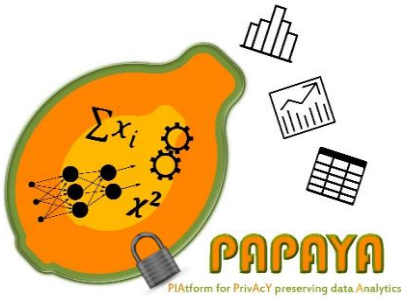


D4.3 – FINAL REPORT ON PLATFORM IMPLEMENTATION AND PETS INTEGRATION

Work Package	WP 4, Platform Design and Development
Lead Author	Boris Rozenberg, Ron Shmelkin (IBM)
Contributing Author(s)	Beyza Bozdemir, Orhan Ermis, Melek Önen (EURC) Sebastien Canard, Bastien Vialla (ORA) Angel Palomares Perez (ATOS) Tobias Pulls, Simone Fischer-Hübner, Tobias Vehkajärvi, Elin Nilsson (KAU)
Reviewers	Orhan Ermis (EURC) Tobias Pulls (KAU)
Due date	30.04.2021
Date	28.04.2021
Version	1.0
Dissemination Level	PU (Public)

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, through the PAPAYA project, under Grant Agreement No. 786767. The content and results of this deliverable reflect the view of the consortium only. The Research Executive Agency is not responsible for any use that may be made of the information it contains.

Revision History

Revision	Date	Editor	Notes
0.1	21.02.2021	Boris Rozenberg (IBM)	ToC
0.2	11.04.2021	All contributing authors	A version for the 1st internal review
0.3	21.04.2021	All contributing authors	A version after the 1st internal review
0.4	28.04.2021	All contributing authors	A version after the 2nd internal review

Table of Contents

Executive Summary

Rather than several standalone modules, the PAPAYA project aims at developing an integrated platform for privacy-preserving data analytics to make them available in a broad spectrum of products and services. The main goal of the platform is to be used by service developers to deploy and run privacy-preserving services, and by service consumers, that are interested to employ privacy-preserving analytics. In the deliverable D4.1 [1] we presented a first version of the platform functional design, architecture, and deployment. In particular, (1) we described the design of the main platform components that were elicited based on the requirements presented in D2.2 [2], (2) we explained how different privacy-preserving primitives being developed in WP3 (see D3.1 [3] and D3.3 [4]) are integrated into the platform in a way that they will be interoperable/compatible with each other and could work together in the integrated platform; and (3) we presented the design of platform dashboards that provide the UI, configuration functionality, and visualization functionality. In the deliverable D4.2 [5] we describe changes to the services specified in D4.1 [1], provided description of several new services created on top of the approaches developed in WP3 and described in detail in D3.3 [4], and explained how to deploy and run all platform components. In this deliverable we report on finalizing the platform implementation and PETs integration: the platform is deployed on the IBM K8s cloud account and it is in use by the project partners. In addition, we provide still missing descriptions for some of the services and provide links for movies demonstrating the services at work.

As already mentioned in D4.1 [1], PAPAYA platform services are specified based on the four generic usage scenarios, namely upload model, create model, apply model and collaborative training. In upload model, an already trained machine learning (ML) model can be uploaded to the PAPAYA platform when the client wants to delegate the computationally intensive task (which is applying a model on the client's sensitive data) in a privacy-preserving manner. The create model is used when the client is not able to create the ML model; therefore, the PAPAYA platform generates a model on the protected data shared by the client. Apply model is the use case where already uploaded or created model is applied on the client's protected data in a privacy-preserving manner. Finally, in collaborative training, two or more participants perform a ML training collaboratively while preserving the privacy of the training data.

The PAPAYA platform consists of the following groups of services and tools:

- Privacy-preserving analytics defined in the deliverables D3.1 [3] and D3.3 [6] such as classification on Neural Networks, privacy-preserving clustering, privacy-preserving statistics, and privacy-preserving collaborative training of Neural Networks.
- Security and transparency services, including the identity access management (IAM) for authentication and authorization services to the different components integrated in the PAPAYA platform, auditing to support auditing and towards being able to hold stakeholders accountable for their use of PAPAYA, and key manager for managing the cryptographic material during the whole lifecycle of the PAPAYA project in the cases where it will be required.

- The PAPAYA platform provides two dashboards for configuration and visualization, namely the Platform Dashboard and the Agent Dashboard. The Platform Dashboard is used for configuration and monitoring the services provided by the platform whereas the Agent Dashboard is used for viewing the data processing logs from an agent and for showing the configuration of the agent.
- The Data Subject Toolbox provides a number of mostly independent tools (Explaining Privacy-preserving Analytics, Data Disclosure Visualization, Annotated Log View and Privacy Engine) related to the privacy of the data subject. Moreover, the PAPAYA platform provides means to its clients to integrate one or more tools from the provided toolbox to create an integrated *data subject dashboard*.

As a proof of concept usage, the platform is deployed on IBM Kubernetes[1] cloud service. In addition to that, all platform services and tools are designed to be generic to be deployable on other cloud platforms.

[1] https://kubernetes.io/

Glossary of Terms

2PC	Two-Party Computation
AA	Auditing Agent
ABY	Arithmetic sharing, Boolean sharing and Yao's garbled circuits framework
AC	Auditing Collector
ALT	Annotated Log view Tool
API	Application Programming Interface
BF	Bloom Filters
BGV	Brakerski-Gentry-Vaikuntanathan homomorphic encryption scheme [7]
BFV	Brakerski/Fan-Vercauteren fully homomorphic encryption scheme [8, 9]
CA	Certificate Authority
CKKS	Cheon-Kim-Kim-Song fully homomorphic encryption scheme [10]
CLI	Command Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CR	Container Registry
DB	Data Base
DC	Data Controller
DP	Differential Privacy
DNN	Deep Neural Network
DS	Data Subject
DSRM	Data Subject Rights Manager
DVT	Disclosure Visualization Tool
ECG	Electro cardiogram
ES	ElasticSearch
FC	Fully Connected
FE	Functional Encryption
FHE	Fully Homomorphic Encryption
GRU	Gated Recurrent Unit
HE	Homomorphic Encryption
IAM	Identity Access Manager
KM	Key Manager
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PE	Privacy Engine
PHE	Partially Homomorphic Encryption
PoC	Proof of Concept
PPM	Privacy Preferences Manager
RAM	Random Access Memory

REST	Representational State Transfer
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SIMD	Single Instruction Multiple Data

1 Introduction

1.1 Purpose and Scope

The purpose of this deliverable is: (1) to complete the platform design and architecture already presented in D4.1 [1] and D4.2 [5] by providing modifications and/or missing details to several already described services; and (2) to provide several movies demonstrating the platform at work. The intended audience for this document consists of two main groups: (1) service developers, who could use the document to understand the design of the existing services and to develop and deploy their own services; (2) service users, who could use the document to understand how to employ the services available on the platform and how to incorporate them in their applications.

It is important to note that the description of the underlying algorithms employed by the services presented in this document is not in the scope of this deliverable: they are described in D3.1 [3] and in D3.3 [6].

1.2 Structure of the Document

The rest of the document is organized as follows:

- **Section 2** provides a high-level overview of the PAPAYA framework, including description of main stakeholders.
- **Section 3** provides an overview of the core PAPAYA services, which are described in details in D4.1 [1] and D4.2 [5] and provides modifications and/or missing details to some of them. Provided services in the scope of this deliverable are: (1) Privacy-preserving classification on Neural Networks; (2) Privacy-preserving collaborative training of Neural Networks; (3) Privacy-preserving clustering; and (4) Privacy-preserving statistics.
- **Section 4** summarizes how security and transparency is achieved in the platform, including authentication, authorization, auditing and key management for cryptographic tools (if it is required), and provides modifications and/or missing details to some of the services.
- **Section 5** describes PAPAYA dashboards, namely the Platform Dashboard and the Agent Dashboard.
- **Section 6** dedicated to the Data Subject Toolbox, which provides versatile tools (Explaining Privacy-preserving Analytics, Data Disclosure Visualization, Annotated Log View and Privacy Engine) related to privacy of data subjects. Tools presented in this section can then be used as part of a *data subject dashboard*.
- We conclude the deliverable in **Section** 7.

2 PAPAYA Framework

2.1 Main Stakeholders

On a high-level, as mentioned in the deliverable D4.2 [5], there are four main stakeholders in the PAPAYA framework:

1. ***Platform clients:*** stakeholders who wish to perform some analytics in a privacy-preserving manner. ***Platform clients*** can be considered as Data Controllers or external queriers who are allowed to request some analytics results while not being the actual owners of the data.
2. ***Platform administrators:*** responsible for platform administration purposes such as resource allocation or monitoring.
3. ***Service providers***: the author of the services available on the platform.
4. ***Data Subjects:*** *end-users* (e.g. application user) of the *Platform clients*.

Details about PAPAYA usage scenarios can be found in D4.1 [1].

2.2 PAPAYA framework architecture

The PAPAYA framework (see Figure 1) is composed of two main groups of components: (1) the platform-side components that are running on the (non-trusted, but semi honest) Kubernetes cloud server; and (2) client side components, that are running on trusted client environment.

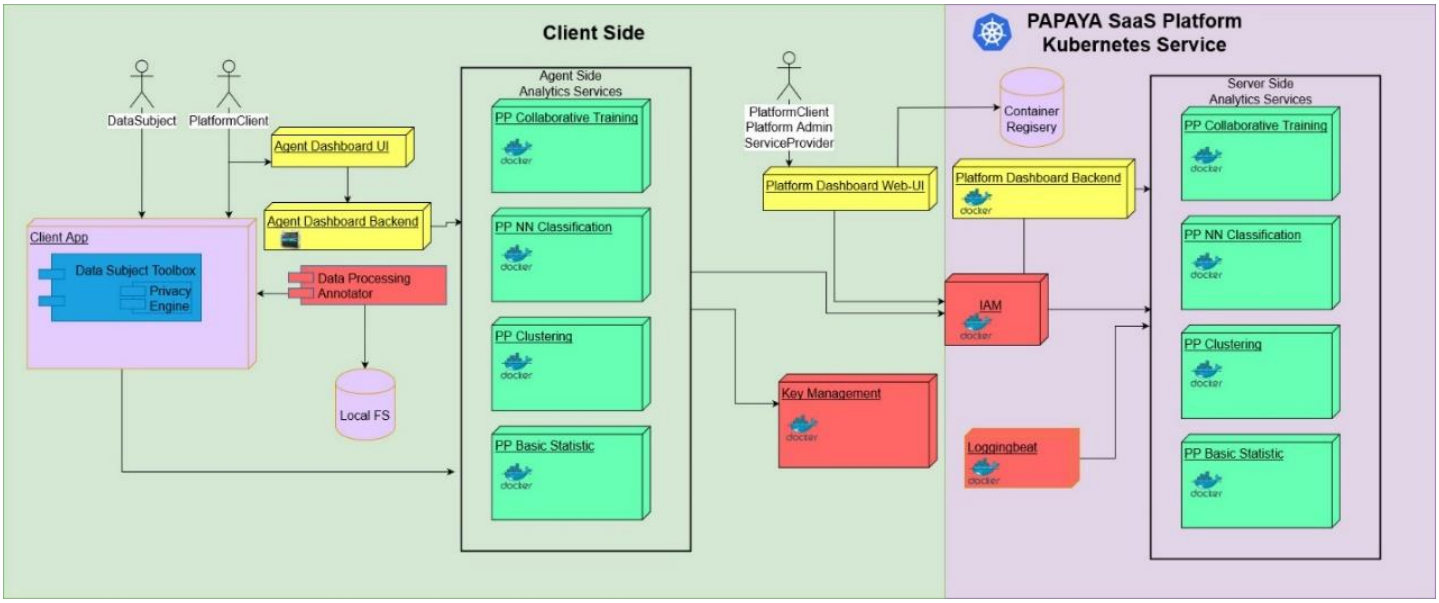


Figure 1 PAPAYA Framework architecture

From the functional perspective, the PAPAYA framework components can be regrouped into the following categories (more details about each category can be found in deliverables D4.1 [1] and D3.1 [3]):

- **The privacy-preserving analytics services** (colored with green in Figure 1) which allow platform clients to perform analytics of interest, in a privacy-preserving manner. We currently support the following analytics:
  1. Privacy-preserving NN classification. We provide four services for applying neural network for the purpose of classification in a privacy-preserving manner: (1) 2PC-based; (2) PHE-based; (3) FHE-based; and (4) Hybrid approach. Each one could be preferable than others in different settings, mainly depending on NN architecture
  2. Privacy-preserving collaborative training of NN. The service allows multiple participants to perform a ML training collaboratively, while preserving the privacy of the training data.
  3. Privacy-preserving trajectory clustering. The service provides means to cluster trajectories in a privacy-preserving manner.
  4. Privacy-preserving basic statistics. The service provides means for privacy-preserving computation of statistics using functional encryption and privacy-preserving counting using Bloom Filters.

  The users of these services are platform clients who can either be Data Controllers or external queriers (in the case for privacy-preserving trajectory clustering who are authorized by Data Controllers to request some analytics results. Each service is divided into two parts:
  1. Server – responsible for performing analytics of interest on encrypted data and runs on a PAPAYA's Kubernetes cluster.
  2. Agent – responsible for communication with the appropriate server-side component and responsible for managing cryptographic operations for the client. The agent should be downloaded from the Container Registry (CR) as a Docker image and deployed on a client side. Deployment and execution are under the responsibility of the platform client.
- **The platform security and transparency services** (colored with red in Figure 1) which provide platform authorization, authentication (Identity and Access Management - IAM), auditing and cryptographic Key Manager (if needed).

