D4.10

Design of library for health data sharing for research - V1

ABSTRACT

This deliverable describes the initial version of the design of the libraries offered by the InteropEHRate Framework as a reference implementation of the protocol and services for citizen-driven research data sharing. A detailed description is provided for the libraries, including their interactions with each other and the applications participating in the data sharing process, namely the Central Node of the Research Network, Research Centres participating in research studies, and S-EHR Apps running on the mobile devices of citizens. The current deliverable is intended for developers and manufacturers that are interested in designing and building either mobile or web applications that aim at exploiting and reusing the functionalities offered by these libraries, in the context of their applications.

Delivery Date	1st April, 2021
Work Package	WP4
Task	T4.4
Dissemination Level	Public
Type of Deliverable	Report
Lead partner	UNITN

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
Contributor	Gábor Bella	UNITN
Contributor	Simone Bocca	UNITN
Contributor	Alessio Zamboni	UNITN
Contributor	Salima Houta	FRAU
Contributor	Sofianna Menesidou	UBITECH
Contributor	Martin Marot	A7
Contributor	Stella Dimopoulou	BYTE
Contributor	Charis Tsigkounis	UPRC
Contributor	Thanos Kiourtis	UPRC
Contributor	Argyro Mavrogiorgou	UPRC
Reviewer	Francesco Torelli	ENG
Reviewer	Patrick Duflot	CHU

LOG TABLE

Version	Date	Change	Author	Partner
0.1	2020-07-27	First draft created	Gábor Bella	UNITN
0.2	2021-01-18	Cleaned up the document & TOC, updated the library descriptions and diagrams	Gábor Bella	UNITN
0.3	2021-01-28	Updated APIs and diagrams to the latest specifications	Gábor Bella	UNITN
0.4	2021-02-28	Received contributions from partners	Salima Houta, Sofianna Menesidou, Martin Marot, Stella Dimopoulou,	FRAU, UBITECH, A7, BYTE, UPRC,

			Charis Tsigkounis, Simone Bocca, Gábor Bella	UNITN
0.5	2021-03-04	Harmonizing document and adding operation details	Gábor Bella	UNITN
0.6	2021-03-10	Finishing the definition of operations	Sofianna Menesidou, Martin Marot, Stella Dimopoulou, Charis Tsigkounis, Gábor Bella	UBITECH, A7, BYTE, UPRC, UNITN
0.7	2021-03-17	Added details on security libraries and key generation, document finished for internal review	Gábor Bella	UNITN
0.8	2021-03-19	Reviewed by CHU	Patrick Duflot	CHU
0.9	2021-03-22	Reviewed by ENG	Francesco Torelli	ENG
1.0	2021-03-23	Produced final version following reviewer suggestions	Gábor Bella, Sofianna Menesidou, Martin Marot, Charis Tsigkounis, Stella Dimopoulou	UNITN, UBITECH, A7, UPRC, BYTE
2.0	2021-03-31	Quality check	Argyro Mavrogiorgou	UPRC
VFinal	2021-04-01	Final check and submission version	Laura Pucci	ENG

ACRONYMS AND TERMS

Acronym	Term	Definition
CN	Central Node	A node of the Research Network (a server) that stores RDDs and provides a central access point to S-EHR Apps for retrieving them.
-	Citizen	Any person potentially participating in a research study and having the minimal technical means to do so, i.e. the S-EHR App installed on their smartphone.
-	Sponsor	The (public or private) legal entity who has ordered the research study and is paying for it.
CRC	Coordinating Research Centre	A medical research centre that initiates a particular research study and is in charge of defining it and carrying it out.
CSR	Certificate Signing Request	A message sent from an applicant to a registration authority of the public key infrastructure in order to apply for a digital identity certificate.
PI of the Research Centre	Principal Investigator of a Research Centre	The researcher (person) in charge of the citizens enrolled for a specific study at a RC.
PI of the Study	Principal Investigator of the Study	The researcher (person) in charge of a specific study at the CRC.
CN Administrator	Central Node Administrator	A single person in charge of overseeing at the Central Node the publishing of new research studies on the Research Network.
RDD	Research Definition Document	A document that contains a human-readable description of the research study to be performed, as well as information written in a formal, computer-processable language that describes the research datasets to be retrieved from citizens' EHRs, enrolment and exit criteria, as well as related metadata.
RDDI	RDD Interface	Application Programming Interface offered by the Central Node, allowing the S-EHR App to access the published RDDs.
RDS	Research Data Sharing	The protocol implemented by the libraries described in this deliverable. Secure IT communication protocol (and APIs) for publishing and retrieving machine processable descriptions of research studies and for sending citizen's consents and health data from S-EHR Apps to research centres (that are RDS nodes), without any cloud storage of health records.
RDSI	RDS Interface	Application Programming Interface allowing the exchange of consent and health data between the S-EHR App and Research Centres.
RN	Research Network	The network of research centres and technical nodes that implement the Protocol.

RRC	Reference Research Centre	A research centre participating in a given study is a RRC for the citizens who are officially attached to it (i.e. send data to it) for the duration of the study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the document

This deliverable describes the APIs of a set of reusable software libraries that will constitute the reference implementation of the *Research Data Sharing Protocol* [D4.8]. This libraries run inside three components of the research data sharing infrastructure defined in [D4.8]:

- inside the S-EHR App on the mobile device of the citizen;
- inside the information system of the Research Centres participating in research studies;
- inside the Central Node of the Research Network that publishes the *Research Definition Documents (RDDs)* of new studies, based on which data sharing is automated.

The libraries described here cover multiple aspects of the data sharing process:

- the logic of publishing, downloading, and interpreting *RDDs*;
- the logic of the study enrolment and data sharing process;
- the communication with citizens through the UI of the S-EHR App;
- pseudonym generation and pseudo-anonymization of the data to be shared;
- security of the data transmissions, including digital signature and encryption.

Differently from the specification of the RDS protocol, the present specification is non-normative, i.e. developers are free to implement the RDS protocol by means of different libraries following a different design as these choices do not impact on the interoperability.

1.2. Intended audience

This document is intended mainly for software developers and architects (both inside and outside the InteropEHRate project) who wish to implement software components or applications that either realise or use the research data sharing services that follow the Research Data Sharing Protocol [D4.8]. More generally, the information contained in this deliverable can help developers understand the technical requirements behind implementing such functionalities.

1.3. Structure of the document

Section 2 describes how the libraries described in this deliverable are used in the context of the InteropEHRate scenarios and pilots, providing motivation for their design. Section 3 introduces the main devices, software components, and (intra-device and inter-device) interfaces through which the libraries communicate with each other. Section 4 describes the library operating the Central Node of the Research Network. Section 5 describes the library in charge of the enrolment and data collection operations inside the Research Centre information system. Section 6 describes the libraries inside the S-EHR App. Section 7 describes the libraries that help applications access Research Definition Documents. Section 8 describes security-related libraries, used in every device participating in the data sharing process. Section 9 provides sequence diagrams that show how the libraries defined in sections 4-8 interact with each other. Finally, Section 10 provides conclusions and the next steps towards the future version 2 of this deliverable.

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

Not Applicable.

2. RELATION TO INTEROPEHRATE SCENARIOS

The libraries described in this deliverable realise functionalities needed for the implementation of the Research Scenario (that is, Scenario 3) of InteropEHRate [D2.3] and the Research Data Sharing Protocol [D4.8]. This scenario and the corresponding project pilot consist of publishing the formal description of (a small number of) research studies on the Central Node of the Research Network used for demonstration. The S-EHR Apps of the citizens participating in the pilot will automatically download these formal and digitally signed descriptions (called *Research Definition Documents*), interpret them, and automatically compare them to the health data stored by the citizens' S-EHR Apps in order to verify their eligibility to participate in the given study. Each eligible citizen is explicitly asked by the S-EHR App whether they are willing to participate in the study and consequently share their health data for this purpose. In case of consent (digitally signed by the citizen), the citizen is officially enrolled into the study and is attached to a *Reference Research Centre* (RRC) to which his/her data are sent. The actual sharing of pseudo-anonymized and encrypted data happens automatically, on one or multiple occasions, initiated by the S-EHR App according to the data collection period defined for the study.

The Research Scenario presupposes that the health data stored on the S-EHR Apps is already interoperable, based on the Interoperability Profiles [D2.9].

3. ARCHITECTURE AND INTERFACES

The figure below, already introduced and described in detail in [D4.8], shows the main software systems, their exposed APIs, and the human actors whose actions and communication are covered by the Research Data Sharing Protocol. On a high level, the protocol involves the following systems:

- the **S-EHR App** that holds a Citizen's health data that can be shared for research purposes; it is also through this app that the Citizen expresses his/her consents to data sharing;
- the **Central Node** (CN) on which machine-processable RDDS are published to be downloaded by S-EHR Apps through a dedicated API;
    - the CN also includes a *Research Portal* that helps in the administration and publishing of RDDs; while the portal is outside the scope of the RDS protocol, it relies on the libraries described in this deliverable;
- the **Research Centre Information System** of each research centre participating to a given research study, and to which citizens send their health data;
- in order to ensure that data sharing is performed in a secure and privacy-preserving manner, the following third parties are also involved in the protocol mechanism:
    - a **Certification Authority** that provides certificates to the S-EHR App for the purposes of asymmetric encryption and digital signature (which the S-EHR App needs to store and manage),
    - a **Pseudonym Provider** that generates random pseudonyms to be used for privacy-preserving data sharing (in case the research study indicates that the pseudonym-based variant of the RDS Protocol should be used);
    - an **eIDAS Node** that will be used for authentication purposes with the Pseudonym Provider.

The security and privacy components are considered already to exist independently of the InteropEHRate project, and the protocol uses their services as is. The first three systems, on the other hand, need to be either adapted to the needs of research data sharing or, in the case of the Central Node, deployed as an entirely new system.

