Horizon 2020 Programme

H2020-SU-TDS-02-2018
Trusted digital solutions and Cybersecurity in Health and Care

ProTego

DATA-PROTECTION TOOLKIT REDUCING RISKS IN HOSPITALS AND CARE CENTERS

Project Nº 826284

ProTego

D5.3 *Description of the final cybersecurity risk mitigation tools*

Responsible:	IBM
Contributors:	IBM, UAH, IMEC, KUL
Dissemination Level:	Public
Version:	1.0
Date:	30/06/2021

Executive summary

This report summarizes the work done by WP5 Cybersecurity Risk Mitigation Tools over the course of the project.

In particular, while presenting an overall view of the complete solution developed by this work package, this deliverable will give particular emphasis to the efforts done in the second half of the project and supplements the D5.2 deliverable which was released at project month eighteen.

This work package focuses on supplying an end-to-end solution for securely exchanging medical information between hospital data centers and end locations, which may be remote care locations or authorized end user devices, such as tablets or smart phones. As such, this work package looks at securing data-in-motion through the development of network slicing technologies, data-at-rest through the development and use of Apache Parquet file encryption, Attribute-Based Encryption (ABE) and its integration into access control schemes, and data-in-process using hardware enclaves such as Full Memory Encryption (FME) enclaves, such as AMD SEV.

Work package 5 has provided ProTego and the scientific community in general, with innovations to protect data. While healthcare and the hospital environment were chosen as our main focus, these innovations can be applied a wide range of other use cases.

Work package 5 was instrumental in the development and acceptance of the Apache Spark Parquet Modular Encryption standard and help spearhead the development of the official code release. Additionally, advanced data protection topics in this work package are contributing to IBM's Open-Source development of "The Mesh for Data".

Two security schemes were developed for ProTego. The CP-ABAC scheme offers security against a compromised access control and key management system, while still allowing to enforce fine-grained access control policies during read requests on sensitive medical data. An alternative solution, C-ABAC, was studied as well to support the revocation of users' attributes in addition to the security properties already offered by C-ABAC. Both schemes offer high security guarantees and are suitable to be applied in the use cases of the ProTego project.

The CA module contributes to the ProTego project mobile device security through a scalable architecture to continuously authenticate users based on their interactions with their mobile phones. Research contributes to the state of the art with an increase in the accuracy and other target metrics using keystroke mechanics with the soft keyboards of mobile devices.

The research done in network slicing proposes to use state-of-the-art airtime-based network slicing to provide 5G-like performance isolation for mission critical applications in the hospitals.

Contributors Table

DOCUMENT SECTION	AUTHOR(S)	REVIEWER(S)
I	IBM	Colin Upstill (ICE), Esteban Municio (IMEC), Antonio Jesús Gamito González (Inetum), Fernando José Rendón Segador (Inetum), Luis Carrascal (Inetum)
II	IBM	Colin Upstill (ICE), Esteban Municio (IMEC), Antonio Jesús Gamito González (Inetum), Fernando José Rendón Segador (Inetum), Luis Carrascal (Inetum)
III	KUL	Colin Upstill (ICE), Eliot Salant (IBM), Antonio Jesús Gamito González (Inetum), Fernando José Rendón Segador (Inetum), Luis Carrascal (Inetum)
IV	UAH	Colin Upstill (ICE), Farhad Sighili (KUL), Antonio Jesús Gamito González (Inetum), Fernando José Rendón Segador (Inetum), Luis Carrascal (Inetum)
V	IMEC	Colin Upstill (ICE), Carlos Cilleruelo (UAH), Antonio Jesús Gamito González (Inetum), Fernando José Rendón Segador (Inetum), Luis Carrascal (Inetum)

Table of Contents

Table of Figures

Table of Acronyms and Definitions

Acronym	Definition
ABE	Attribute Based Encryption
API	Application Programming Interface
BYOD	Bring Your Own Device
DEK	Data encryption key
DGW	Data Gateway
DPIA	Data Protection Impact Assessment
EHR	Electronic Health Record
EDR	Endpoint Detection and Response
FHIR	Fast Healthcare Interoperability Resources
FME	Full Memory Encryption
GDPR	General Data Protection Regulation
HL7	Health Level 7
JWT	Java Web Token
KEK	Key encryption key
KMS	Key management service
k8s	Kubernetes
MPLS	Multiprotocol Label Switching
M4D	The Mesh for Data
MEK	Master encryption key
OS	Operating System
QoS	Quality of service
SDN	Software defined network
SEV	Secure Encrypted Virtualization
SGX	Software Guard Extensions
SIEM	Security Information and Event Management
SSL	Secure Socket Layer
TLS	Transport Layer Security
VLAN	Virtual Local Area Network

VPN	Virtual Private Network
VM	Virtual Machine
WP	Work package

I. INTRODUCTION

Work Package 5, Cybersecurity Risk Mitigation Tools, focuses on supplying an end-to-end solution for securely exchanging medical information between hospital data centers and end locations, which may be remote care locations or authorized end user devices, such as tablets or smart phones. As such, this work package looks at securing data-in-motion through the development of network slicing technologies, data-at-rest through the development and use of Apache Parquet file encryption and Attribute-Based Encryption (ABE), and data-in-process through the use of hardware enclaves such as Full Memory Encryption (FME) enclaves, like AMD SEV. WP5 also provides a framework to manage authentication of both users and software entities in the system, as well as access control to sensitive medical data.

Leveraging emerging technologies such as 5G networks and the HL7 FHIR standard for electronic health records, WP5 supplies a key part of the infrastructure upon which ProTego's use cases run. In addition, WP5 goes beyond infrastructure creation and tackles issues of both usability and deriving extra value from the existing data – for example, by supplying a framework for performing analytics on the stored medical data.

WP5 has been designed to provide WP4, Cyber Risk Assessment Tools, with the SIEM input that it will require for Risk Analysis when suspected tampering of stored encrypted Parquet files is detected or the Access Control Framework (Section III) returns an access violation.

II. Trusted Data Gateways and Exchanges (IBM)

II.1.1. Overall achievements

In the second half of the project, this task rounded out its work on the advanced encryption of medical data, adding an innovative double key wrapping to its Open-Source work on Parquet Modular Encryption including a comprehensive performance study. We then showed how our solution for the Data Gateway can be run in a secure hardware partition to provide protection for data-in-process. Advance security topics were pursued by contributing to the development of the on-going development work IBM is leading in The Mesh for Data.

II.1.2. Second half work

II.1.2.a. Secure Enclaves (Trusted Execution Environments, TEEs)

Background:

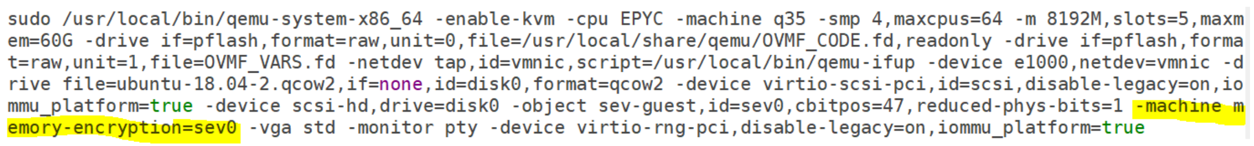
Our planned work on Full Memory Encryption (FME) enclaves in the second half of the project aims to protect data-in-process and complements our work on Parquet encryption for data-at-rest.

Intel's Software Guard Extensions (SGX) came out in 2015 and introduced hardware support for the execution of relatively small functions inside of designated enclaves. However, due to a number of issues such as the need to rewrite applications to work with SGX and the limitations on enclave size, commercial adaptation of SGX was very limited.

AMD's Secure Encrypted Virtualization (SEV) integrates main memory encryption with AMD-V virtualization architecture to support encrypted virtual machines. Encrypted virtual machines are protected from snooping not only from other guest VMs on the system, but also from the hypervisor itself, which typically runs at a higher Linux privilege level. Therefore, code running inside an SEV-encrypted VM is safe from both physical attacks on the host machine (e.g. the removal and subsequent dumping of non-volatile, DVDIMM memory), and from attacks from a malicious hypervisor on the host machine. In addition to hardware support, operating system support is also required for SEV. As opposed to Intel SGX technology, software running on SEV does not require any changes to operate.

In the second half of the project, we experimented with SEV in order to demonstrate that it indeed supports our ProTego components.

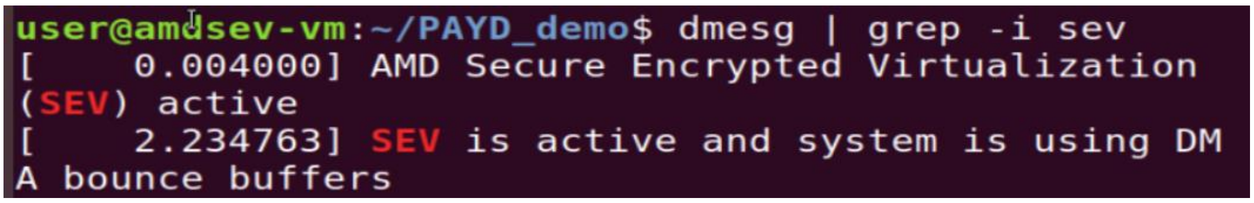
In order to initiate a SEV VM on Ubuntu, the following command was used:



```
sudo /usr/local/bin/qemu-system-x86_64 -enable-kvm -cpu EPYC -machine q35 -smp 4,maxcpus=64 -m 8192M,slots=5,maxm
em=60G -drive if=pflash,format=raw,unit=0,file=/usr/local/share/qemu/OVMF_CODE.fd,readonly -drive if=pflash,forma
t=raw,unit=1,file=OVMF_VARS.fd -netdev tap,id=vmnic,script=/usr/local/bin/qemu-ifup -device e1000,netdev=vmnic -d
rive file=ubuntu-18.04-2.qcow2,if=none,id=disk0,format=qcow2 -device virtio-scsi-pci,id=scsi,disable-legacy=on,io
mmu_platform=true -device scsi-hd,drive=disk0 -object sev-guest,id=sev0,cbitpos=47,reduced-phys-bits=1 -machine m
emory-encryption=sev0 -vga std -monitor pty -device virtio-rng-pci,disable-legacy=on,iommu_platform=true
```

Figure 1: Starting up a SEV secure enclave on Ubuntu

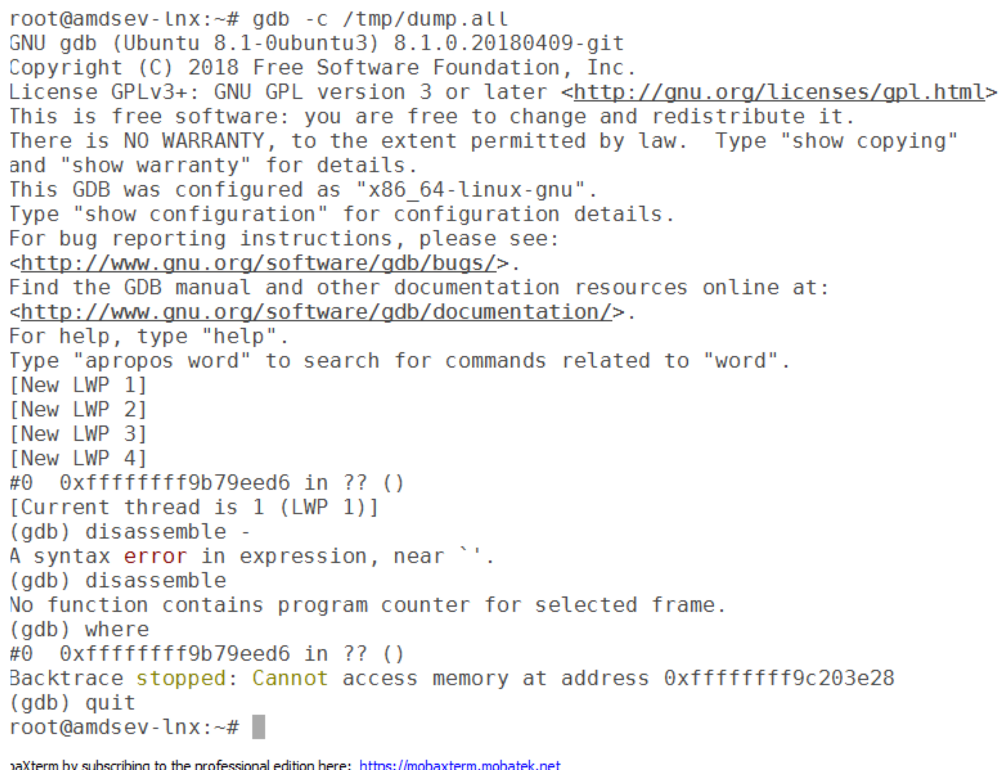
We able to confirm that SEV is active in the VM as shown below:



```
user@amdsev-vm:~/PAYD_demo$ dmesg | grep -i sev
[    0.004000] AMD Secure Encrypted Virtualization
(SEV) active
[    2.234763] SEV is active and system is using DM
A bounce buffers
```

Figure 2: Verification that SEV is active

Attempts to snoop into memory in this secure enclave failed, as can be seen below.

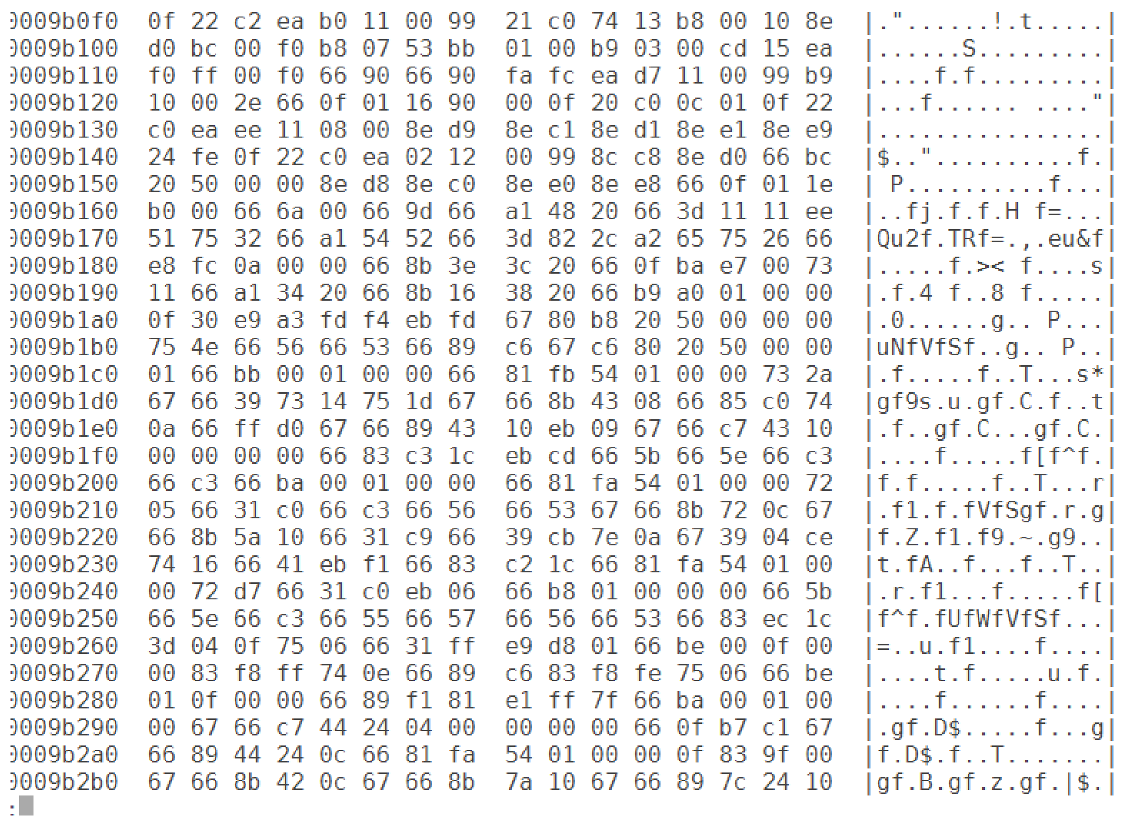


```
root@amdsev-lnx:~# gdb -c /tmp/dump.all
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
[New LWP 1]
[New LWP 2]
[New LWP 3]
[New LWP 4]
#0  0xffffffff9b79eed6 in ?? ()
[Current thread is 1 (LWP 1)]
(gdb) disassemble -
A syntax error in expression, near `'.
(gdb) disassemble
No function contains program counter for selected frame.
(gdb) where
#0  0xffffffff9b79eed6 in ?? ()
Backtrace stopped: Cannot access memory at address 0xffffffff9c203e28
(gdb) quit
root@amdsev-lnx:~# 

aXterm by subscribing to the professional edition here:  https://mobaxterm.mobatek.net
```

