part 8: Security, 1998

- **[NIST 2003]** National Institute of Standards and Technology. NIST SP 800-59, Guideline for Identifying an Information System as a National Security System, 2003 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-59.pdf
- **[NIST 2014]** National Institute of Standards and Technology. NIST SP 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations, 2014 https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf
- **[NIST ENTR]** National Institute of Standards and Technology. NIST SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
- **[SCHIEDERMEIER 2019]** M. Schiedermeier, O. Hasan, T. Mayer, L. Brunie, H. Kosch, A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks, 2019, arXiv:1909.06462v1
- **[SSPEAR 2014]** S. Gisdakis, T. Giannetsos, and P. Papadimitratos. 2014. SPPEAR: security & privacy-preserving architecture for participatory-sensing applications. In Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks (WiSec '14). Association for Computing Machinery, New York, NY, USA, 39–50. DOI:https://doi.org/10.1145/2627393.2627402
- **[Zwingelberg 2011]** Harald Zwingelberg, Marit Hansen. Privacy Protection Goals and Their Implications for eID Systems. PrimeLife. 2011: 245-260

APPENDIX A

This section summarises all the different security libraries and the corresponding offered APIs by the different protocols (e.g. D2D, R2D and RDS) and by the different components (e.g. S-EHR App, HCP App, RRC/CN). Table 2 provides a detailed mapping of the component, the protocol library, the security library and the remote interface, while Table 3 provides the security-related necessary legacy remote APIs for completion.

Protocol Family	Component	Protocol Libraries	Security Libraries	Remote Security Interfaces	APIs Description
All	S-EHR App				fetchCertificate
	HCP App, RRC, CN				fetchCertificate
D2D	S-EHR App	M-D2D-E	M-D2D-SM	MD2DI-Security	sendSEHRCertificate, signPayload, verifySignature, createKeyStore, loadKeyStore, encrypt, decrypt
	HCP App	T-D2D-E	T-D2D-SM	TD2DI-Security	sendHCPCertificate, signPayload, verifySignature, encrypt, decrypt
R2D Access	S-EHR App	M-R2D-E	M-R2D-SM	MR2DI-Security	login, logout, isAuthenticated,
	Trusted Proxy Server	T-R2D-E	P-R2D-SM	TR2DI-Security	authenticate, metadata, response
R2D Backup	S-EHR App	R2DWriter	M-R2D-SM	MR2DI-Security	login, logout, isAuthenticated, encrypt, decrypt, signPayload, verifySignature,

					generateSymmetricKey
R2D Emergency	HCP App	R2DReader	T-R2D-SM	TR2DI-Security	encrypt, decrypt, login, logout, isAuthenticated
RDS	S-EHR App	M-RDS	M-RDS-SM	MRDSI-Security	signPayload, verifySignature, bobInitKeyPair, bobKeyAgreement, bobKeyAgreementFin, bobPubKeyEnc, generateSymmetricKey, encrypt, retrievePseudonym, retrievePseudoIdentity, login, logout, isAuthenticated, setPseudo, anonymizeData, pseudonymizeData
	RRC, CN	S-RDS, IRS	T-RDS-SM	TRDSI-Security	signPayload, verifySignature, aliceInitKeyPair, aliceKeyAgreement, alicePubKeyEnc, aliceKeyAgreementFin, generateSymmetricKey, decrypt

Table 2 - InteropEHRate security libraries and remote APIs used by the InteropEHRate protocols

Legacy Remote API	Using Protocols	Description
CAI	D2D, R2D, RDS	The interface offered by the Certification Authorities to retrieve digital certificates.
PPI	RDS	The interface offered by the Pseudonym Provider for creating Pseudonyms.
eIDASI	R2D Access, RDS	The interface offered by the eIDAS Nodes for cross-border identification and authentication of Citizens.