As depicted in Figure 1, the systems communicate with each other through the following communication interfaces:

- the **RDDI** (Research Definition Document Interface) through which RDDs are downloaded from the Central Node by the S-EHR App;
- the **RDSI** (Research data Sharing Interface) through which consent to data sharing is expressed and health data are securely transmitted;
- the **CAI** (Certification Authority Interface) through which the S-EHR App connects to a Certification Authority to retrieve its certificates;
- the **PPI** (Pseudonym Provider Interface) through which the S-EHR App retrieves research-study-specific pseudonyms (only for studies using pseudonym-based pseudonymisation);
- the **eIDASI** (eIDAS Node Interface) through which the S-EHR App connects to an *eIDAS Node* in order to retrieve an authentication SAML assertion, used later for authentication with the Pseudonym Provider (only for studies using pseudonym-based pseudonymisation).

Again, among these interfaces, the last three are considered already to be specified, implemented, and provided by the respective components. The present deliverable is thus only concerned with implementing the first to interfaces (RDDI and RDSI).

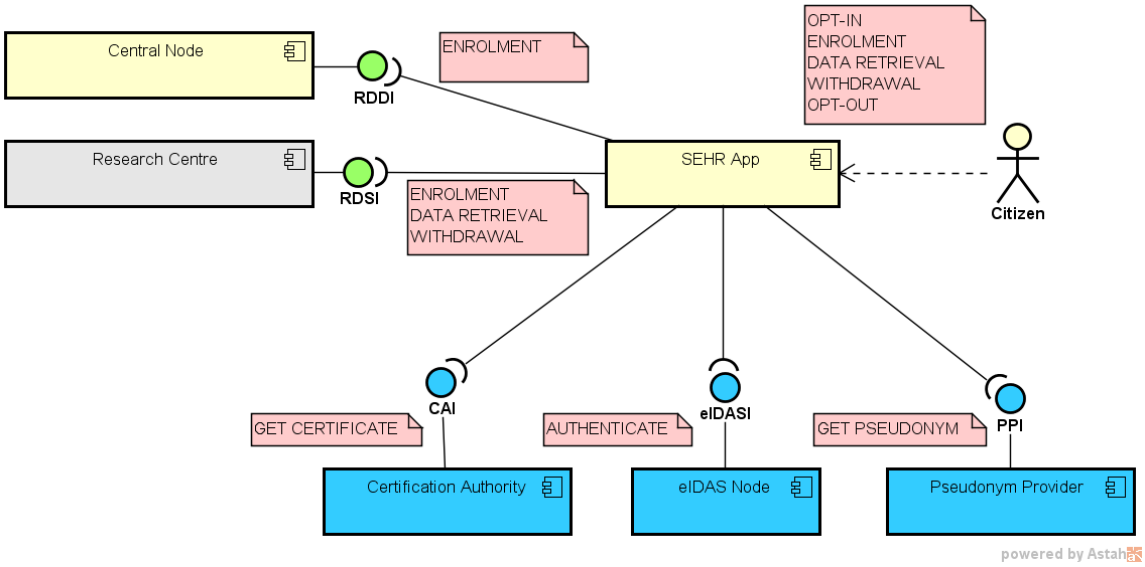

Figure 1 - Systems, actors, and communication channels of the Protocol

The interfaces specified by the RDS protocol are depicted in green colour. The pre-existing systems and interfaces are depicted in light blue, while the Research Centre that is also a pre-existing system but that must expose an interface specified by the RDS protocol is depicted in grey. New systems required by the RDS protocol are depicted in yellow.

3.1. Libraries

Figure 2 below shows the library components that will be integrated within the reference implementation of the S-EHR App, the Central Node, and the Research Centre above, as well as the internal interfaces they expose to each other. The main goal of this deliverable is to specify the requirements and behaviour of these libraries.

- **RDS-Logic:** implements the main Protocol logic on the citizen's smartphone;
- **Core S-EHR App Logic:** this is an interface that needs to be implemented by existing vendor-specific S-EHR Apps in order to support the logic that implements research data sharing, including user interfaces, health data storage and retrieval, etc.;
- **RDS-Anonymization:** library providing data (pseudo-)anonymization functionalities to the RDS-Logic component;
- **RDS-Security:** library providing security functionalities (encryption, digital signature) integrated within all three systems (Central node, Research Reference Centre, S-EHR App);
- **RDDI-Server and Client:** server and client libraries implementing the services of the RDDI interface (download of RDDs from the Central Node);
- **RDSI-Server and Client:** server and client libraries implementing the services of the RDSI interface (citizen consent and sharing of health data).

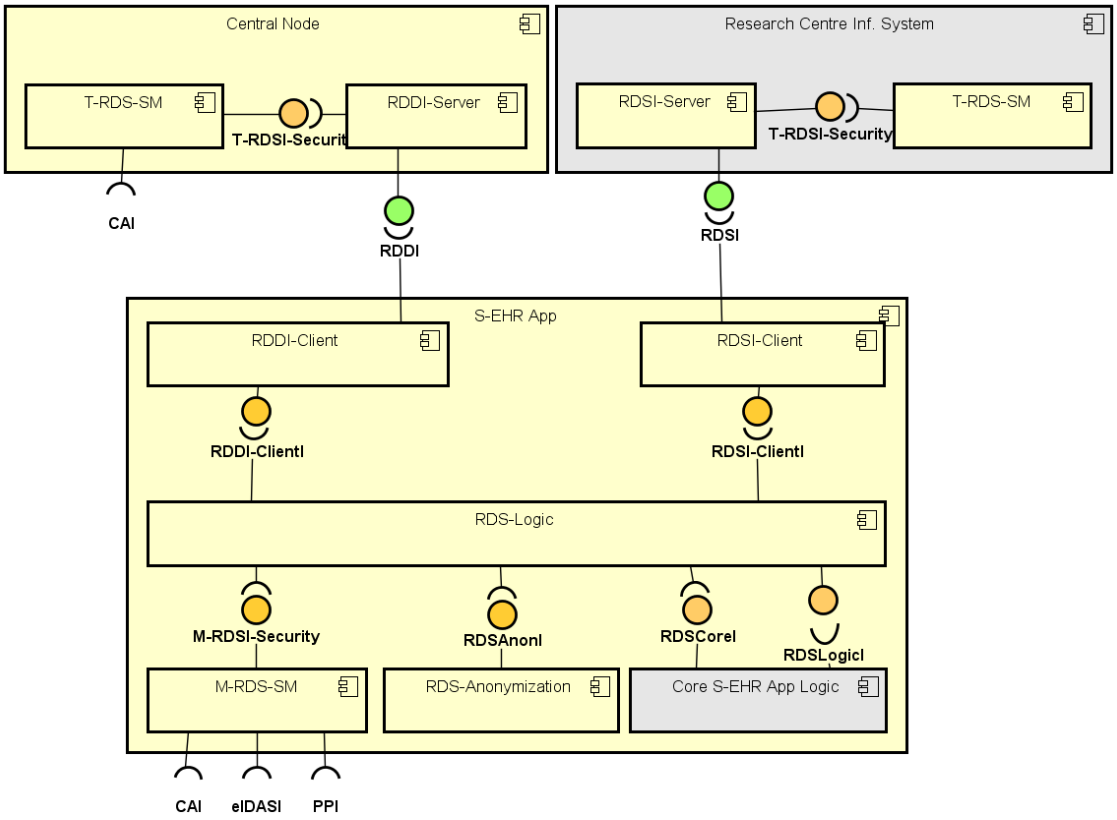

Figure 2 - Research-related subcomponents of the S-EHR App, the Central Node, and the Research Center Information System. Orange-coloured local interfaces are introduced in this deliverable. Green-coloured remote interfaces are defined as part of the Research Data Sharing Protocol.

4. CENTRAL NODE LIBRARIES

Summary: the Central Node provides the services exposed through the *Research Dataset Definition Interface* (RDDI), through a library called *RDDI-Service*. It also uses the RDS-Security library to secure the RDDI communication channel. RDDI is a RESTful interface already defined by the RDS protocol [D4.8]; in this document, it is presented in an evolved version.

Interface provided: RDDI

Used by: RDS-Logic through the RDDI-Client library, as well as the Research Portal deployed on the Central Node itself.

RDDI	Description
getOpenStudies	An open endpoint that allows any caller, but primarily a S-EHR App, to retrieve the digitally signed list of currently open studies, to which enrolment is possible. Returns a list of RDDs, each describing a study.

Table 1 - Summary of operations of the RDDI Interface

Operation submitStudy

Name	**submitStudy**
Description	Allows a Researcher (Principal Investigator) to upload a new study on the Central Node, in order to be published, in the form of a Research Definition Document (RDD). The RDD is uploaded on the Central Node, where it is syntactically validated with respect to the RDD InteropEHRate profile, moreover the content of the RDD is checked to verify the validity of the information that have to be published.
Caller	An online Research Portal operated by the Researcher who wants to submit a new study.
Arguments	• RDD: a formal Research Definition Document to be published • Researcher Credentials: the *Research Portal username and password* of the Researcher who wants to submit the study, which allows him/her to use the Central Node submit service.
Return Value	True on success, exceptions in case of wrong upload. The Research Portal shows the RDDs correctly uploaded.
Exceptions	• Upload failed exception: the file provided for the upload is not a Research Definition Document. • Structure not allowed exception: the RDD provided for the upload doesn't respect the RDD InteropEHRate profile. • File not selected exception: the file for the upload is missing.

Preconditions	• The Researcher has to have an account with valid username and password registered in the Central Node.