- **The PAPAYA dashboards** (colored with yellow in Figure 1):

  1. **Platform Dashboard** allows: (1) **service providers** to deploy privacy-preserving services; (2) **platform clients** to choose/run privacy-preserving services and review operational and auditing logs; and (3) **platform administrators** to configure and manage the platform.
  2. **Agent Dashboard** allows **platform clients** to visualize the configuration of the agent running on the client side and review operational and auditing logs.

- **Data Subject Toolbox** (colored with blue in Figure 1) consists of a number of mostly independent tools (DS Tool 1: Explaining Privacy-preserving Analytic, DS Tool 2: Data Disclosure Visualization, DS Tool 3: Annotated Log View and DS Tool 4: Privacy Engine), which provide versatile tools for data protection by design by platform clients (acting as Data Controllers) towards data subjects whose personal data is processed in their services.

3 Platform Core Services

3.1 Apply Neural Network Model

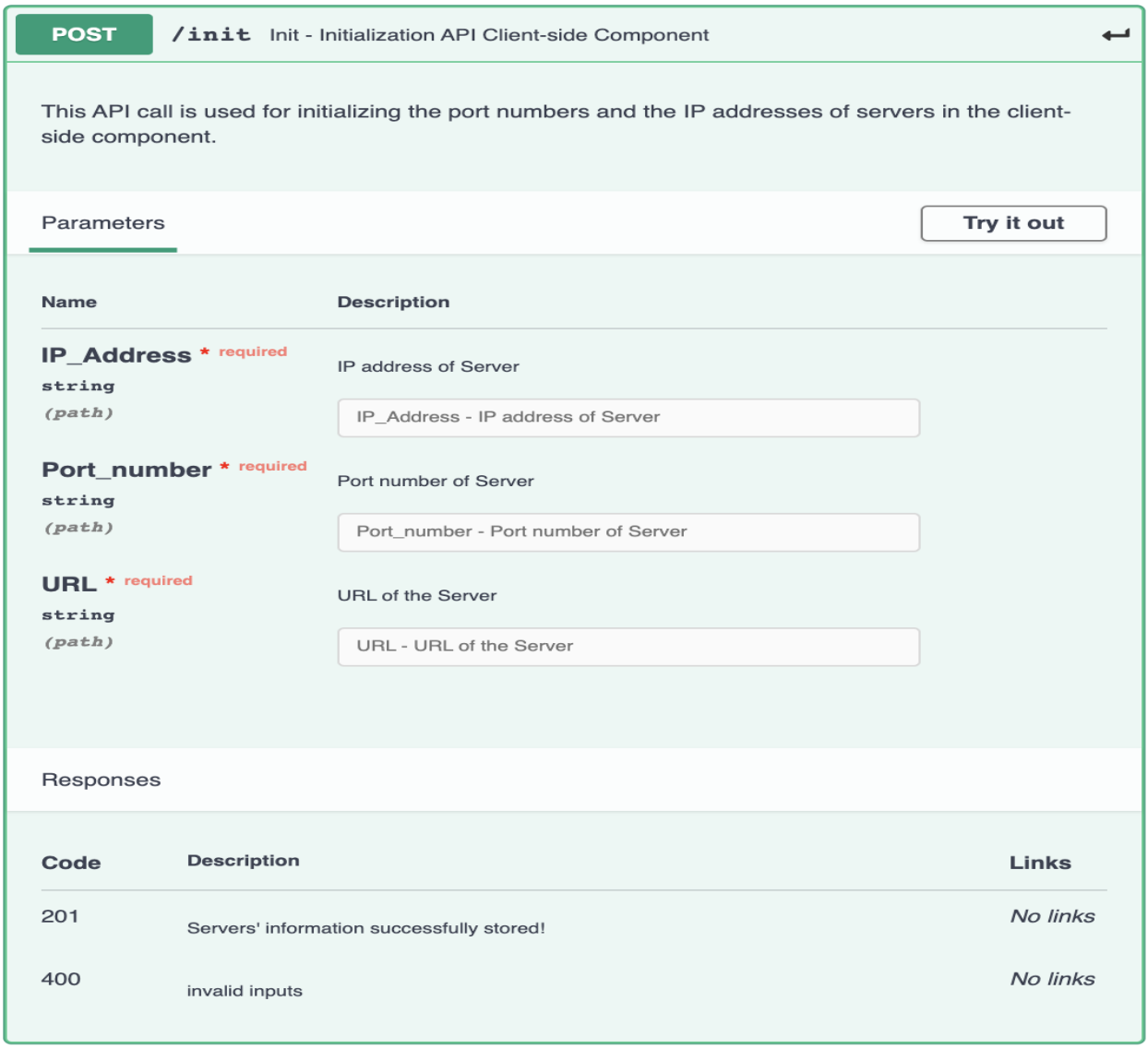
In this section, we complete the description provided in D4.1 [1] and D4.2 [5] for several services for applying neural network for the purpose of classification in a privacy-preserving manner. The detailed description of the underlying technology these services based on can be found in deliverables D3.1 [3] and D3.3 [6].

Prior to using any of the services described in this section, a Neural Network model should be trained locally based on the clear text data with all the required optimization and the dimensionality reduction. After achieving the desired accuracy, the trained model (i.e. architecture and weights) should be saved in a supported format and passed to the service later as described in each of the following subsections.

3.1.1 Privacy-preserving NN classification based on 2PC

This 2PC-based NN classification solution uses the ABY library[2] as the secure two-party computation (2PC). The solution uses a NN model that is particularly trained for arrhythmia classification. The detailed design and behavioral analysis for the service are introduced in the deliverable D4.1 [1], and the integration evaluation are presented in the deliverable D4.2 [5]. A demo of this service is available at https://www.papaya-project.eu/dissemination/demos.There is only slight update on the one of the client-side component's APIs. We have updated the API definition of the init/ call as shown below:

[2] https://github.com/encryptogroup/ABY

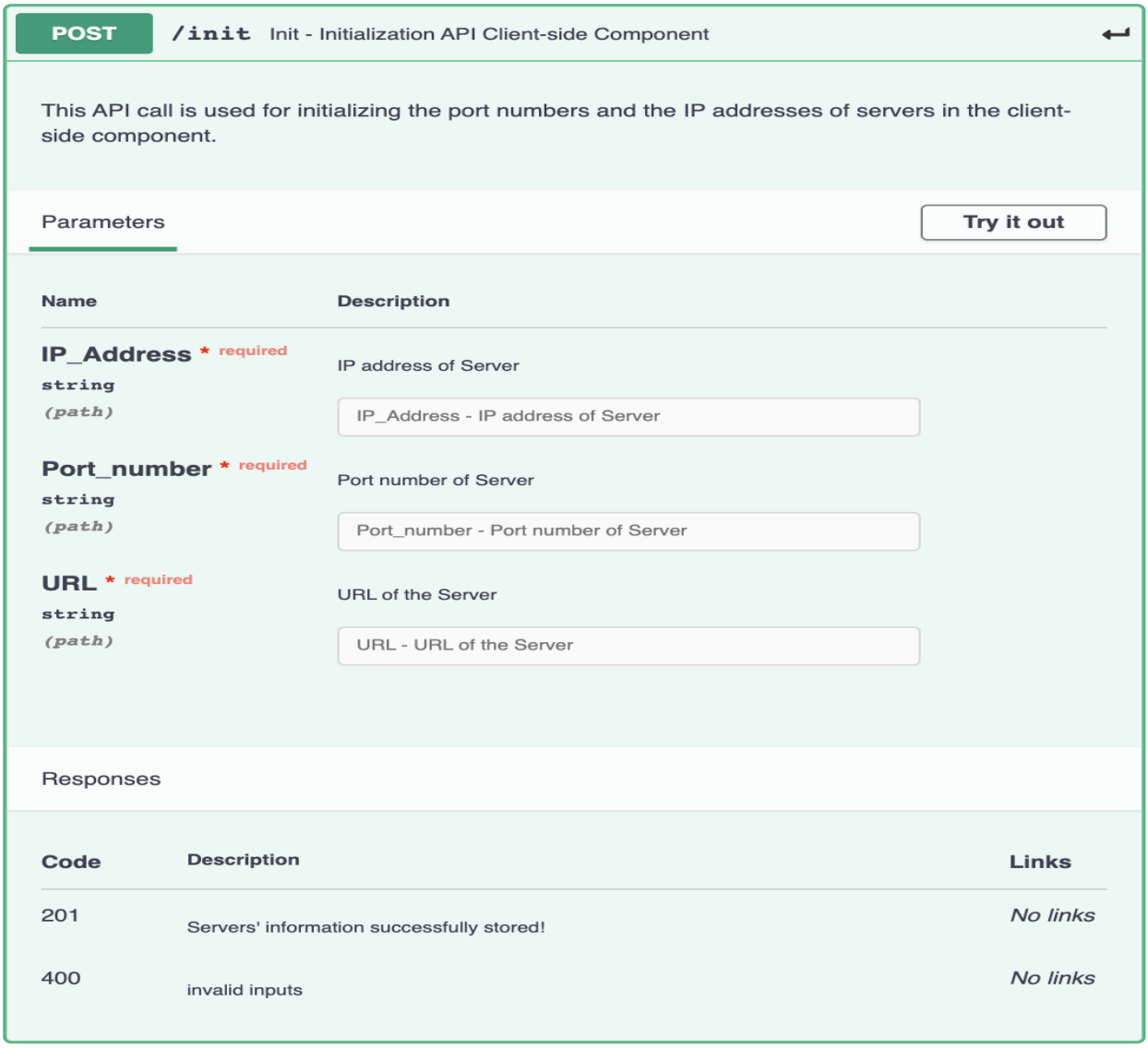


3.1.2 Privacy-preserving NN classification based on PHE

The service introduces a privacy-preserving NN classification utilizing a partially homomorphic encryption (PHE) scheme. The detailed design and behavioral analysis are presented in the deliverable D4.1 [1], and the APIs are introduced in the deliverable D4.2 [5]. In this deliverable, we present the integration evaluation of this service.

3.1.2.1 APIs

We have updated the API definition of one of the client-side APIs, namely init/. The new definition of the API is as shown in below:



3.1.2.2 Service Integration Evaluation

The Paillier encryption and socket programming are used to provide this service within two components: the server-side and client-side components implemented as Docker containers. The client-side component is deployed on the local machine, and the server-side component is deployed on the PAPAYA platform. Once the server-side component is deployed to the PAPAYA platform, the server-side component's public IP address and the port number are stored in the client-side component via the init/ API call, as illustrated in Figure 2. Later, the input file to be classified is uploaded to the client-side component using classify/ API call of the client. Then, the client-side component sends the classification request to the server-side component. Once the

two components are ready for classification, the server executes the classification and performs the linear layers until it reaches the square layer as an activation layer. Then, the server and the client execute the secure square layers via socket calls.

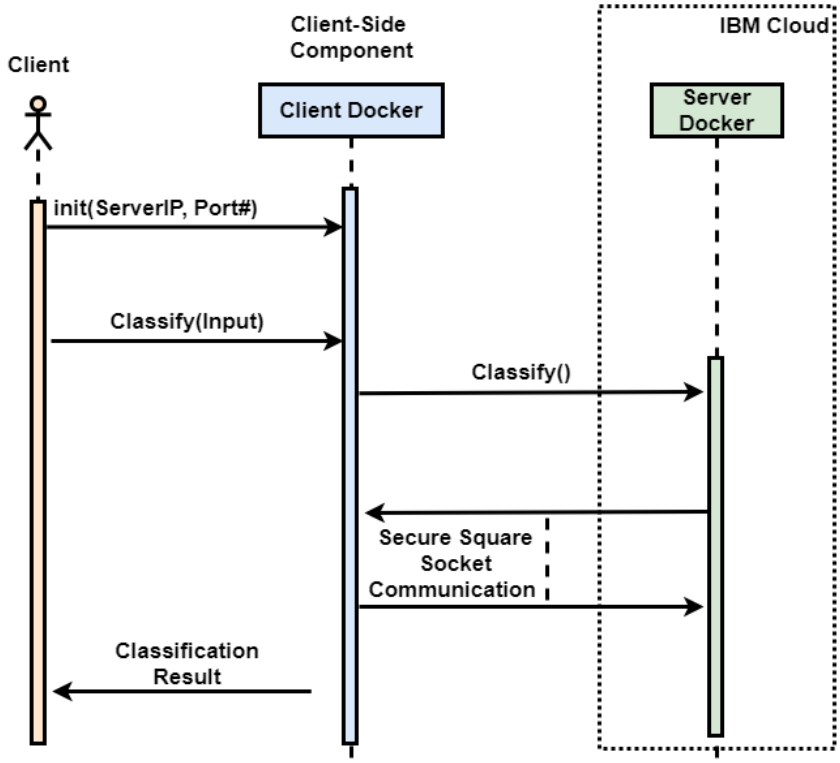


Figure 2 Service Integration Evaluation for PP NN Classification based on PHE

3.1.3 Solution based on Homomorphic Encryption

This solution uses Homomorphic Encryption (HE) to build a privacy-preserving neural network inference solution. This solution uses the CKKS scheme [10] implemented in Microsoft, open source, SEAL library [11]. The solution takes a pre-trained neural network composed of the supported layers (Dense, Convolution), and activation functions and provide an encrypted version for secure inference. The detailed design of this service is presented in D4.1 [1]. A demo of this service used in the use case UC5 is available at https://www.papaya-project.eu/dissemination/demos.

In the following subsection we describe how we integrated and evaluated the service.