Table 3 - Legacy security remote APIs used by the InteropEHRate security protocols

APPENDIX B

This section summarises the workflow of actions of the login Interface, including the exposed APIs of the Trusted Proxy Server, that acts as a "bridge" between the Healthcare EHR and the eIDAS infrastructure for providing a connection point to an eIDAS (proxy-based) node depending on specific Healthcare EHR requirements for certifying a requesting user. This solution simplifies the secure interaction of any Healthcare EHR, regardless of the programming language used for the EMR implementation, and the secure exchange of all required information for the eIDAS-based user certification and authentication. In a nutshell, the Trusted Proxy Server: a) enables Healthcare EHR entities to communicate with the eIDAS infrastructure without using SAML 2.0 - a time consuming development process, and b) supports platform independent EMRs since the communication is implementation-agnostic as it is based on REST APIs. The Trusted Proxy Server, as its name suggests, acts as a proxy between the actual Healthcare EHR and the eIDAS infrastructure. It handles all the interactions with the eIDAS, and requires from the Healthcare EHR to receive the Country and the UserAttributes. At the end of an eIDAS authentication process, it receives the identification attributes from the eIDAS Node and generates a signed JWT token containing those attributes. Finally, the token is sent back to the Healthcare EHR for the subsequent user authentication which, if successful, will result in the transmission of the healthcare records back to the user. The Healthcare organisation is responsible for deploying this Trusted Proxy Server.