Operation getOpenStudies

Name	**getOpenStudies**
Description	Allows any caller (but primarily a S-EHR App through a RESTful API call) to retrieve a digitally signed list of the currently open studies, to which enrolment is possible. The RDD list which is retrieved through this service, contains all the RDDs published on the Research Network (using the publishStudy operation), which allow the enrolment of citizens (the enrolment period declared in the RDD is open).
Caller	The RDDI-Client library running on the mobile device, through a RESTful API call. RDDI-Client, in turn, exposes a method with an identical signature to the RDS-Logic library.
Arguments	• startDate: an optional filter in order only to retrieve open studies with enrolment periods having started after the date given here.
Return Value	• a digitally signed list of RDDs (Research Definition Documents) that are represented as FHIR bundles • the certificate of the Central Node that certifies the authenticity of the digital signature
Exceptions	• Empty list exception: there are no RDDs available.
Preconditions	• In order to be able to check the digital signature in the return value, the caller needs to have access to the public key of the Central Node.

5. RESEARCH CENTRE LIBRARIES

Summary: the Research Centre Information System provides the services exposed through the *Research Data Sharing Interface* (RDSI), through a library called *RDSI-Service*. It also uses the RDS-Security library to secure the RDSI communication channel. RDSI is a RESTful interface already defined by the RDS protocol [D4.8]; in this document, it is presented in an evolved version.

Interface provided: RDSI

Used by: the RDSI-Client library on the mobile device (itself locally called by RDS-Logic).

RDSI Endpoint	Description
sendEnrollmentConsent	Send the Citizen's electronically signed consent of enrolling into a specific study. The consent also includes the newly generated study-specific pseudonym or pseudo-identity, as well as the S-EHR App ID. The receiving RC checks the signature validity of the signedConsent, signs and returns the contract signed by both parties.
sendExitNotification	Send a notification that the Citizen is exiting a study due to the exit criteria being met. If the RRC fails to satisfy the call, a corresponding RESTful API Error is returned.
sendHealthData	Allows a S-EHR App to send citizen health data to the RRC. The receiving RC verifies and decrypts the encrypted and signed payload *healthData* and retrieves the FHIR bundle contained within. If the RRC fails to satisfy the call, a corresponding RESTful API Error is returned.
retrievePseudoIdentity	Allows a S-EHR App to receive a pseudo identity which has been generated at the RRC.

Table 2 - Methods of the RDSI Interface

Operation sendEnrollmentConsent

Name	**sendEnrollmentConsent**
Description	Send the Citizen's electronically signed consent of enrolling into a specific study. The receiving RC checks the signature validity of the signed consent, signs and returns the contract signed by both parties.
Caller	The RDS-Logic library running on the S-EHR App, through the intermediary of the RDSI-Client library.
Arguments	• studyID: the ID of the study in which the Citizen is enrolling; • citizenPseudo: pseudonym or pseudo-identity generated for the Citizen; • citizenCertificate: certificate of the Citizen issued by a Certification Authority and sent to the RC so that it can verify the digital signature; • enrollmentCriteriaData: encrypted data values corresponding to the enrolment criteria, so that the RC can cross-check their validity; • signedConsent: a digitally signed document containing the Citizen's

	consent to participate in the study.
Return Value	• The consent contract where the Research Centre has added its own digital signature, and which is now signed by both parties; • the certificate of the Research Centre that certifies the authenticity of the digital signature.
Exceptions	• invalid content (study ID, pseudo-identity, consent form); • digital signature cannot be verified; • enrolment criteria not met
Preconditions	• The S-EHR App of the Citizen must have verified the eligibility of the Citizen to participate in the study through checking the enrolment criteria. • The S-EHR App must have access to the Citizen's private key in order to sign the consent. • The S-EHR App must have generated a pseudonym or pseudo-identity to be used in the study, which conforms to the RDD of the study.

Operation sendExitNotification

Name	**sendExitNotification**
Description	Send a notification that the Citizen is exiting a study due to the exit criteria being met **or upon explicit withdrawal by the Citizen.**
Caller	The RDS-Logic library running on the S-EHR App, through the intermediary of the RDSI-Client library.
Arguments	• studyID: the ID of the study in which the Citizen is enrolling; • citizenPseudo: the study-specific pseudonym or pseudo-identity of the Citizen; • reason: either WITHDRAWAL or DROPOUT, the first one meaning a voluntary withdrawal by the Citizen while the second is the consequence of the exit criteria being met; • reasonText: in the case of DROPOUT, a text string explaining the reason for the exit (e.g. "blood pressure < 120"); • citizenSignature: digital signature confirming the exit.
Return Value	none
Exceptions	• invalid content (study ID, pseudo-identity, reason, signature); • citizen not enrolled and thus cannot exit
Preconditions	• The Citizen must have enrolled into the study previously.

Operation sendHealthData

Name	**sendHealthData**
Description	Allows a S-EHR App to send citizen health data to a Research Centre. The receiving RC verifies and decrypts the encrypted and signed payload healthData and retrieves the FHIR bundle contained within.
Caller	The RDS-Logic library running on the S-EHR App, through the intermediary of the RDSI-Client library.
Arguments	• studyID: the ID of the study in which the Citizen is enrolling; • citizenPseudo: the study-specific pseudonym or pseudo-identity of the Citizen; • healthData: a FHIR bundle containing the health data (resources, attributes, values) necessary for the study, in an encrypted form
Return Value	-
Exceptions	• invalid content (study ID, pseudo-identity, healthData);
Preconditions	• The Citizen must have enrolled into the study previously. • The S-EHR App must have access to the Citizen's private key to encrypt the health data, and the called Research Centre must have access to the Citizen's public key to be able to decrypt it.

Operation retrievePseudoIdentity (for pseudo-identity-based studies)

Name	**retrievePseudoIdentity**
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	• studyID: the ID of the study for which the pseudo-id shall be generated
Return Value	A string containing the pseudo-identity generated.
Exceptions	• invalid content (study ID); • if the study does not require a pseudo-identity (i.e. it requires a pseudonym obtained through a different channel)

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. • The RDD specifies that the RRC must generate a pseudo-identity.

6. S-EHR APP LIBRARIES

A large part of the Protocol logic is implemented in the S-EHR App, running on the Citizen's phone. The main component implementing the logic is called *RDS-Logic* in Figure 2. RDS-Logic interacts with a large number of components inside the S-EHR App:

- **Communication with the Central Node and the RRC.** The RDS-Logic library communicates with the Central Node and the information system of the Citizen's chosen RRC through the corresponding client libraries *RDDI-Client* and *RDSI-Client*.
- **Use of security services.** The RDS-Logic accesses security services through a local instance of the *RDS-Security* library.
- **Anonymization and pseudonymisation.** The RDS-Logic accesses anonymization and pseudonymisation services through the *RDS-Anonymization* library that implements the *RDSAnonI* interface.
- **Interaction with the S-EHR App.** The RDS-Logic interacts with the rest of the S-EHR App logic (including the app UIs) through two interfaces: the *RDSCoreI* interface exposed by the SEHR App and, for operations initiated by vendor-specific logic such as the UI, the *RDSLogicI* interface implemented by the RDS-Logic library.

6.1. RDS-Logic Interface and Library

Summary: this library implements the major part of the client-side logic of the RDS Protocol.

Interface provided: RDSLogicI

Used by: the S-EHR App.

RDSLogicI	Description
setOptInStatus	Set an internal flag that indicates whether the citizen has opted in or out of participating in research studies.
getOptInStatus	Get the value of the flag that indicates whether the citizen has opted in or out of participating in research studies.
checkNewStudies	Based on the last check date, ask the Central Node if there are new/updated studies available.
fetchStudies	return the list of previously retrieved studies the citizen is participating and could participate
enrollInStudy	Execute all operations necessary for study enrolment
withdrawFromStudy	Execute all operations necessary for withdrawing from a study.

Table 5 - Summary of operations of the RDS-Logic Interface

Operation setOptInStatus

Name	**setOptInStatus**
Description	Set an internal flag that indicates whether the Citizen has opted in or out of participating in research studies.
Caller	S-EHR App
Arguments	• boolean
Return Value	-
Exceptions	• none
Preconditions	• none

Operation getOptInStatus

Name	**getOptInStatus**
Description	Get the value of the flag that indicates whether the citizen has opted in or out of participating in research studies.
Caller	S-EHR App
Arguments	-
Return Value	A flag that indicates whether the Citizen has opted in to participating in future studies or not.
Exceptions	• none
Preconditions	• none

Operation checkNewStudies

Name	**checkNewStudies**
Description	Based on the date of the previous check, request the RDS-Logic library to poll the Central Node for new/updated studies. The date of the previous check is stored in the RDS-Logic library.
Caller	S-EHR App

Arguments	-
Return Value	An integer number representing the number of studies retrieved where the Citizen is eligible to participate, based on the verification of enrolment criteria.
Exceptions	• NetworkException: if the Central Node is not reachable through the network. • An exception should be raised if the operation is invoked without opting in.
Preconditions	• Should only be called if the Citizen has opted in.

Operation fetchStudies

Name	**fetchStudies**
Description	Return the list of previously retrieved ongoing and available studies (in which the citizen is either already participating or could in theory participate)
Caller	S-EHR App
Arguments	-
Return Value	A list of study metadata to be used for display purposes, extracted by the RDS-Logic library from the RDDs (in order to hide RDD format details from the S-EHR App logic).
Exceptions	• NotPartOfResearchNetworkException: the Citizen has not opted in to the Research Network.
Preconditions	• none

Operation enrollInStudy

Name	**enrollInStudy**
Description	Execute all operations necessary for study enrolment
Caller	S-EHR App
Arguments	• studyId: the ID of the study in which to enrol • referenceRC: choice of the Reference Research Centre made by the Citizen
Return Value	-

Exceptions	• UnknowStudyException: the ID does not correspond to any available study • ReferenceCenterInvalidException: the citizen did not choose any RRC, or an invalid RRC is indicated as the choice. • NotPartOfResearchNetworkException: the Citizen has not opted in to the Research Network. • NetworkException: the network connection to the Research Centre was unsuccessful.
Preconditions	• The study identified in the input must be a valid study with enrolment open. • The Citizen has selected the study, consented to the enrolment, digitally signed it, and selected a Reference Research Centre.