Figure 3: Failure to snoop memory in secure enclave

This can be contrasted with the same command, executed outside of a secure enclave:



```
0009b0f0  0f 22 c2 ea b0 11 00 99  21 c0 74 13 b8 00 10 8e  |."......!.t.....|
0009b100  d0 bc 00 f0 b8 07 53 bb  01 00 b9 03 00 cd 15 ea  |......S.........|
0009b110  f0 ff 00 f0 66 90 66 90  fa fc ea d7 11 00 99 b9  |....f.f.........|
0009b120  10 00 2e 66 0f 01 16 90  00 0f 20 c0 0c 01 0f 22  |...f...... ...."|
0009b130  c0 ea ee 11 08 00 8e d9  8e c1 8e d1 8e e1 8e e9  |................|
0009b140  24 fe 0f 22 c0 ea 02 12  00 99 8c c8 8e d0 66 bc  |$.."..........f.|
0009b150  20 50 00 00 8e d8 8e c0  8e e0 8e e8 66 0f 01 1e  | P..........f...|
0009b160  b0 00 66 6a 00 66 9d 66  a1 48 20 66 3d 11 11 ee  |..fj.f.f.H f=...|
0009b170  51 75 32 66 a1 54 52 66  3d 82 2c a2 65 75 26 66  |Qu2f.TRf=.,.eu&f|
0009b180  e8 fc 0a 00 00 66 8b 3e  3c 20 66 0f ba e7 00 73  |.....f.>< f....s|
0009b190  11 66 a1 34 20 66 8b 16  38 20 66 b9 a0 01 00 00  |.f.4 f..8 f.....|
0009b1a0  0f 30 e9 a3 fd f4 eb fd  67 80 b8 20 50 00 00 00  |.0......g.. P...|
0009b1b0  75 4e 66 56 66 53 66 89  c6 67 c6 80 20 50 00 00  |uNfVfSf..g.. P..|
0009b1c0  01 66 bb 00 01 00 00 66  81 fb 54 01 00 00 73 2a  |.f.....f..T...s*|
0009b1d0  67 66 39 73 14 75 1d 67  66 8b 43 08 66 85 c0 74  |gf9s.u.gf.C.f..t|
0009b1e0  0a 66 ff d0 67 66 89 43  10 eb 09 67 66 c7 43 10  |.f..gf.C...gf.C.|
0009b1f0  00 00 00 00 66 83 c3 1c  eb cd 66 5b 66 5e 66 c3  |....f.....f[f^f.|
0009b200  66 c3 66 ba 00 01 00 00  66 81 fa 54 01 00 00 72  |f.f.....f..T...r|
0009b210  05 66 31 c0 66 c3 66 56  66 53 67 66 8b 72 0c 67  |.f1.f.fVfSgf.r.g|
0009b220  66 8b 5a 10 66 31 c9 66  39 cb 7e 0a 67 39 04 ce  |f.Z.f1.f9.~.g9..|
0009b230  74 16 66 41 eb f1 66 83  c2 1c 66 81 fa 54 01 00  |t.fA..f...f..T..|
0009b240  00 72 d7 66 31 c0 eb 06  66 b8 01 00 00 00 66 5b  |.r.f1...f.....f[|
0009b250  66 5e 66 c3 66 55 66 57  66 56 66 53 66 83 ec 1c  |f^f.fUfWfVfSf...|
0009b260  3d 04 0f 75 06 66 31 ff  e9 d8 01 66 be 00 0f 00  |=..u.f1....f....|
0009b270  00 83 f8 ff 74 0e 66 89  c6 83 f8 fe 75 06 66 be  |....t.f.....u.f.|
0009b280  01 0f 00 00 66 89 f1 81  e1 ff 7f 66 ba 00 01 00  |....f......f....|
0009b290  00 67 66 c7 44 24 04 00  00 00 00 66 0f b7 c1 67  |.gf.D$.....f...g|
0009b2a0  66 89 44 24 0c 66 81 fa  54 01 00 00 0f 83 9f 00  |f.D$.f..T.......|
0009b2b0  67 66 8b 42 0c 67 66 8b  7a 10 67 66 89 7c 24 10  |gf.B.gf.z.gf.|$.|
:
```

Figure 4: Memory snooping outside of a secure enclave

We were able to seamlessly run our ProTego components in an SEV enclave, without needing code refactoring.

It should be noted that the technology behind TEEs is still evolving, as a number of side-channel attacks on secure enclaves have been discovered in a controlled laboratory environment (Intel® SGX Technology and the Impact of Processor Side-Channel Attacks, 2020). Additionally, industrial-strength deployment of secure enclaves is still a difficult task, as an enclave must be attested as being secure in order to avoid the attack scenario where a malicious, insecure enclave can spoof a user into believing that it is a TEE, potentially exposing sensitive applications.

Efforts such as the Red Hat sponsored Open Source project, Enarx, (Bursell, 2019) which is working to provide cross-silicon support for secure enclaves, as well as provide the attestation of the TEE, are underway, and it is expected that the use of TEEs will become more commonplace in the future as implementation barriers are lowered.

While secure enclaves vastly reduce the threat of attacks on data-in-process, workloads on the cloud may still be vulnerable to data leakage due to the current state-of-the-art in networking technology. To this end, in the second half of the project, WP5 joined forces with IBM's on-going, Open Source The Mesh for Data (M4D) project (IBM, n.d.) (See Section II.1.2.c.)

II.1.2.b. Performance Evaluation for Parquet Modular Encryption Double Key Wrapping

As part of our work on Parquet Modular Encryption, we developed and designed an encryption key management system which uses a unique double wrapping method to improve efficiency.

When writing a Parquet file, a random data encryption key (DEK) is generated for each encrypted column and for the footer. These keys are used to encrypt the data and the metadata modules in the Parquet file.

The data encryption key is then encrypted with a key encryption key (KEK), also generated inside Spark/Parquet for each master key. The key encryption key is encrypted with a master encryption key (MEK), either locally if the master keys are managed by the application, or in a Key Management System, for example Vault, as is the case in ProTego.

Encrypted data encryption keys and key encryption keys are stored in the Parquet file metadata, along with the master key identity. Each key encryption key has a unique identity (generated locally as a secure random 16-byte value), also stored in the file metadata. Key encryption keys are cached by access token, so there is no need to interact with the KeyProtect service for each encrypted column or file if they use the same master encryption key. (See Figure 5)

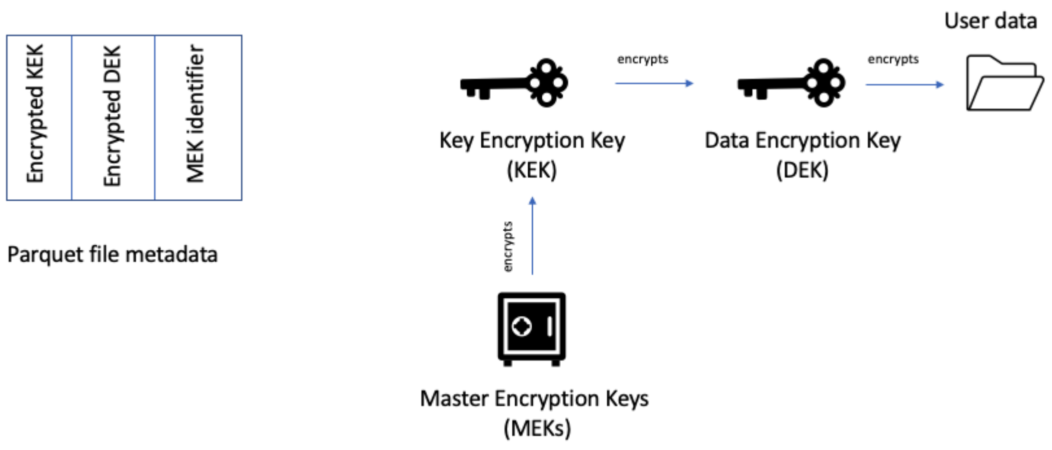


Figure 5: Parquet double key wrapping

When reading a Parquet file, the identifier of the master encryption key (MEK) and the encrypted key encryption key (KEK) with its identifier, and the encrypted data encryption key (DEK) are extracted from the file metadata.

The key encryption key is decrypted with the master encryption key stored in Vault. The key encryption keys are cached by access token, so there is no need to interact with the KeyProtect service for each decrypted column or file if they use the same key encryption key. Then the data encryption key (DEK) is decrypted locally, using the key encryption key (KEK).

II.1.2.c. The Mesh for Data

The Mesh for Data is a cloud native platform to unify data access and governance, enabling business agility while securing enterprise data being developed as Open Source by IBM. Based on Kubernetes, M4D smoothly integrates with the ProTego technologies developed in the first half of the project, and brings together access and governance for data, greatly reducing risk of data loss or leakage. The incorporation of M4D in our framework is a natural complement to the rest of the work done in WP5 – Parquet Modular Encryption protects data at rest, secure enclaves protect data-in-process, and The Mesh for Data protects the data at the policy level.

The Mesh for Data is a sophisticated framework, consisting of a Control Plane and a Data Plane. The M4D Control Plane, as the name suggests, controls the deployment of components that are required to connect a data requester to the desired data source, and includes:

- A Policy Manager to implement a Policy Decision Point (PDP) which decides whether data under the given request and environmental conditions is allowed to flow between two services.
- A Blueprint Manager which determines what components need to be deployed to provide a secured data path between the requester and the data source,
- A Credential Manager which is responsible for supplying the control plane with access tokens, to allow connection to a data source, without exposing sensitive tokens to user applications.

- A Catalog for holding registered data assets and the metadata associated with these assets.
- A Plotter which is used to plan and distribute multi-cluster deployments.

The Control Plane is agnostic to data requirements of a hosted application, whereas the Data Plane, triggered by the Control Plane, implements the actual data communication mechanisms and connectivity. This architecture is depicted in Figure 6.

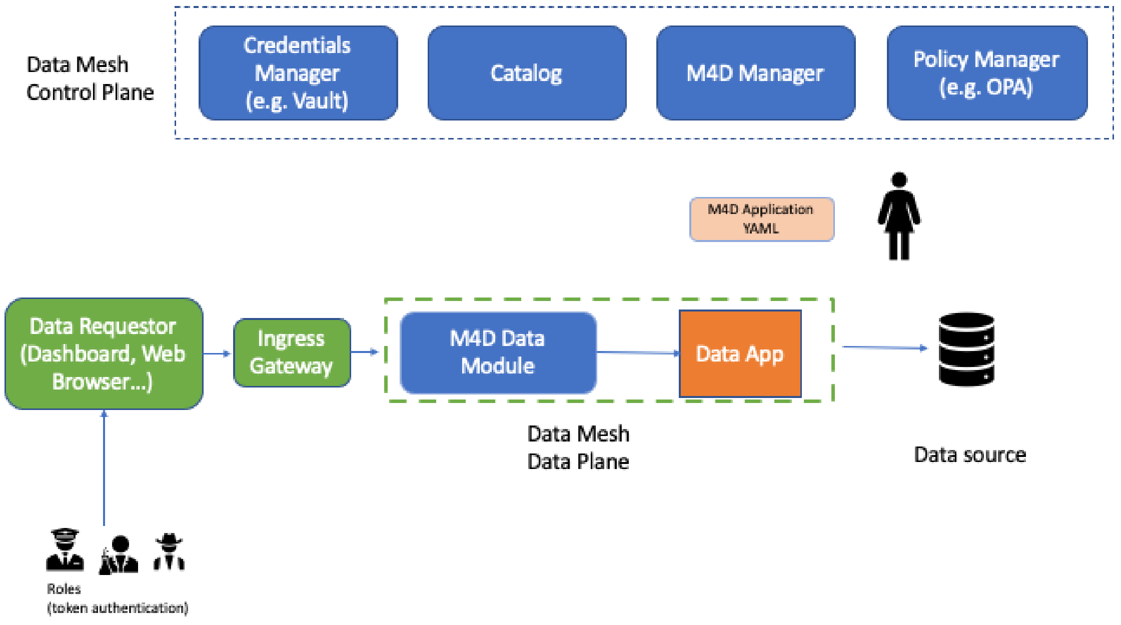


Figure 6: The Mesh for Data architecture

The Data Plane relies on a Kubernetes custom resource (CRD) called an M4DModule which handles the communication protocol required between the data requester and data source. An example of an M4DModule is the Arrow Flight module, which is responsible for marshalling large amounts of data between two points, using the Apache Arrow protocol. The M4DModule is responsible for communicating with the Policy Manager to determine what data can be returned to the requestor. The REST M4DModule, developed for use in ProTego, receives information in the HTTP header (which may be encoded in a Java Web Token (JWT) or passed as a HTTP header variable) which encodes information about requestor, such as role and organization. Before allowing a requester access to a REST endpoint, the ProTego REST M4DModule passes this information to the Policy Manager, to determine whether to block or allow URL access. If access is allowed, the M4DModule will once again query the Policy Manager with the received user credentials to determine what fields, if any, in the data being passed back needs to be redacted before being returned to the user, as will be described in more detail in the next section.

ProTego relies heavily on a REST M4DModule which it is developing for accessing data using the REST protocol.

The M4D provider will supply the M4DModule - the application developer does not need to be aware of its existence. The application developer will need to fill out a template YAML file, called an M4DApplication, which describes the purpose for which the data is being requested, and a pointer to the catalogued data source, called a dataset id, including a description of the format of the stored data which is required for selection of the appropriate M4DModule.

It is a simple matter for an application developer to create the M4DApplication to bring up the required ProTego WP5 secure enclave framework and deploy it - the M4D system will set up the rest.

An illustration of how this works is given in Figure 7. Users request access to FHIR resource via a dashboard, which communicates with Keycloak to obtain a JWT which encodes the role of the requester. The dashboard sends the REST request for the FHIR resource to the published address of the k8s ingress end point, (istio ingress gateway), which validates the JWT and then forwards the REST request to the M4DModule. The Module will then query the Policy Manager whether or not to allow the given role access to the end point, as will be described in section 0 If the query is allowed, the Module will then proxy the original request to the original REST target – which, for the ProTego use cases, is a FHIR server. The Module receives the returned resources from the FHIR server, and once again queries the Policy Manager to determine if any fields need to be redacted before finally returning the data to the caller.

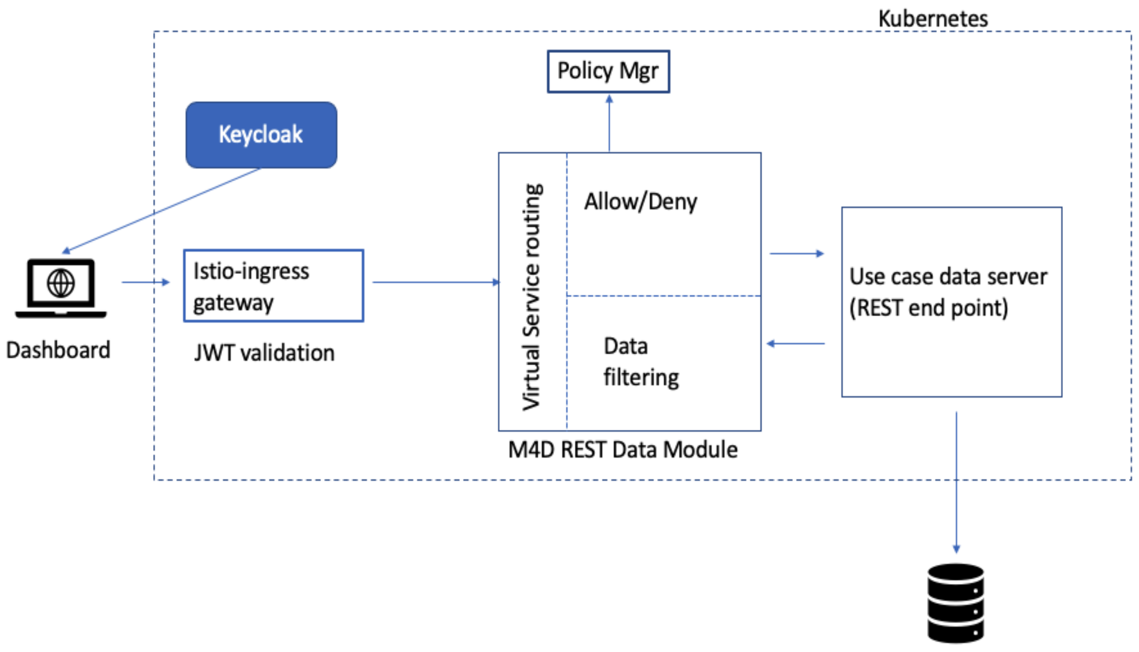


Figure 7 - the ProTego REST M4DModule in the use cases

Implementing access control, deciding whether or not a requester should be given access to a data source, the ProTego REST M4DModule enables fine-grained access control on data returned from the source, based on role-based policy.

As an example of the power that this enables, we can consider access to FHIR resources in a healthcare environment today – either a user is allowed or not to access a FHIR server. Given access to a FHIR server, the user is free to access any stored resource. Taking this a level deeper, any user accessing a given FHIR resource is allowed to see all the information in the resource.

Using M4D technology, ProTego is able to improve upon this model.

M4D implements a *Policy Manager*, which uses Open Policy Agent (OPA) (Open Policy Agent, n.d.). OPA includes both a language for expressing policy, called Rego, and an engine for evaluating the policy against given input – i.e. a Policy Decision Point (PDP).