3.1.3.1 Service integration evaluation

We used the threat detection use case described in D2.2 [12] to evaluate our implementation. The module that creates the homomorphic version of the model and run it is hosted on the PAPAYA platform in IBM cloud. The evaluation procedure is described in D4.2.
For the tests we developed two applications, one for the company side, and one for the client side (see Figure 3). The first application aims at allowing the company to send a neural network to the platform such that it can be hosted by the platform et evaluate using homomorphic encryption. The client application has basic cryptographic tools to encrypt the data and decrypt the result. It also managed all the connections with the platform. Both applications are packaged in docker. We run those applications locally using our own data, everything worked as expected.

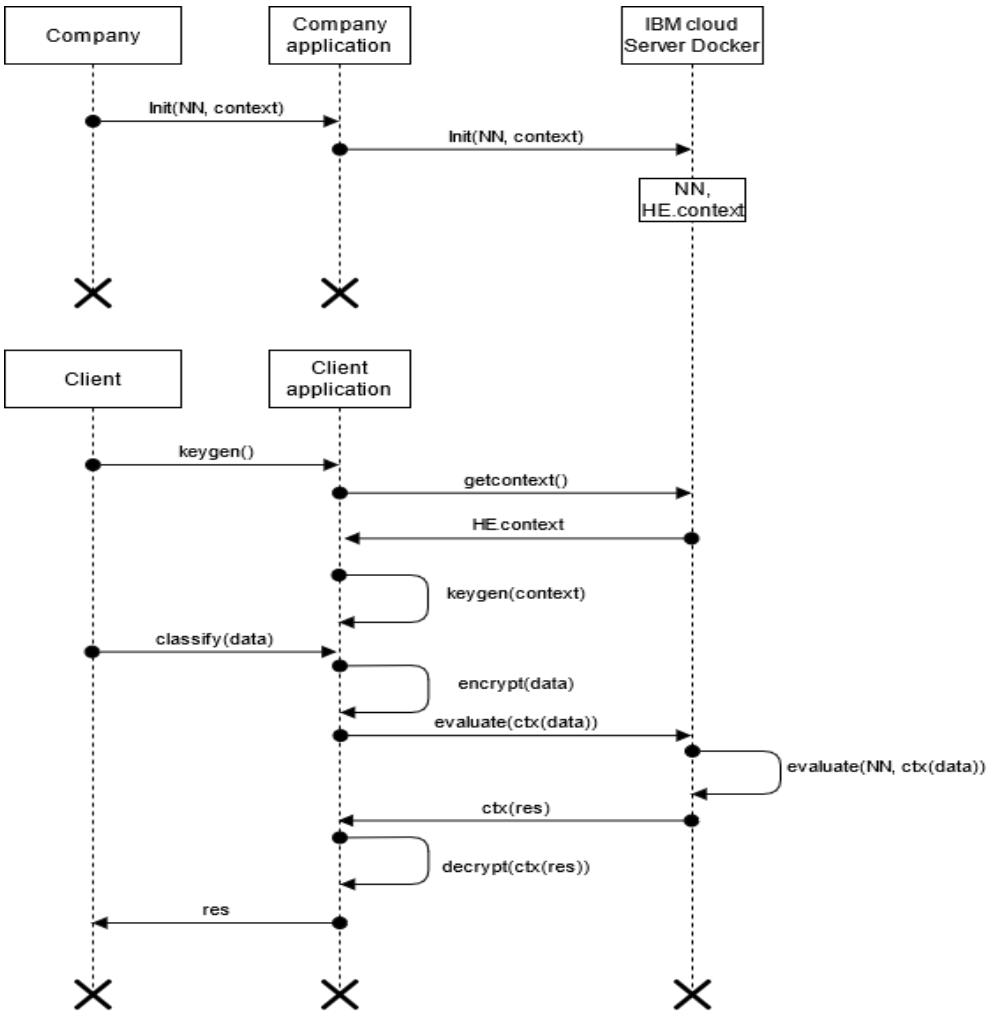


Figure 3 PP NN Classification based on PHE - testing

3.1.4 Privacy-preserving NN classification based on hybrid approach

The hybrid solution uses both, the HE and 2PC, to build a privacy-preserving NN classification framework and maximize the efficiency of classification on deep NN. The solution is *practically* generic, namely, it supports different types of DNN (i.e., MLP, CNN, and RNN) with any number of layers, any number of neurons in each layer and any activation function (from the set of supported activation functions), while the performance still acceptable (grows linearly with the DNN's depth).
We presented a detailed design in D4.1 [1], and described how we integrated and evaluated the service in D4.2 [5].

3.2 Collaborative Training of Neural Network

Privacy-preserving collaborative training of Neural Network allows multiple participants to perform ML training collaboratively, while preserving the privacy of the training data. We presented a complete specification of the service based on Shokri and Shmatikov approach [13] in D4.1 [1], and an extension based on the method presented by Abadi et. al. [14] (presented in D3.3 [6]) in D4.2 [5]. We also described how we integrated and evaluated the service and provided the service's APIs in D4.2 [5].
The service was evaluated in a course of WP5. During the service validation was raised a need for a synchronization mechanism to synchronize the training between many participants. In addition, we modified several APIs in a course of validation in WP5.
A demo involving this service is available at https://www.papaya-project.eu/dissemination/demos.
In this deliverable we present the updated behavioral analysis for the service and the updated APIs.

3.2.1 Behavioral analysis

As described in Figure 4, during the initialization call, the initiator may specify the waiting period by using the *training_join_period* variable. After the minimal number of participants joined the training, the server will start this training period **on a first train call**. During that time period all participants will be able to join and start the training. After that time period the server will not allow neither to join nor to start the training. Additionally, we extended the *get_status* call introduced in D4.1 [1]. The new version of this call provides the agent's status, which is based on the server's status. Using this API call the client may synchronize the beginning of the training.
The returned status can be one of the following:

- *WaitingToJoin* – the agent is waiting for to join the training.
- *WaitingForMinParticipants* – the agent joined the training; however, the minimal number of participants did not send a join request to the server.
- *ReadyToTrain* – minimal number of participants asked to join/start the training and the agent can start the training.
- *Training* – running the training.
- *TrainingFinished* – the training successfully finished.

- *TrainingTerminated* – something went wrong during the training process.

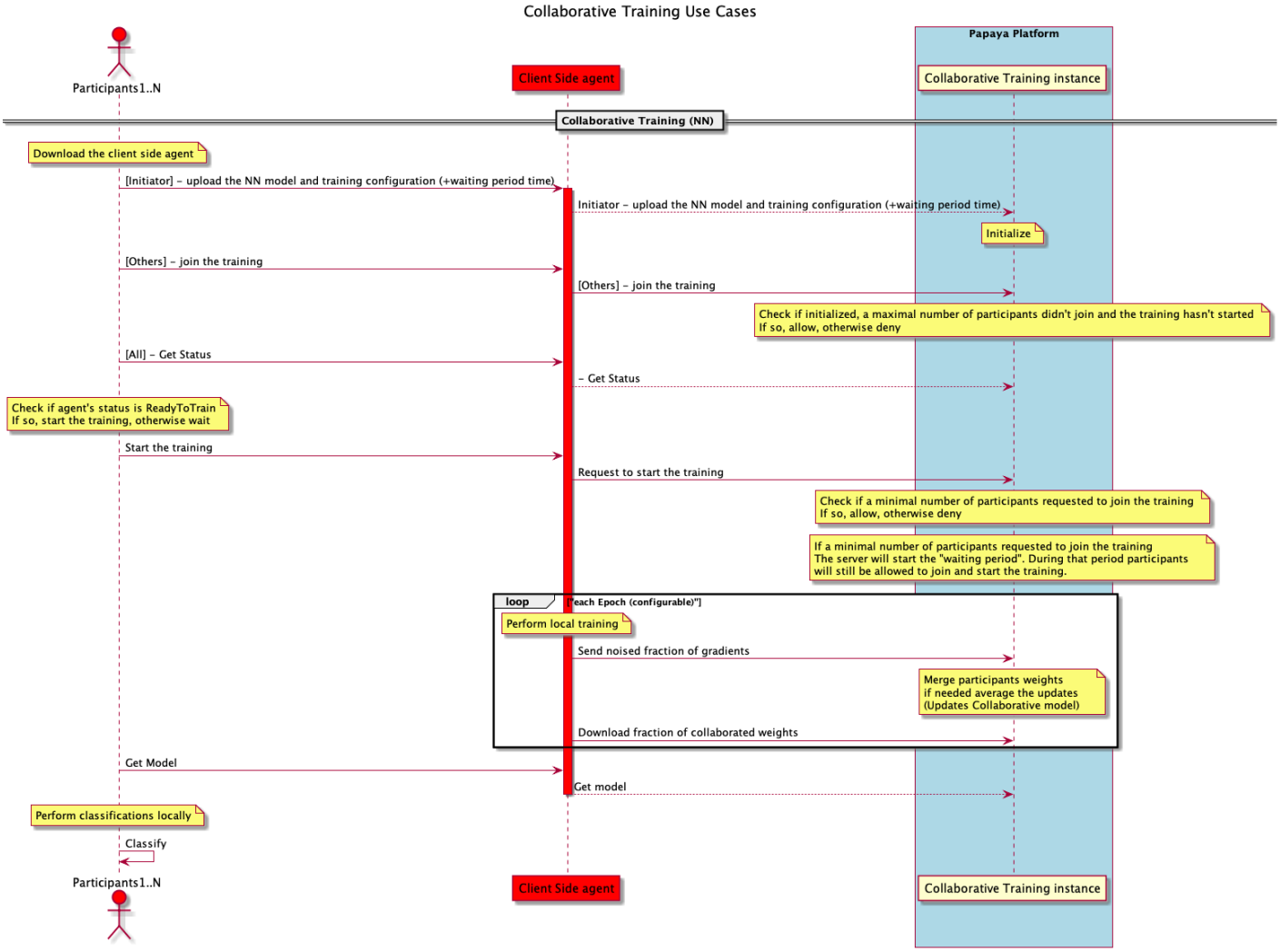


Figure 4: Collaborative training - sequence diagram

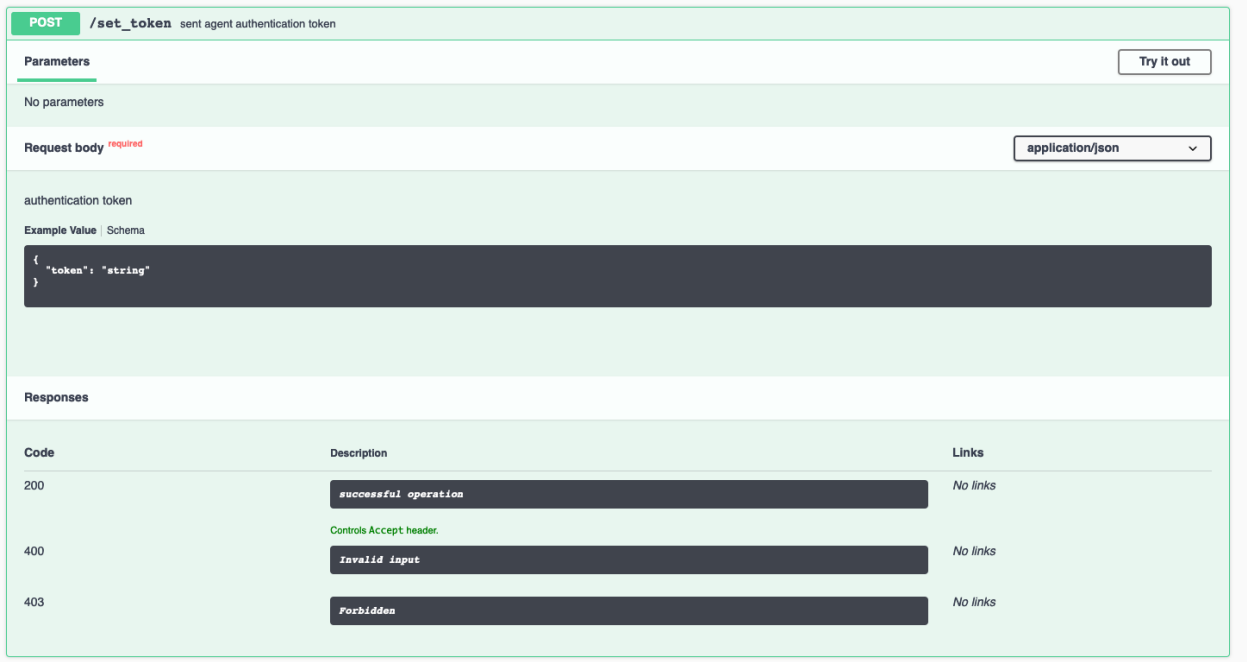
3.2.2 APIs

Based on the MCI validation (the whole specification is described in D5.4 [15]), there are minor changes in the following APIs:

- The "*init*" and "*train*" calls. The updated version of these calls expects receiving an URL for the model and dataset files, and the agent will download these files from the provided URL.
- The get_*model* call provides two options: agent allows downloading either the local latest model or the final collaborative model. Additionally, the new version of this call allows

downloading the model instead of providing the path to agent's local file system where the model was saved.

In addition, a new *set_token* API was added to the service to allow the integration with the IAM. The *set_token* API allows to set authentication token, issued by the platform client from the IAM, to the agent's request. This token will be set at a *Bearer Tokens* [3] to all requests sent to the server-side component. Following is the API specification:



3.3 Clustering

3.3.1 Privacy-preserving clustering based on 2PC

The service introduces a privacy-preserving trajectory clustering based on 2PC. The service uses the ABY library as the secure two-party computation library. This service operates in two modes, namely the client-server mode and two non-colluding servers mode. The behavioral analysis, implementation constraints, and the deployment and configuration information were already presented in the deliverable D4.2 [5]. A demo of this service is available at https://www.papaya-project.eu/dissemination/demos. In this deliverable, we present the updated APIs and the evaluation of the integration with respect to the modes of the service.

