D4.3

Final Design and Development of Privacy Aware Analytics for Secure Services

WP4 – Privacy-Aware Analytics for Security and Services

SIFIS-Home

Secure Interoperable Full-Stack Internet of Things for Smart Home

Due date of deliverable: 30/06/2023
Actual submission date: 30/06/2023

Responsible partner: CNR
Editor: Paolo Mori
E-mail address: paolo.mori@iit.cnr.it

30/06/2023
Version 1.3

Project co-funded by the European Commission within the Horizon 2020 Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

The SIFIS-Home Project is supported by funding under the Horizon 2020 Framework Program of the European Commission SU-ICT-02-2020 GA 952652

Authors: P. Mori, W. Alabbasi, M. Simoni, A. Saracino (CNR), O. Isohanni, J. Jämsä (CEN), H. Lundtrom, S. Moriz (SEN)

Approved by: Luca Ardito (POL), Samuli Stenudd (RIOTS)

Revision History

Version	Date	Name	Partner	Section Affected Comments
0.1	07/12/2022	P. Mori	CNR	Initial ToC
0.2	24/02/2023	H. Lundstrom, S. Moriz	SEN	Description of xAnomaly Analytics
0.3	19/03/2023	W. Alabbasi	CNR	Description of Face Recognition, Person Recognition Analytics
04	1/04/2023	W. Alabbasi	CNR	Description of new version of Privacy Aware Speech Recognition and Voice Anonymization Analyics
0.5	14/04/2023	O. Isohanni, J. Jamsa	CEN	Description of new version of Netspot Analytics
0.6	28/04/2023	M. Simoni	CNR	Description of Multi Level Instrusion Detection Analytics
0.7	01/06/2023	W. Alabbasi, P. Mori. A. Saracino	CNR	Added Section: Preserving Privacy of Data Exploited in Analytics
0.8	04/06/2023	W. Alabbasi	CNR	Description of Anomaly Detection in Audio Signals Analytics
0.9	09/06/203	P. Mori	CNR	Added Introduction and Conclusion Sections
1.0	14/06/2023	P. Mori	CNR	Version ready for internal revision
1.1	26/06/2023	P. Mori, W. Allabasi,	CNR	Addressed Reviewers' comments
1.2	27/06/2023	H. Lundstrom	SENS	Addressed Reviewers' comments
1.2	29/06/2023	P. Mori	CNR	Final Version

Executive Summary

Deliverable D4.3 is the final outcome of Work Package 4 "Privacy-Aware analytics for Security and Services", and it provides the description of the analytics that have been designed and developed in the second half of the SIFIS-Home project period, i.e., after M18, as well as the updated versions of a couple of analytics that were designed before M18 but they have been significantly updated in the second half of the project period.

Thanks to the analytics designed in the second half of the project, at the end of Work Package 4 we have covered the analytics that have been described in deliverable D4.1. For each of these analytics, this document gives a detailed description of the aim, of the requirements they address (among the one defined in deliverable D1.2 "Final Architecture Requirements Report"), of the input data they need and the results they return, of the analytic training and testing, along with some consideration about the privacy of the input dataset and of the results. The analytics are grouped under four activity areas, following the WP4 task's structure: i) anomalies and misbehavior detection; ii) network intrusion detection; iii) policy enforcement; and iv) privacy aware speech recognition. All the analytics designed and developed in the project period have been integrated in the SIFIS-Home framework, and this document describes how this integration has been implemented. In this way, all the other components of the SIFIS-Home framework can invoke such analytics and can benefit of their results.

When analytics are executed outside the smart home cyber-perimeter, for instance exploiting Cloud services, data privacy problems could arise, and hence the data are anonymized before being sent to the Cloud based analytics. In deliverable 4.2 we described a set of techniques for preserving privacy of sensitive data. However, applying such techniques to the data could reduce the data utility in such a way that the accuracy of the results provided by the analytics is significantly affected. Hence, in this deliverable we also defined a methodology to fine tune the application of privacy preserving techniques in order to preserve data privacy as much as possible but, at the same time, keep the result accuracy level over a given threshold.