M4D creates a Kubernetes Custom Resource called, Asset, which describes each data source, specifying a general classification for the type of data that the data source supplies (e.g. “HR data”, “accounting”…) as well as specification of which of the fields in the data source should be considered as Personal Identifiable Information (PII). An example of how this could be done for a FHIR Patient resource is shown below. Note that the *assetMetadata.tags* is given the value of “personal”. This is an arbitrarily chosen classification, however we will use this value in the policy definition, as will be described.

Under componentsMetadata, the field “id” is classed as PII.

```
apiVersion: katalog.m4d.ibm.com/v1alpha1
kind: Asset
metadata:
name: patient
spec:
assetDetails:
dataFormat: fhir
connection:
type: REST
assetMetadata:
tags:
- personal
componentsMetadata:
id:
tags:
- PII
```

Figure 8 - Specifying a data asset

We now have a characterization of a data source, and as explained in section II.1.2.c. , a specification of the role of the data requester from the passed JWT. Both of these are used as inputs to the Policy Manager, which firstly determine if a given user (or role) is allowed to access the requested data source based on the characterization of the type of the data specified in the corresponding Asset (e.g. the tag value for the assetMetadata field in the Asset definition).

If a user is allowed to access a data source, then the ProTego REST M4DModule requests from the Policy Manager to get a list of all fields which must be redacted. The Policy Manager invokes the OPA engine on the defined policy – which itself exists as a Kubernetes Custom Resource holding the Rego code which defines the policies.

An example of a simple policy is given in Figure 9 - Access policy definition. In this case, the Policy Manager will be asked to assess the rule “blocklist” to determine whether or not to give the requester access to this data asset, and the rule “filters” to get a list of all fields that need to be redacted. In this case, we are allowing all users access to all FHIR resources but redact the PII fields (as defined in the corresponding Asset CRD) for users in the “admin” role.

```
apiVersion: v1
data:
  main: |
    package katalog.example
    import data.kubernetes.assets


    filters[output] {
      count(rule)==0
      asset := assets[input.request.asset.namespace][input.request.asset.name]
      output = {"name": "No filtering by default!", "action": "Allow"}
    }

    filters[output] {
      count(rule)>0
      output = rule[_]
    }

    rule[{"name": "Redact PII columns if admin for personnel data", "action": "RedactColumn", "columns": columns}] {
      asset := assets[input.request.asset.namespace][input.request.asset.name]
      contains(input.request.role, "admin")
      columns := [c | asset.spec.assetMetadata.componentsMetadata[i].tags[_] = "PII"; c = i]
    }

    blockList[output] {
      output = {"name":"Allow all URLs", "action": "None"}
    }

kind: ConfigMap
metadata:
  creationTimestamp: null
  labels:
    openpolicyagent.org/policy: rego
  name: policy
  namespace: katalog-system
```

Figure 9 - Access policy definition

II.1.2.c.1. Improving Vault integration

ProTego uses Hashicorp's Vault Key Management System to secure store encryption keys used by Parquet Modular Encryption in the use cases.

Requests sent to Vault must be accompanied by an authentication token. In the first half of the project, this initial authentication secret was created together with the Vault instance, and then coded into the application code. While this was considered acceptable for a proof-of-concept demonstration of using Parquet Modular Encryption in a use case environment, it is of course not a product-level solution; not only must the authentication key be manually transferred to the application, but the key will must be coded in the application.

In order to design a better solution to this problem, in the second half of the project, starting with a Kubernetes deployment of ProTego components, WP5 collaborated with M4D to improve upon this solution.

Building upon Vault's Kubernetes auth method, the following high-level protocol was used:

1. A YAML file for the deployment of a k8s service account was created and added to the Helm chart which deploys the Vault server.
2. The Kubernetes auth method was enabled in Vault Docker image.
3. Deployment of the Data Gateway (DGW) (see deliverable D5.2 for a functional description of the DGW) in the k8s environment automatically creates a Kubernetes *service account*, which encodes a Java Web Token (JWT) and CA certification which is discoverable from the DGW pod. The values for the JWT, CA cert and Kubernetes cluster are stored in the Docker Vault image.
4. A k8s ConfigMap is created which creates a Vault Agent configuration file, specifying the use of Kubernetes auth method on the Vault server. Additionally, a location in the pod ("sink") for storing the returned Vault authentication token is specified. The name of the ConfigMap is attached to the YAML for the deployment of the DGW.

Now, the rest is automatic – when the DGW is deployed, it will request authentication from the Vault server based on its JWT and CA certificate. When Vault receives this request, it will contact the Kubernetes Authentication Service running on the cluster to validate the DGW. Successful validation will result in Vault returning a client authentication token to the DGW pod, which will automatically be stashed in the sink location. The DGW code can access the token, and subsequently store this as a Hadoop configuration for use by the Parquet Modular Encryption code.

III. Access control framework (KUL)

III.1.1. Basic access control framework

During the first half of the ProTego project, the efforts in task 5.3 focused on designing and developing a base solution for managing and protecting access to confidential data. This solution has been described in deliverable D5.2 and has been further enhanced during the second half of the ProTego project. Below, we first briefly revisit this basic access control framework. In the rest of this section, we then discuss our improvements to this solution.

III.1.1.a. Functionality of the basic protocol

The access control framework is responsible for the authorization and Key Management Service (KMS) to enhance the ProTego Data Gateway (DGW) security. The main purpose of the access control framework is to decide whether a data consumer is allowed to retrieve the plaintext data or not (authorization service). Let's look at how a data producer/consumer can upload/access (resp. read) the medical data.

Once the data producer wants to send the data to the DGW, it first authenticates itself to the external Identity and Access Management (IAM) component and obtains an authorization token (the IAM is responsible for managing the attributes of all users of the system and generating the authorization token based on these attributes and the expiration time). The data producer then sends to the DGW the token together with all the medical data it wants to upload. The DGW now encrypts this data and sends the key encryption key (KEK) to the access control component. The KEK is then forwarded – via the access control interface – to the open-source Vault key management service (KMS). The KMS will use an internal key – the master encryption key (MEK) - to encrypt the KEK and the data producer's parameters (i.e. the content of the authorization token) – into a wrapped KEK. This wrapped KEK is then sent back to the DGW for storage along with the encrypted medical data.

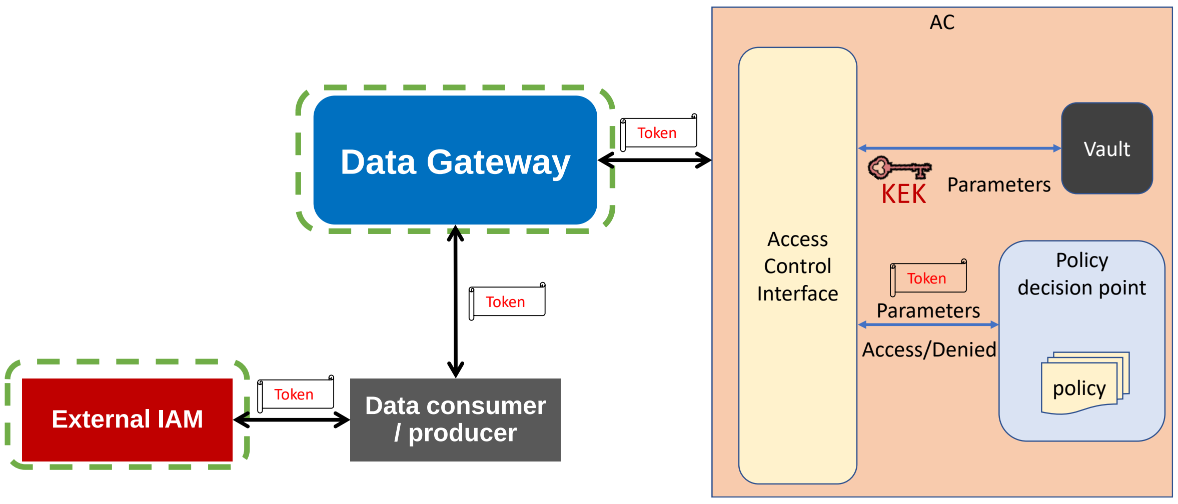


Figure 10 - Protocol to access medical data

As depicted in Figure 10, when a data consumer wants to access the encrypted medical data, it first authenticates itself to the external IAM and obtains an authorization token. Then, it sends the authorization token to the DGW. The DGW relays this token to the access control component. It is important to note that the DGW, in addition to the authorization token, also sends the master key identity and the wrapped KEK associated with the medical data that the data consumer wants to retrieve to the access control component. At this point, the access control component checks the validity of the token (i.e., being a legitimate token that is not yet expired) and for which the medical data access is requested. Next, the control interface forwards the outcome of this evaluation (token valid or not) to the Policy Decision Point (PDP). The Policy Decision Point will need additional information (i.e. the data producer's parameters associated with the KEK) to make a proper security decision (i.e. to grant access or not). This input is processed in two stages. First, the Control Interface component relays the wrapped KEK and the master key identity to the KMS to fetch the unwrapped KEK and the corresponding data producer's parameters (i.e. the owner's identity, the owner's role and the group to which the owner belongs). Second, the access control interface component forwards these parameters to the Policy Decision Point. Based on all this information and applicable security policies, the Policy Decision Point will now be able to decide if access should be granted to the data consumer. This decision is based on pre-defined security policies that are stored in the Policy Decision Point component.

III.1.1.b. Limitations of the current solution

Although the basic access control framework described above provides the necessary security functionality, it also has some limitations.

First of all, as mentioned above, the current authorization service is based on static security policies. Since the access policy is stored in the Policy Decision Point and also constant parameters are statically linked to the KEK, the policies cannot be automatically changed when the system evolves. This means that the current version of the access control framework (presented in Figure 10) does not meet the requirements for dynamic security policies (e.g. when the admin of the system wants to change the policies). Therefore, we propose a new representation of a policy, called dynamic policy access control, which could be generated and bound dynamically to the wrapped KEK.

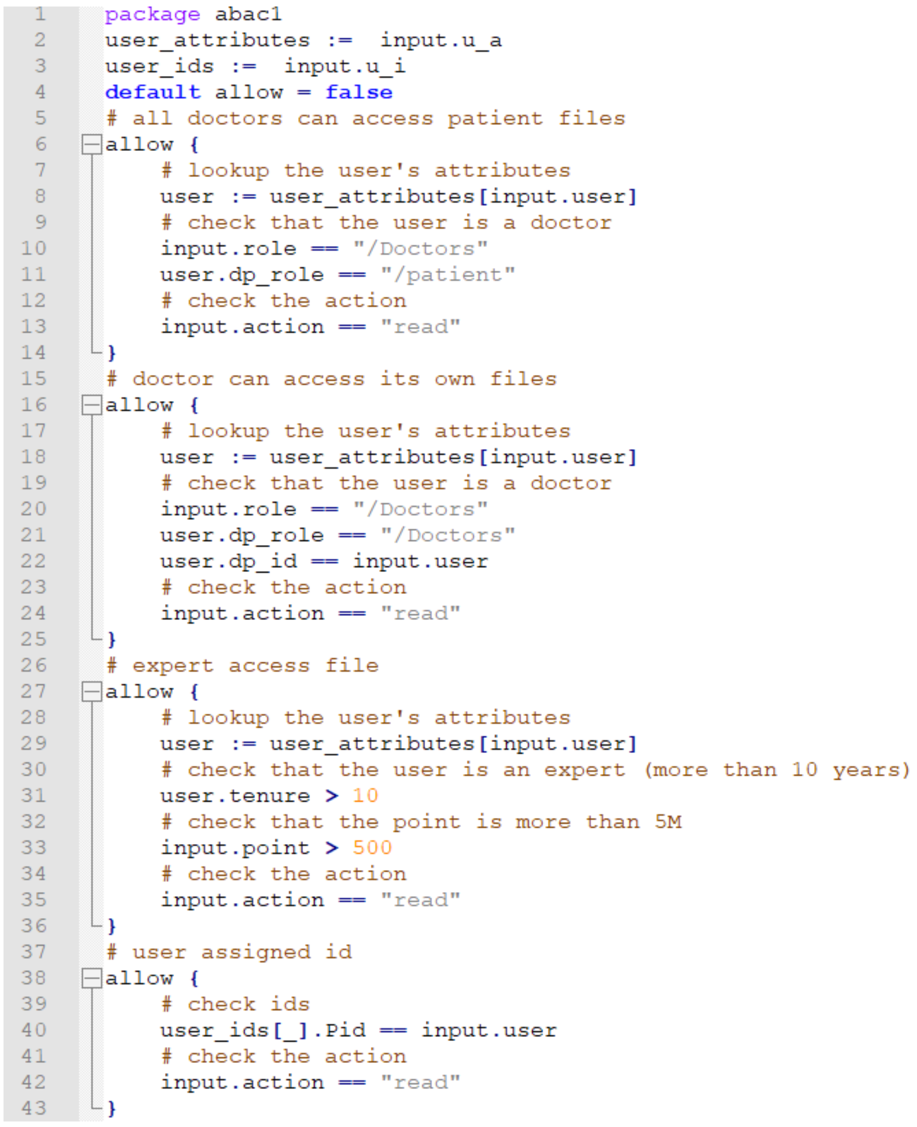
Moreover, if an adversary would compromise both DGW and access control components, then they would be able to decrypt all the encrypted data. To improve upon this, the encryption/decryption keys need to be constructed based on not only secret parameters stored in the access control component but also secret parameters (e.g. secret key) from a non-colluding third party (for example the data producer). To this aim, we proposed an enhanced access control solution that relies on Ciphertext-policy attribute-based encryption (CP-ABE). In this scheme, the medical data first gets protected based on security policies defined by the data producer. Afterwards, the medical data gets an additional round of protection, based on the system's security policies. We also propose a Cryptographic attribute-based access control (C-ABAC) scheme that can provide a user revocation mechanism. An important property of the proposed revocation mechanisms, as will be discussed later, is that all the non-revoked data consumers' secret keys will not be updated when a revocation event occurs. It is worth noting that in the conventional revocation mechanisms, all encrypted data need to be re-encrypted to prevent the revoked data consumer from using their secret key for decryption. Thus, all other non-revoked data consumers should receive updated secret keys, which is not efficient.

The next sections present an overview of our 3 proposed improvements: i) Dynamic policy attribute-based access control (D-ABAC) protocol, ii) Ciphertext-policy attribute-based access control (CP-ABAC) scheme and iii) Cryptographic attribute-based access control (C-ABAC) scheme. The D-ABAC protocol replaces our basic access control solution explained above. The D-ABAC scheme can then be further extended with either the CP-ABAC protocol or the C-ABAC protocol.

III.1.2. Dynamic policy attribute-based access control (D-ABAC) protocol

III.1.2.a. Description

To propose a new representation of a policy, called dynamic policy access control, we enhanced our access control architecture by relying on Open Policy Agent (OPA) (Open Policy Agent, n.d.). OPA is an open-source engine that unifies policy enforcement across the stack. OPA provides a high-level declarative language (Rego) that lets us specify policy as code and simple APIs to offload policy decision-making from our access control component. When our component needs to make policy decisions it queries OPA and supplies structured data (e.g., JSON) as input. Then OPA outputs policy decisions by evaluating the query input against policies and data. The policies that the admin of the system defines can be inspected and transformed using Rego. It is important to note that policies from several existing policy systems can be implemented with the OPA using the Rego language. Figure 11 shows an example of the Attribute-based access control (ABAC). With attribute-based access control, one makes policy decisions using the attributes of the users (e.g. a user being a doctor) involved in the request.



```
package abac1
user_attributes :=  input.u_a
user_ids :=  input.u_i
default allow = false
# all doctors can access patient files
allow {
    # lookup the user's attributes
    user := user_attributes[input.user]
    # check that the user is a doctor
    input.role == "/Doctors"
    user.dp_role == "/patient"
    # check the action
    input.action == "read"
}
# doctor can access its own files
allow {
    # lookup the user's attributes
    user := user_attributes[input.user]
    # check that the user is a doctor
    input.role == "/Doctors"
    user.dp_role == "/Doctors"
    user.dp_id == input.user
    # check the action
    input.action == "read"
}
# expert access file
allow {
    # lookup the user's attributes
    user := user_attributes[input.user]
    # check that the user is an expert (more than 10 years)
    user.tenure > 10
    # check that the point is more than 5M
    input.point > 500
    # check the action
    input.action == "read"
}
# user assigned id
allow {
    # check ids
    user_ids[_].Pid == input.user
    # check the action
    input.action == "read"
}
```

Figure 11 - Rego example for ABAC

In this example, user_attributes is an attribute set of the user. We could also have some dynamic attributes such as “tenure” (e.g., doctor joined the hospital more than 10 years ago) for the users. According to the policies defined in this example, the OPA will return “true” in the cases where i) the data consumer has the role of doctor, ii) the data consumer is the data producer, or iii) their identity was assigned to the KEK at the time of wrapping. In the latter case, the data producer must send these assigned identities (the ‘input’ shown in Figure 10) to the access control component through DGW.