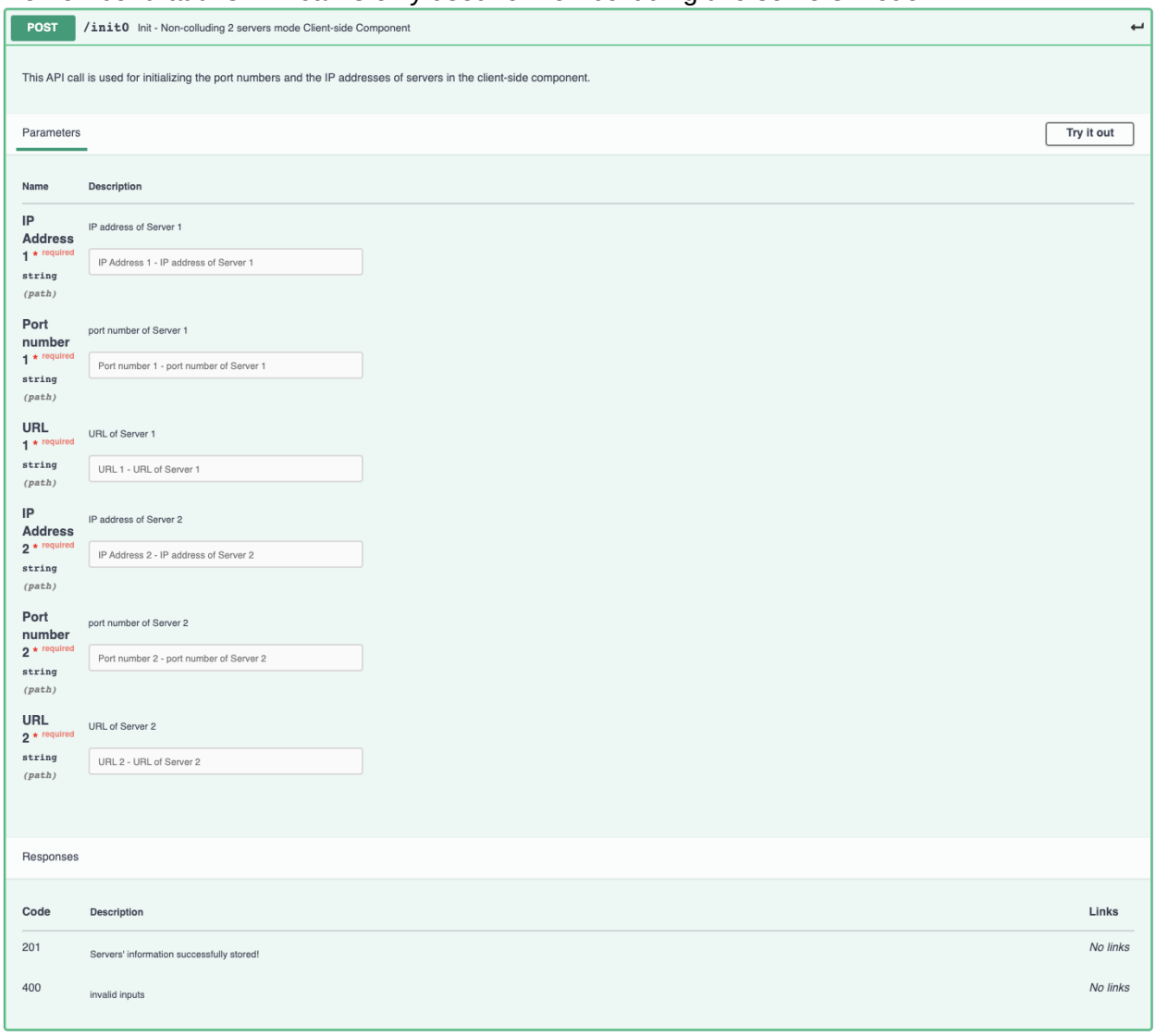
[3] https://swagger.io/docs/specification/authentication/bearer-authentication/

3.3.1.1 APIs

Client-side Component APIs: The client-side component has three different API calls, namely init0/, init1/ and start/. The first two init API calls are used for initializing the information of the servers in the client-side component for the two non-colluding servers mode and the client-server mode, respectively. The last API call, start/, is used for initiating the clustering algorithm to provide inputs to the client-side component.

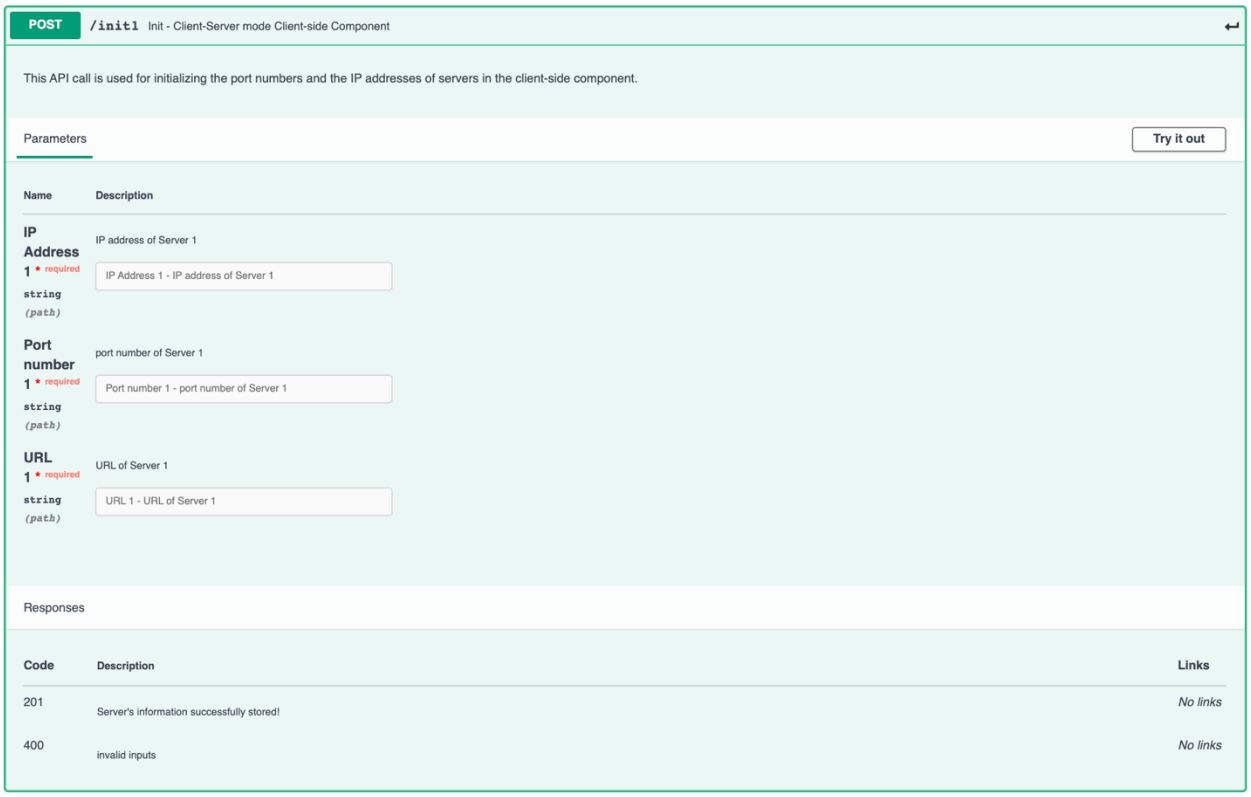
POST init0/

This API call is used for initializing the information of servers on the client-side component. Remember that this API call is only used for non-colluding two servers mode.



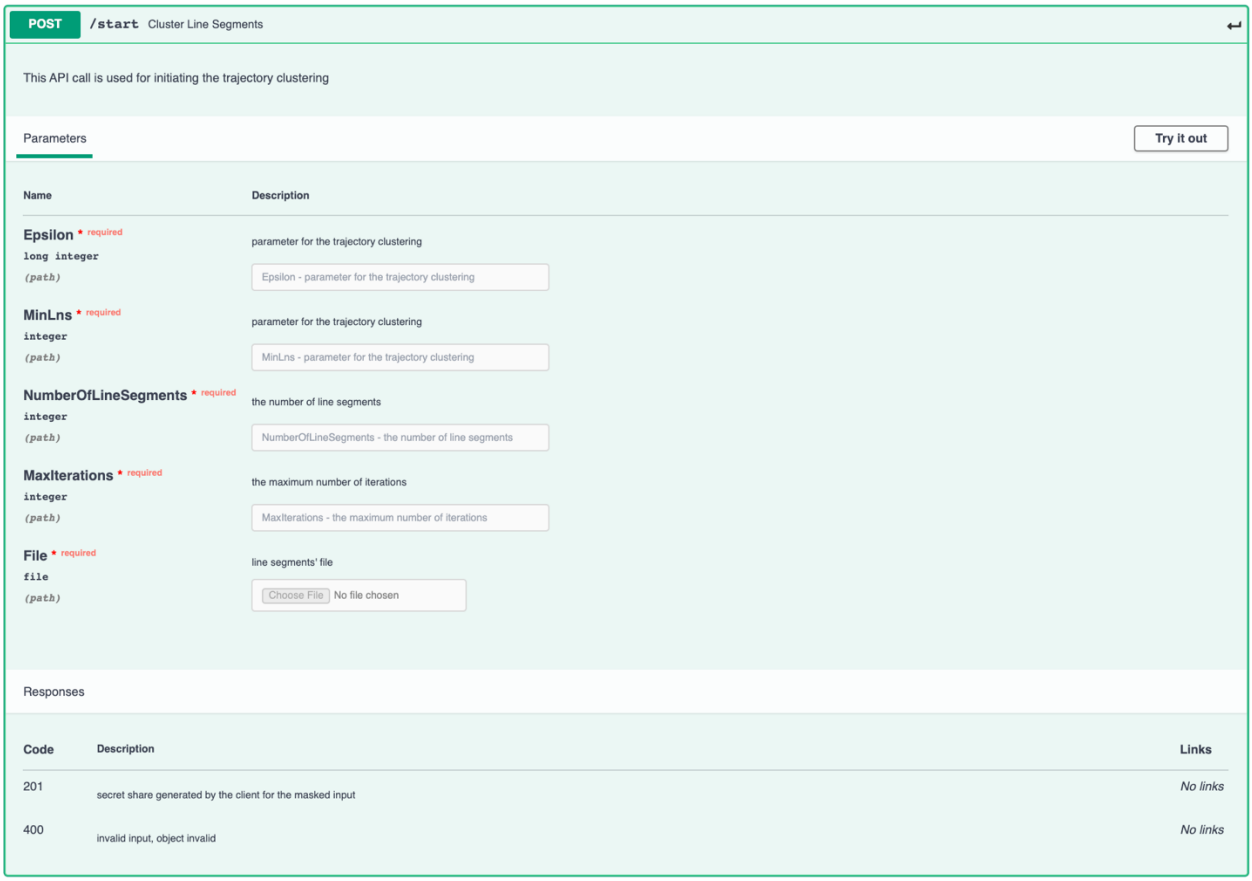
POST init1/

This API call is used for initializing the information of the server on the client-side component. Remember that this API call is only used for client-server mode.

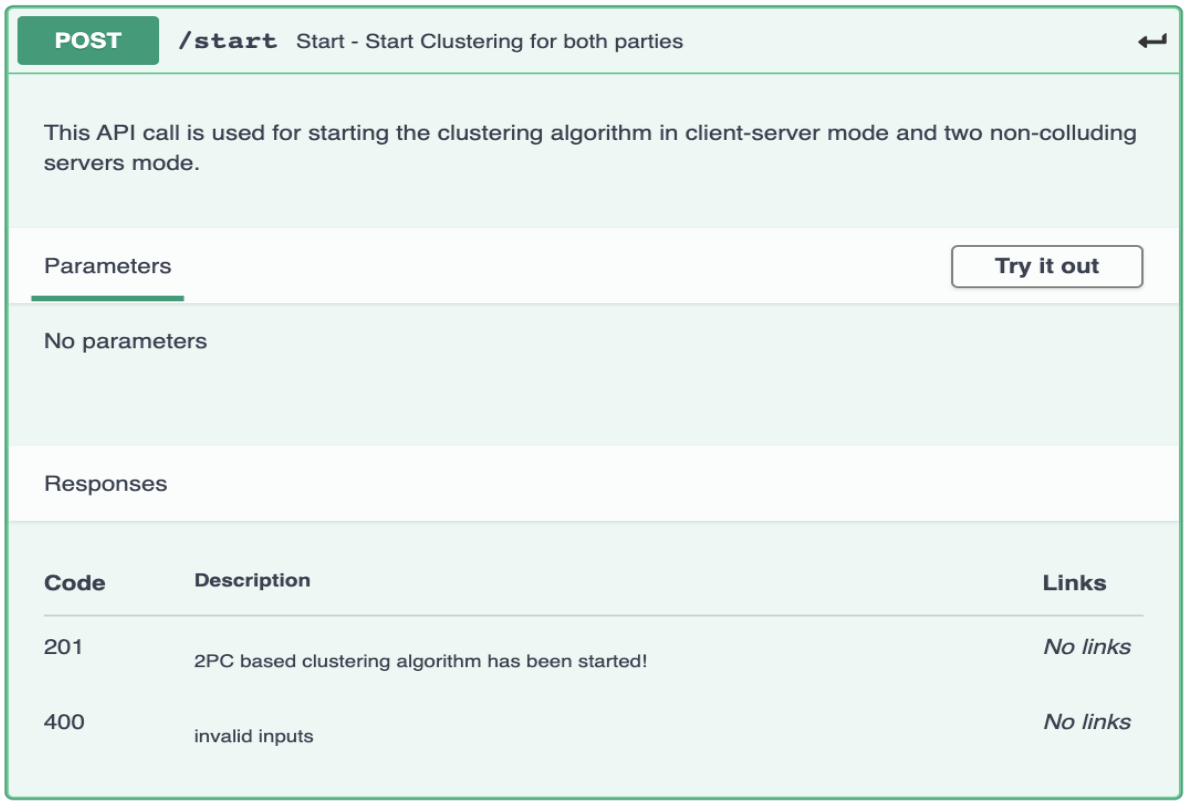


POST start/

Selection of modes for the clustering service is realized automatically based on the API call. Therefore, if init0 API is called in the initialization phase then, the clustering algorithm operates on the non-colluding two servers mode. Otherwise, the clustering algorithm operates on the client-server mode.