Table of contents

1 Introduction

Deliverable D4.3 is the final output of Work Package 4, “Privacy Aware Analytics for Security and Services” due at the end of the work package activities (M33) and it consists of two parts. The first part of deliverable D4.3 consists of this document, which describes the analytics that have been designed and developed following the requirements we defined in deliverable D1.2 "Final Architecture Requirements Report", in order to be applicable to the Smart Home environment taken into account in the SIFIS-Home project. In particular, this deliverable focuses on the analytics that have been designed and developed after M18, but also reports an updated description of a couple of analytics that have been designed and developed before M18, but that undergone relevant modifications in the last period of the project. Instead, the analytics that have been designed and developed before M18, and that did not undergo relevant modifications in the last period of the project, are described in D4.2 and, for the sake of simplicity, we didn’t report them in this deliverable.
The following is the complete list of analytics produced by WP4 of the SIFIS-Home project, grouped according to the activity areas defined by WP4 task's structure. For each analytics we specify the deliverable where it has been described:

Anomaly and Misbehaviour Detection and Prevention Analytics:

- Device Fault Detection – D4.2
- Device Activity Monitoring in Centralized Cloud – D4.2
- xAnomaly – D4.3
- Face Recognition and Person Recognition – D4.3
- Parental Control – D4.2
- Object Detection – D4.3
- Multilevel Anomaly Detection – D4.3

Network Intrusion Detection Analytics:

- Netspot Network Anomaly Detection – D4.3
- Anomaly Detection for IoT Devices in a Consumer Home Network – D4.2

Policy Enforcement

- Policy Enforcement – D4.2

Privacy Aware Speech Recognition and Smart Services Analytics

- Privacy Aware Speech Recognition – D4.3
- Voice Recognition and Verification – D4.3
- Anomaly Detection in Audio Signal Analysis – D4.3

In deliverable D4.1 we provided a list and a high-level description of the analytics that will be designed and developed within WP4, and the previous list covers them.
Besides designing and developing a number of new analytics, in the second period of the project we also integrated all the analytics within the Data Analysis Toolbox, which is a component of the SIFIS-Home framework (see the next Section). This integration allows the other components of the SIFIS-Home framework, such as the System Protection Manager, to invoke the execution of the aforementioned analytics, and to get their results through the DHT in order to react to intrusions by

taking proper countermeasures.

Finally, this document studies the privacy issues concerning data collected in the smart home environment when they are sent outside the smart home cyber-perimeter to be elaborated by external analytics. In deliverable D4.2 we provided a number of privacy preserving techniques that can be adopted to preserve the privacy of data collected in the smart home. However, applying privacy preserving techniques to such data could affect the accuracy of the results of the analytics where these data are exploited. Moreover, another factor that should also be taken into account according to the last directives, such as the EU proposal for the Artificial Intelligence Act[1], is the decision explainability. As a matter of fact, understanding the decision-making criteria applied by artificial intelligence based analytics is very important for both technical and ethical reasons. For this reason, in Section 5 of this document we studied a methodology that take into account privacy requirements concerning the data exploited in the analytics, but also the desired accuracy of the result in order to define the optimal trade-off between data utility, privacy, and explainability. We applied this methodology to face analysis. In particular, we exploited two common techniques for preserving data privacy, Differential Privacy and Autoencoders (already introduced in D4.2) and one technique for ensuring model explainability (Saliency Mas enhanced with Smoothgrad), and we performed a set of experiments to demonstrate and validate the proposed methodology.

The second part of deliverable D6.3 consists of the source code of the previously mentioned analytics, which is available in the project repository at the following link: https://github.com/sifis-home/.

2 Role of Analytics in the SIFIS-Home Framework

To ensure security and privacy in the SIFIS-Home framework, machine learning, deep learning statistics and rule-based techniques are adopted to analyze the behaviour of users and devices in the smart home, to detect intrusions at cyber-physical level (e.g., face-sound recognition), at network level (Network Intrusion Detection Analytics), and at system level (Multi-Level Intrusion Detection). Such analysis mechanisms exploit features extracted through continuous monitoring of the full smart-home stack, e.g., of device system calls of the smart home devices, network events, sensor data, DHT operations, collected sounds, pictures and videos. The results of such analysis are used by the SIFIS-Home framework to detect dangerous or not desirable situations in the smart home, in order to timely react and take proper countermeasures to prevent, avoid or mitigate the impact of the malfunctioning/attack.