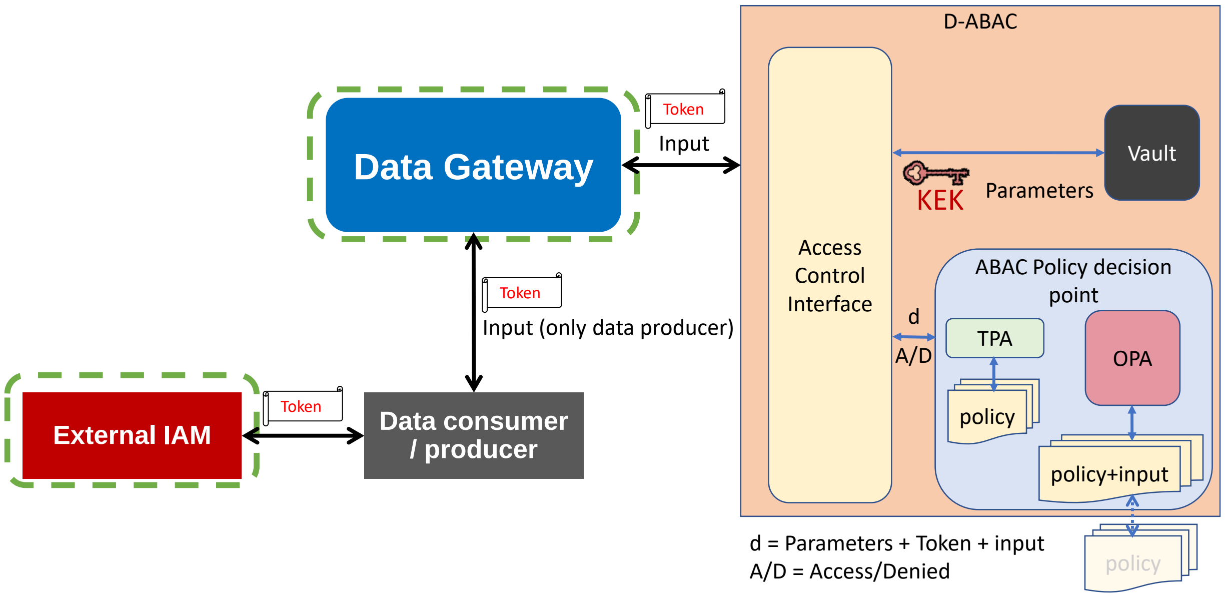


Figure 12 - Dynamic ABAC Protocol to access medical data

High-level overview of the D-ABAC architecture:

First of all, it is important to stress that our proposed D-ABAC solution replaces our initial basic access control framework. In the D-ABAC scheme, we rely on a new component, denoted as the ABAC Policy Decision Point, to realize dynamic ABAC. As depicted in Figure 12, the ABAC Policy Decision Point component includes Token-based Policy Agent (TPA) and Open Policy Agent (OPA). It also supports updating the security policies.

Token-based Policy Agent (TPA):

The TPA is based on a simple access control rule that was defined in the Policy Decision Point component of the basic access control framework. This simple access control rule states that the user that creates the data (i.e. data producer) always can access their own data (it means that the TPA evaluates the identity of the data consumer and will grant access to this user if the data consumer is the data producer). By using this simple access control rule, already many access control requests (i.e. the data producer’s requests to access their produced data) can be handled. Therefore, TPA is a fast decision point that improves the efficiency of the access control system.

Open Policy Agent (OPA):

In the cases that the requester (data consumer) is not the data producer, the ABAC Policy Decision Point sends the request (Query) to the OPA using specific CLIs. The Query which is the file with the JSON format includes the Token parameters, data producer’s parameters, and data producer’s input (note that parameters and input of the data producer are fetched from the Vault, similarly as in the basic access control framework). OPA generates policy decisions by evaluating the query input and security policies stored in policy files.

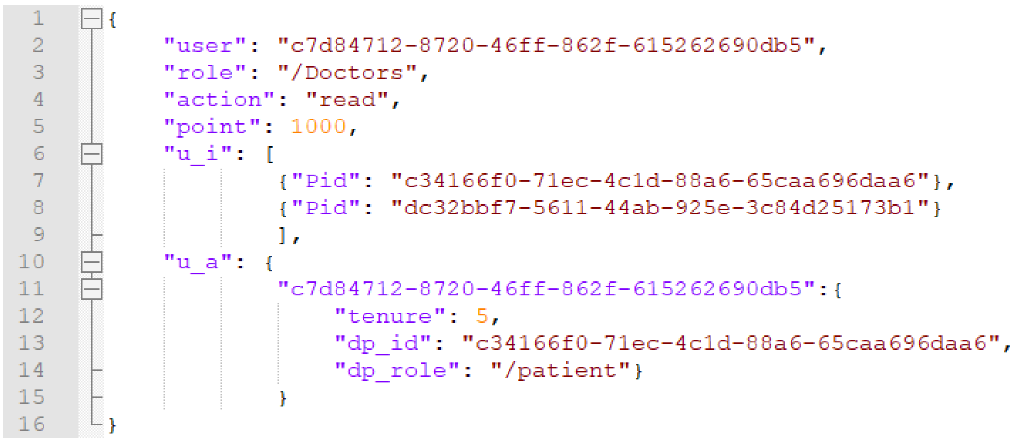
Updating the security policies:

When OPA starts for the first time, it will not have any security policies. Policies can be added, removed, and modified at any time. Since OPA accepts external data, the system will support dynamic policy access control. This dynamic property is based on external input and policy files.

Input file

Often policies require external data that is not available to the OPA. The query can include external data (necessitating of course that the policy is written accordingly). The Policy Decision Point acts as below:

- It sends a query to OPA including the external input data (Figure 13 shows an example of the input data) and token parameters.
- OPA makes a decision based on the query and the external policy file.

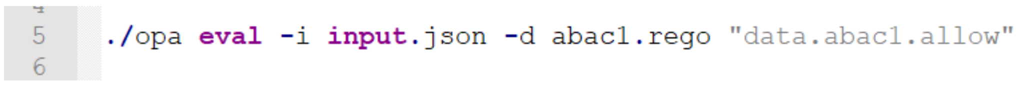


```
{
        "user": "c7d84712-8720-46ff-862f-615262690db5",
        "role": "/Doctors",
        "action": "read",
        "point": 1000,
        "u_i": [
                {"Pid": "c34166f0-71ec-4c1d-88a6-65caa696daa6"},
                {"Pid": "dc32bbf7-5611-44ab-925e-3c84d25173b1"}
                ],
        "u_a": {
                "c7d84712-8720-46ff-862f-615262690db5":{
                    "tenure": 5,
                    "dp_id": "c34166f0-71ec-4c1d-88a6-65caa696daa6",
                    "dp_role": "/patient"}
                }
}
```

Figure 13 - Example of input data (JSON format)

Policy file

The policy file is expressed in the Rego language (see Figure 11). This file is under the control of the security administrator of the system. The administrator can add or remove one or more policies inside this file at any time. The OPA will make its access control decision based on this file and the query. Figure 14 demonstrates the example of the query including the name of the external policy and input files.



```
./opa eval -i input.json -d abac1.rego "data.abac1.allow"
```

Figure 14 - Example of a query to OPA

III.1.2.b. Implementation

The rest of this subsection walks through the implementation of D-ABAC in more detail.

We built the new Docker image (`kulacc:v6`) to add D-ABAC properties to the current version of the Docker image (`kulacc:v2`). This version of the Docker image has the following functionalities.

- **Enable or disable JWToken verification**

It is possible to enable or disable token verification using the "TOKEN_VERIFY" environment.

- Running the Docker image with `-e TOKEN_VERIFY=1` and `-e PUBLIC_KEY="public key string"` enables JWToken verification.
- Running the Docker image with `-e TOKEN_VERIFY=0` or without "TOKEN_VERIFY" environment disables JWToken verification (should only be used for testing).

- **Policy environments**

  It is possible to choose the policy file using the "POLICY_RULE" environment.

  - Running the Docker image with `-e POLICY_RULE=0` enforces the pre-defined internal policies.
  - Running the Docker image with `-e POLICY_RULE=1` enforces the external Rego file as policies.

    Note: to read the external Rego file from the host machine, the Docker image should run with the `-v` flag. The command `-v <directory on the host machine>:/var` sets up a bind-mount volume that links the `/var` directory from inside the container to the directory on the host machine. Docker uses `:` to split the host's path from the container path, and the host path always comes first. The Rego file should be in the `<directory on the host machine>` directory. Whenever the system administrator changes the content of this file, the ABAC Policy Decision Point component of the access control container uses the new content as policy rules (necessitating of course that the policy is written accordingly).

III.1.3. Ciphertext-policy attribute-based access control (CP-ABAC) scheme

Since both DGW and access control components are executed and managed in the same cloud environment, and therefore most likely managed by the same entity, an adversary who compromises this cloud environment would be able to obtain all necessary secret keys to decrypt all the encrypted medical data that is stored externally. To mitigate this problem, one should only be able to retrieve the encryption/decryption keys by using secrets from the access control component and secrets from another external entity that does not collude with the DGW or access control framework. For the latter, we will rely on the data producer itself. In summary, one can only retrieve the necessary keys to encrypt/decrypt the medical data when both the DGW/access control framework and the data producer contribute. None of these parties in isolation is able to get the encryption/decryption keys.

To this end, we propose an enhanced access control scheme called CP-ABE. In CP-ABE, a ciphertext is encrypted with access policies over attributes – called access structure. Any data consumer whose attributes satisfy the access policy can decrypt the ciphertext; otherwise, the decryption fails. Associating the access policy with the ciphertext means that the ciphertext chooses which key can recover the plaintext, giving the data producer more control of its outsourced data. Generally speaking, three entities are responsible for running the CP-ABE scheme: (i) attribute authority, (ii) data producer, and (iii) data consumer. The role of the attribute authority is to generate system public and master secret keys and issue attribute secret keys to each data consumer based on their corresponding attribute list. A data producer chooses an access policy themselves and integrates that into the ciphertext. Then, the data consumer has the means for decrypting the ciphertext to retrieve the original data based on their attribute secret keys.

It is important to stress that the CP-ABE protocol is a security enhancement of our access control framework. Therefore, it can be combined with either the basic access control framework (developed during the first half of the ProTego project) or the D-ABAC solution (see Section III.1.2.). In the ProTego project, we have opted for the latter, as this provides the most functionality.

In our enhanced scheme, the medical data first gets encrypted based on security policies defined by the data producer. After this encryption, the result is then forwarded to the DGW. The rest of the process is similar as in the D-ABAC solution described in Section III.1.2. When requesting access to the data, the DGW will now send an encrypted medical file to the data consumer (assuming that the access control framework granted access). The data consumer can then decrypt this data and retrieve the plaintext medical data when it has all the necessary attributes, as was defined by the security policies of the data producer.

Note: In the enhanced scheme there are two types of polices; i) policies defined by the data producer, and ii) policies defined by the organization (e.g., hospital). The data consumer can access to the data iff they can satisfy both of these policies.

Let us now look more into detail in this scheme.

High-level overview of the CP-ABAC architecture:

As demonstrated in Figure 15, our proposed CP-ABAC relies on attribute-based encryption (ABE) (Sahai, 2005). Integrating ABAC into the ABE technique provides us a possibility to encrypt the outsourced information (e.g. FHIR data) and protect it from attacks when both the DGW and access control framework are compromised. Using ABE, even if an adversary compromises both DGW and access control components, they would not be able to read the sensitive information. The algorithms and steps of the proposed CP-ABAC scheme are as follows.

Algorithms:

The scheme consists of six algorithms: Setup, AES-Enc, CP-ABE-Enc, ABE-KeyGen, CP-ABE-Dec, and AES-Dec.

Setup

This algorithm generates a pair of public parameters (pp) and a master secret key based on the system security parameter and an attribute space. The IAM runs this algorithm, stores the master secret key, and publishes pp.

Note: In the CP-ABAC scheme, one attribute authority administrates all system attributes. This authority has the master secret key that is used to derive all users' decryption secret keys. When CP-ABAC is combined with D-ABAC, the IAM plays the role of the attribute authority.

AES-Enc

This algorithm picks a random key K_AES. It then uses this key and the AES encryption algorithm to encrypt the FHIR data. The output of this algorithm is the ciphertext CT_AES. The data producer runs this algorithm.

CP-ABE-Enc

This algorithm is the encryption algorithm that takes as input the public parameters, K_AES as a message, and an access structure $\mathbb{A}$ over the universe of attributes (i.e. a security policy based on the attributes). The algorithm will encrypt K_AES and produce a ciphertext CT_ABE such that only a data consumer that possesses a set of attributes that satisfies the access structure will be able to decrypt the message (and hence retrieve K_AES). We will assume that the ciphertext implicitly contains $\mathbb{A}$. The data producer also runs this algorithm. Note that we use CP-ABE-Enc to encrypt the encryption key rather than the FHIR data, for efficiency reasons.

Note: access structure $\mathbb{A}$ will be defined by the data producer.

ABE-KeyGen

The key generation algorithm takes as input the master secret key and a set of user attributes `DC_Att` that describe the key. It outputs the data consumer's private key `K_ABE`. The IAM runs this algorithm for each data consumer.

CP-ABE-Dec

The decryption algorithm takes as input the public parameters `pp`, a ciphertext `CT_ABE`, which contains an access structure $\mathbb{A}$, and a data consumer private key `K_ABE`, which is a private key for the attribute set `DC_Att`. If the set `DC_Att` of attributes satisfies the access structure $\mathbb{A}$, then the algorithm will decrypt the ciphertext and return `K_AES` as an output message. Using this output (i.e. this key), the data consumer can then decrypt the encrypted FHIR data (see algorithm below). The data consumer runs this algorithm.

AES-Dec

This algorithm takes the key `K_AES` and the ciphertext `CT_AES` as input. It then uses this key and the AES decryption algorithm to decrypt the `CT_AES`. The output of this algorithm is the FHIR data. The data consumer also runs this algorithm.

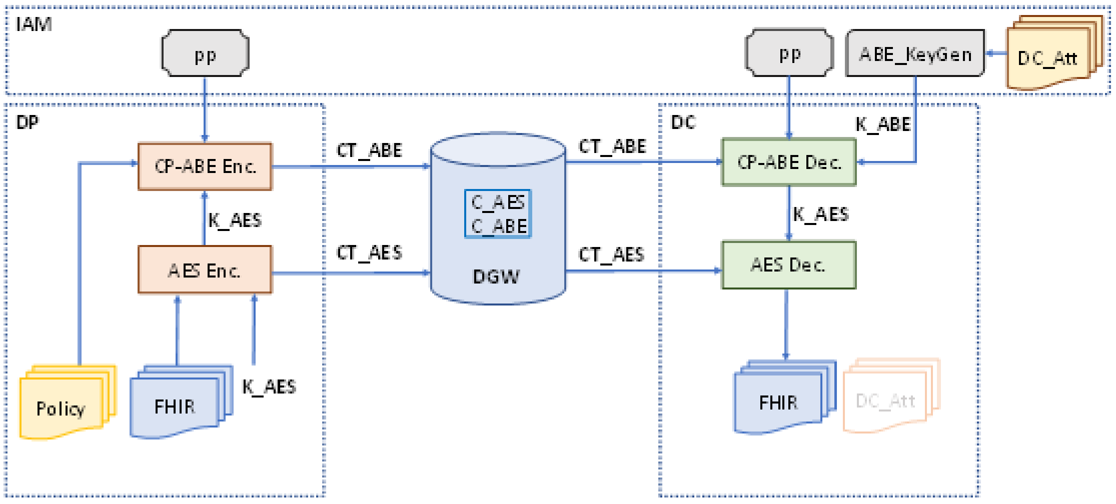


Figure 15 - Ciphertext-policy ABAC scheme to access medical data

Different steps within CP-ABAC:

The steps of our proposed CP-ABAC scheme are described as follows. Note that steps 1 and 4 are only used for initialization, steps 2 and 3 for data storage (by the data producer), and steps 5 and 6 for data retrieval (by the data consumer).

Step 1

The IAM takes the security parameter and the attribute space and runs the Setup algorithm. The Setup algorithm returns the master secret key and public parameters `pp` of the system. At this point, IAM stores the master secret key and publishes the public parameters `pp`.

Step 2

Once the data producer (DP) generates the FHIR data, it picks the random key `K_AES`. And runs the AES-Enc algorithm to encrypt the FHIR data. The output of this algorithm is the ciphertext `CT_AES`.

Step 3

At this point, the DP defines the access structure $\mathbb{A}$ over the universe of attributes and runs the CP-ABE-Enc to encrypt the key `K_AES`. The output of this algorithm is the ciphertext

`CT_ABE`. After running this algorithm, the DP determines the access rights to the encrypted message in access sets and grants access to the ciphertext only to a specific group of DCs. Finally, DP uploads the tuple <`CT_AES` , `CT_ABE` > to the DGW. From this stage onwards, the rest of the encryption and access control works exactly as before (see D-ABAC solution).

Step 4

During initialization, each data consumer (DC) needs to get the secret key corresponding to its attributes. Therefore, it will query the IAM. The IAM then runs the ABE-KeyGen algorithm and produces the secret decryption key `K_ABE` under the attribute set of the data consumer `DC_Att`. Finally, the IAM sends the secret decryption key `K_ABE` to DC. **It is worth mentioning that this step of the scheme can be performed offline and once.**

Step 5

When the data consumer (DC) with a valid attribute set wants to access the FHIR data, it will communicate to the DGW and the access control framework, similarly as in the D-ABAC solution. When access was granted, the DGW will provide the DC with the tuple <`CT_AES` , `CT_ABE` >. Using its secret key `K_ABE`, the data consumer (DC) performs the CP-ABE-Dec algorithm on `CT_ABE`. If the collection of DC attributes satisfies the access structure $\mathbb{A}$ associated with the ciphertext, then the algorithm outputs a decrypted message `K_AES,` else, decryption fails.