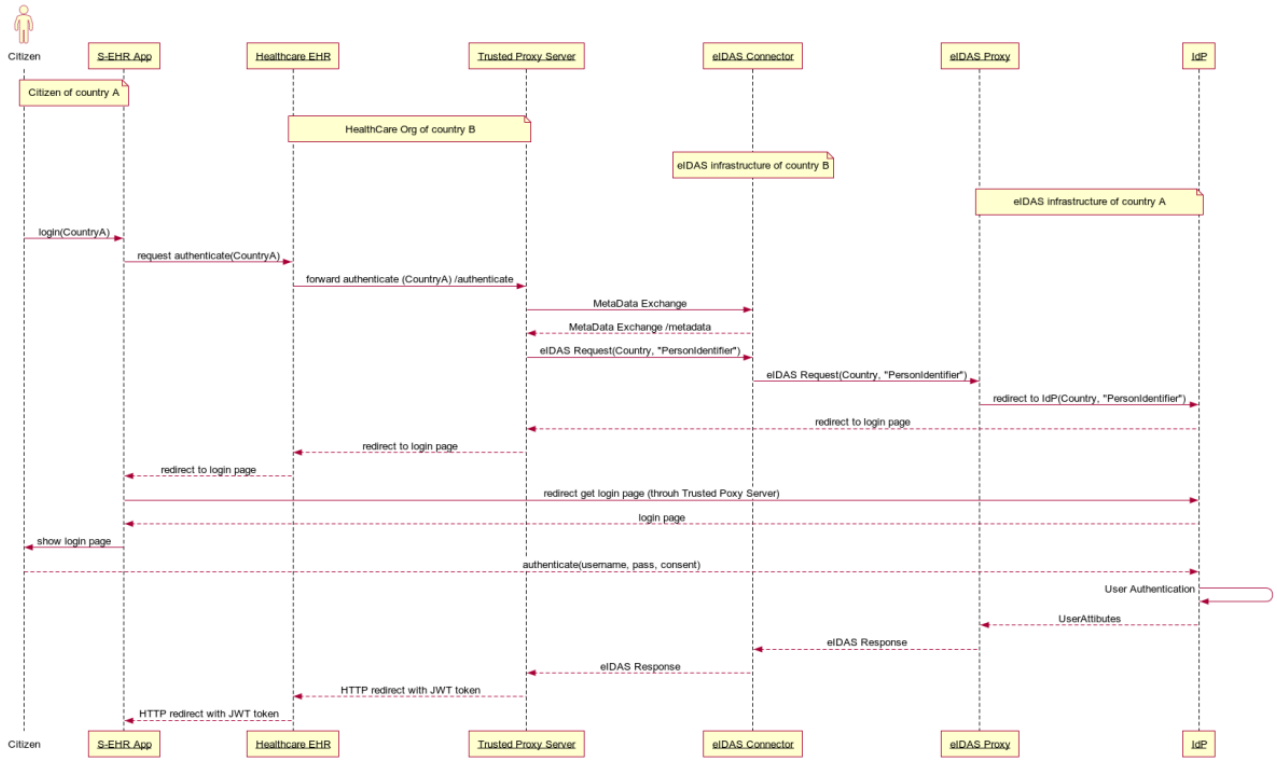

Figure 14 - eIDAS-based authentication flow

The high-level workflow of actions, as depicted in Figure 14, is summarized below.:

- **Step 1**: We assume an (already) eIDAS registered citizen that tries to get access to the R2D Access Server in order to retrieve his/her healthcare records for the first time (login(CountryA)).
- **Step 2**: The S-EHR App performs an authentication request to Healthcare EHR including the information regarding the Citizen's Country. R2D Access library is responsible for exposing these APIs as part of the Healthcare EHR.

- **Step 3**: The Healthcare EHR redirects the authentication request to the Trusted Proxy Server leveraging the exposed /authenticate REST endpoint. As aforementioned, the Trusted Proxy Server is a trusted service implemented in the context of InteropEHRate for assisting the Healthcare EHR developers to securely and efficiently handle the requests to/from the backend eIDAS Infrastructure. The Trusted Proxy Server is responsible for constructing the SAML authentication request and interacting with the nearest (proxy) eIDAS Node (eIDAS Connector) of the Healthcare EHR country. Initially there is the exchange of some metadata and then the actual eIDAS request.
- **Step 4**: Then the eIDAS Connector redirects the request to the eIDAS node of the Country of Origin of the User (eIDAS Proxy) - in the case that the country of origin is different to the country where the request was made.
- **Step 5**: The output of the eIDAS authentication flow is an eIDAS SAML authentication response. The User is authenticated using the eIDAS flow, while the user is redirected to the corresponding login page of his home country IdP.
- **Step 6**: The eIDAS Node of the User Country of Origin (eIDAS Proxy) dispatches the process for forwarding the eIDAS SAML authentication response to the eIDAS Node that initiated the authentication request (eIDAS Connector).
- **Step 7**: Upon successful authentication, the eIDAS Node which receives the authentication response then forwards it to the Trusted Proxy Server at the /returnPage endpoint. The Trusted Proxy Server processes and decrypts the authentication response and generates a JWT token that contains all the retrieved eIDAS attributes (plus the UUID of the session, generated at the beginning of the process). The Trusted Proxy Server creates the appropriate identification assertions for the eIDAS authenticated User, so that the Healthcare EHR can easily consume them. If an error occurs during authentication, such as a failure in the authentication of the User to the IdP, the Trusted Proxy Server handles it by displaying an appropriate message.
- **Step 8**: The Trusted Proxy Server provides the user info, through a signed JWT, to the Healthcare EHR and the internal user validation process will check the UserAttributes provided. The Healthcare EHR retrieves the JWT from the HTTP request, verifies its signature and authenticates the User or handles the error. The healthcare organization will only need to interact with the exposed endpoints of the Trusted Proxy Server to integrate the provided services with the eIDAS infrastructure.
- **Step 9**: If the procedure is successful, the user can then access and download the requested records provided by Healthcare EHR. Next time the user needs to authenticate himself though eIDAS all previous steps will be performed again and there is no need for any token storage.

The implementation API of the Trusted Proxy Server is the following.

R2DAccess / Trusted Proxy Server API

POST /authenticate	Handles the authentication request, constructs and propagates SAML 2.0 authRequest to the eIDAS infrastructure.

GET /metadata	Publish Healthcare EHR metadata to the eIDAS infrastructure.
POST /returnPage	Process authResponse (as received from eIDAS Node). If the response is successful, it provides authResponse translated from SAML 2.0 to JSON, and redirects to Healthcare EHR along with a JWT token. If the response fails generates a failure report.