The SIFIS-Home architecture includes a specific component, namely Data Analytics Toolbox, which hosts all the software component implementing the aforementioned analytics. This component provides the tools to analyze textual, tabular, and multimedia data in order to perform predictions, analyze voice and gesture commands, detect intrusions and misbehaviors, as well as for providing advanced smart services to the smart home users. The Data Analytics Toolbox is part of the Application Toolboxes module, shown in Figure 1. Please refer to Deliverable D1.3 for a detailed description of the Application Toolboxes module and of the rest of the SIFIS-Home framework.

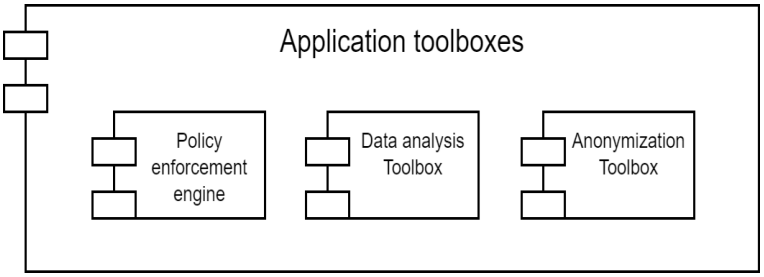

Figure 1: Application Toolboxes module

As shown in Figure 1, the Application Toolboxes module also includes the Anonymization Toolbox engine, which contains software tools that are used to preserve privacy of data during analysis. Depending on the data type and the desired level of privacy, the most appropriate anonymization tool can be chosen from the toolbox, in order to generalize or suppress data information, as well as for supporting differential privacy for privacy preserving data analysis. These anonymization techniques have been described already in D4.2. In this deliverable we focused on two of these techniques, Differential Privacy and Autoencoders, and we studied a methodology to guide the choice of the privacy degree to use for the data exploited for a given analytics in order to guarantee a good privacy level and, at the same time, a good accuracy of the analytics results. This methodology also take into account the explainability aspect of analytics, since understanding why a decision has been taken by an artificial intelligence based analytics is very important for both technical and ethical reasons. As a matter of fact, since the analytics are based on Artificial Intelligence, they could inadvertently inherit biases from the data they are trained on, and hence they could produce discriminatory (and even wrong) results. Introducing explainability, it is possible to identify such discriminatory patterns and correct them, thus ensuring that the system treats all individuals fairly and without discrimination.

3 Analytics Design

This section gives a detailed description of the analytics that have been designed and developed in Work Package 4 after M18. We recall that these analytics are grouped into four areas, consistently with the tasks of Work Package 4: i) Anomaly and Misbehavior Detection, ii) Network Intrusion Detection, iii) Policy Enforcement, and iv) Privacy Aware Speech Recognition. The description is structured in the same way for all the analytics. First, we describe the main aim of the analytics in the context of the SIFIS-Home scenario and, with reference to deliverable D1.2 "Final Architecture Requirements Report", we list the requirements that the analytics contributes to satisfy, as well as the requirements that the analytics introduces. Then, we describe the data that are given as input to the analytics, how such data are collected from the sensors in the SIFIS-Home framework, how the analytic is trained and tested with such data and the results provided, as well as some privacy consideration on the input data and results. Moreover, we describe the hardware requirements of the analytics, in order to understand what classes of devices it can be executed. Finally, we present some implementation details of the analytics, along with some preliminary results that have been obtained by executing the analytics on the dataset previously described.

3.1 *Anomaly and Misbehavior Detection*

The analytics of the Anomaly and Misbehavior Detection area have been defined to preserve data correctness and integrity, in addition to preventing anomalous actions and behaviors in smart

environments. Data collected from connected devices and network data are processed to provide security as a service, by ensuring multi-level anomaly and misbehavior detection and prevention.