Server-side Component APIs: There is only one API call for the server-side component(s), namely *start/*. This API call is used for initiating the execution of the clustering algorithm on the server-side component(s). The rest of the communications for executing the clustering algorithm in client-server mode and the two non-colluding servers mode are realized by using socket communication.



3.3.1.2 Service Integration Evaluation

In this section, we present the service integration evaluation for privacy-preserving trajectory clustering based on 2PC. As defined above, the service operates on two modes: client-server and two non-colluding servers modes. Therefore, we evaluate the integration of those modules separately. As illustrated in the Figure 5, client first sends the server information to the client-side component using *init1/* API call. Later, client calls cluster/ API to execute the clustering together with the necessary parameters for the algorithm (epsilon, MinLns, number of lines to be clustered, number of iterations and the data set file). All the communications between client-side and server-side components are realized by the ABY sockets.

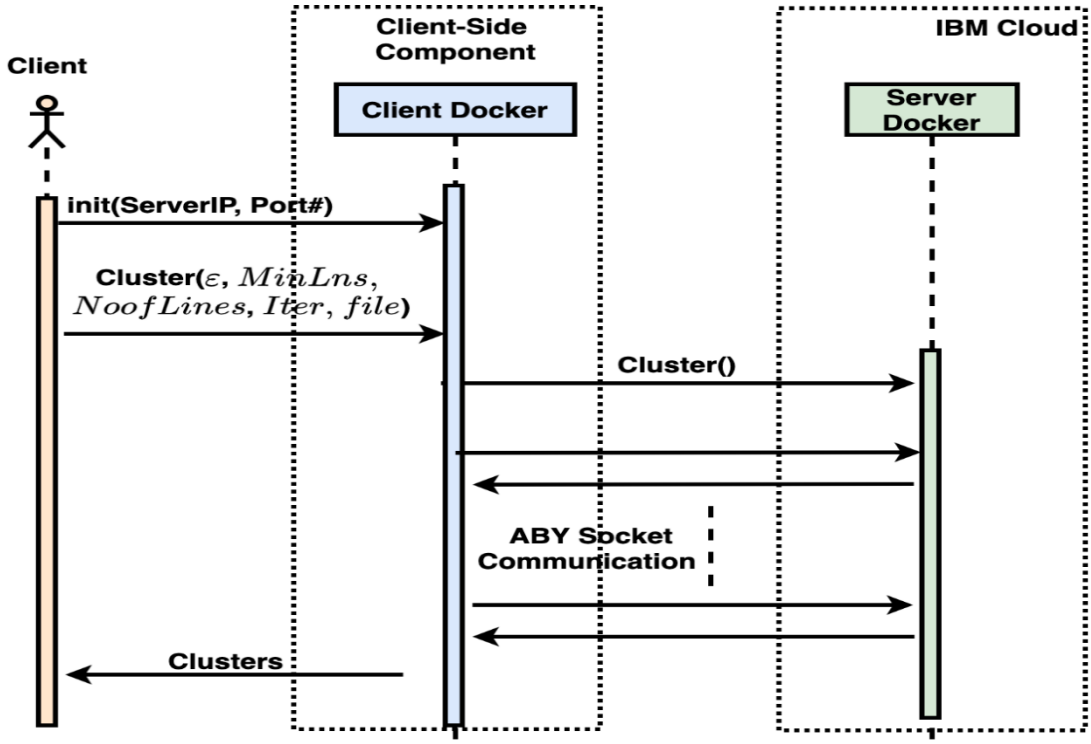


Figure 5 PP Trajectory Clustering Integration Evaluation for Client-Server Mode

Although 2PC is one of the most popular cryptographic techniques to develop a privacy-preserving machine learning solutions, such approaches suffer from the communication costs. Therefore, we propose an outsourcing scenario for the privacy-preserving trajectory clustering solution, where the execution of the clustering algorithm is realized by the non-colluding two servers. The evaluation of the integration is as shown in Figure 6. First, client invokes the init0/ API call to initialize the information of the servers on the client side and then calls the cluster/ API together with the necessary parameters for the clustering algorithm. Later, client generates shares for the clustering execution with the Server 1 using the ABY library. Client-side component sends its own share to the Server 2 as a file using TCP socket communication. Server 2 re-construct shares using ABY library. Finally, Server 1 and Server 2 execute clustering algorithm and returns the results to the client-side component separately.

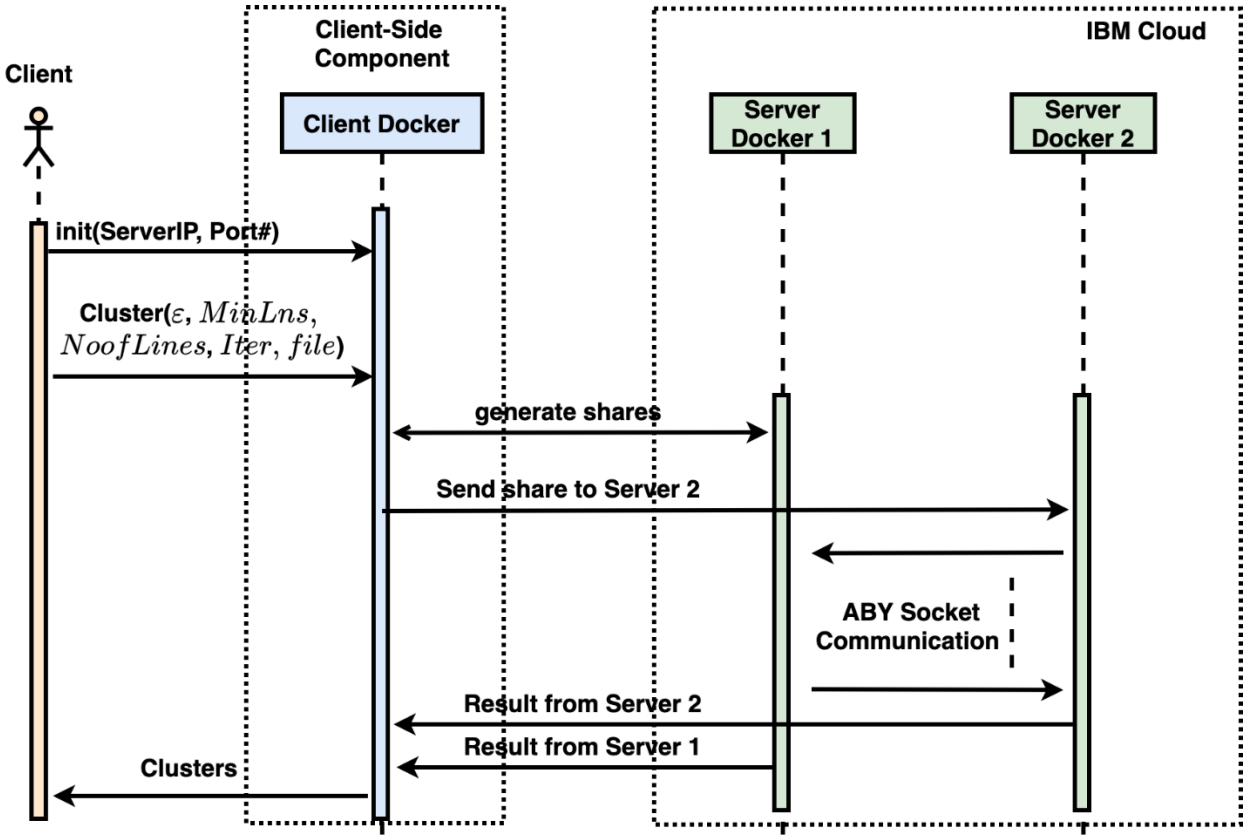


Figure 6 PP Trajectory Clustering Integration Evaluation for Non-colluding Two Servers Mode

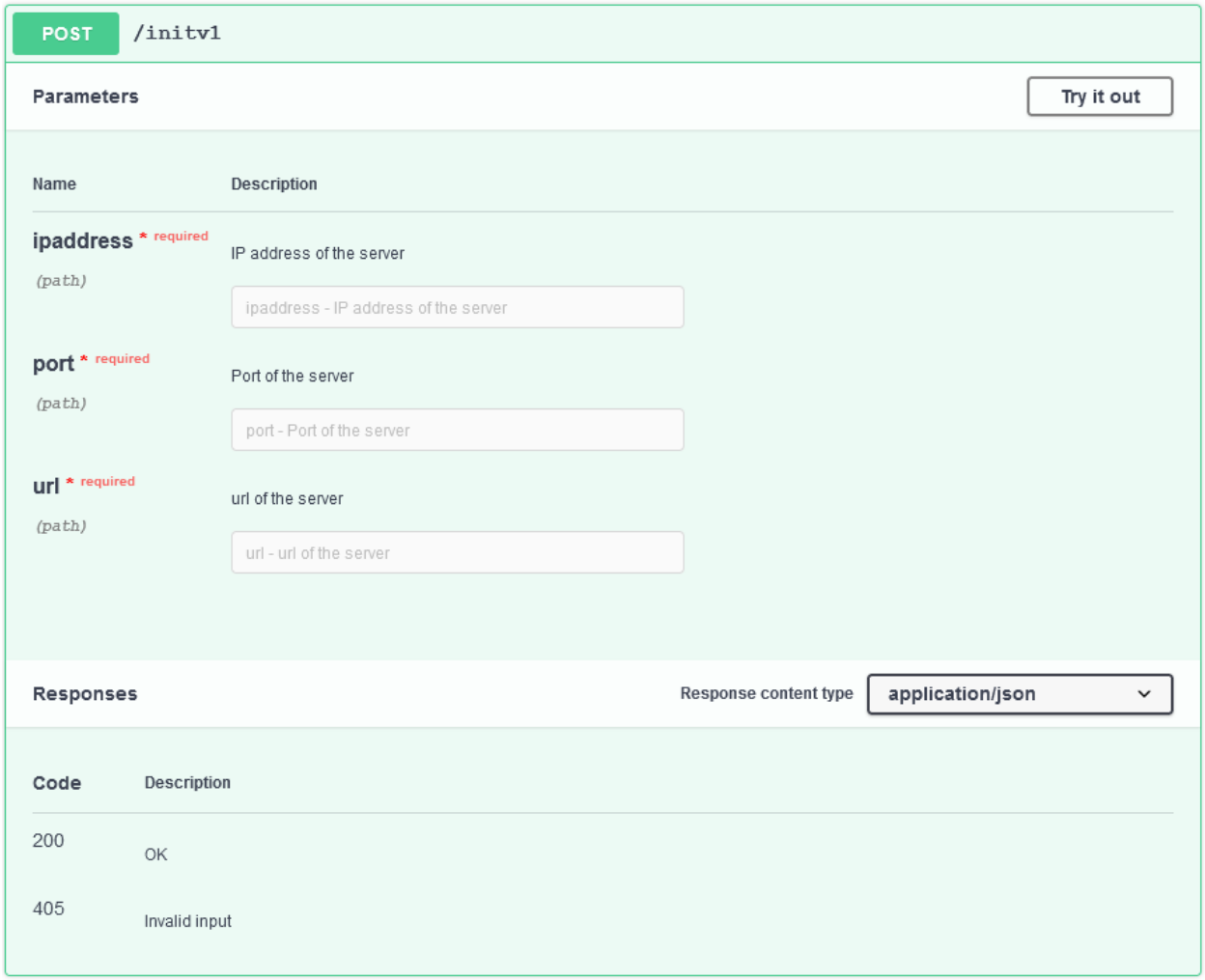
3.3.2 Privacy-preserving clustering based on MinHash

For this service we deployed a new version based on the work done for Privacy-preserving clustering based on 2PC described in the previous section. This new version reuses most of the code and architecture of the 2PC based clustering. For the deployment, configuration, and description of the architecture we refer to D4.1 [1] and D4.2 [5]. As for the 2PC clustering, this solution provides two implementations, a client-server version and a two non-colluding servers version. In the following subsections we provide the service API and describe how we evaluated the service integration. A demo is available at https://www.papaya-project.eu/dissemination/demos.