Step 6

The final decryption algorithm AES-Dec uses the key `K_AES`, retrieved in step 5, to securely decrypt FHIR data. This step is also performed by the data consumer.

Key Revocation:

Since the data producer (DP) encrypts the data based on a chosen access structure and does not obtain the data consumer (DC)'s certificate online, the DP is not able to check if the DC is revoked. The most popular solution in ABE-based access control systems is to re-encrypt the data based on the new access structure or to update the key. However, this type of method has several shortcomings. Several different DCs might match the decryption policy. Updating the key may therefore force the IAM to maintain a large amount of private key storage, i.e. a key for every time period. Therefore, to provide the revocation functionality we propose another attribute-based scheme called cryptographic attribute-based access control (C-ABAC), which we will discuss below.

III.1.4. Cryptographic attribute-based access control (C-ABAC) scheme

In this subsection, we propose an alternative solution for the CP-ABAC scheme described above. As mentioned, CP-ABE schemes have the limitation that they only have limited support for dynamic key revocation. Therefore, to provide key revocation, we propose a novel scheme denoted as cryptographic attribute-based access control (C-ABAC). In this scheme, we rely on the IAM to decrypt the encryption key that was chosen by the data producer. Note that similarly to the CP-ABAC solution, also the C-ABAC scheme is a security extension of the current access control framework (so either the basic access control framework or the D-ABAC solution). In the ProTego project, we opted to combine C-ABAC with D-ABAC.

High-level overview of the C-ABAC architecture:

As shown in Figure 16, our proposed C-ABAC scheme relies on encrypting the `K_AES` using the public key `pk` of the IAM. Thus, the decryption of the `K_AES` is now carried out in the IAM. The algorithms and steps of the proposed CP-ABAC scheme are as follows.

Algorithms:

The scheme consists of six algorithms: Setup, AES-Enc, Public-key-Enc, IAM-PDP, Public-key-Dec, and AES-Dec.

Setup

This algorithm generates the pair of a public key (`pk`) and a secret key based on the system security parameter. The IAM runs this algorithm, stores the secret key, and publishes `pk`.

AES-Enc

This algorithm picks the random key `K_AES`. It then uses this key and the AES encryption algorithm to encrypt the FHIR data. The output of this algorithm is the ciphertext `CT_AES`. The data producer runs this algorithm.

Public-key-Enc

This algorithm is the asymmetric encryption algorithm that takes as input the IAM public key `pk`, `K_AES` as the input message, and an access structure $\mathbb{A}$ over the universe of attributes (i.e. the security policy). The algorithm will encrypt `K_AES` and produce a ciphertext $CT\_pk=E_{pk}(K\_AES,\mathbb{A})$. The data producer also runs this algorithm.

Note: access structure $\mathbb{A}$ will be defined by the data producer.

Public-key-Dec

The Public-key-Dec algorithm takes as input the master secret key and a message $CT\_pk=E_{pk}(K\_AES,\mathbb{A})$. It decrypts the message and obtains the tuple <$K\text{-}AES,\mathbb{A}$>. Then it forwards the tuple to the IAM Policy Decision Point (IAM-PDP) algorithm. The IAM runs this algorithm.

IAM-PDP

The IAM-PDP algorithm takes as input the tuple <$K\text{-}AES,\mathbb{A}$> and outputs the private key `K_AES` to the data consumer iff the data consumer's attributes satisfy the access structure $\mathbb{A}$. If these security policies are met, `K_AES` is given to the data consumer. The IAM also runs this algorithm.

AES-Dec

This algorithm takes the key `K_AES` and the ciphertext `CT_AES` as input. It then uses this key and the AES decryption algorithm to decrypt the `CT_AES`. The output of this algorithm is the FHIR data. The data consumer runs this algorithm.

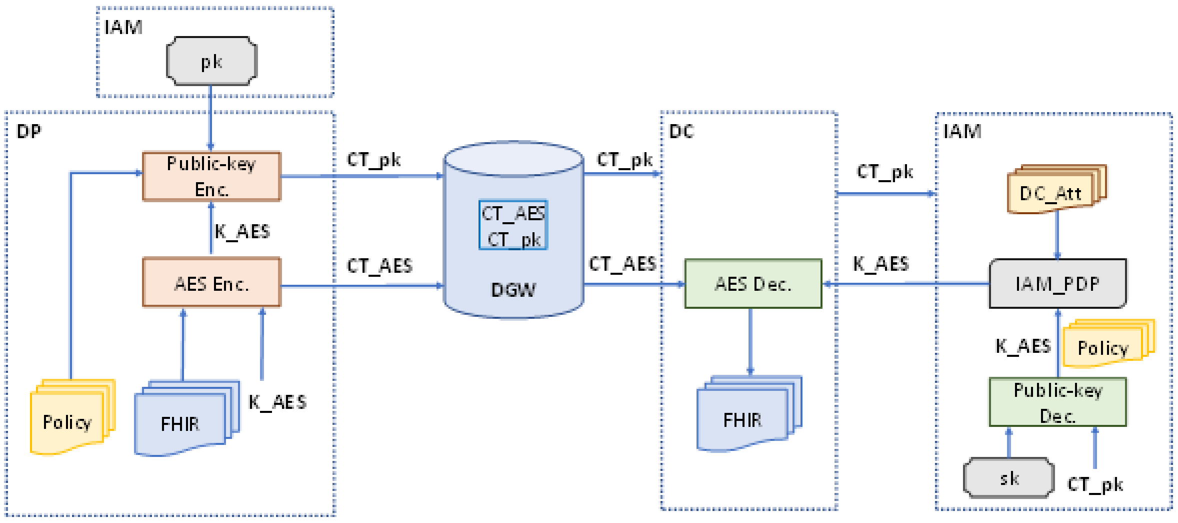


Figure 16 - Cryptographic ABAC scheme to access medical data

Different steps within C-ABAC:

The steps of our proposed C-ABAC scheme are described as follows. Note that step 1 is only used for initialization steps 2 and 3 for data storage (by the data producer), and steps 4 to 6 for data retrieval (by the data consumer).

Step 1

The IAM takes the security parameter and runs the Setup algorithm. The Setup algorithm returns the secret key and public key pk of the IAM. At this point, IAM stores the secret key and publishes the public key pk.

Step 2

Once the data producer (DP) generates the FHIR data, it selects a random key K_AES. and runs the AES-Enc algorithm to encrypt the FHIR data using this key. The output of this algorithm is the ciphertext CT_AES.

Step 3

At the time of sending data to the DGW, the DP defines the access structure $\mathbb{A}$ over the universe of attributes and runs the Public-key-Enc to encrypt both K_AES and $\mathbb{A}$ as CT_pk=E_{pk}(K_AES,$\mathbb{A}$). Finally, DP uploads the tuple <CT_AES , CT_pk> to the DGW. From this stage onwards, the rest of the data storage process is similar as in the D-ABAC solution.

Step 4

When the data consumer (DC) wants to access the FHIR data, it first communicates to the DGW and access control framework, similarly as in the D-ABAC solution. If access is granted, the DGW will send the tuple <CT_AES , CT_pk> to the DC. After receiving this tuple, the DC sends the ciphertext CT_pk to the IAM for decryption.

Step 5

Upon receiving the request, the IAM runs the Public-key-Dec algorithm to decrypt the message CT_pk=E_{pk}(K_AES,$\mathbb{A}$) using the IAM secret key. Then it forwards the tuple <K_AES,$\mathbb{A}$> to its internal IAM Policy Decision Point (IAM-PDP) algorithm. At this point, the IAM-PDP algorithm outputs the private key K_AES to the data consumer iff the data consumer’s attributes satisfy the access structure $\mathbb{A}$. Then, K_AES is given to the data consumer.

Step 6

The DC uses the decryption algorithm AES-Dec, using the key `K_AES,` to securely decrypt the FHIR data.

Key Revocation:

The IAM runs the IAM-PDP algorithm for each transaction. Since the IAM-PDP algorithm checks at runtime the dynamic access structure $\mathbb{A}$ and validates the data consumer (DC)'s credentials online, the IAM will automatically detect when a DC is revoked, and will not provide a revoked data consumer the encryption key `K_AES.` In other words, the IAM performs an online revocation check on behalf of the data producer.

III.1.5. Access structure

In both the CP-ABAC and C-ABAC constructions, the credentials of the data consumer will be represented by a set of descriptive attributes. An access structure will specify through an access tree structure (Bethencourt, 2007): a security policy that the data consumer's attributes set must satisfy to decrypt.

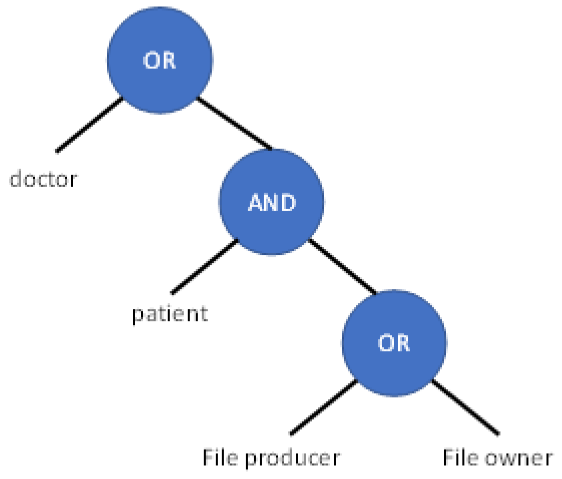


Figure 17 - Example of the access tree structure with four attributes located in the leaf nodes

The threshold gate access tree is an access structure where interior nodes are threshold gates and the leaf nodes are the attributes. An access tree could represent a Boolean equation that contains AND and OR gates instead of threshold gates. Note that a Boolean equation access tree is a special case of a threshold gate access tree (e.g., the AND gate is a (2, 2)-threshold gate while the OR gate is a (1, 2)-threshold gate). A data consumer will be able to access data iff it has all the attributes in the nodes of the tree such that the access structure encoded in the tree is satisfied. For instance, the access policy demonstrated in Figure 17 can be described using the following Boolean equation: (doctor OR (patient $\boldsymbol{AND}$ (file producer OR file owner)).

III.1.6. Deployment of the enhanced access control schemes

In this section, we explain how CP-ABAC or C-ABAC can be combined with the current access control framework (so either the basic access control framework or the D-ABAC solution). We also compare these two extensions in terms of efficiency and functionality.

CP-ABAC extension: Figure 18 shows the high-level overview of the CP-ABAC and D-ABAC combination. The scheme works as following steps. In the initialization phase of the system, the IAM runs **Setup** and **ABE-KeyGen** algorithms of the CP-ABAC scheme and sends the necessary data to the other entities in the system (for more details see CP-ABAC scheme). When the data producer wants to send the data to the DGW, it first authenticates itself to the external IAM and obtains an authorization token. The data producer then runs the CP-ABAC scheme (only **AES-Enc** and **CP-ABE-Enc** algorithms) and generates `CT_ABE` and `CT_AES` based on the access structure $\mathbb{A}$. It then sends to the DGW the token together with tuple <`CT_AES` , `CT_ABE` > and input data (for more details see D-ABAC and CP-ABAC schemes). The DGW now encrypts this data and sends the key-encryption key (KEK) to the access control component. From this stage onwards, the rest of the data storage process is similar to the D-ABAC solution.

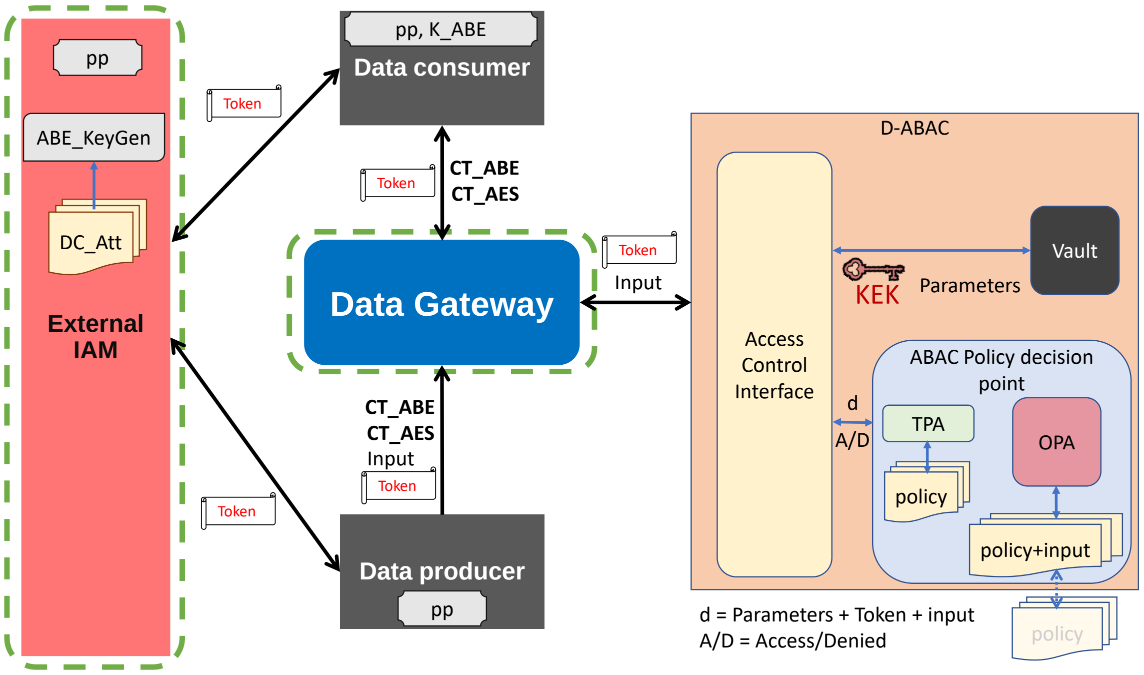


Figure 18 - CP-ABAC extension on D-ABAC scheme

When a data consumer wants to access the encrypted medical data first authenticates themselves to the external IAM and obtains an authorization token. Then, the data consumer sends the authorization token to the DGW. The DGW relays this token to the access control component for the access request similarly as in the D-ABAC solution. When access is granted, the DGW will provide the data consumer with the tuple <`CT_AES` , `CT_ABE` > (this decision is based on the OPA component with external security policies and input files). At this point, the data consumer uses the `K_ABE` and runs the CP-ABAC scheme (only **CP-ABE-Dec** and **AES-Dec** algorithms). More precisely, If the set `DC_Att` of attributes satisfies the access structure $\mathbb{A}$, then the **CP-ABE-Dec** algorithm will decrypt the ciphertext and return `K_AES` as an output message. Using `K_AES` and **AES-Dec** algorithm, the data consumer can then decrypt the encrypted FHIR data (for more details see CP-ABAC scheme).

CP-ABAC extension:

Figure 19 shows the high-level overview of the C-ABAC and D-ABAC combination. The scheme works as following steps. In the initialization phase of the system, the IAM runs the **Setup** algorithm of the C-ABAC scheme and sends its public key `pk` to the data producer (for more details see C-ABAC scheme). When the data producer wants to send the data to the

DGW, it first authenticates itself to the external IAM and obtains an authorization token. The data producer then runs the C-ABAC scheme (only **AES-Enc** and **Public-key-Enc** algorithms) and generates `CT_pk` and `CT_AES`, where, $CT\_pk = E_{pk}(K\_AES, \mathbb{A})$. It then sends to the DGW the token together with tuple <`CT_AES` , `CT_ABE` > and input data (for more details see D-ABAC and C-ABAC schemes). The DGW now encrypts this data and sends the key-encryption key (KEK) to the access control component. From this stage onwards, the rest of the data storage process is similar to the D-ABAC scheme.