3.1.1 xAnomaly

3.1.1.1 Aim of the Analytics

Our goal is to detect intrusion attempts by creating an anomaly detection system for IoT data. Anomaly detection is the process of identifying unusual patterns in data that deviate from normal behavior. Supervised anomaly detection is a type of anomaly detection that learns from labelled data and can be used to identify previously seen anomalies. On the other hand, unsupervised anomaly detection learns from unlabelled data and can be used to identify previously unseen anomalies. As new intrusion attempts are likely to differ from previous ones, an anomaly detection system for IoT data needs to be able to handle previously unseen anomalies. Therefore, we have chosen to work with unsupervised models. We have chosen to base our system on autoencoders (AEs), a standard choice in the industry (Pang et al. 2021, Ruff et al. 2021).

3.1.1.2 Requirements related to the analytics

The xAnomaly analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-15: The SIFIS-Home system shall provide means of identifying anomalous situations and behaviours inside the smart home
- Non-Functional Requirements:
  - PE-10: An abnormal (suspicious) behavior caused by a malware shall be identified and notified within 60 seconds
- Security Requirements:
  - SE-17: Anonomalous device behaviours should be identified and signalled in less than 60 seconds.

3.1.1.3 Input Data

The input data, X, is equivalent to the output data, X. In our proof-of-concept (PoC), the input data is time series windows of length t, xi =x0 ... xt−1, that we create by subdividing the original series into one-step windows.

3.1.1.4 Analytic Design

An autoencoder is a neural network model where the input data is encoded to a bottleneck layer and decoded to reconstruct the input data. You can picture encoding as finding a compression function and decoding as finding a decompression function. The compression function is enforced by the bottleneck layer being small, which forces the data to be represented in a smaller space.

Encoder : $z = \phi e (x; \Theta e)$
Decoder : $\hat{x} = \phi d (z; \Theta d)$

The encoder network, $\phi e ()$, with parameters Θe , outputs the bottleneck layer nodes, z. The bottleneck layer, z, is used as input in the decoder network, $\phi d ()$, with parameters Θd , to create the reconstruction, $\hat{x}$, of the original series. Training proceeds in a normal fashion by minimizing the loss, which is often referred to as the reconstruction error in the autoencoder context. In our PoC, the encoder and decoder networks are fully-connected layers. We chose this as a starting point because it allows us to take into account time dependencies while circumventing the vanishing gradient problem and, at the same time,

it does not add complexity to the model (as noted in Xu, 2018).

Anomaly scores are approximated by using the empirical distribution of the reconstruction error. In our PoC, we train using the mean absolute error to limit the influence of anomalies in our training data. We calculate an anomaly threshold by letting the user set how many anomalies they want to be flagged per year, which implicitly sets specific type 1 and type 2 error rates. We calculate the corresponding anomaly threshold by estimating it from the empirical distribution function of the reconstruction error on the input data.

It is straightforward to construct anomaly scores for autoencoders, whereas the literature has not yielded a conclusive answer on how to construct them for variational autoencoders (Xu et al. 2018, An and Cho 2015). Therefore, we have refrained from using variational autoencoders so far. The benefit of variational autoencoders in our context would be to allow different types of anomalies to cluster in the bottleneck space, which would not be a beneficial feature until a user-feedback mechanism exists.

3.1.1.5 Training and Testing

Available Dataset
We have used a non-public dataset consisting of approximately half a year worth of time series data of infrared light readings sampled at irregular intervals (it sends more data if it previously noted a change).

Training strategy
We have trained the model on the above-mentioned data.

3.1.1.6 Output data
The analytics outputs a zero if an anomalous event is detected, otherwise nothing. This is to minimize data transfer.

3.1.1.7 Hardware Requirements
The training and verification of xAnomaly has been performed on a standard Sensative Strips Presence sensor connected to the SIFIS Home network but could have used any device or measurement that delivers time series measurements.

3.1.1.8 Privacy Considerations
We have based our autoencoder on a single time series to allow us to deploy the models in a privacy-preserving manner at a fog node.