Operation withdrawFromStudy

Name	**withdrawFromStudy**
Description	Execute all operations necessary for withdrawing from a study.
Caller	S-EHR App
Arguments	• studyId: the ID of the study from which to withdraw
Return Value	Boolean
Exceptions	• UnknowStudyException: the ID does not correspond to a currently ongoing study. • NetworkException: the network connection to the Research Centre was unsuccessful. • NotEnrolledException: the Citizen has not enrolled into the study. • NotPartOfResearchNetworkException: the Citizen has not opted in to the Research Network.
Preconditions	• The study indicated in the input must be an ongoing study in which the Citizen is participating.

6.2. Anonymization / Pseudonymisation Interface and Library

Summary: provide in-device data anonymization and pseudonymisation functionalities over research data.

Interface provided: RDSAnonI

Used by: RDS-Logic library inside the mobile device.

The design principle underlying the definition of this library is a distinction between structured data (essentially in FHIR format, but also data structures included in DICOM images) and unstructured data (image bitmaps, PDF documents, etc.). The design decision is that structured data is anonymized inside the mobile device (i.e. through this library) while unstructured data is not, due to the potential complexity of the task: dealing with multilingual data, text embedded in image bitmaps, etc. For unstructured data, it is supposed that pre-anonymized versions of such datasets will be uploaded into the S-EHR App together with their non-anonymized versions. When running the RDS service, the S-EHR App will automatically return the anonymized versions of unstructured data. In case anonymized versions are not available, the S-EHR App will not return the datasets.

RDSAnonI	Description
setPseudo	Set the pseudo-identity or pseudonym (depending on the technique requested in the RDD) used to pseudonymize a citizen's health data.
getPseudo	Get the previously computed pseudo-identity or pseudonym (depending on the technique requested in the RDD) from the library.
anonymizeData	Anonymize structured data.
pseudonymizeData	Pseudonymize structured data.
retrievePseudonym	Retrieve a pseudonym from a Pseudonym Provider.

Table 3 - Summary of operations of the RDS-Anonymization/Pseudonymisation Interface (with Pseudo-identities)

Operation setPseudo

This operation is used to set either a pseudonym or a pseudo-identity, according to the needs of the research study as defined in the RDD. The pseudo has been previously retrieved either from a Pseudonym Provider or from a Research Centre.

Name	**setPseudo**
Description	Depending on the requirements of the study, set either a pseudo-identity or a pseudonym for the pseudonymisation of a citizen's health data.
Caller	RDS-Logic
Arguments	• pseudoType: indicates whether the library should use a pseudo-identity or a pseudonym • pseudo: sets the pseudo-identity/pseudonym which will substitute a citizen's personal information • studyID: the research study to which the pseudo-id applies
Return Value	Void
Exceptions	• Thrown if any of the input values is null.
Preconditions	• none

Operation getPseudo

This operation is used to get a previously set pseudonym or pseudo-identity.

Name	**getPseudo**
Description	Get the pseudo associated to the study indicated as input.
Caller	RDS-Logic
Arguments	• studyID: the research study to which the pseudo-id applies
Return Value	• The pseudo as a String; • a flag indicating whether the pseudo is a pseudonym or a pseudo-identity.
Exceptions	• InvalidStudyException: if the study ID in input does not have a pseudo associated.
Preconditions	• none

Operation anonymizeData

Name	**anonymizeData**
Description	Anonymize structured data.
Caller	RDS-Logic
Arguments	• data: a FHIR bundle containing FHIR resources, attributes, and values, that need to be anonymized • typeOfFile: the type of the data file (structured or unstructured)
Return Value	The same FHIR bundle as in the input, except that identifying data provided in the input are removed.
Exceptions	• FHIRParsingException: in case the input cannot be parsed as a FHIR bundle conform to the InteropEHRate Interoperability Profiles.
Preconditions	• none

Operation pseudonymizeData

Name	**pseudonymizeData**

Description	Pseudonymise structured data.
Caller	RDS-Logic
Arguments	• data: a FHIR bundle containing FHIR resources, attributes, and values, that need to be pseudonymized • typeOfFile: the type of the data file (structured or unstructured)
Return Value	The same FHIR bundle as in the input, except that identifying data provided in the input are replaced with the pseudo-id/pseudonym.
Exceptions	• FHIRParsingException: in case the input cannot be parsed as a FHIR bundle conformant to the InteropEHRate Interoperability Profiles. • PseudoException: in case the pseudo-identity or the pseudonym is not set.
Preconditions	• The pseudo-id/pseudonym must be set.

Operation retrievePseudonym (for pseudonym-based studies)

Name	**retrievePseudonym**
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.
Arguments	• anAssertion: Anonymous assertion token
Return Value	A string containing the pseudonym generated.
Exceptions	• Invalid content (study ID).
Preconditions	• The S-EHR App has already been authenticated by an eIDAS node.

6.3. Core S-EHR App Interface and Library

Summary: this is an interface not implemented by the RDS reference libraries, but to be implemented by the S-EHR App logic (core S-EHR App Logic). Its function is to allow the RDS-Logic library to query the content of the S-EHR App and to notify the S-EHR App UI about relevant events.

Interface provided: RDSCoreI

Used by: RDS-Logic library inside the mobile device.

RDSCoreI	Description
notifyPendingStudies	Upon opt-out, if there are ongoing research studies, notifies the citizen of this fact.
checkCriteria	Checks whether the citizen's health data match the criteria as formulated by the PathQuery.
notifyEnrollment	Notify the citizen of the successful enrolment in a study.
notifyExit	Notify the citizen of exiting a study due to a particular reason.
notifyDataRetrieval	Notify the citizen that data is being sent as part of a specific study.
queryData	Retrieve citizen health data according to the query in input.

Table 4 - Operations of the RDS-Core S-EHR App Interface

Operation notifyPendingStudies

Name	**notifyPendingStudies**
Description	Upon opt-out, if there are ongoing research studies, requests the S-EHR App UI to notify the Citizen of this fact.
Caller	RDS-Logic
Arguments	• pendingStudyList: metadata of the ongoing research studies
Return Value	-
Exceptions	• none
Preconditions	• none

Operation notifyNewStudies

Name	**notifyNewStudies**
Description	Requests the S-EHR App UI to notify the Citizen that new studies are available to him/her.
Caller	RDS-Logic
Arguments	• studyList: list of research studies available to the Citizen
Return Value	-
Exceptions	• none
Preconditions	• It has previously been verified that the Citizen matches the enrolment criteria of all studies given as input.

Operation checkCriteria

Name	**checkCriteria**
Description	Checks whether the citizen's health data match the (enrollment or exit) criteria as formulated by the FHIR Group object given as input.
Caller	RDS-Logic
Arguments	• condition: a FHIR Group object that represents the enrolment or exit criteria
Return Value	• A boolean value that is true if the criteria are met, false otherwise; • the data values corresponding to the Group object, represented inside a FHIR bundle.
Exceptions	• FHIRConverterException: if the input Group object values cannot be properly converted.
Preconditions	• none

Operation notifyEnrollment

Name	**notifyEnrollment**
Description	Request the S-EHR App UI to notify the Citizen of the successful enrolment in a research study
Caller	RDS-Logic
Arguments	• study: metadata to be displayed about the study
Return Value	-
Exceptions	• none
Preconditions	• Enrolment into the study must have been successful, with "success" returned by the Research Centre.

Operation notifyExit

Name	**notifyExit**
Description	Request the S-EHR App UI to notify the citizen of exiting a study due to the exit criteria specified by the RDD being met.

Caller	RDS-Logic
Arguments	• study: metadata to be displayed about the study • reason (optional): human-readable reason for the exit
Return Value	-
Exceptions	• none
Preconditions	• none

Operation notifyDataRetrieval

Name	**notifyDataRetrieval**
Description	Request the S-EHR App UI to notify the citizen that data is being sent to a Research Centre as part of a specific study.
Caller	RDS-Logic
Arguments	• study: metadata to be displayed about the study • data (optional): human-readable list of data elements (attributes) being sent
Return Value	-
Exceptions	• none
Preconditions	• none

Operation queryData

Name	**queryData**
Description	Retrieve citizen health data according to the query in input
Caller	RDS-Logic
Arguments	• query: a FHIR PathQuery that represents the data elements (attributes) to be queried
Return Value	A ResultSet that contains FHIR bundle(s) that include(s) the data values queried, may be empty if no data is found.

Exceptions	• FHIRParseException: if the input PathQuery could not be parsed.
Preconditions	• none

Operation notifyWithdrawal

Name	**notifyWithdrawal**
Description	Request the S-EHR App UI to notify the citizen of successfully having withdrawn from a study.
Caller	RDS-Logic
Arguments	• study: metadata to be displayed about the study
Return Value	-
Exceptions	• none
Preconditions	• none

6.4. RDSI-Client Library

Summary: this library implements connectivity to the server-side RDSI interface, running on the Research Centre Information System.

Interface provided: the interface mirrors exactly that of the RDSI-Server. See Section 5 for a detailed list of operations.

Used by: the RDS-Logic library.

RDSI Endpoint	Description
sendEnrollmentConsent	Local call corresponding to the same remote RDSI operation.
sendExitNotification	Local call corresponding to the same remote RDSI operation.
sendHealthData	Local call corresponding to the same remote RDSI operation.
retrievePseudoIdentity	Local call corresponding to the same remote RDSI operation.

Table 2 - Methods of the RDSI Interface

Operation sendEnrollmentConsent

The local call mirrors the eponymous remote operation defined in section 5.

Operation sendExitNotification

The local call mirrors the eponymous remote operation defined in section 5.

Operation sendHealthData

The local call mirrors the eponymous remote operation defined in section 5.

Operation retrievePseudoIdentity (for pseudo-identity-based studies)

The local call mirrors the eponymous remote operation defined in section 5.