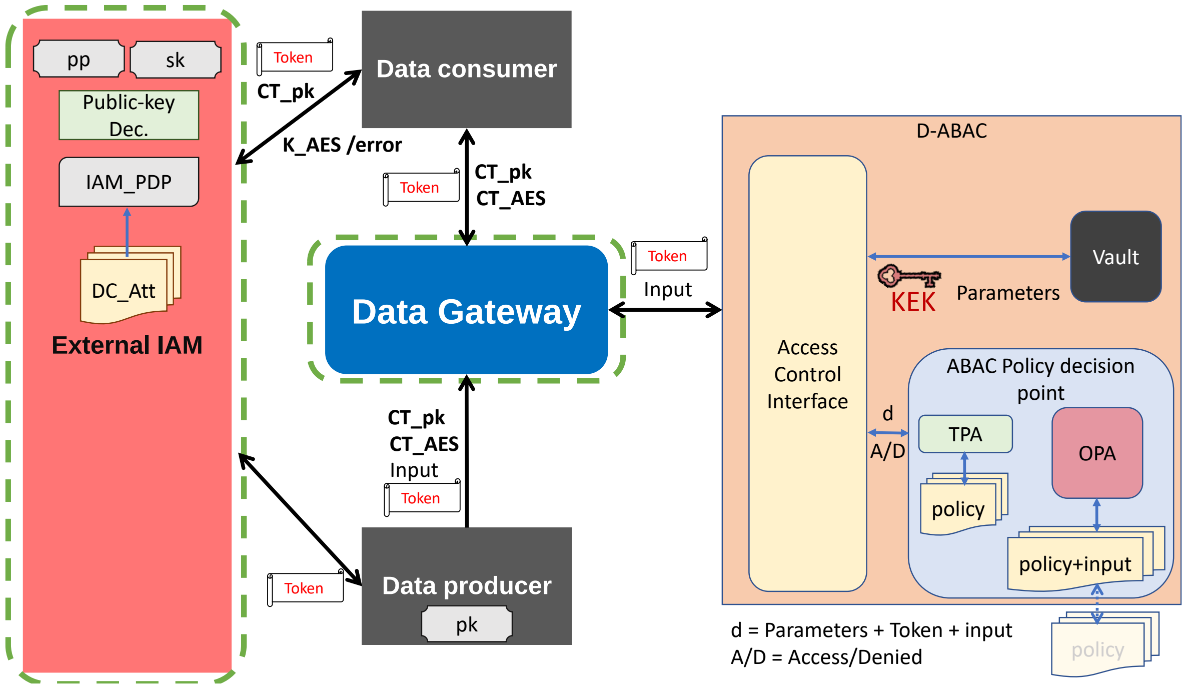


Figure 19 - C-ABAC extension on D-ABAC scheme

When a data consumer wants to access the encrypted medical data first authenticates themselves to the external IAM and obtains an authorization token. Then, the data consumer sends the authorization token to the DGW. The DGW relays this token to the access control component for the access request similarly as in the D-ABAC solution. When access is granted, the DGW will provide the data consumer with the tuple <`CT_AES` , `CT_pk` > (this decision is based on the OPA component with external security policies and input files). At this point, the data consumer sends the `CT_pk` to the IAM. At this point, IAM runs the C-ABAC scheme (only **Public-key-Dec** algorithms) to obtain the tuple <$K\text{-}AES, \mathbb{A}$>. Then it runs the **IAM-PDP** algorithm of the C-ABAC scheme to retrieve the private key `K_AES`. Finally, IAM forwards this key to the data consumer iff the data consumer's attributes satisfy the access structure $\mathbb{A}$. Using `K_AES` and **AES-Dec** algorithm, the data consumer can then decrypt the encrypted FHIR data (for more details see C-ABAC solution).

CP-ABAC extension vs. C-ABAC extension:

As mentioned earlier, there are some strong points and shortcomings in both CP_ABAC and C_ABAC extensions. Table 1 compares the two extensions. It is important to note that both extensions are robust against an attack where the DGW and access control component collude, and are resistant against an attack where either the DGW or the access control components are compromised.

Table 1 - CP-ABAC extension vs. C-ABAC extension

Scheme	Strong points	Shortcomings
CP-ABAC	- no need for extra computation on the IAM side (the IAM only runs the Setup phase that can be offline and once). - no need to add extra flow to the current scheme.	- does not support key revocation. - computational costs compared to C-ABAC on both data producer and data consumer sides (1 AES and 1 ABE encryption/decryption on both data producer data consumer sides).
C_ABAC	- efficient key revocation. - low computation costs on both data producer and data consumer sides (1 AES and 1 public-key encryption on the data producer side and only 1 AES decryption on the data consumer side).	- needs extra computations in the IAM-side (i.e., the IAM runs the Public-key-Dec and IAM-PDP algorithms). - needs an extra flow between data consumer and IAM (higher communication cost compared to CP-ABAC).

IV. MOBILE DEVICE SECURITY (UAH)

During the first half of the ProTego project, we designed and developed the initial architecture for the ProTego Continuous Authentication (CA) model. This process was described in deliverable D5.2. During the second half of the project, this architecture was reviewed and extended. Also, during this part of the project, we completed different experiments to test the CA method for mobile devices resulting in the final modular CA system. The following subsections detail each of these.

IV.1.1. Overall achievements

IV.1.2. Continuous Authentication Integration

The Continuous Authentication (CA) solution covers the integration with the hospitals involved in the project, Marina Salud (MS) and Ospedale San Rafaelle (OSR). Each hospital has its own architecture and proprietary systems. The Continuous Authentication solution is aware of these singularities and provides a generic solution, which can be used in multiple scenarios.

Like the rest of the components of WP5, the CA module was Dockerized and prepared for the deployment in Kubernetes platforms, through the creation of Helm charts and also, for its deployment using Docker-compose.

1. Kubernetes deployment. The CA module implements a new functionality that facilitates creating a custom configuration profile for each partner's environment. One of the most significant challenges was the authentication of the partners' users for all the architecture components. Now, it is possible to use the same unique identifier of the partner's legacy system to associate user metrics and values. We redesigned the system to use several authorization providers so that it can rely on data from Amazon Cognito in the case of Marina Salud, and authorization tokens provided by OSR from Keycloak. The setup of the authorization provider can be made with a new initialization feature.
2. To accommodate possible future improvements and to improve versatility, several Continuous Authentication AI models can be registered. Each can perform different assessments of the user's behavior and produce different metrics. So, the decision system can be fed with more inputs to return more reliable authentication decisions (trust values).

On the other hand, the Android Agent of the CA module was completed and improved. It acts as an Endpoint Detection and Response (EDR) client that runs on a mobile phone. Each hospital has different approaches, to develop its mobile apps:

1. Marina Salud uses a mobile app called Pocket EHR developed on Flutter.
2. For OSR's Foodcoach mobile app, we developed a mobile app using Java/Kotlin and Android Web Views.

As Flutter does not allow the use of Java/Kotlin code, we needed to find a common solution integrable with both systems. So, in order to support both mobile apps the EDR client can be called through broadcast receivers. This provides a single-entry point to allow different technologies to communicate with our application. Furthermore, if the hospital wants to use their own user database and credentials, they need to share each user's identity, clients have to create a valid JWT token (i.e., signed with an authorization provider registered in the EDR backend) and then sent it through the broadcast receiver.

```
intent = Intent().setClassName("eu.protego.ca.edr.android",
"eu.protego.ca.edr.android.core.receiver.ProTegoJWTReceiver")
intent.action = "android.intent.action.SEND"
```

Figure 20 – CA component launch using a broadcast receiver

Finally, the trust values produced by the CA solution can be sent through a Kafka topic to the SIEM, which can trigger alerts.

IV.1.3. Final Architecture

Figure 21 shows the design architecture of the system. The redesign aims to meet the following objectives:

1. Separate the trust assessment from the CA decision-making system so that different types of clients can use the system.
2. Separate the authorization from the main CA logic to facilitate using several authentication systems for different users (e.g. MS does not use the same authentication methods, and its users are different than OSR or other Hospitals).
3. Enable the implementation of local CA systems but also combining their predictions in a holistic model. This facilitates the implementation of sensible models (from the point of view of data protection) in the mobile phone and combines their trust values with other (cloud) metrics to produce a global result.

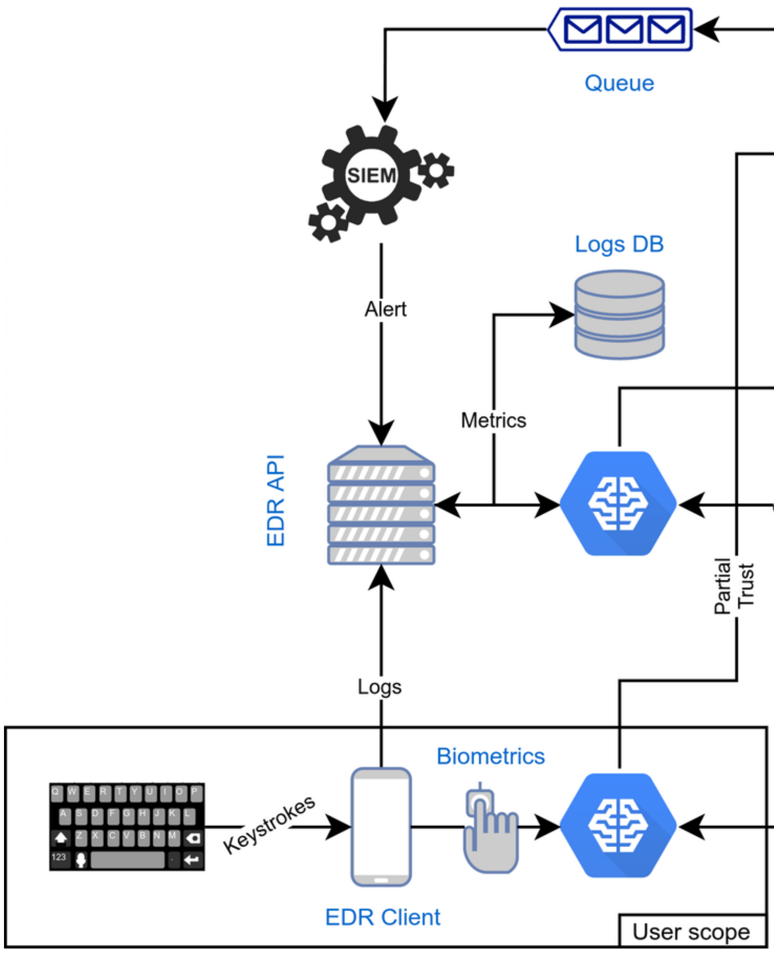


Figure 21 – Architecture diagram

Figure 22 shows how the system retrieves user metrics (like keyboard measurements, navigation between applications, or touch patterns) and processes them (every 5 minutes) to generate user AI models. After that, it evaluates the new incoming user metrics using the AI model (every 30 seconds), producing a trust value for each one. All trust values are uploaded to the CA decision system (named JBCA), which generates a final decision combining them. The EDR system queries the decision to maintain the session or lock the mobile device. The decision result is also sent to the SIEM and could be checked by the partners' backends.

A new AI model is generated when one of the following conditions is met: (1) there is no previous model, or (2) there is a previous model, and the trust value of the user's identity is over 50%. So, the new model is not created using the metrics of an impostor.

If an unauthorized user interacts with the mobile phone, the metrics new interaction will return a negative prediction of user identity, and the CA decision system will generate trust values under 50%, triggering the additional responses.

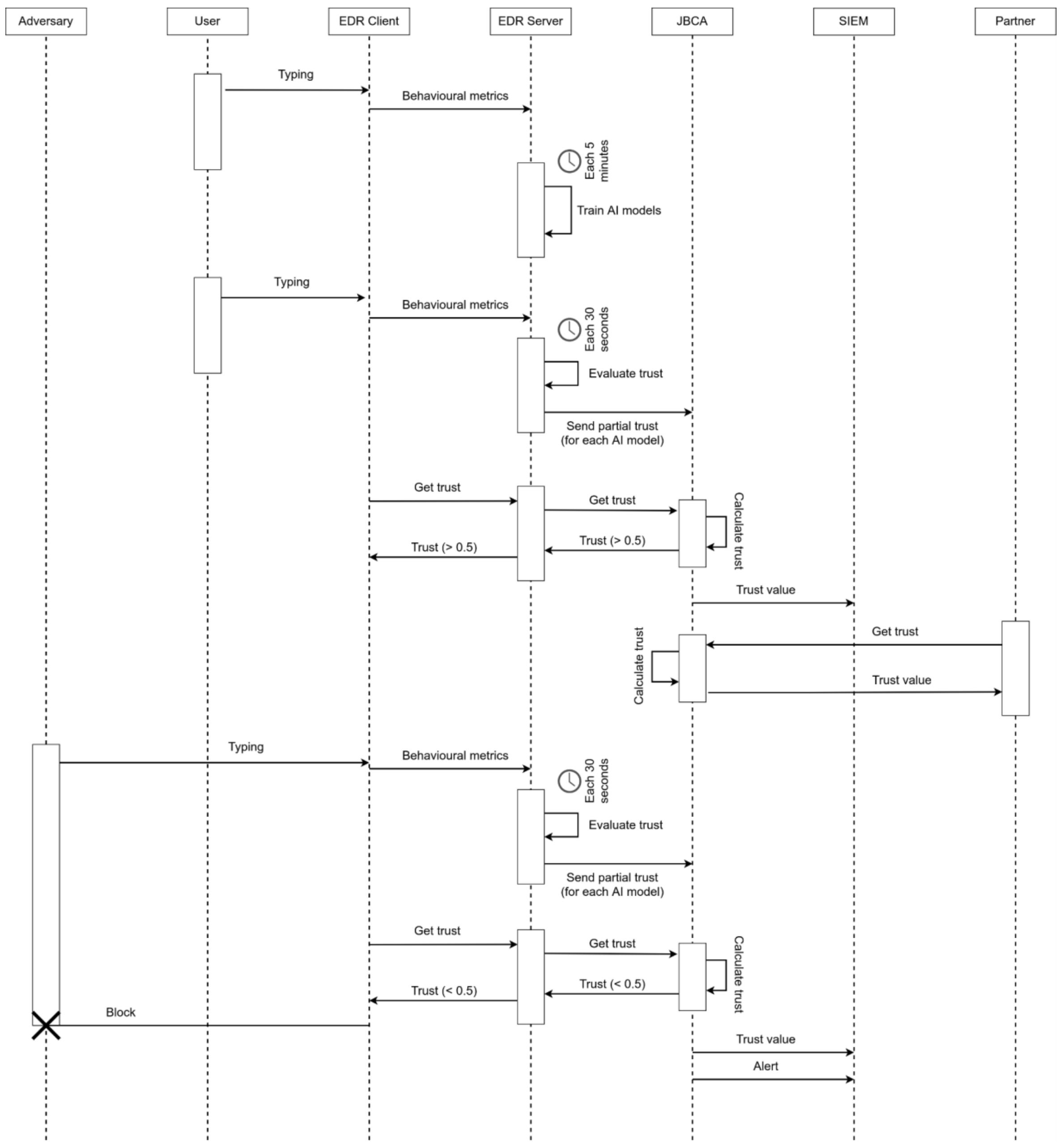


Figure 22 - Sample flow diagram

IV.1.4. Experiments with AI Models

After a successful evaluation from the ethical committee of the project and the ethical committee of the University of Alcalá, we performed a set of experiments with a larger population. The COVID19 pandemic made data collection for the experiments difficult but meeting all the safety guidelines we got a sample of over 50 users. Experiments focused on the keystroke metrics for mobile phones by implementing an EDR agent to collect them. Participants in the experiment completed the following steps:

1. Install a mockup CA EDR app and log in using an anonymous user created by the research team.
2. Input a predefined text. All users type the same text. Participants were not given instructions to follow any particular pattern or style. For example, they could write using capital letters or not and or make particular choices concerning grammatical rules or conventions.
3. Participate in a ten-minute conversation using their mobile phone, about a selected topic with another participant or a research team member.

Data gathered from these experiments is used to build and compare decision models and decide which machine learning (ML) classifiers better fit the CA decision problem. Furthermore, to extend our experiments and data, we used a public dataset called HMOG (al., 2016), transforming the data into the keystroke metrics supported by the EDR mobile agent. HMOG dataset records the interactions of 100 users during 24 sessions (8 reading sessions, 8 writing sessions, and 8 map navigation sessions). It includes sensor information, touch, gestures, and keypresses on the virtual keyboard. The HMOG dataset recorded raw data for each keypress event, including the timestamp, type (down or up), and key code. For the ProTego research, the keypress event information of all eight writing sessions of the HMOG dataset was transformed using a Python script that computed the pressingTime, timeBetween-Press, and timeReleaseNextPress based on the timestamps and type of keypresses reported in the dataset. These metrics were described in deliverable 5.2 and are used by the EDR mobile agent. Figure 23 presents these keystroke measurements graphically. The transformed HMOG dataset used in the ProTego research includes 712418 keypress events for the 100 users (range 4306-11917 events).

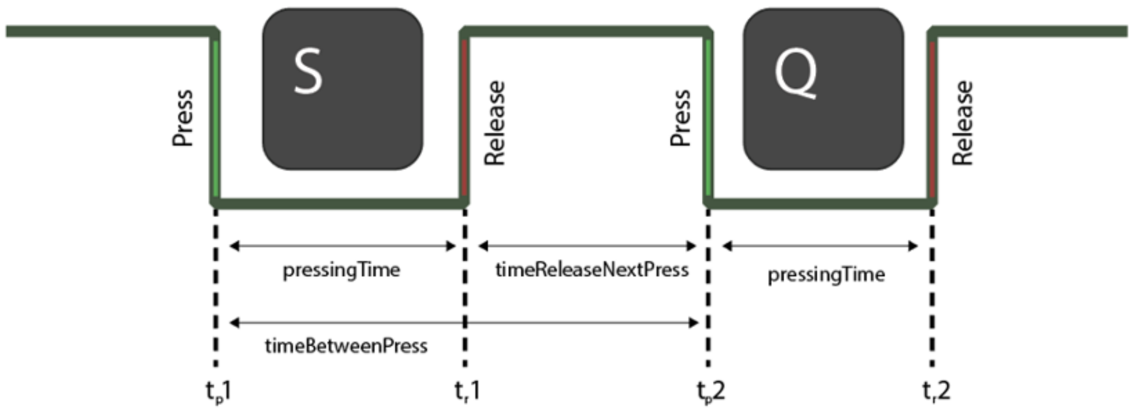


Figure 23 - Keystroke events and measurements

Initial experiments reported in deliverable 5.2 suggested that RFC (Random Forest Classifier) returned better results than other classifiers for the CA classification problem. Experiments performed in this last stage of research as part of WP5 showed that ensemble trees (RFC, GBC, and ETC) return similar results (Figure 24 shows a box plot of metrics of classifiers for the CA problem). We tested various ML classifiers, including ensemble trees, decision trees, instance-based, probabilistic, and neural models.