3.3.2.1 APIs

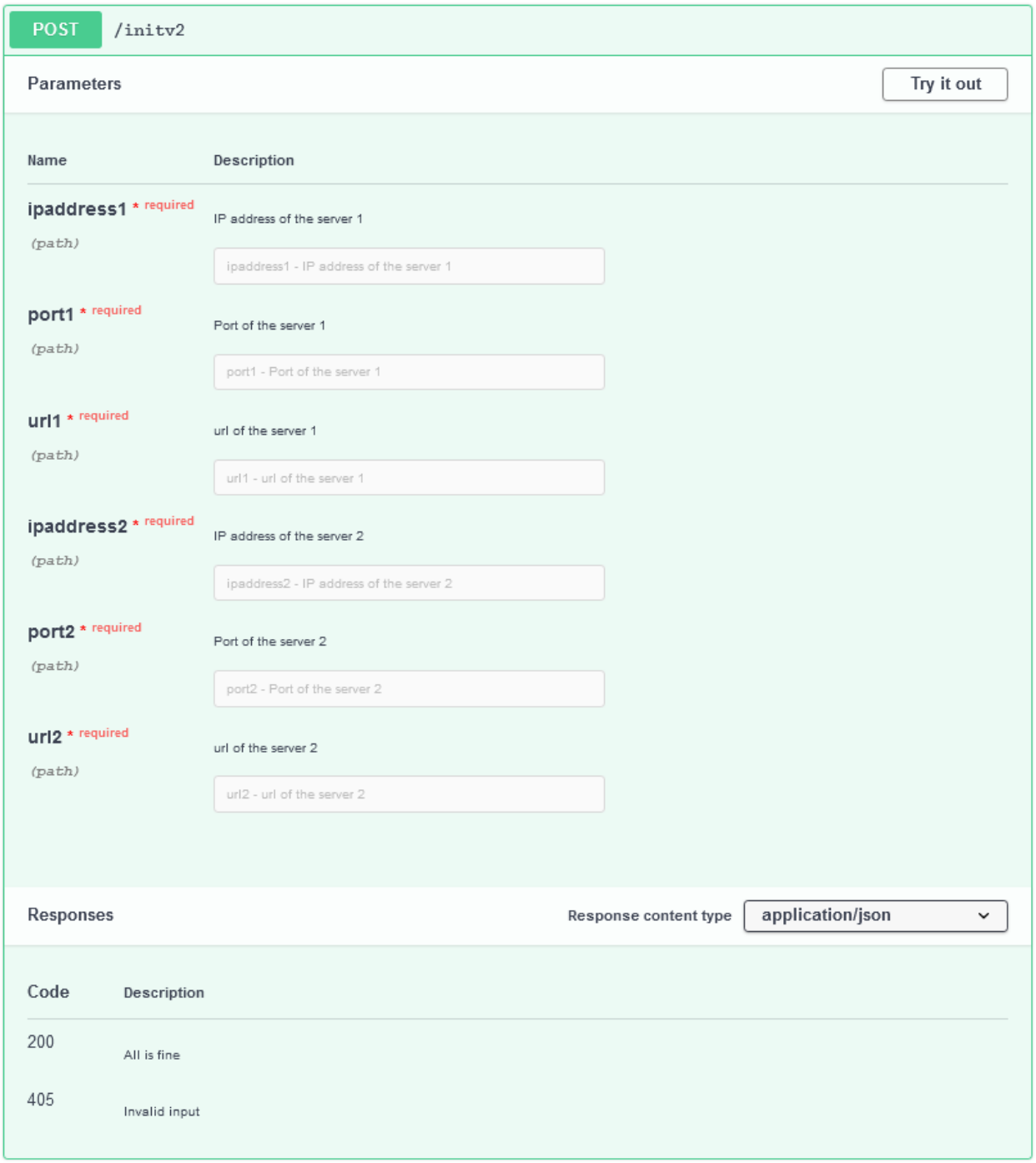
Client-side component APIs:

POST /initv1:
This call is used to initialize the client by providing the parameters to connect to the server. This call will use the client-server version.

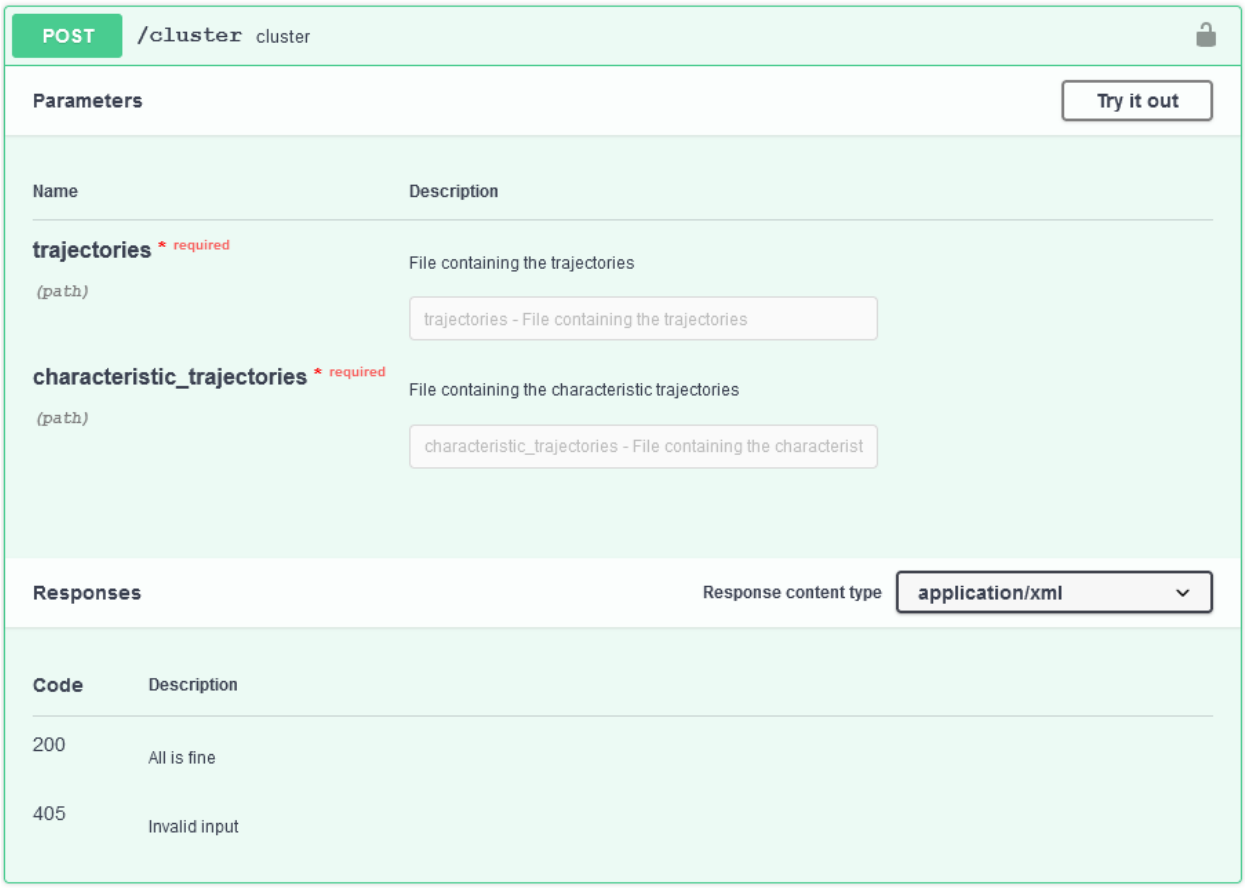

POST
/initv1
Parameters
Try it out
Name
Description
ipaddress * required
(path)
IP address of the server
ipaddress - IP address of the server
port * required
(path)
Port of the server
port - Port of the server
url * required
(path)
url of the server
url - url of the server
Responses
Response content type
application/json
Code
Description
200
OK
405
Invalid input

POST /initv2:

This call is used to provide the information related to the two servers used for the two non-colluding servers versions.



POST /cluster:
This call is used to upload the data to the client application.



3.3.2.2 Service integration evaluation

We first evaluated the client-server version, (see Figure 7). The server component is hosted on IBM cloud, and the client-side component runs on local computer. We tested the service with datasets of different size. We first initialized the client component with the IP address and the port of the server component, and then we launched the clustering by calling the Cluster API call. We got the correct result for each test, everything worked as expected.

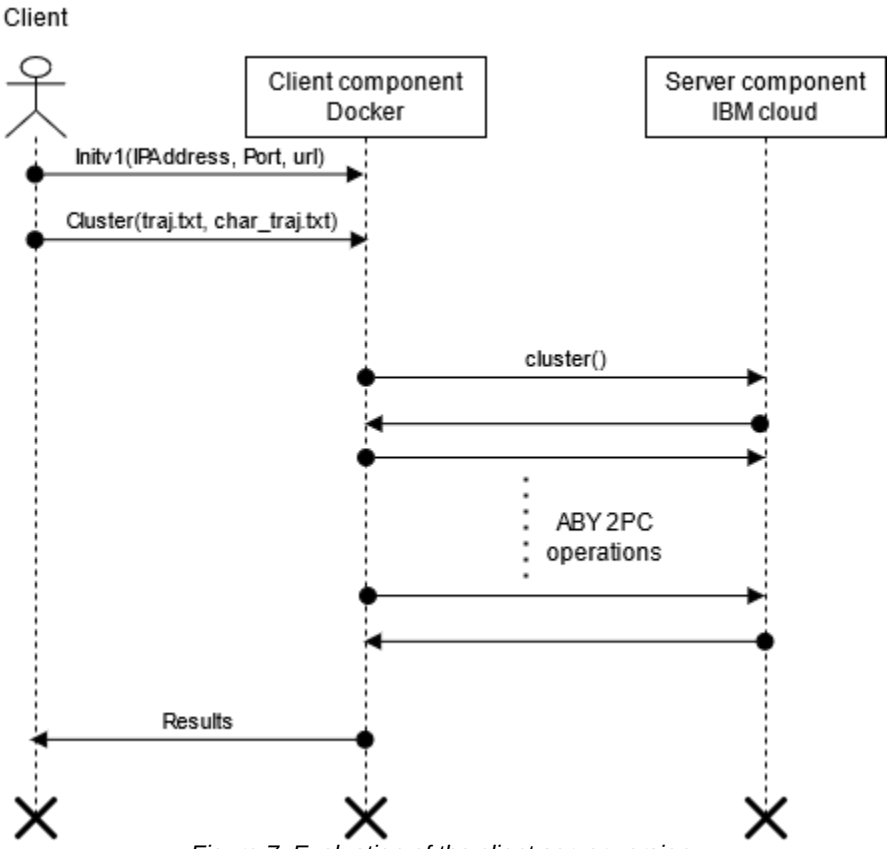


Figure 7: Evaluation of the client server version

Next, we evaluated the two servers version, (see Figure 8). The two servers' components are hosted on IBM cloud, and the client-side component runs on local computer. We tested the service with datasets of different size. We first initialize the client component with the IP address and the port of the two servers' component, and then we launched the clustering by calling the Cluster API call of the client. We got the correct result for each test, and everything worked as expected.

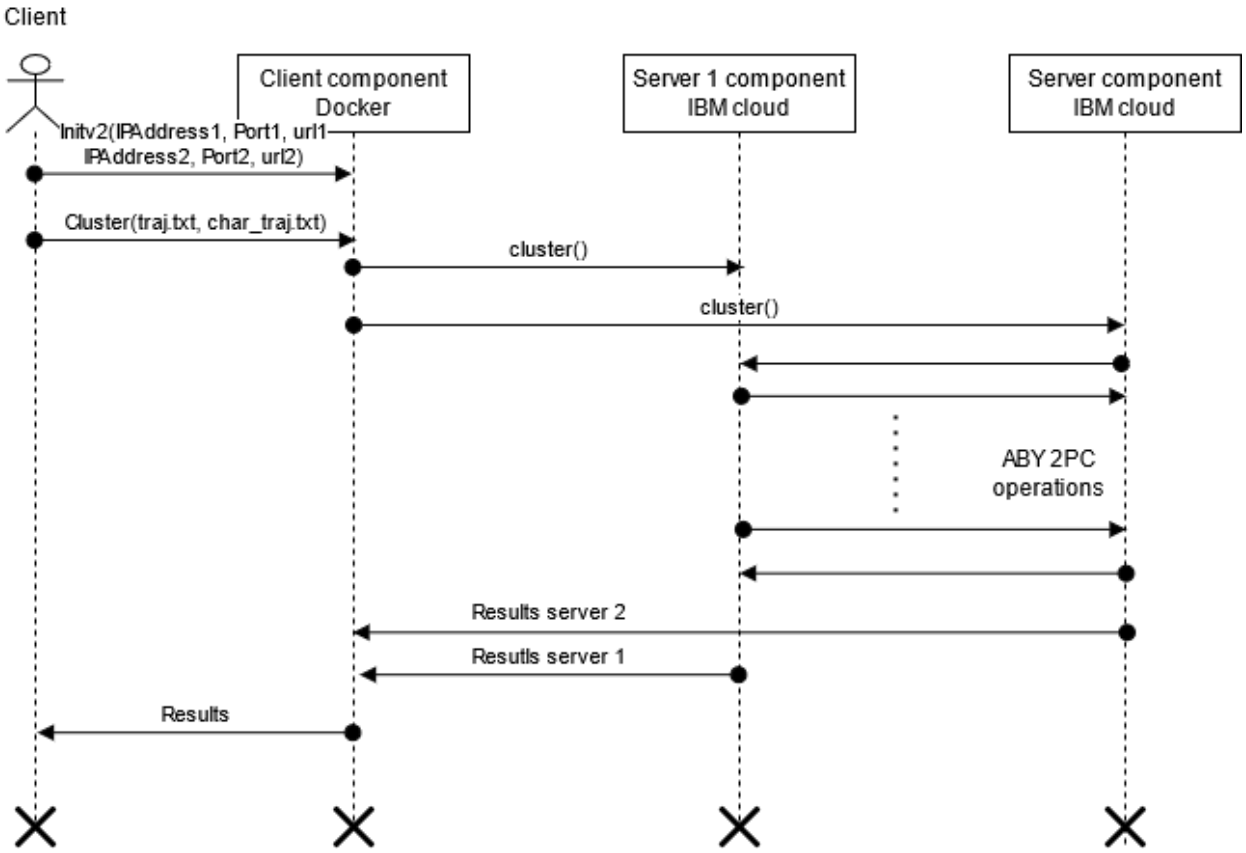


Figure 8: Evaluation of the two non-colluding servers version

3.4 Basic Statistics

3.4.1 Privacy-preserving statistics based on Functional Encryption

This solution uses functional encryption to allow a requestor to compute statistics on users' mobile usage. Functional encryption ensures data confidentiality and also that only the computation agreed by the user can be done. A detailed description of the design is given in D4.1 [1].
The solution is tightly designed to the corresponding use case, UC4 described in D2.1 [12], and cannot be generalized to other ones. Furthermore, the implementation is specialized for Orange operational infrastructure and does not works without it. So, we decided not to deploy it on the PAPAYA platform. However, a demo involving this service is available at https://www.papaya-project.eu/dissemination/demos.