6.5. RDDI-Client Library

Summary: this library implements connectivity to the server-side RDDI interface, running on the Central Node.

Interface provided: the interface mirrors that of the RDSI-Server.

Used by: the RDS-Logic library.

RDDI	Description
getOpenStudies	Local call corresponding to the same remote RDDI operation.

Operation getOpenStudies

The local call mirrors the eponymous remote operation defined in section 4.

7. RDD ACCESS LIBRARY

Summary: the goal of the RDD Access library is to provide services for accessing and manipulating structured RDD content. This involves the syntactic validation of RDDs, as well as high-level data access that allows applications to retrieve RDD content while hiding data representations details from them.

Interface provided: RDDAccessI

Used by: the RDS-Logic library on the client side, the RDD Server on the server side.

RDDAccessI	Description
validateResource	Validate a FHIR resource against the RDS-IG.

Table 6 - Summary of operations of the RDDAccess Interface

7.1. RDD Validation

The RDD Validation library provides operations that can be used to validate FHIR resources against the RDS Implementation Guide. This includes a check if the structure and the selected code values of the resources are valid. The following figure shows the UML class diagram representing the classes provided by the library.

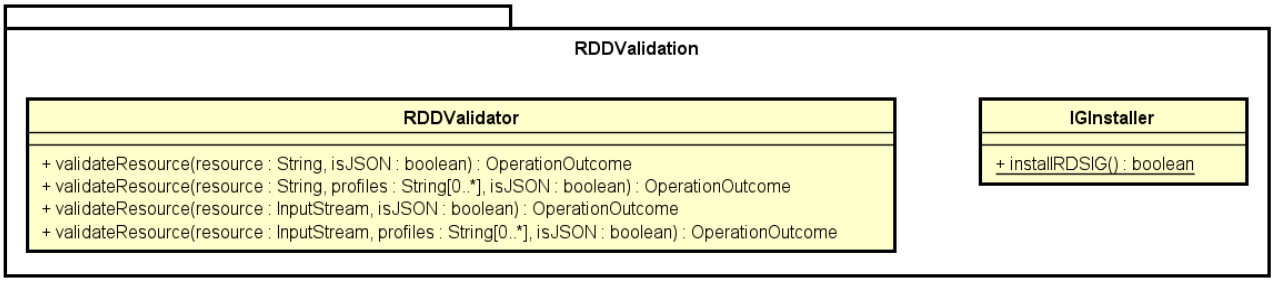

Figure 3 - Components and operations of the RDD Validation library

The library's validateResource operation is used to validate FHIR resources and supports JSON and XML formats. If an instance of this class is created, it will first check if the package is already installed and install the contained version when necessary. If it finds the package, it will not check if the installed version is the same as the contained one. If it is necessary to update the package it can be manually deleted and will be reinstalled during the next validation. Alternatively, the installRDSIG operation of the IGInstaller class can be used to overwrite it. The following table provides a description of the exposed operation.

Operation validateResource

Name	**validateResource**
Description	Validate a FHIR-Resource against the RDS-IG [D2.9]. It can validate resources in JSON and XML formats.

Arguments	• resource: the resource that will be validated. The resource can be provided in the following formats:
	  ○ String
	  ○ InputStream
	• profiles (optional, not recommended): a list of Strings, that contains profiles the resource should be validated against. If no profiles are provided, it will be validated against the profiles listed in the Resource.meta.profile attribute
	• isJSON: a boolean that specifies the resource format. If the boolean is true the input is treated as a JSON file, otherwise it is treated as an XML file

Return Value	On success, the operation will return an OperationOutcome. This OperationOutcome contains all errors, warnings, and notes that were found during validation.
Exceptions	• IOException: caused by malformed resources. • EOperationOutcome exception: caused by the failure to create the OperationOutcome.
Preconditions	• The device has access to the internet and is connected to the terminology server.

7.2. RDD Access

The goal of the RDD Access library is to hide format, version, and implementation details with respect to reading data from Research Definition Documents. Instead of having to parse RDDs directly, applications can use the RDD Access library to read relevant data elements from it.

The library will be described in detail in v2 of this deliverable.

8. SECURITY LIBRARIES

In the context of the RDS scenario, the security functionalities are implemented through two libraries: one used by the S-EHR App on the mobile device, invoked by the RDS-Logic component, and the other one used on the server side, invoked by the RDDI-Server and RSI-Server components. The mobile-side library, called M-RDS-SM, includes functionalities for key agreement, digital signature, and encryption, while the server-side library, T-RDS-SM, provides operations for key agreement, digital signature, and decryption.

8.1. Mobile RDS Security Management Library (M-RDS-SM)

Summary: provides security functionalities to S-EHR App and calls external security provider services (e.g. CA, eIDASI and PP).

Interface provided: MRDSI-Security

Used by: the RDS-Logic library inside the mobile device.

The M-RDS-SM library is shared by the S-EHR App components that implement the Protocol. Through the *MRDSI-Security* interface, it provides essential services for protocol security, such as digital signature, AES encryption and Diffie-Hellman key agreement.

MRDSI-Security	Description
fetchCertificate	Storage of the Certificate in Android keystore.
signPayload	Digitally sign the input content using an asymmetric private key.
verifySignature	Verify a digital signature using an asymmetric public key.
aliceInitKeyPair	Create a Diffie-Hellman key pair (to be used by the S-EHR App)
aliceKeyAgreement	Create and initialize a Diffie-Hellman KeyAgreement object
aliceKeyAgreementFin	Generate a shared secret.
alicePubKeyEnc	Encode the public key given in input.
generateSymmetricKey	Generate an AES session key that will be used for encrypted communication between the RRC and the S-EHR App.
encrypt	Encrypt using AES.

Table 7 - Summary of operations of the MRDSI-Security Interface

Operation fetchCertificate

Name	**fetchCertificate**
Description	This call generates an X.509 certificate signed by the citizen's CA upon a CSR request. The Certificate is stored in the keystore of the device.
Caller	RDS-Logic
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

Operation signPayload

Name	**signPayload**
Description	Digitally sign the input content using an asymmetric private key.
Caller	RDS-Logic
Arguments	• content: the content to be signed • privateKey: the private key used for the signature
Return Value	The signed content, represented as a single string.
Exceptions	• IOException • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException
Preconditions	• successfully fetched credentials from a CA

Operation verifySignature

Name	**verifySignature**
Description	Verify a digital signature using an asymmetric public key.
Caller	RDS-Logic

Arguments	• signedContent: the signed content in which the signature is to be verified • publicKey: the public key used for verification
Return Value	A boolean value, “true” meaning that the signature was successfully verified.
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• successfully fetched credentials from a CA

Operation aliceInitKeyPair

Name	**aliceInitKeyPair**
Description	Create a Diffie-Hellman key pair (to be used by the S-EHR App).
Caller	RDSI-Server
Arguments	-
Return Value	KeyPair aliceKpair: Diffie Hellman key pair
Exceptions	• Exception[1]
Preconditions	• N/A

Operation aliceKeyAgreement

Name	**aliceKeyAgreement**
Description	Create and initialize a Diffie-Hellman KeyAgreement object.
Caller	RDSI-Server
Arguments	• KeyPair aliceKpair: Diffie Hellman key pair
Return Value	KeyAgreement aliceKeyAgree: Diffie HellmanKeyAgreement object
Exceptions	• Exception[1]
Preconditions	• Successful Diffie Hellman key pair generation

Operation aliceKeyAgreementFin

Name	**aliceKeyAgreementFin**
Description	Generate a shared secret.
Caller	RDSI-Server
Arguments	• byte[] bobPubKeyEnc: Encoded public key • KeyAgreement aliceKeyAgree: Diffie Hellman KeyAgreement object
Return Value	KeyAgreement keyagreement: The same shared secret is generated
Exceptions	• Exception[1]
Preconditions	• Successful generation of Diffie Hellman key agreement

Operation alicePubKeyEnc

Name	**alicePubKeyEnc**
Description	Encode the public key given in input.
Caller	RDSI-Server
Arguments	• KeyPair aliceKpair: Diffie Hellman key pair
Return Value	byte[] alicePubKeyEnc: Encoded public key
Exceptions	• Exception[1]
Preconditions	• Key pair successfully created

Operation generateSymmetricKey

Name	**generateSymmetricKey**
Description	Generate an AES session key that will be used for encrypted communication between the RRC and the S-EHR App.
Caller	RDS-Logic
Arguments	• byte[] sharedSecret: generated shared secret • int size: Size in bits of the key pair
Return Value	SecretKeySpec symKey: the symmetric key generated.

Exceptions	• Exception[1]
Preconditions	• The same secret is generated

Operation encrypt

Name	**Encrypt**
Description	Encrypt using AES.
Caller	RDS-Logic
Arguments	• String payload: the content to be encrypted. • String symKey: the symmetric key to be used for encryption.
Return Value	String cipher: the encrypted payload.
Exceptions	• Exception[1]
Preconditions	• Symmetric key agreement established

8.2. Terminal RDS Security Management Library (T-RDS-SM)

Summary: provides security functionalities to both the Central Node and Research Centre and calls external security provider services (e.g. the CA).

Interface provided: TRDSI-Security

Used by: the RDDI-Server library inside the Central Node and the RDSI-Server library inside the Research Centres.

The T-RDS-SM library is shared by the *Central Node* and *Research Center* components that implement the Protocol. Through the *TRDSI-Security* interface, it provides essential services for protocol security, such as digital signature, AES decryption and Diffie-Hellman key agreement.