3.1.1.9 Execution of the Analytics by the Data Analysis Toolbox
Since the Analytics interact with SIFIS Home Cloud Interface (Yggio), the integration with the Data Analysis Toolbox is done through Yggio. A service which forwards a selected set of messages from Yggio to the DHT and vice versa has been developed to this aim.

3.1.1.10 Implementation Details

Our PoC is an autoencoder model that takes infrared light readings from a sensor under my office desk as the input time series (e.g. it notices that I am in front of my computer writing this). The choice is

motivated by the ease of testing deployment, both software engineering- and model-related, from an easily accessible sensor.

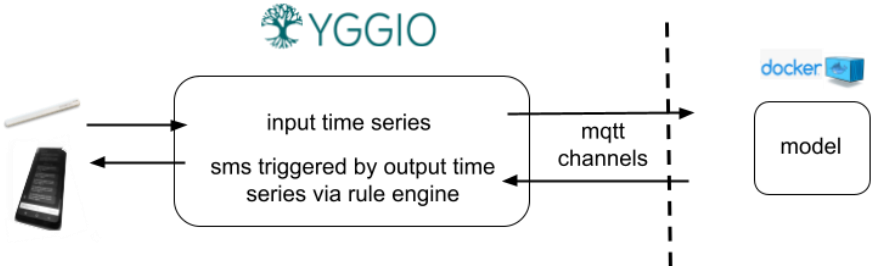

Figure 2: Yggio SIFIS Home Cloud Interface

The figure above displays the architectural overview. SIFIS Home Cloud Interface (Yggio) handles most of the functionality. The input time series is streamed in real-time via an MQTT channel to a Python-based Docker container. The container uses the Paho MQTT client library to receive input data points and the Tensorflow Keras library to query a trained VA-model. If an anomaly is detected, an MQTT message is sent back to Yggio and stored in an output time series. Yggio is then configured to send a SMS to my mobile phone.

3.1.1.11 Preliminary Results

We did not specify the exact model, as there are many possible specifications and currently, we do not have a cost-effective way of evaluating anomalies. The challenge is further complicated by the fact that IoT data is difficult for humans to interpret, meaning that it is hard to create labelled data. Furthermore, for effective labelling, we need to be careful in explaining why certain data points constitute anomalies and why others do not. For example, a temperature reading of -10 at 12:00 may be considered an anomaly, but at 5:00 it would not be. There are many papers that try to prove that their model specification is superior by showing that it works well in a specific case, but we want a more general type of anomaly detection system. Therefore, we aim to find commonalities among model types that work well. To lower the cost of evaluating anomalies, we plan to have users evaluate the model output. As interpretability is crucial for success, we have named our project "Explainable Anomaly" (xAnomaly). User input will allow us to redefine our model specifications.

3.1.2 Face Recognition – Person Recognition

3.1.2.1 Aim of the Analytics

Accurate person recognition is essential for various purposes, including identifying home residents, guests, and potential intruders. It enables the provision of personalized smart services based on individual identities. Typically, face recognition models are trained using images of the home's resident users. When a person enters the home or a specific room, their face image is detected and compared against the trained model, allowing for classification based on extracted facial features. This process ensures reliable and efficient person recognition in a home environment.

3.1.2.2 Requirements related to the analytics

The Face Recognition analytic is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-1: this analytic satisfies the functional requirement F-1 by identifying the resident users and administrators inside the smart home through biometrics.

    - F-3: this analytic satisfies the functional requirement F-3 by matching read biometrics against a database of stored ones.
- Non-Functional Requirements:
    - PE-2: The user recognition shall happen in less than 5s.
    - PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection.
    - US-26: The presence of a GPU is needed to perform DL-based analysis.
    - DE-06: the identification through biometrics should be performed correctly in more than 95% of cases.
    - DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.
- Security Requirements:
    - SE-21: Gaussian Blurring parameters are configurable for data analysis and can be changed.
    - SE-22: this analytic is able to work with anonymized data, but the accuracy of the results might be decreased.

3.1.2.3 Input Data

The face recognition algorithm utilizes video frames captured by surveillance cameras within a controlled environment. This includes both the camera of the controlled device itself and other surveillance cameras deployed in the surroundings. In addition, recorded videos and captured images can also be used as input data. The algorithm requires a database directory that contains the identities of authorized users. By processing this input data, the face recognition system can accurately identify and verify individuals within the controlled environment.