3.4.2 Privacy-preserving Counting using Bloom Filters

This solution uses Bloom Filters to count elements of a set without the need to store the elements ids. Bloom Filters allows to get the cardinality of the set as well as cardinality of union and / or intersections. While the construction of the bloom filters is down in plaintext, this solution uses homomorphic encryption to compute the union, intersection, and cardinality. We used this service in the use case UC4, described in D2.1 [12], to extract statistical indicators on visitors of tourist sites during the 2024 Olympics games.
A detailed description of the design for this solution is given in D4.1 [1] and D4.2 [5].
The use case that uses this PET, relies on a sensitive service in Orange. We designed our implementation by considering specific requirements of that Orange service. The implementation is too specific to Orange infrastructure to be deployed on the PAPAYA platform. Nonetheless, a demo involving this service is available at https://www.papaya-project.eu/dissemination/demos.

4 Platform Security and Transparency

4.1 IAM

The Identity Access Manager (IAM) carries out the Authentication and Authorization services to protect the accesses to the PAPAYA platform. To do so, it is necessary to implement and deploy different components in addition to the IAM server.

The main IAM components and their relationships with other components of the PAPAYA platform is described in detail within deliverable D4.1 [1] and D4.2 [5]. As shown in the Figure 9, all access to the PAPAYA framework will be done towards the Security Gatekeeper components. Each Security Gatekeeper component will contact the IAM to verify if every access is authenticated and authorized and, if the access is granted, it will be redirected to the Computation Component.

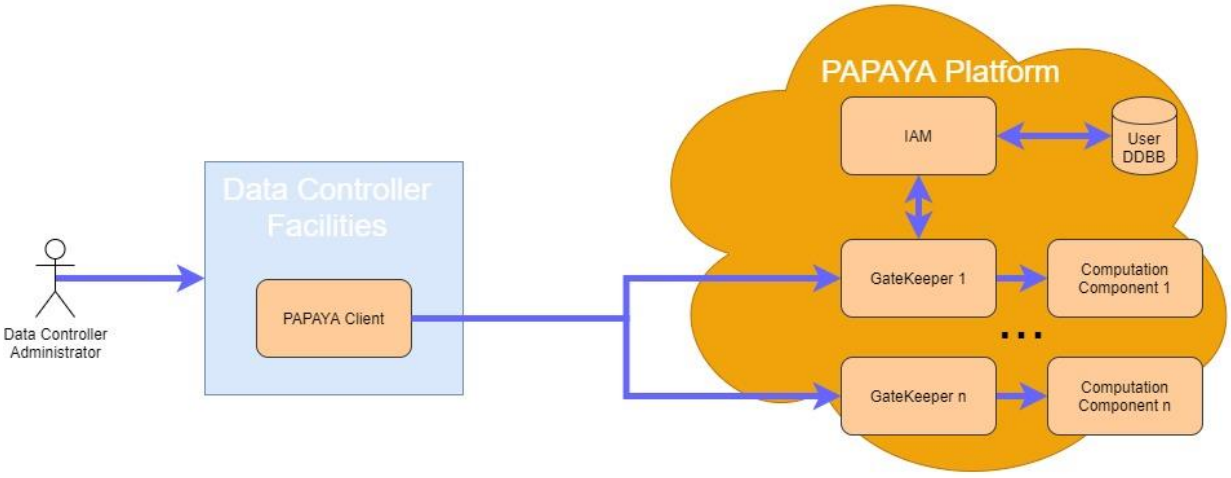


Figure 9: IAM and Security Gatekeeper in PAPAYA platform

The deployment of IAM encompasses the following steps:

1. **Set up Keycloak:** this includes:
   - secret creation that will be used later on ingress
   - application to the deployment and service
   - define the ingress in order it will be accessible from outside and to share the certificates of the initial application. If everything is ok, the IAM is accessible.
2. **Set up Gatekeeper:** this encompasses the steps:
   - Generation of gatekeeper config map
   - Apply the Deployment + Service for the gatekeeper + example service
   - Finally, to access to the service from outside, apply the corresponding ingress
3. **Testing the access:** in order to test that the service has been deployed correctly using the Authentication and Authorization services provided by the Gatekeeper filter and the help of the IAM it its necessary that the papaya client will obtain the corresponding token, to be included within the request call headers to the computation services, otherwise the client won't be able to access to the service and will obtain and 403 response from it.

A demo involving the IAM component is available at https://www.papaya-project.eu/dissemination/demos.

4.2 Auditing

Towards being able to hold stakeholders accountable for their use of the PAPAYA platform and services, data processing is logged both as part of the platform and locally at agents.

4.2.1 Platform auditing

4.2.1.1 Overview

We deployed the platform auditing in two stages:

1. In the first stage we deployed the commonly used Elastic stack in the platform's K8s cluster.
2. In the second stage we improved the security of the setup by making the resulting logs (i) tamper proof and (ii) verifiable—in terms of authenticity and time—by third parties.

As part of the Elastic stack, we use the Filebeat[4], Elasticsearch[5], and Kibana[6] components. Containers that run the analytics services (Section 3) log their operations to standard output, and Filebeat is configured in K8s (as part of pods) to collect all of the output and send it to the Elasticsearch instances we run as part of the platform for storage.

[4] https://www.elastic.co/beats/filebeat
[5] https://www.elastic.co/elasticsearch/
[6] https://www.elastic.co/kibana

From the platform dashboard, only administrators are able to access an instance of Kibana to view all operational logs of the PAPAYA platform stored in Elasticsearch (since the PAPAYA platform is running on IBM account, this functionality is terminated due to IBM's internal security requirements).
For the platform users we allow to retrieve logs and observer them for each application that has been deployed by the user. The logs are presented in a dedicated view in the platform dashboard, such that we limit each user to observe logs only of his deployed services.

In the second stage we deployed one additional component of the Elastic stack: Logstash[7]. Logstash is a log pre-processor that receives logs from Filebeat, performs some processing, and then forwards them to Elasticsearch. We create a custom Logstash filter based on an existing secure logging scheme built for the Elastic stack [16], that provides basic authenticity and time verification, but no completeness of the logs (not implemented due to time constraints, but its design is documented in [16]). The filter efficiently signs the logs such that they can be shared and verified by third parties. Figure 10 summarizes the flow of logging data for platform auditing and the involved components.

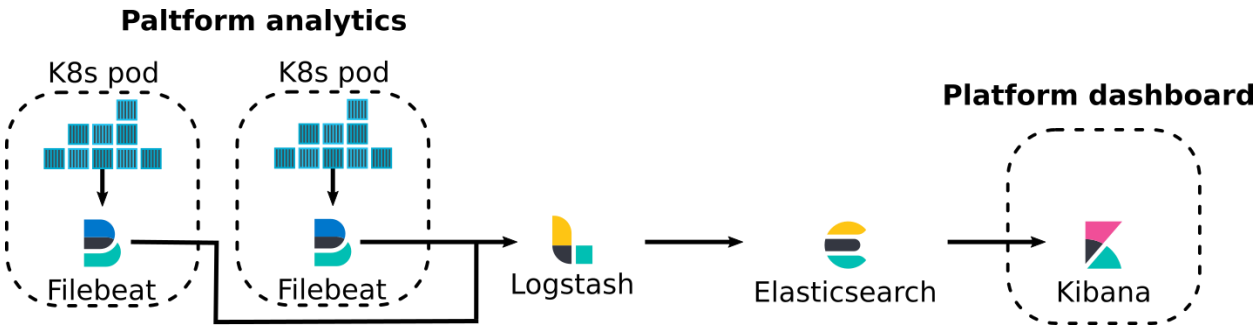


Figure 10: Platform auditing components and the flow of logs.

4.2.1.2 Integration evaluation

We validated the integration of the platform auditing as follows:

1. Created a new user for the platform.
2. Created number of instances of analytic services.
3. For several services, we ran their agents and perform some analytics.
4. As the newly created user, accessed the platform dashboard. Verified that all expected logs are available.
5. We performed the same verification as in 4, for number of users, verifying that it cannot see the newly generated logs.

For the platform administrator, we ensured that all operational logs can be observed by using Kibana UI, that is integrated in the platform dashboard. Finally, as administrator, we verified the existence of signatures on logs.

[7] https://www.elastic.co/logstash

4.2.2 Agent auditing

Agent Auditing is considered in the development and operational phases of a platform agent, as defined in D4.2 [5]:

- In the development phase, the Agent Dashboard can be used by the user of the platform agent to quickly view the generated logs generated by the agent. This is demonstrated as part of the Agent Dashboard (see Section 5.2).
- In the operational phase, the user of the platform agent is assumed to already be operating some logging infrastructure for its existing systems (in which the agent is integrated), as described in D4.2 [5]. Creating project-specific tooling here is therefore not wise. We note that the infrastructure and tooling we use for platform auditing, as just described in Section 4.2.1, could be re-used by a platform user.

Due to time constraints we opted not to create a small proof-of-concept of the operational phase based on our platform tooling, as initially planned in D4.2 [5]. The difference between the agent auditing and platform auditing is just the use of an appropriate method to transport logs (from Docker or whatever container runtime that is used) from containers into Logstash[8].

4.3 Key Manager

The Key Manager (KM) component is devoted to providing a collateral service of the use of a privacy-preserving platform, carrying out the management of the cryptographic material during the whole lifecycle of the PAPAYA project in the cases where it will be required. D4.1 [1] described the architecture, a detailed design of the Key manager, its main components, and the integration of them within PAPAYA platform. D4.2 [5] provided the deployment details. There have been no changes to this component.

[8] See https://logz.io/blog/docker-logging/ for examples of how to use Filebeat or a logging driver. Accessed 2020-03-29.

5 PAPAYA Dashboards

There are two dashboards in PAPAYA, namely the Platform Dashboard and the Agent Dashboard. The platform and agent dashboards are, as the name suggests, tied to the respective architectural components and their respective back-ends provide for the dashboards. The dashboards are accessed through web views in a web browser by their respective target users.

5.1 Platform Dashboard

The Platform Dashboard is implemented as a Web application hosted in a container that runs on the PAPAYA's K8s cluster. We presented a detailed design in D4.1 [1], and described how we integrated and evaluated the Platform Dashboard in D4.2 [5].
In this deliverable we provide specification of several new APIs (with respect to D4.1 [1] and D4.2 [5]) which expose functionality that have been implemented based on the platform requirements, reviews feedback and analytics constraints. We also describe how we integrated and evaluated the new functionality. A demo involving the Platform Dashboard is available at https://www.papaya-project.eu/dissemination/demos.

5.1.1 APIs

- On creation of a new user, password re-entering was added.
- Added an option to deploy service analytics integrated with IAM.
    - This functionality is supported only for services with HTTP communication channel
- Added operational log view based on described in Section 4.2.1

5.1.2 Integration evaluation

We evaluated an observation of operational logs as described in Section 4.2.1.2.
The IAM functionality we evaluated as following:

1. We developed an analytics service without IAM integration and checked that it is reachable by the agent without any authentication.
2. We deployed an analytics service with IAM and checked that It's unreachable by agent without providing authentication token and reachable by agent which provides authentication token via request header.

5.2 Agent Dashboard

The Agent Dashboard has been implemented as planned in D4.2 [5], targeting the development phase of a developer integrating a platform agent into an existing system. As a command line tool, the Agent Dashboard provides two options:

1. Show the logs of the agent in the browser of the user, and
2. Show the agent configuration in the terminal (JSON structured and highlighted) and open an explanation of the agent in the browser of the user. The explanation is a port of the explanation tools, as part of the Data Subject Toolbox, from mobile to browser interfaces.

More details on and a demo of the Agent Dashboard is available in D5.4 [15] and at https://git.cs.kau.se/papaya/agentdashboard.

6 Data Subject Toolbox

Data protection by design is a key consideration of any Data Controller and data processor since the GDPR came into effect. The use of complex cryptographic systems—such as privacy-preserving data analytics—pose a challenge for Data Controllers and processors when dealing with data subjects. In particular, when it comes to the rights to data subjects to be informed about how their personal data is processed and to remain in control of this processing. Towards addressing these challenges, the PAPAYA framework includes a Data Subject Toolbox with a number of small, independent tools that can be used by Data Controllers and data processors in their interaction with data subjects.