TRDSI-Security	Caller	Description
fetchCertificate	RDDI-Server, RSI-Server	Storage of Certificate in the keystore
signPayload	RDDI-Server, RSI-Server	Digitally sign the input content using an asymmetric private key
verifySignature	RSI-Server	Verify a digital signature using an asymmetric public key
bobInitKeyPair	RSI-Server	Create a Diffie-Hellman key pair (to be used by the RRC)
bobKeyAgreement	RSI-Server	The RRC creates and initializes its DH KeyAgreement object
bobPubKeyEnc	RSI-Server	The RRC encodes its public key, and sends it over to S-EHR App.

bobKeyAgreementFin	RSI-Server	The RRC generates the (same) shared secret.
generateSymmetricKey	RSI-Server	AES session key that will be used for encrypted communication between the RRC and the S-EHR App
decrypt	RSI-Server	AES decryption method

Table 8 - Summary of operations of the TRDSI-Security Interface

Operation fetchCertificate

Name	**fetchCertificate**
Description	This call generates an X.509 certificate signed by the server's CA upon a CSR request.
Caller	RDS-Logic
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	• NoSuchProviderException • NoSuchAlgorithmException
Preconditions	• Internet Connection • Public/Private key generation and storage in keystore

Operation signPayload

Name	**signPayload**
Description	Digitally sign the input content using an asymmetric private key.
Caller	RDDI-Server, RDSI-Server
Arguments	• content: the content to be signed • privateKey: the private key used for the signature
Return Value	The signed content, represented as a single string.

Exceptions	• IOException • SignatureException • InvalidKeyException • NoSuchAlgorithmException • InvalidKeySpecException
Preconditions	• None

Operation verifySignature

Name	**verifySignature**
Description	Verify a digital signature using an asymmetric public key.
Caller	RDSI-Server
Arguments	• signedContent: the signed content in which the signature is to be verified • publicKey: the public key used for verification
Return Value	A boolean value, “true” meaning that the signature was successfully verified.
Exceptions	• UnsupportedEncodingException • NoSuchAlgorithmException • InvalidKeyException • SignatureException
Preconditions	• None

Operation bobInitKeyPair

Name	**bobInitKeyPair**
Description	Create a Diffie-Hellman key pair (to be used by the RRC).
Caller	RDS-Logic
Arguments	• byte[] alicePubKeyEnc: Encoded public key
Return Value	KeyPair bobKpair: Diffie Hellman key pair generation
Exceptions	• Exception[1]
Preconditions	• Successful generation of public key

Operation bobKeyAgreement

Name	**bobKeyAgreement**
Description	Create and initialize a Diffie-Hellman KeyAgreement object.
Caller	RDS-Logic
Arguments	• KeyPair bobKpair: Diffie Hellman key pair
Return Value	KeyAgreement bobKeyAgree: Diffie Hellman key agreement object
Exceptions	• Exception[1]
Preconditions	• Successful generation of Diffie Hellman key agreement object

Operation bobKeyAgreementFin

Name	**bobKeyAgreementFin**
Description	Generate a shared secret.
Caller	RDS-Logic
Arguments	• PublicKey alicePubKey: Public key • KeyAgreement bobKeyAgree: Diffie Hellman key agreement object
Return Value	KeyAgreement keyagreement: The same shared secret is generated
Exceptions	• Exception[1]
Preconditions	• Successful generation of Diffie Hellman key agreement

Operation bobPubKeyEnc

Name	**bobPubKeyEnc**
Description	Encode the public key given in input.
Caller	RDS-Logic
Arguments	• KeyPair bobKpair: Diffie Hellman key pair
Return Value	byte[] bobPubKeyEnc: Encoded public key
Exceptions	• Exception[1]

Preconditions	• Key pair successfully created

Operation generateSymmetricKey

Name	**generateSymmetricKey**
Description	Generate an AES session key that will be used for encrypted communication between the RRC and the S-EHR App.
Caller	RDSI-Server
Arguments	• byte[] sharedSecret: generated shared secret • int size: Size in bits of the key pair
Return Value	SecretKeySpec symKey: the symmetric key generated.
Exceptions	• Exception[1]
Preconditions	• The same secret is generated

Operation decrypt

Name	**Decrypt**
Description	Decrypt using AES.
Caller	RDSI-Server
Arguments	• String encryptedPayload: the encrypted content. • String symKey: the symmetric key used for decryption.
Return Value	String plainContent: the decrypted payload.
Exceptions	• Exception[1]
Preconditions	• Symmetric key agreement established

9. LIBRARY INTERACTIONS

This section presents the interactions of the libraries described in the deliverable in order to implement the RDS Protocol phases defined and described in detail in [D4.8]. The phases are:

- **OPT-IN:** the Citizen accepts to be solicited to participate in future research studies;
- **OPT-OUT:** the Citizen decides not to be solicited anymore for future studies;
- **ENROLLMENT:** the eligibility of the Citizen to a newly published study is checked; in the positive case, the consent of the Citizen is asked, and the consenting Citizen is enrolled into the study;
- **DATA RETRIEVAL:** for an ongoing study in which the Citizen is enrolled, health data is retrieved from his/her mobile device and sent to his/her Reference Research Centre;
- **WITHDRAWAL:** the Citizen decides to interrupt his/her participation in an ongoing research study.

In addition, the **SETUP** and **PUBLISHING** phases are included below that, while not strictly part of the RDS Protocol, are a prerequisite for fulfilling the preconditions of the operations in the other phases.

9.1. Setup Phase

In this phase, public key certificates are retrieved from the Certification Authority for all three kinds of peers involved in the RDS Protocol: S-EHR Apps, the Central Node, and Research Centres. The S-EHR App also retrieves the public key of the Central Node in order to be able to verify the integrity of RDDs downloaded in the ENROLLMENT phase.

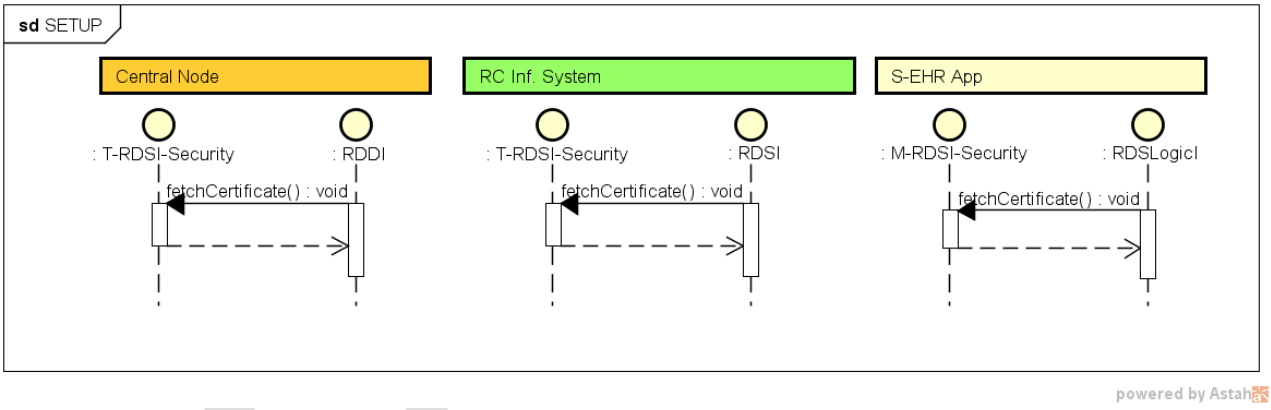

Figure 4 - Sequence diagram of the SETUP phase

9.2. Publishing Phase

Below is the sequence diagram of the RDD PUBLISHING process, happening on the Central Node. In this phase, the *PI of a study* submits a study proposal, in the form of an RDD, to the Central Node using a tool that is outside of the scope of this deliverable: for example, a web portal---we will call this tool the *Central Node Portal*. The Central Node Portal will exploit the libraries described in this deliverable. The Central Node validates the RDD from a syntactic point of view. Then the *Central Node Administrator* also manually reviews the non-syntactic aspects of the proposal and, if no issues are detected, he/she publishes the digitally signed RDD on the Central Node.

The following libraries participate in the process:

- the RDDI Server where Research Definition Documents are published;

- the TRDSI-Security library for the digital signature of RDDs;
- the RDDAccess library for the syntactic validation of RDDs.

The process is as follows:

1. the PI of the Study submits a new RDD through the Portal;
2. the CN verifies the syntactic validity of the RDD before accepting the submission;
3. the CN Administrator reviews and officially accepts the submission;
4. the RDD is digitally signed by the CN and is published on the CN.

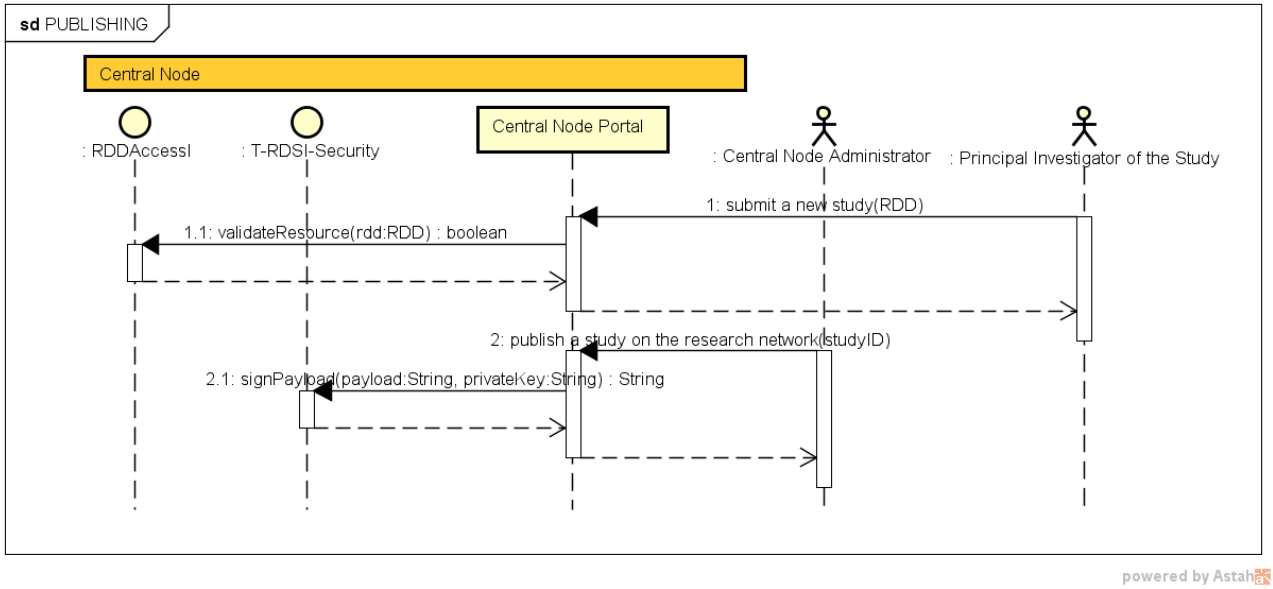

Figure 5 - Sequence diagram of the PUBLISHING phase

9.3. Opt-In and Opt-Out Phases

Below is the sequence diagram of the OPT-IN and OPT-OUT functionalities implemented on the mobile device. The diagram depicts the interaction of the RDS-Logic library (implementing the RDS Protocol logic inside the S-EHR App) and the core S-EHR Application that controls the UI through which the Citizen interacts. Upon OPT-OUT, the Citizen is notified of ongoing studies from which he/she is not withdrawn automatically.