3.1.2.4 Analytic Design

The used face recognition model is DeepFace [Facebook, SER20]. It is an open-source model which uses a deep learning mechanism for face recognition. The core of the architecture is a Convolutional Neural Network (CNN) model that takes image data as input and produces the relative representation as output and verifies it with the representations of authorized identities. The model has been trained on a large dataset for face recognition based on a distance metric between face representations. This model has been validated with a set of experiments on a well-known dataset, the Labelled Faces in the Wild (LFW) dataset. Figure 3 shows the entire workflow in a sample scenario, it starts with image capture and anonymization using Gaussian blurring on the user side. The resulting anonymized images are forwarded to the server to be processed for face recognition starting with a face detector OpenCV to detect all faces within an image. Detected faces are aligned and then converted to vectors. Finally, the faces are verified by comparing their representations with the representations of face images stored in the database. We use VGG-Face deep-learning model for face recognition. The pipeline of the analytic is composed of four different components as shown in Figure 3.

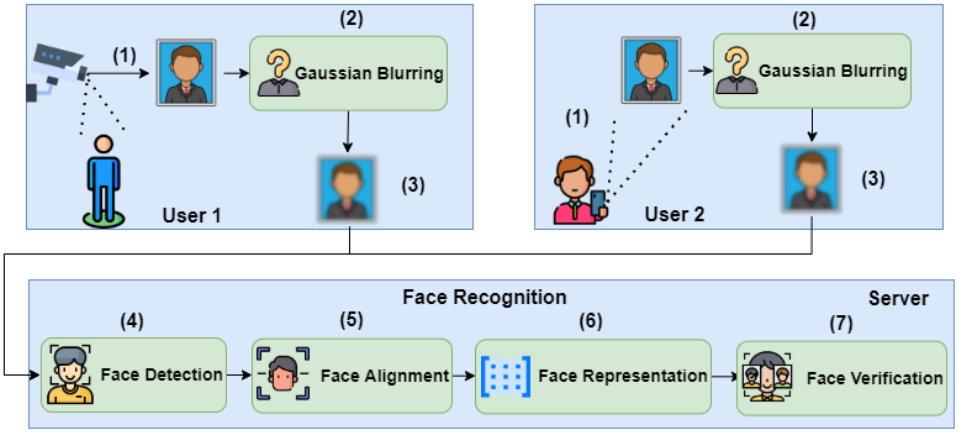

Figure 3: Privacy-Preserving Face Recognition

The SIFIS-Home face recognition analytic is composed of a pipeline of the following four components:

Face Detection: using the Haar Cascade classifier. Haar-like features are used to detect facial regions like eyes and nose, and they are determined using the integral image representation mechanism, and the AdaBoost classifier is used to select only relevant Haar-like features that are known to improve the binary face classification algorithm results.

Face Alignment: face images might have several poses and expressions, which may affect the accuracy of the face detection and face verification models. Thus, to decouple these poses and expressions from the face identity and to reduce their effect on the face detection and classification algorithms, face alignment is used. For face alignment, we use a simple trigonometric method. This method detects eyes and eye coordinates as a first step and draws a triangle between eyes based on their centre locations. The angle of the lower eye needs to be computed, and this is done based on the horizontal line drawn between eye centres, computing the length of the three edges of the triangle between eyes using the Euclidean distance. Then, the angle is calculated with the cosine rule. Finally, the image is rotated based on this angle.

Face Representation: converts face images into vector embeddings, so that vectors of similar images for the same person are closer in distance among them. Trained Convolutional Neural Networks (NN) for face recognition tasks do represent face images in the layer before the output layer. Therefore, these trained models on huge datasets can sufficiently represent new face images into high dimensional representations, and to extract and represent facial features of complex concepts.

Face Verification: compares face representations produced by the layer before the output layer of the CNN used to perform face recognition. The output layer returns the classification of a face image compared to another one, whether they belong to the same person or not, based on semantic closeness. To measure semantic similarity, we use the cosine similarity metric, which finds the cosine of the angle between two representations.