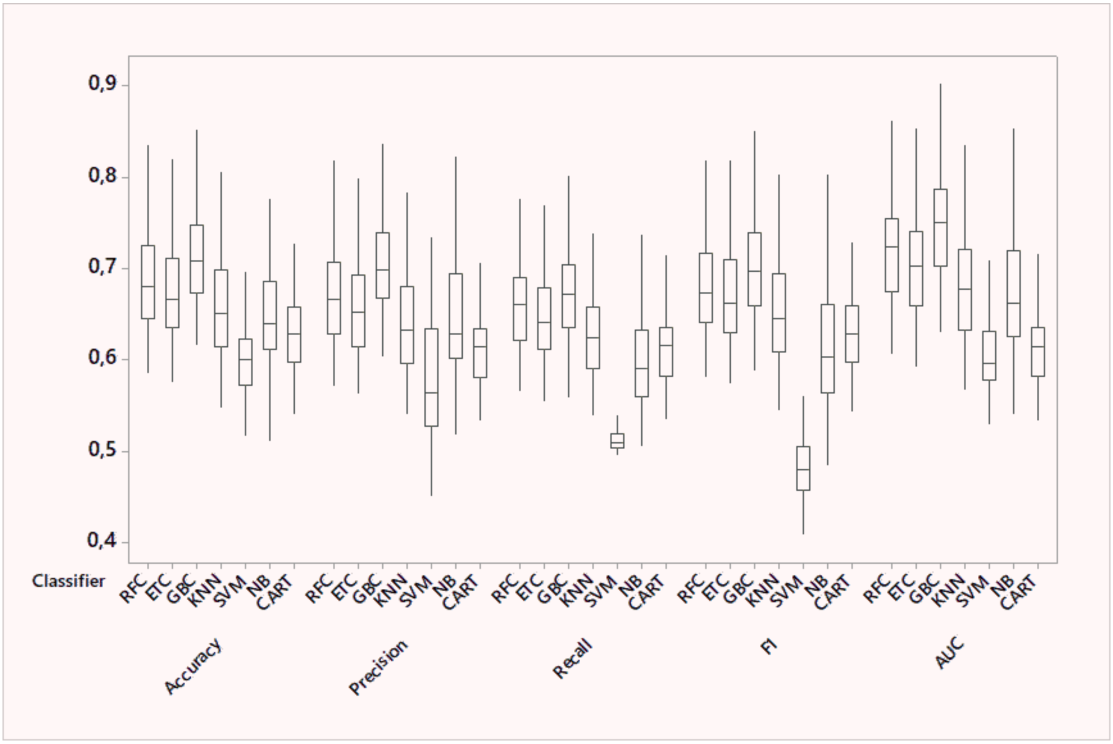


Figure 24 - Box plot of metrics of ML classifiers for the CA problem

Moreover, we run an analysis of variance (ANOVA) to compare the differences between the 100 users of the HMOG dataset for each classifier statistically. Results show that the differences are significant across different metrics: accuracy (F=46.73, p<0.001), precision (F=43.29, p<0.001), recall (F=90.58, p<0.001), F1 (F=160.99, p<0.001), and AUC (F=65.26,p<0.001). Figure 25 shows the confidence intervals for the accuracy metric, showing the differences graphically. Overlapping intervals (for each pair of classifiers) mean that there are no differences. Non-overlapping means that there are statistical differences between the two methods. GBC outperforms all other classifiers statistically, although differences with other ensemble classifiers (RFC, ETC) are marginal. So, all ensemble trees can be used to produce reliable CA AI models.

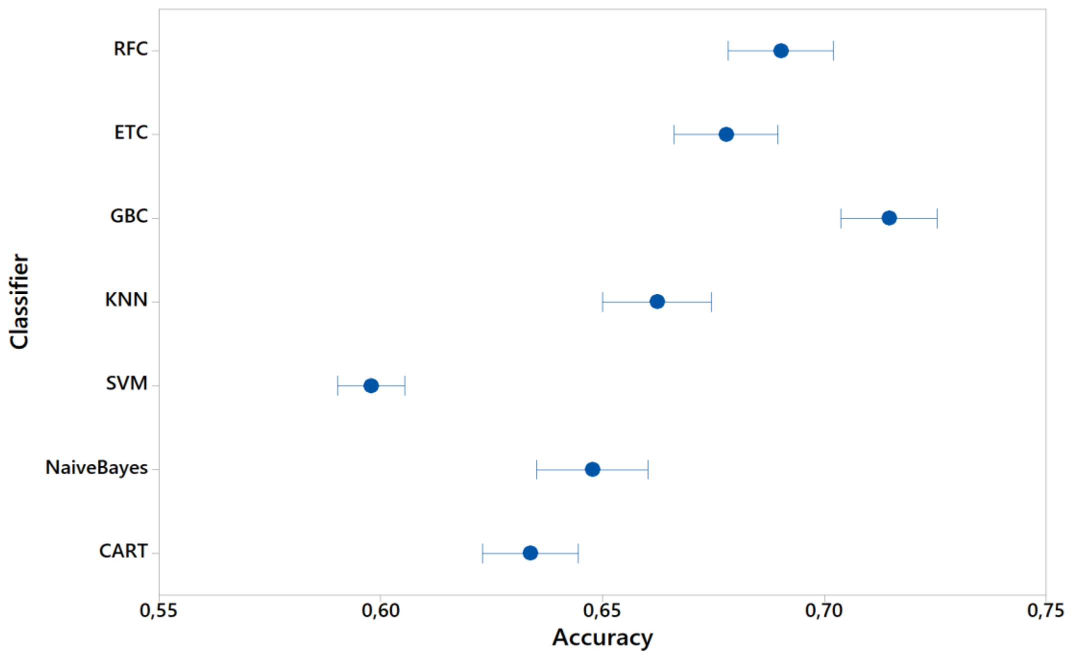


Figure 25- Interval plot of Accuracy for all classifiers (95% confidence interval of the mean).

To improve these results, we also tested the combination of several machine learning models through ensemble learning. Ensemble learning is the process by which multiple models, such as classifiers or experts, are generated and combined. However, ensemble learning did not improve the accuracy of prediction models substantially. Just a marginal 1% or 2% accuracy increase is obtained when combining the classifiers that returned the single best performance or a combination of classifiers that use different methods (e.g. ensemble tree + instance-based + probabilistic).

Results may suggest that the prediction is not very accurate since even for the best ML classifiers around 29% of the cases are incorrectly classified. However, with keystroke mechanics, every single event (i.e. key pressed) produces a prediction. So a few individual predictions can be combined to produce a more reliable result, thus mitigating the number of false positives and false negatives. The combination of multiple events and other possible sources of information can significantly improve the reliability of the CA system. The CA architecture of the ProTego project supports the integration of new sources of information coming from existing or new EDR agents. As part of the final stages of the project, we are working on fine-tuning the AI models to improve the results reported by the current scientific literature (al., 2016) (Frank, 2012) of mobile CA systems (around 5% increase for accuracy). We are collating the results of internal experiments to report them in scientific publications. Further trials at hospitals will also validate the findings.

IV.1.5. Privacy analysis

Data protection is a priority for ProTego. Experimentation and the CA solution collect behavioral information that has to be adequately managed. As mentioned earlier, all experiments and data collection for the CA experiments were supervised and approved by the Ethical Committee of the project and of the University of Alcala. The process guarantees that data is gathered and stored securely and anonymously. However, even though all information is stored anonymously, we performed a privacy analysis of the behavioral information data from two different perspectives:

- Compliance: the experiments and system meet privacy and ethical standards. They are appropriately reported and approved.
- Side channel attacks: a statistical analysis shows that even though data was anonymized, cryptoanalysis techniques could return information about user activity. Figure 26 shows a sample of how the touch events' coordinates can be clustered (e.g., using k-means algorithm), so they can be used to unveil the text typed. The research team at the University of Alcalá is conducting further research to implement enhanced privacy for the ProTego CA module, which could also be applied to any system that collects data related to user interaction with mobile devices.

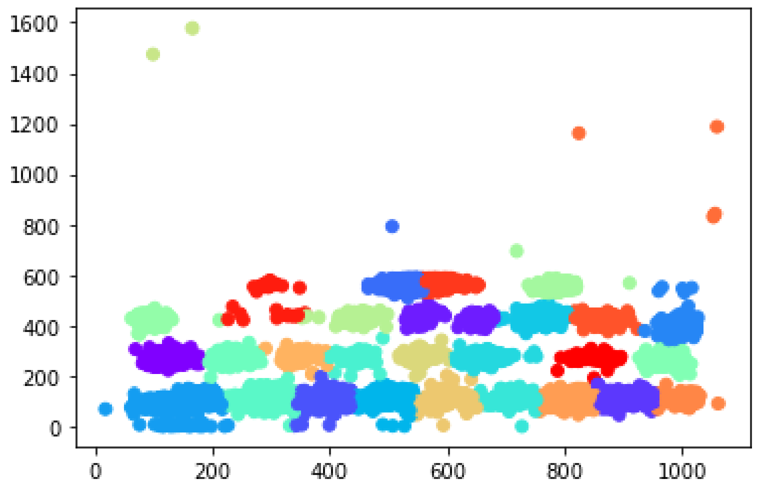


Figure 26 - Clustering sample (where X and Y are screen coordinates)

V. NETWORK SLICING ISOLATION

During the first half of the ProTego project, T5.4 efforts focused on providing network slicing isolation to the communication infrastructure. In this section we will present the final solution. We will first review the main network slicing ideas and later on, we will discuss the implementation details and additional features considered for the tool.

V.1.1. Description of the solution

The ProTego network slicing tool is an extra element of the ProTego toolkit to increase security at network level within the hospital network infrastructure. This allows to virtualize in logical topologies the available physical network elements. In ProTego, the Network slicing solution has two main functions: performance isolation and privacy isolation:

- Performance Isolation: one of the most important functions in network slicing is to provide data channel capacity allocation in a deterministic manner. This means that flows will have a certain agreed QoS independently of the network state (e.g. traffic congestion). Network slices are created on top of the physical infrastructure and virtualize the access to the network resources for the different flows. This way, resources will be used by each network slice in a predefined manner, managing priorities and schedules to achieve the targeted QoS. In the context of ProTego, secured flows need to have a guaranteed QoS (measured by e.g. bandwidth, latency and packet loss) to perform the required security processes. The network slice will provide such a per-flow isolated communication channel ensuring that each use case has its own required QoS.
- Privacy Isolation: the second function ensures that physical wired network can be mapped in secured, isolated logical topologies. Besides adding security and data channel confidentiality to each slice, the privacy isolation function logically separates each flow from each other, allowing per-flow topology management and per-flow encryption.

By applying network slicing, the ProTego network can provide strict performance policies, achieving two main goals: protection against Denial of Service attacks (by having isolated slices in terms of performance, the damage of a Denial of Service attack in one slice is limited to that slice, without impacting other existing network slices) and guaranteed performance for mission critical data flows (e.g. a slice for critical medical data can be configured with the necessary resources to guarantee enough bandwidth and low latency, and not be affected by other traffic with lower priority such as access revocation messages or SIEM flows).

The ProTego network slicing architecture is depicted in Figure 27 - Proposed Network Slicing architecture for ProTego. The first layer processes input from external actors, such as Mobile Network Operators or the private hospital network operator. This input is transferred through a predefined Northbound interface that configures well-known standard network slicing parameters (bandwidth, latency, packet loss, etc), or through a standard network slice type file, as specified by GSMA [GSMA]. These commands (or type files) are processed by the Slice Processor, which translates high-level network slice parameters to network specific configuration (e.g. airtime in the radio interfaces or topologies in the wired network). The Slice processor generates these performance and privacy rules, sending them to the Performance Isolation Engine and the Privacy Isolation Engine. The engines are then able to map the requirements to API calls to the virtualized resources, i.e. Software Defined Radio (SDR) controllers, SDN controllers, and the Secure Interface Setup. The SDR controller is responsible for configuring the radio parameters and prioritization queues in each Access

Point (AP) to reach the necessary performance for the specified service. The SDN controller will configure the SDN-enabled network (in terms of topology and intents) to steer the traffic securely and without performance degradation inside the hospital infrastructure. The Secure Interface Setup will configure the secure network interfaces with standard encryption techniques (e.g. AES-256-CBC + pub/priv keying) to enable the confidentiality control of the sensitive data. These secure network interfaces will be available for the SDN switches to control the critical service flow through them and, therefore, perform an extra and customizable security layer for hospital services.

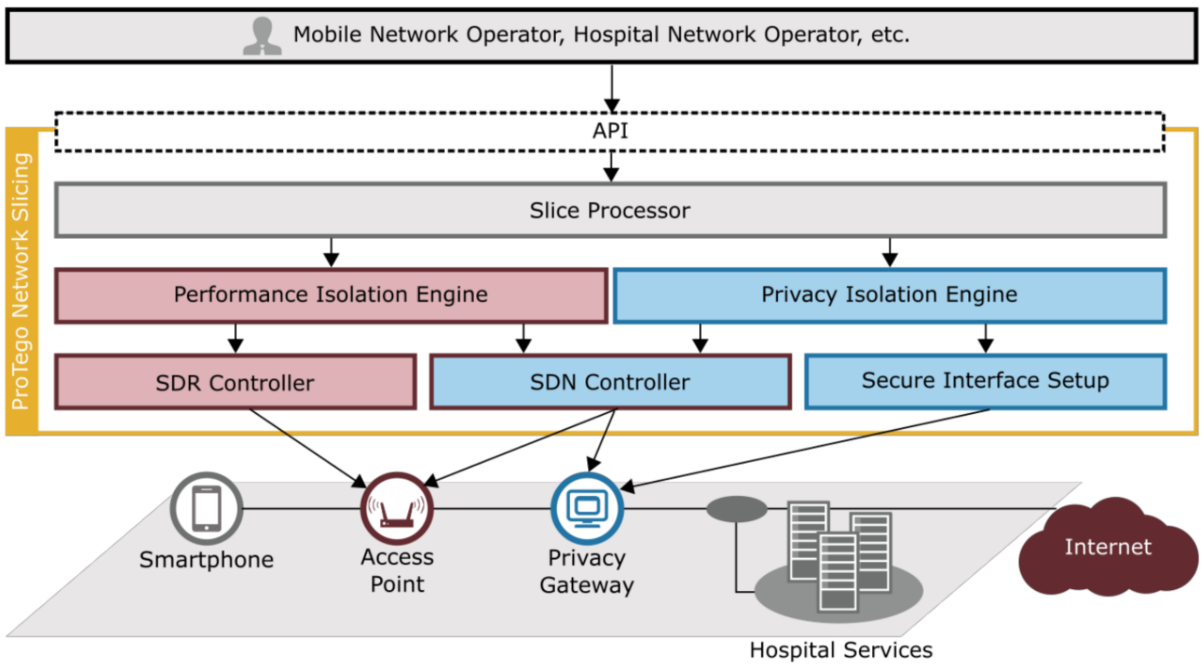


Figure 27 - Proposed Network Slicing architecture for ProTego.

The implementation of the ProTego Network slicing tool is done on the basis of two use cases: traffic differentiation within hospital data traffic, and traffic differentiation between devices from outside the hospital and devices from the hospital infrastructure:

- Traffic differentiation inside the hospital infrastructure. Network slicing can be used to potentially create multiple network slices on the hospital premises, each of the slices would have its own set of properties and requirements. This depends on the particular use case. The objective of creating network slices on the hospital premises is to take advantage of the two main functions of network slicing (performance and privacy isolation). There are two main categories of data to be protected in communication:
    - Critical medical data - Such as from/to surgery room devices, doctors, etc.
    - Critical security data - Such as data to SIEM, etc.
- Of course, each of those categories can have one or more network slices built for them. For example, for medical data flows, there can be two network slices, one for patients and one for the doctors. The doctors network slice would also take advantage of the data channel bandwidth guarantee, meaning the network slice used by doctors would have more resources allocated to it than the one used by patients. This ensures that critical medical data is available to doctors even in situations when the network is congested.

- Between devices outside the hospital infrastructure and devices in the hospital infrastructure. In this use case, network slicing can protect the critical medical data communication. On one hand, devices that are inside the hospital infrastructure can have a secure and guaranteed way of connecting to the network (this can be used by doctors or patients in the hospital). On the other hand, devices from outside the hospital (e.g., visitors), will have a limited and constrained access to the network. Using the main functions of network slicing, we can guarantee bandwidth to certain critical network slices, as well as have additional security in terms of network slice isolation.