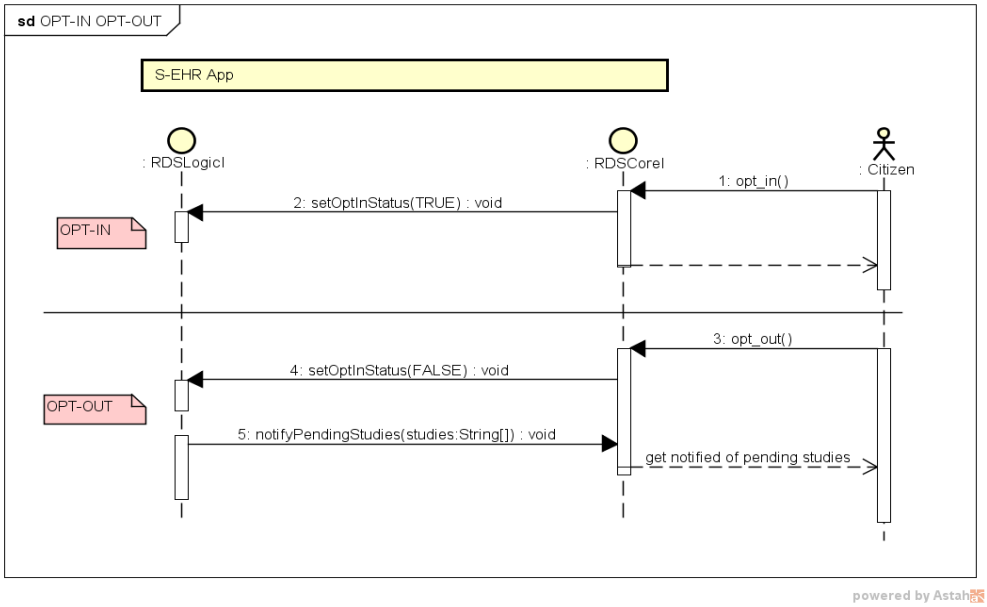

Figure 6 - Sequence diagram of the OPT-IN and OPT-OUT phases

9.4. Enrolment Phase

Below is the sequence diagram of the ENROLLMENT phase of the RDS Protocol. The diagram depicts the interaction of the RDS-Logic library (implementing the RDS Protocol logic inside the S-EHR App) with:

- the core S-EHR Application that controls the UI through which the Citizen interacts;
- the anonymization library for the purpose of generating a pseudonym or a pseudo-identity;
- the server-side and mobile-side security libraries that implement digital signature and encryption;
- the RDDI Server from which Research Definition Documents are downloaded, and the corresponding Client library;
- the RDSI Server library (running inside the Research Centre Information System) that registers enrolment requests, and with which the S-EHR App communicates through the corresponding Client library;
- the RDDAccess library for syntactically validating the RDD downloaded.

The process is as follows:

1. at regular intervals, the S-EHR App downloads the RDDs of open studies from the RDD Server;
2. the integrity of the RDDs downloaded is checked through verification of the digital signature and the syntactic validity of the RDD;
3. by checking the enrolment criteria, the S-EHR App verifies the eligibility of the Citizen to participate in any of the new studies;
4. eligible Citizens are asked for their explicit consent to participate in the study, as well as to choose a Reference Research Centre (RRC) to which they will be attached;
5. a pseudo-identifier is generated for the Citizen for future use:
    a. either a pseudo-identity is retrieved from the RRC,
    b. or a pseudonym is retrieved from the Pseudonym Provider, where the Citizen is authenticated through a SAML assertion previously retrieved from the eIDAS Node;

6. an encryption key pair is generated between the RRC and the S-EHR App in order to ensure subsequent secure exchange of data;
7. the consent document is digitally signed by the Citizen;
8. the signed consent is sent to the RRC;
9. the RRC, in turn, signs the consent and sends back a copy to the Citizen.

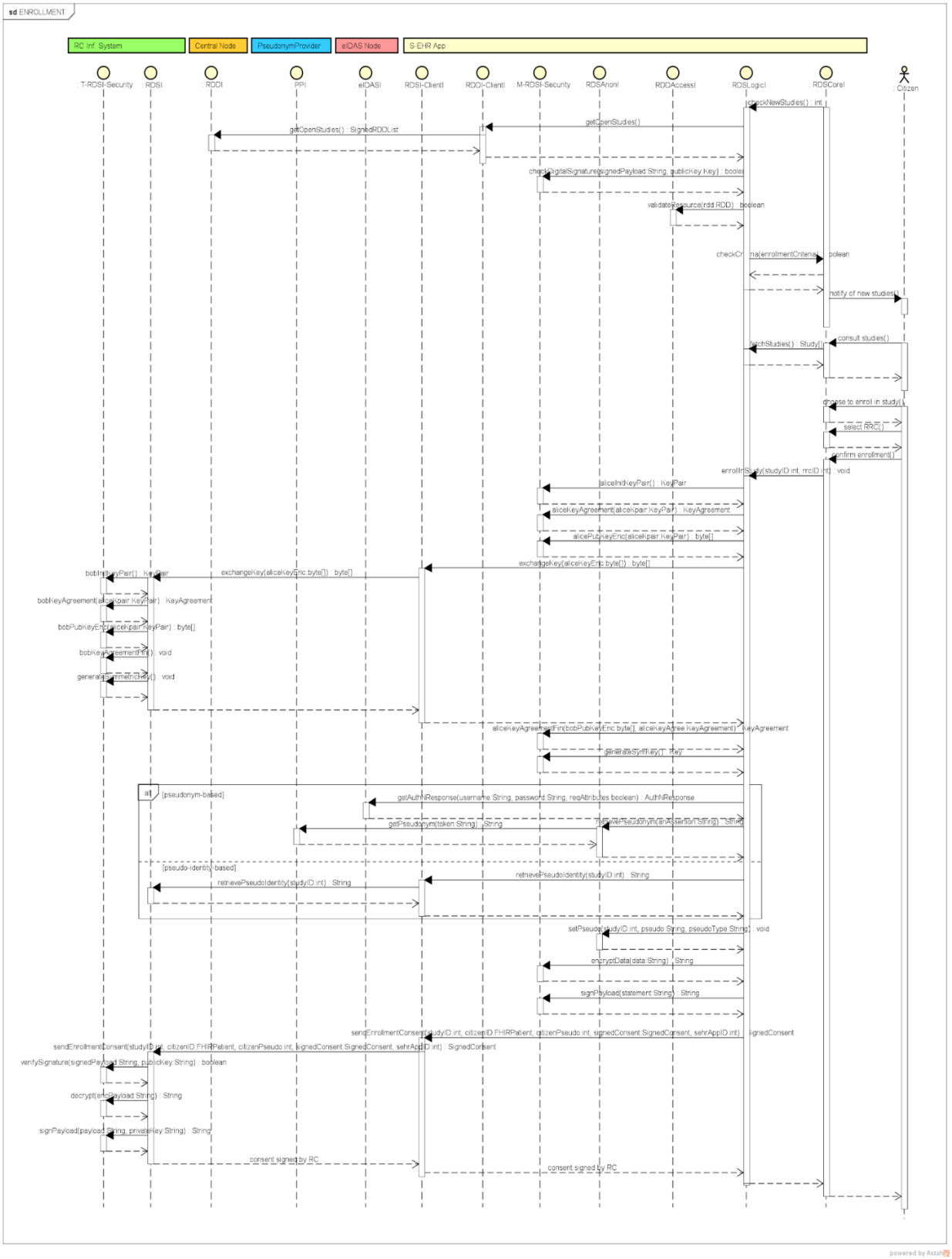

Figure 7 - Sequence diagram of the ENROLLMENT phase

9.5. Data Retrieval Phase

Below is the sequence diagram of the DATA RETRIEVAL phase of the RDS Protocol. The diagram depicts the interaction of the RDS-Logic library (implementing the RDS Protocol logic inside the S-EHR App) with:

- the core S-EHR Application that controls the UI through which the Citizen interacts;
- the anonymization library for the purpose of generating a pseudonym or a pseudo-identity;
- the security library implementing digital signature and encryption functionalities;
- the RDSI Server library (running inside the Research Centre Information System) to which health data is sent, and with which the S-EHR App communicates through the corresponding Client library.

The process is as follows:

1. the exit criteria are regularly checked to ensure that the Citizen still conforms to the study requirements;
2. in case of non-conformance, the Citizen exits the study and the RRC is informed of this fact;
3. at intervals defined in the RDD, health data are queried from the Citizen's smart health record;
4. the data retrieved are pseudo-anonymized;
5. the data are encrypted and digitally signed;
6. the pseudo-anonymized, encrypted, and signed data are sent to the RRC which acknowledges reception;
7. the Citizen is notified of the successful data retrieval operation.