6.1 Explaining Privacy-preserving Analytics

This category of tools consists of a number of smaller, largely independent tools that is intended to be combined and mixed based on the needs of the user of the PAPAYA platform and the selected analytic(s) in use. Broadly, the tools can first be categorized into tools for sharing results from risk assessments (so called risk management artefacts) or for explaining and exemplifying how privacy-preserving data analytics from PAPAYA works.
The tools for conveying risk share particular results from a performed PIA using a modified version of the CNIL PIA tool[9], where users are informed briefly about the different kinds of risks to their personal data in the system and how the controls in place (such as the use of PAPAYA's privacy-preserving data analytics) changes those risks (reducing or increasing them).
The tools for for explaining and exemplifying how privacy-preserving data analytics from PAPAYA works target the different data analytics developed in the project (described in Section 3). We focused in particular on tools for explaining with multiple layers of details privacy-preserving neural networks for classification and collaborative training.
Our work progressed as planned in D4.2 [5]. Further details of the results can be found in D5.4 [15] on the general results and in D3.4 [17] for further details on the user interface design development. D5.2 [18] and D5.3 will also demonstrate use-case specific integration of the tools.
The data subject tools for explaining privacy-preserving analytics can be accessed via the following repositories:

- Data subject tool for explaining 2PC: https://git.cs.kau.se/papaya/gatsby-papaya-encryption-data-analysis
  A clickable demo of the tool is available at: https://hex.cse.kau.se/~tobivehk/gatsby-papaya-encryption-data-analysis/
- Data subject tool for explaining MPC: https://git.cs.kau.se/papaya/react-native-papaya-mpc
- Data subject tool for explain differential privacy for collaborative learning: https://git.cs.kau.se/papaya/react-native-papaya-differential-privacy

[9] https://github.com/LINCnil/pia/ , accessed 2021-03-30.

A clickable demo of the tool is available at: https://hex.cse.kau.se/~tobivehk/react-native-papaya-differential-privacy/

- Data subject tool for explaining Functional Encryption: https://git.cs.kau.se/papaya/react-native-papaya-functional-encryption
  A recorded demo of the tool is available at: https://kauplay.kau.se/media/t/0_nyrj5pcf
  In addition, a video for explaining Functional Encryption was produced that can be used as an alternative format of presentation of the same content. It is available at: https://kauplay.kau.se/media/t/0_z3c7ib6d

For producing the risk management artefacts, a PIA needs to be conducted with the extended PIA tool, which is available as AppImage on Linux, and installation on Windows at https://hex.cse.kau.se/~jonamagn/.
Source code for the modified PIA tool is also available at https://github.com/papaya-h2020/pia.
Finally, the UI components can be found at https://git.cs.kau.se/papaya/react-native-papaya-riskmatrix.
See D5.4 [15] and the respective repositories for documentation.

6.2 Data Disclosure Visualization Tool

The goal of the tool is to provide transparency to data subjects concerning what actors process or have access to what types of their personal data within the context of some system. The tool is designed to be integrated into an existing mobile app and consists of three views with complementary information to data subjects: the trace view, the actor view, and the data view.

- The trace view provides an interactive view that shows which actors in the system know what personal data about the data subject.
- The actor view describes the actors in the system.
- The data view describes the personal data in the system.

Further details will be provided in D5.3 and D5.4 [15]. The tool is used for the demonstrator in UC2.
Code is available at https://git.cs.kau.se/papaya/react-native-papaya-trace-viewer.

A clickable demo is available at: https://hex.cse.kau.se/~tobivehk/react-native-papaya-trace-viewer/

A recoded demo video is available at: https://kauplay.kau.se/media/t/0_252nchhk

6.3 Annotated Log View Tool

The tool provides an overview to data subjects of data *processing* performed by a data processor or Data Controller after personal data has been disclosed (ex-post transparency). This overview takes the form of a timeline of data processing events, designed to be integrated into an existing mobile app, e.g., as part of some privacy settings, dashboard, or controls within the app. There

are two types of events shown in the timeline view: generic events and PAPAYA events. Generic events are intended to be tailored by the user of the tool to describe their data processing that does not use the PAPAYA platform, while the PAPAYA events are tied to the particular data analytics used from the PAPAYA platform. Interacting with a PAPAYA event shown an integrated view of the relevant explanation tool (see Section 6.1).
Compared to the initial design in D4.2 [5], we changed how data is provided to the tool. Instead of creating annotated logs as data processing occurs, we decided to simply specify a JSON data format as input to the tool. This provides the most flexibility and makes no assumption about how the system as a whole processes the data.
Further details are provided in D5.4 [15]. The tool is planned to be integrated in UC4.
Code is available at https://git.cs.kau.se/papaya/react-native-papaya-timeline.

6.4 Privacy Engine

The functionality provided by the Privacy Engine (PE) can be grouped in: Privacy Preferences Manager (PPM) and the Data Subject Rights Manager (DSRM).

The main aim of the PPM is to provide the data subject with the functionality necessary to define their preferences on the use of their personal data, while the DSRM is devoted to help the Data Controllers to comply with current legislation, helping them with the management of the events triggered by the data subject when exercising their rights (e.g. right to erase)

As described in Section 6.4 of D4.2 [5], the deployment of the set of components of PE (Back-End Servers and Front-End Interfaces) is as follows:

- The back-end components can be easily deployed using dockers. There are two images correspondent to the main functionalities of PPM which are PPM and DSRM. This is the command to execute to deploy PPM:

```
$ docker run -p 8080:8080 de.icr.io/papaya-de/privacy_engine-ppm-server:latest
```

  And this is the command to deploy DSRM container:

```
$ docker run -p 8080:8080 de.icr.io/papaya-de/privacy_engine-dsrm-server:latest
```

- In the case of the Front-End interfaces, they have been developed using the NativeScript framework and finally created a mobile APK which can be easily deployed in the mobile devices as a common APK. This link guides how to integrate these interfaces with the final pilot applications: https://www.nativescript.org/faq/how-do-i-add-nativescript-to-an-existing-ios-or-android-app

The Privacy Engine is included in two different Use Cases: *US2 - Privacy-preserving stress management* and *US4 - Privacy-preserving Mobile usage analytics*. Thereby PE evaluation will be included in the plan developed for the evaluation of both Use Cases. There is a video of this component here: https://www.papaya-project.eu/dissemination/demos.

7 Conclusions

In this deliverable, we presented modifications and extensions to the platform functional design, architecture and deployment presented in D4.1 [1] and D4.2 [5]. So, this deliverable completes the specification presented in those deliverables: we described in detail the core platform services dealing with privacy-preserving computations as well as the services responsible to ensure data privacy, security, and transparency of all the processes while operating the platform. We explained how different services can be integrated into the platform in a way that they will be interoperable/compatible with each other and could work together in the integrated platform. On the other hand, all platform services and tools are designed to be generic and can be deployed independently of each other, thus providing additional route for exploitation.

By using IBM Kubernetes cloud service, we show that the analytics are possible to run using modern cloud environments; by adding Identity and Access management (IAM), we ensure that access to the analytics can be properly authenticated and authorized; by adding auditing mechanisms, we ensure that the analytics generate appropriate logs that can be centrally collected. We also presented the design of platform dashboards that provides UI, configuration, and visualization functionality, and described how we deployed all the services on the IBM Kubernetes cloud account and how we evaluated their integration. Finally, to demonstrate main functionality of the PAPAYA platform, we also provided a number of demos (available at https://www.papaya-project.eu/dissemination/demos). These demos could help to disseminate and exploit the PAPAYA platform services and tools.

8 References

[1] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla and T. Pulls, "D4.1-Functional Design and Platform Architecture".

[2] S. Fischer-Hübner, B. Kane, J. S. Pettersson, T. Pulls, L. Iwaya, L. Fritsch, B. Rozenberg, R. Shmelkin, A. Palomares Perez, N. Ituarte Aranda and J. Carlos, *D2.2 - Requirements Specification,* 2019.

[3] B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla, B. Rozenberg and R. Shmelkin, *D3.1 - Preliminary Design of Privacy Preserving Data Analytics,* 2019.

[4] S. Canard, B. Vialla, B. Bozdemir, O. Ermis, M. Önen, M. Barham, B. Rozenberg, R. Shmelkin, I. Adir and R. Masalha, *D3.3 - Complete Specification and Implementation of Privacy preserving Data Analytics,* 2020.

[5] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla and T. Pulls, *D4.2 - Progress report on platform implementation and PETs integration,* 2020.

[6] S. Canard, B. Vialla, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, B. Rozenberg and R. Shmelkin, *D3.3 - Complete Specification and Implementation of Privacy preserving Data Analytics,* 2020.

[7] B. Zvika, G. Craig and V. Vinod, "Fully Homomorphic Encryption without Bootstrapping".

[8] Z. Brakerski, "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP," in *Annual Cryptology Conference*, 2012.

[9] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *Cryptology ePrint Archive,* vol. Report 2012/144, 2012.

[10] J. H. Cheon, A. Kim, M. Kim and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*, Hong Kong, China, 2017.

[11] Microsoft Research, Redmond, WA., *Microsoft SEAL,* : https://github.com/Microsoft/SEAL, 2018.

[12] S. G. M. M. M. A. a. S. C. Eleonora Ciceri, *D2.1: Use Cases and Requirements. PAPAYA Deliverable D2.1,* 2019.

[13] R. Shokri and V. Shmatikiv, "Privacy-preserving deep learning," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, Allerton, 2015.

[14] M. Abadi, A. Chu, I. Goodfellow, B. H. McMahan, I. Mironov, K. Talwar and L. Zhang, "Deep learning with differential privacy," in *the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[15] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, S. Canard, B. Vialla, A. Palomares Perez, N. Ituarte, T. Pulls, S. Fischer-Hübner, E. Ciceri and M. Mosconi, "D5.4 – PAPAYA PLATFORM GUIDE," 2021.

[16] T. Pulls and D. Rasmus, "Steady: A Simple End-to-End Secure Logging System," https://eprint.iacr.org/2018/737, 2018.

[17] S. Fischer-Hübner, "Transparent Privacy Preserving Data Analytics (PAPAYA deliverable D3.4)," PAPAYA EU H2020 project., 2020.

[18] S. Canard, "Telecom Use Case Validation (PAPAYA Deliverable D5.2)," PAPAYA EU H2020 project, 2021.

[19] S. Halevi, "HElib," [Online]. Available: https://github.com/homenc/HElib.

[20] "JustGarble," [Online]. Available: https://github.com/irdan/justGarble.

[21] C. r. a. O. S. University, "libOTe," [Online]. Available: https://github.com/osu-crypto/libOTe.

[22] N. Smart and F. Vercauteren, "fully Homomorphic SIMD Operations".

[23] wiki, "Garbled circuits," [Online]. Available: https://en.wikipedia.org/wiki/Garbled_circuit.

[24] wiki, "Oblivious transfer," [Online]. Available: https://en.wikipedia.org/wiki/Oblivious_transfer.

[25] F. Chollet, "Keras," [Online]. Available: https://keras.io/.

[26] M. Abdalla, F. Bourse, A. De Caro and D. Poincheval, "Simple Functional Encryption Schemes for Inner Products," in *IACR International Workshop on Public Key Cryptography*, 2015.

[27] M. Abdalla, D. Catalano, D. Fiore, R. Gay and B. Ursu, "Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings," in *Annual International Cryptology Conference*, 2018.

[28] S. Agrawal, B. Libert and D. Stehlé, "Fully Secure Functional Encryption for Inner Products, from Standard Assumptions," in *Annual International Cryptology Conference*, 2016.

[29] J. Chotard, E. D. Sans, R. Gay, D. H. Phan and D. Pointcheval, "Decentralized Multi-Client Functional Encryption for Inner Product," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2018.

[30] E. D. Sans, R. Gay and D. Pointcheval, "Reading in the Dark: Classifying Encrypted Digits with Functional Encryption," IACR Cryptology ePrint Archive, 2018.

[31] C. E. Z. Baltico, D. Catalano, D. Fiore and R. Gay, "Practical Functional Encryption for Quadratic Functions with Applications to Predicate Encryption," in *Annual International Cryptology Conference*, 2017.

[32] J.-G. L. a. J. H. a. K.-Y. Whang, "Trajectory Clustering: A Partition-and-Group Framework," in *SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Beijing, China , 2007.

[33] Y. Ishai, J. Kilian, K. Nissim and E. Petrank:, "Extending Oblivious Transfers Efficiently," in *CRYPTO*, 2003.

[34] chiraag, "Gazelle MPC," [Online]. Available: https://github.com/chiraag/gazelle_mpc/tree/master/src/lib.

[35] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2006.

[36] C. C. a. A. M. F. Tom Bohman, "Min-Wise Independent Linear Permutations," *Electr. J. Comb.*, vol. 7, 2000.

[37] Y. Z. a. M. R.-N. Maede Rayatidamavandi, "A Comparison of Hash-Based Methods for Trajectory Clustering," *15th {IEEE} Intl Conf on Dependable, Autonomic and Secure Computing*, pp. 107-112, 2017.

[38] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, S. Canard, B. Vialla and T. Pulls, "D4.3 Final report on platform implementation and PETs integration," To be submitted in 2021.

[39] T. Pulls, L. Fritsch, L. Iwaya, F. Karegar, A. Palomares and J. C. Prez Ban, "D3.2 - Risk Management Artefacts for Increased Transparency," PAPAYA report document, 2019.

[40] S. Fischer-H¨ubner, M. T. Beckerle, J. S.¨ Pettersson and P. Murmann, "D3.4 - Transparent Privacy preserving Data Analytics," PAPAYA report document, 2020.