V.1.2. Implementation of the solution

The proposed network slicing solution for ProTego is based on the 5G-EmPOWER framework [1], a state-of-the-art framework to perform network slicing on cellular and WiFi networks. Since most common and cost-effective networks deployed nowadays in hospitals are WLAN/LAN networks, we focus our solution for WiFi networks.

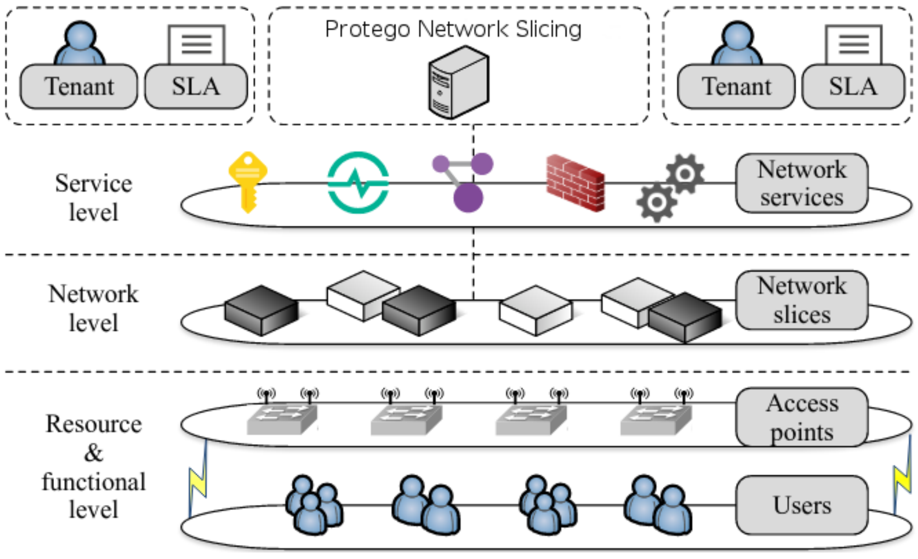


Figure 28 Network Slicing layered architecture performed in ProTego.

Figure 28 shows our network slice enabled layered architecture. A tenant (e.g. the hospital) can use their infrastructure to offer specific Service Level Agreement (SLA) to each user/service group. These SLAs are translated into QoS requirements to be provided by the network, e.g. minimum throughput, maximum allowed latency, etc.

The network slicing in a hospital environment will be managed by the hospital's network operator and slices can be divided per service or per device. Services in a hospital have different priorities, for instance, a nutrition application (low priority), and a hearth-beat monitoring application (high priority). Also, some devices may need to experience different performance, for example, a doctor's laptop can be assigned with more throughput than a patient's smartphone. Therefore, the network slicing will use the available tools to prioritize and schedule the traffic of one service/device over another service/device with lower priority.

[1] [Empower] 5G-EmPower web site. Smart Networks and Services (SENSE) Research Unit at FBK. [Online]. Available: https://5g-empower.io/ (2021, June)

5G-EmPOWER allows for SDN control and radio resource management on Wi-Fi networks which enables slicing techniques [Empower]. 5G-EmPOWER is composed of two main components: the EmPOWER controller and the Wireless Termination Points (WTPs). The EmPOWER controller is the component responsible for the management of the Radio Access Network (RAN) by deploying the necessary configuration in the WTP, which will prioritize and schedule the network traffic by modifying per-flow the most common parameters in the IEEE 802.11 standard family (e.g. DIFS, SIFS, etc.). Unless IEEE 802.11e which provides QoS differentiation over four traffic classes, EmPOWER permits a more deterministic differentiation of the radio resources. The WTPs are deployed in the Wi-Fi access points where the clients connect, and the network slice is monitored and managed.

In order to perform the wireless network slicing, 5G-EmPOWER utilizes virtualization techniques (logical links can be built over one or more physical links) and specific hardware (ath9k Wi-Fi cards) to add programmability to the Wi-Fi access points and provide different, fine-grained QoS for each of the connected users. The QoS is centrally managed by dynamically creating priority queues and steering the data packets to the queue that matches the necessary performance level.

In order to identify the service traffic, 5G-EmPOWER uses OpenVSwitch (OVS) to tag network packets which will enable the Wi-Fi scheduler to apply the right prioritization rule for that traffic flow. OVS is a OpenFlow-enabled virtual switch that enables traffic steering and network isolation using OpenFlow rules.

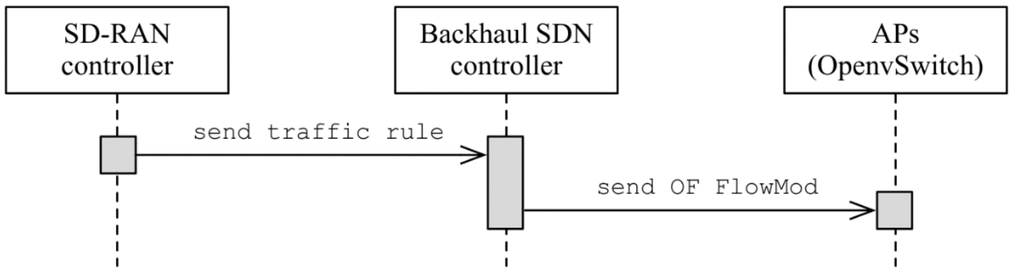


Figure 29 Sequence diagram of the traffic rule creation process.

Thanks to the programmable behavior of OVS, the 5G-EmPOWER controller can control the network slicing at the different APs and allows the STAs to handover seamlessly from one AP to another. Figure 29 illustrates the sequence of communication from the 5GEmPower controller to the OVS. For privacy isolation, the OVS is also used to steer the traffic among different network interfaces that can tunnel the network traffic, apply encryption techniques or just send raw data to the network.

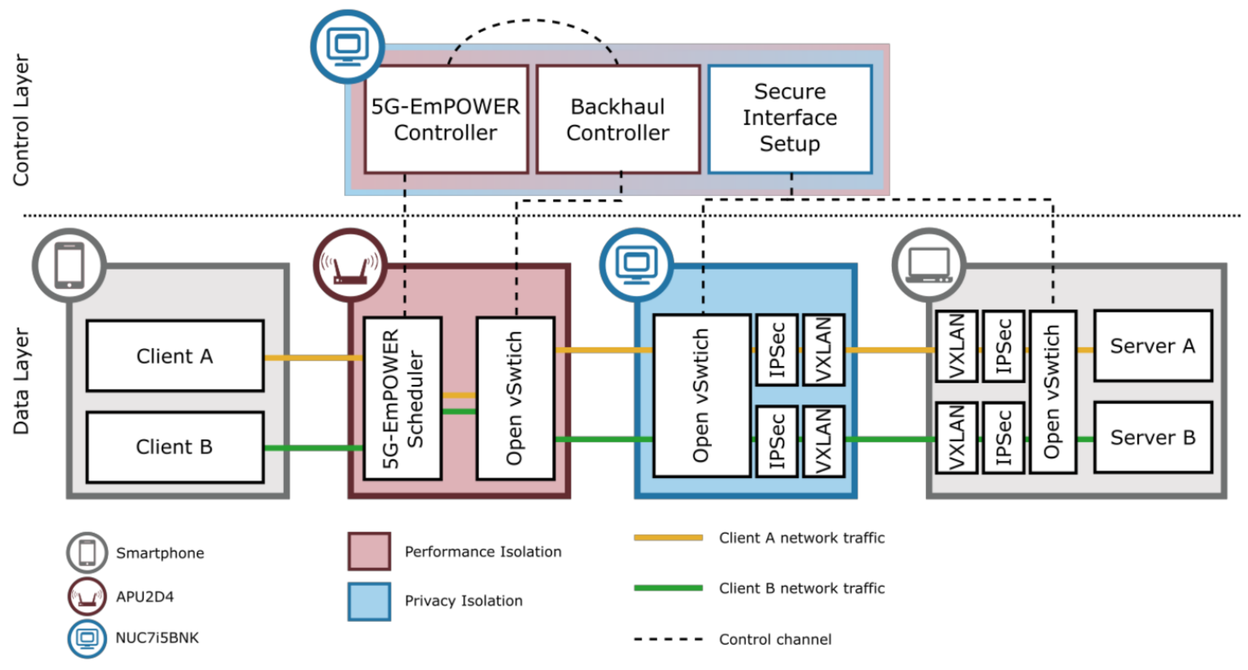


Figure 30 - Slicing enablers for ProTego.

The proposed 5G-EmPOWER based network slicing solution is illustrated in Figure 30. It presents the control and data channels of the different service flows we foresee in a hospital network environment. In the envisioned scenario, the wireless clients connect to the Wi-Fi APs with the 5G-EmPOWER WTP enabled. These clients perceive only one Wi-Fi Service Set IDentifier (SSID) since the APs are virtualized. This way, the 5G-EmPOWER controller can directly control the APs as WTPs, commanding them what to do with each packet. This way, packets can be filtered accordingly to select which WTP is effectively carrying the traffic. Additionally, that WTP will manipulate accordingly their Wi-Fi priority queues and airtime metrics to assign the required wireless slice to each client. This process is illustrated in Figure Figure 31, where the queue structure is presented along with the data traffic flow within a Wi-Fi access point.

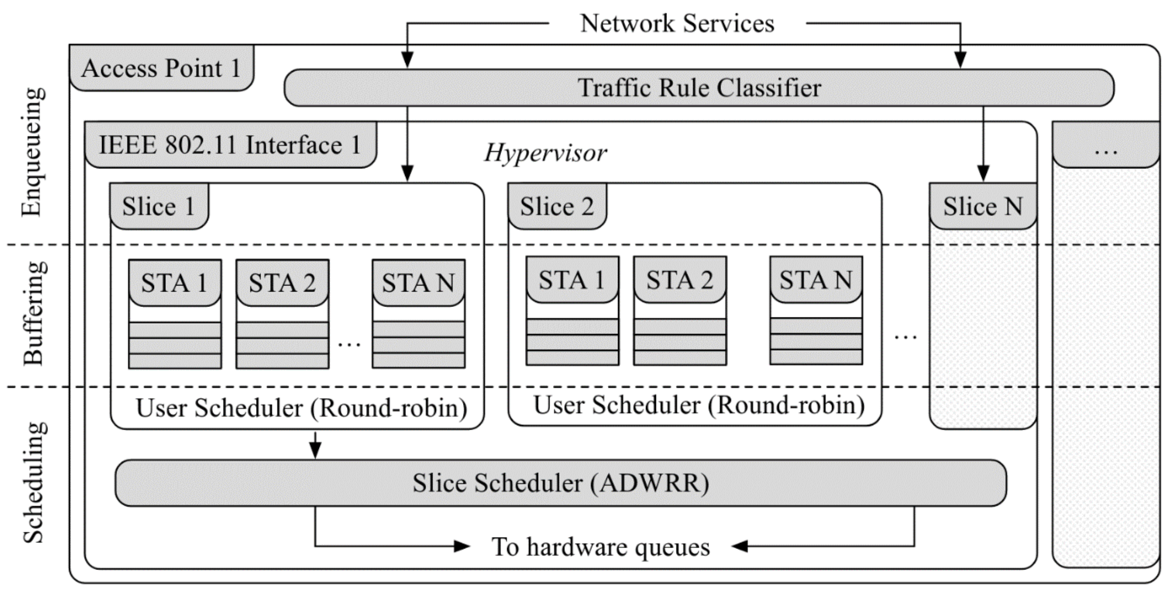


Figure 31 – Slice queue structure and data traffic flow in a Wi-Fi access point.

In the wired segment, traffic is forwarded by an Openflow enabled switched network (e.g., with OVS) to the corresponding VXLAN (running each an IPSec instance). Traffic in this switch network is controlled by the Backhaul controller. VXLAN creation and configuration is done by the Secure Interface Setup which is in charge of deploying the different virtual local networks over the hospital internal network. The control plane (5G-EmPOWER controller, backhaul controller and the Secure Interface Setup) can be deployed in different containers, so that different control services are isolated one from another and may be dynamically dimensioned according to the demand. We refer to D7.2 for preliminary results of this tool in a controlled small-scale testbed and to D7.3 for integration results in a real hospital network.

V.1.3. Network Slicing Tool integration

In addition to the stand-alone network slicing solution described above we have considered specific add-ons for the network slicing tool to further improve the risk mitigation:

- Logging system integrated with SIEM: the 5G-EmPOWER will rely now on SIEM to record logs coming from the network slicing tool. In D5.2 (public) we proposed and developed a blockchain-based plug-in prototype for the tool to securely record changes in the network slice configuration (e.g., when a certain slice is created, or a policy is updated). However, our approach was not cost-effective and had some limitations: required redundant resources to setup the blockchain or rely in an external blockchain, slow transaction time, etc. Since ProTego already has a SIEM as a logging system to detect potential vulnerabilities, we opted to replace the blockchain-based plug-in with one based on the SIEM. This way, not only the changes in the configuration of the network slices will be recorded in the SIEM (and analyzed to detect potential vulnerabilities) but also the other logging messages from the network slicing tool will be reported, such as the detection of anomalous traffic or detection of malicious scans in the WLANs controller by the tool. In order to achieve this, the 5G-EmPOWER controller is now integrated with the SIEM system (i.e. communicates with the Kafka server) to report both business-as-usual logs and network-triggered alerts.
- Token-based network slicing authentication: the current network slicing prototype has the limitation of using information contained in the data packets to classify and assign the traffic to each slice. This is mainly the MAC address and the DSCP/ToS code existing in the data packets. This creates the problem that attackers can spoof such fields to impersonate other devices and services, interfering in the pre-defined network isolation setup defined by the network operator. While the functions of the network slicing will still apply (e.g. the performance isolation will still avoid DoS attacks to propagate from one slice to another), it would be convenient to provide an authentication system to the network slicing tool. We foresee that the most efficient way to perform this is to implement a two-way authentication system that first registers the device in the system (i.e., checks that the devices have the right to use the network and assigns them an authentication credential) and second makes the devices use a token derived from the obtained credentials in every packet to let the network slicing solution check the authenticity of the packet (MAC and DSCP/ToS are legit).

VI. SUMMARY

Work package 5 has provided ProTego and the scientific community in general, with innovations to protect data. While healthcare and the hospital environment were chosen as our main focus, these innovations can be applied a wide range of other use cases.

Work package 5 was instrumental in the development and acceptance of the Apache Spark Parquet Modular Encryption standard and help spearhead the development of the official code release. Additionally, advanced data protection topics in this work package are contributing to IBM's Open Source development of "The Mesh for Data".

The research in work package 5 advances the state-of-the-art in access control for e-health systems and provides a practical solution that offers a good trade-off between efficiency, flexibility and security. The CP-ABAC scheme offers security against a compromised access control and key management system, while still allowing to enforce fine-grained access control policies during read requests on sensitive medical data. Moreover, an alternative solution (C-ABAC) has been studied as well to support the revocation of users' attributes in addition to the security properties already offered by C-ABAC. Both schemes offer high security guarantees and are suitable to be applied in the use cases of the ProTego project.

The CA module contributes to the ProTego project mobile device security through a scalable architecture to continuously authenticate users based on their interactions with their mobile phones. Research contributes to the state of the art with an increase in the accuracy and other target metrics using keystroke mechanics with the soft keyboards of mobile devices. Further, the architecture supports the addition of new EDR agents that can provide additional data to feed AI user models and also report their own trust values, which are combined to produce a single value of user trust. To the best of our knowledge this is the first integrated mobile CA EDR solution based on a flexible dynamic AI models, user trust values and an identity threshold.

Additionally, the network slicing component contributes to ProTego by enabling network isolation in the hospitals' internal networks. The research done proposes to use state-of-the-art airtime-based network slicing to provide 5G-like performance isolation for mission critical applications in the hospitals. This is done by "softwarizing" the IEEE 802.11 networks and centralizing its control to offer to each network slice a guaranteed performance. We also contribute by integrating the proposed solution with existing security tools to offer additional privacy isolation to each network slice.

VII. Works Cited

al., Z. S. (2016, May). *H-MOG Data Set: A Multimodal Data Set for Evaluating Continuous Authentication Performance in Smartphones*. Retrieved from http://www.cs.wm.edu/~qyang/hmog.html

Bethencourt, J. A. (2007). Ciphertext-Policy Attribute-Based Encryption. *IEEE symposium on security and privacy (SP'07)* (pp. 321-334). IEEE.

Bursell, M. (2019, August 16). *Trust no one run everywhere - introducing Enarx*. Retrieved from https://next.redhat.com/2019/08/16/trust-no-one-run-everywhere-introducing-enarx/

Frank, M. e. (2012). Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security 8.1* , 136-148.

IBM. (n.d.). *Mesh for Data*. Retrieved from github: https://github.com/mesh-for-data/mesh-for-data

Intel® SGX Technology and the Impact of Processor Side-Channel Attacks. (2020, March 10). Retrieved from Fortanix: https://fortanix.com/blog/2020/03/intel-sgx-technology-and-the-impact-of-processor-side-channel-attacks/

Open Policy Agent. (n.d.). Retrieved from Policy-based control for cloud native environments: https://www.openpolicyagent.org/

Sahai, A. a. (2005). Fuzzy identity-based encryption. *Annual international conference on the theory and applications of cryptographic techniques* (pp. 457-473). Springer.