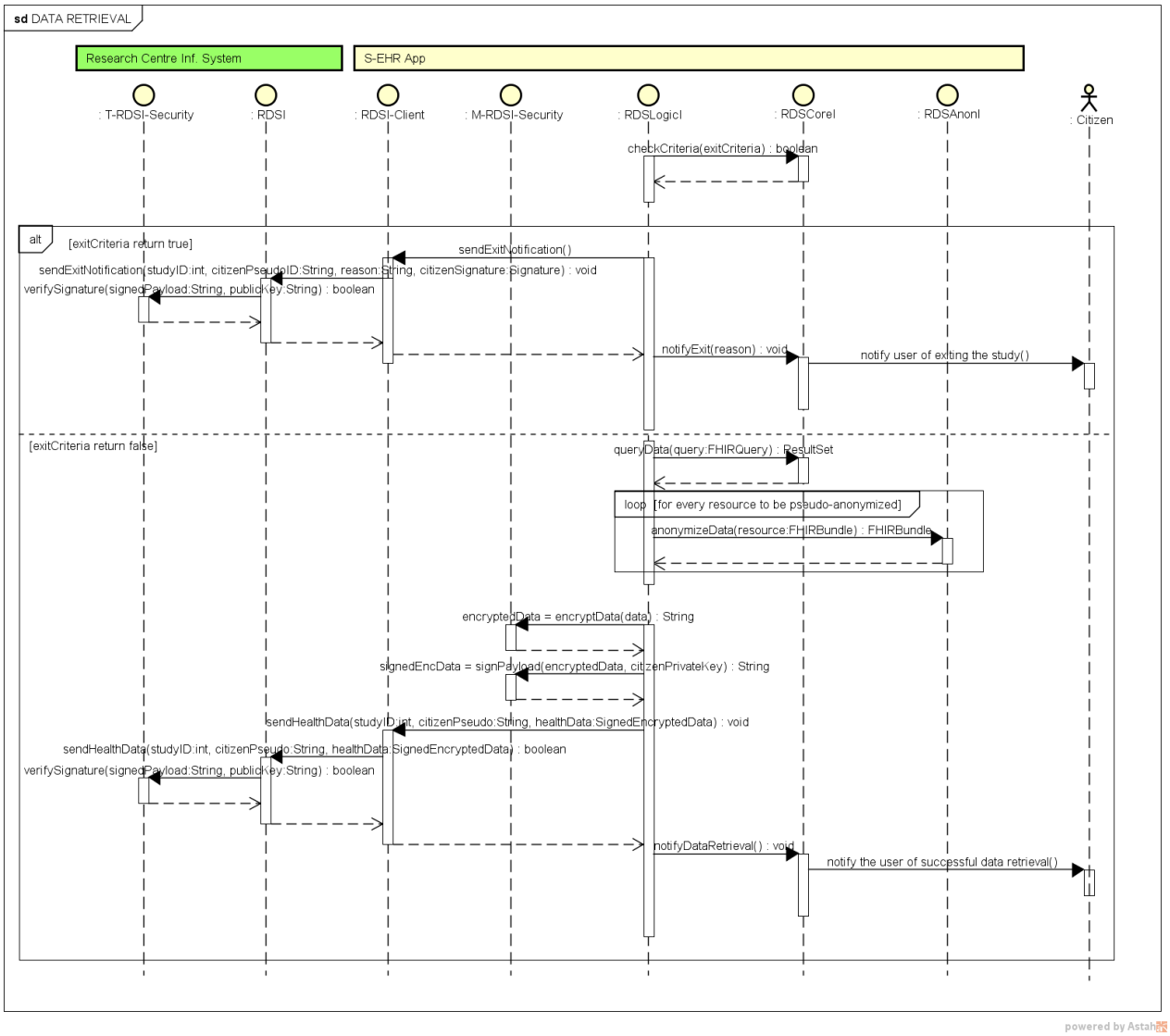

Figure 8 - Sequence diagram of the DATA RETRIEVAL phase

9.6. Withdrawal Phase

Below is the sequence diagram of the WITHDRAWAL phase of the RDS Protocol. The diagram depicts the interaction of the RDS-Logic library (implementing the RDS Protocol logic inside the S-EHR App) with:

- the core S-EHR Application that controls the UI through which the Citizen interacts;
- the security library implementing digital signature and encryption functionalities;
- the RDSI Server library (running inside the Research Centre Information System) to which the withdrawal is sent, and with which the S-EHR App communicates through the corresponding Client library;
- the TRDS-Security library, used on the server side to verify the digital signature.

The process is as follows:

1. the Citizen tells the S-EHR App through the UI of the withdrawal request;
2. the withdrawal request is digitally signed;
3. the withdrawal request is sent to the RRC who verifies the signature, acknowledges the withdrawal, and updates its own records;
4. the Citizen is notified of the success of the process.

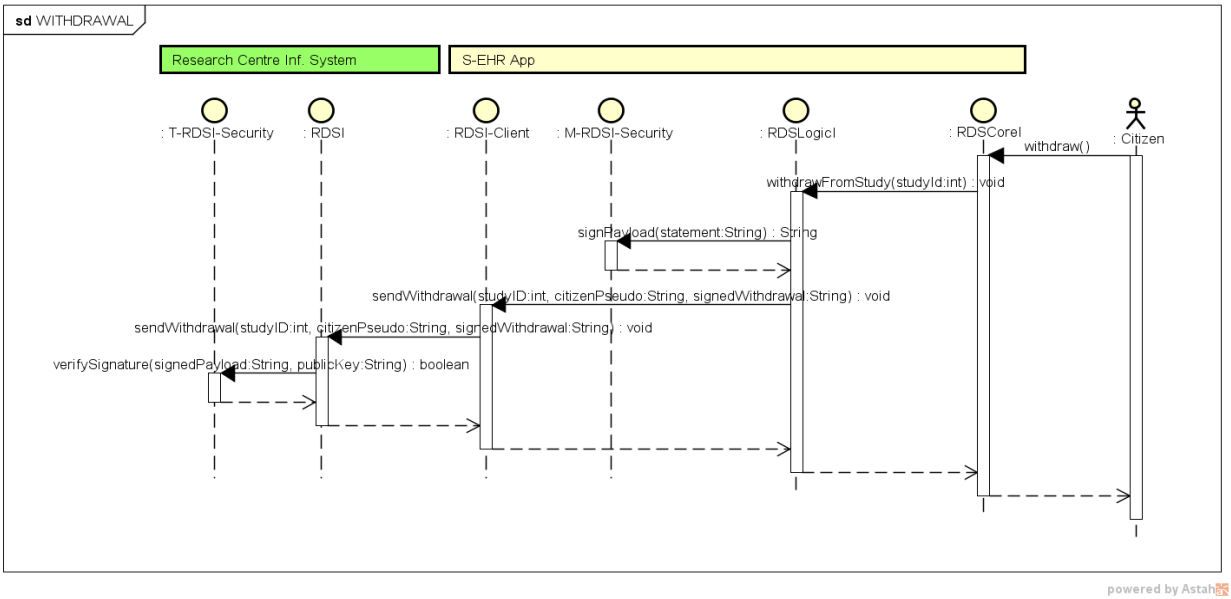

Figure 9 - Sequence diagram of the WITHDRAWAL phase

10. CONCLUSIONS AND NEXT STEPS

The contents of this document initially serve as guidance to producing reference implementations and demonstrators developed in the InteropEHRate project to showcase the Research Data Sharing Protocol [D4.8] and the underlying interoperability infrastructure. Based on this implementation experience, a consolidated version 2 of this deliverable will be produced, providing technical specifications for future applications and describing the InteropEHRate reference implementations.

Accordingly, the current version 1 of this deliverable provides a first design of the libraries based on an initial phase of software development. We expect its second version to introduce more implementation details and possibly minor modifications, based on implementation experience gathered in the meantime. We already foresee the following future extensions:

- The second version of the RDS Protocol (described in [D4.9] and of this deliverable will introduce *research questionnaires* that will be included in the RDD, proposed to participating citizens, and the answers to which will be sent to research centres.
- Accordingly, the contents of the RDD (defined separately in [D2.9]) and the corresponding *RDD Access* library will also need to evolve.
- The RDD Access library, currently in progress, will be fully specified.
- We expect evolution of the anonymization requirements and design: due to the novelty and complexity of anonymization executed on mobile devices, the approach underlying the solution described in this deliverable is not yet fully consolidated, and will be subject to further research.
- The identification of S-EHR App *products* may be necessary for *a posteriori* traceability of data contributions, therefore this information may be sent to research centres in the enrolment phase.
- The enrolment request currently does not send citizen identification data to research centres, only pseudo-identification. However, in exceptional cases of emergency (such as the discovery of a serious medical condition based on the health data shared), the research centre may need to contact citizens directly, and for this may need personal contact details. The sharing of such details will be considered for inclusion in the next version of this deliverable.
- In relation to the previous point, *reverse pseudonymisation* will be considered for the next version, where the Pseudonym Provider will maintain and provide to research centres the identity of citizens in specific cases of emergency.
- The usage of interaction with the CA will also be included in the next version.
- The second version of the RDS protocol will also consider the case if the RC decides that the citizen should exit from the study (e.g. for health complications or complex exit criteria that cannot be checked by the S-EHR App).

REFERENCES

- **[D2.3]** InteropEHRate Consortium, *D2.3-Requirements Specification V3*, 2021. www.interopehrate.eu/resources/#dels
- **[D2.9]** InteropEHRate Consortium, *D2.9-FHIR profile for EHR interoperability v3*, 2021. 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
- **[D4.9]** InteropEHRate Consortium, *D4.9-Specification of protocol and APIs for research health data sharing - V2*, 2021. www.interopehrate.eu/resources/#dels