3.1.2.5 Training and Testing

Available Dataset
The used dataset for testing is the Labeled Faces in the Wild (LFW) dataset, which is a database of face photographs designed for studying the problem of unconstrained face recognition [HRB08]. The data set contains more than 13,000 images of faces collected from the web. 1,680 of the pictured persons have two or more distinct photos in the dataset.

Training strategy
The DeepFace framework was trained on a large-scale dataset called the Social Face Classification (SFC) dataset. The SFC dataset consists of millions of labeled images of individuals sourced from the internet, including various social media platforms. Then, we tested the framework face recognition models on the anonymized Labeled Faces in the Wild (LFW) dataset with different privacy degrees. We reported the results in our paper [AMS22].

3.1.2.6 Output data
The output of the face recognition analytic is the identity of a person's face, including the identity label, since the analytic has access to a database of known individuals, it provides the identity of the person in the detected face by matching it with the stored identities. Also, matching similarity is provided.

3.1.2.7 Hardware Requirements
The recommended Hardware Requirements for running this analytic are as follows:

- GPU: for better cost and performance, TX 2070 or an RTX 2080 Ti is recommended, with 16-bit models and memory >=11 GB. A blower-style fan for cooling is also recommended.
- RAM: 16 GB RAM or above is recommended, as appropriate in the presence of high-performance GPUs.
- CPU: CPU PCIe lanes and motherboard PCIe lanes must support the desired number of GPUs. A minimum of 2 threads per GPU must be supported, i.e., usually one core per GPU.

3.1.2.8 Privacy Considerations

- Input data
  In order to preserve the privacy of data, Gaussian blurring is used for image/frame anonymization [PUL19]. Gaussian blurring convolves an image with a Gaussian filter of different sizes for anonymization. The privacy degree is controlled using the radius of blur metric. Gaussian blurring uses a low-pass filter which performs the smoothing function of an image.
- Results
  To further protect the data, individual identities can be hashed in the database, so that the returned identity would be protected.

3.1.2.9 Execution of the Analytics by the Analytics Toolbox
This analytic is invoked through physical analysis and multi-level analysis APIs to analyze image/video data. The data are passed to be pre-processed for format adaption, then the privacy-preserving technique is applied to the dataset to be forwarded to the analytics engine, which performs data processing using machine learning and deep learning toolboxes. Afterwards, the analysis results are interpreted,

aggregated, and presented to the user.

3.1.2.10 Implementation Details

This analytic is implemented using the DeepFace, Keras, TensorFlow, PyTorch, and Numpy libraries.

Face Detection: The framework utilizes the Haar cascade classifier, a pre-trained model available in OpenCV, to detect faces in images.
Face Alignment: face alignment is performed after face detection to ensure that the detected faces are properly aligned. The facial landmarks, such as the positions of the eyes, nose, and mouth, are detected using the dlib library. These landmarks are then used to estimate the pose and alignment of the face.
Face Representation: the VGG-Face model is used to extract high-level facial features from the aligned face images. These features capture the unique characteristics of each face and can be used for various face recognition tasks. The VGG-Face model has been trained to recognize a wide range of identities, making it suitable for face representation.
Face Verification: To perform face verification, the analytic calculates the cosine similarity between the face representations extracted from the two input face images. A threshold value is then applied to the distance or similarity score to determine whether the faces are considered a match or not.

3.1.2.11 Preliminary Results

The Privacy-Aware-Face-Recognition experiments and results have been reported in the two published papers [AMS22, MAS22]. The results include face detection rates, face verification accuracy, and trade-off score. The trade-off score takes into account the face detection accuracy, face verification accuracy, and the privacy gain resulting from the privacy mechanism (Gaussian Blurring at different levels). Face Detection Rate Results for the OpenCV detector are presented in Figure 4, which shows the results obtained by the Face Detection component on the LFW testing Set by also considering different degrees of privacy and comparing blurred images to original images or blurred imagesB2B). The figure shows that there is a negative relationship between the privacy degree and the face detection rate, since increasing the privacy degree results in a lower face detection rate.

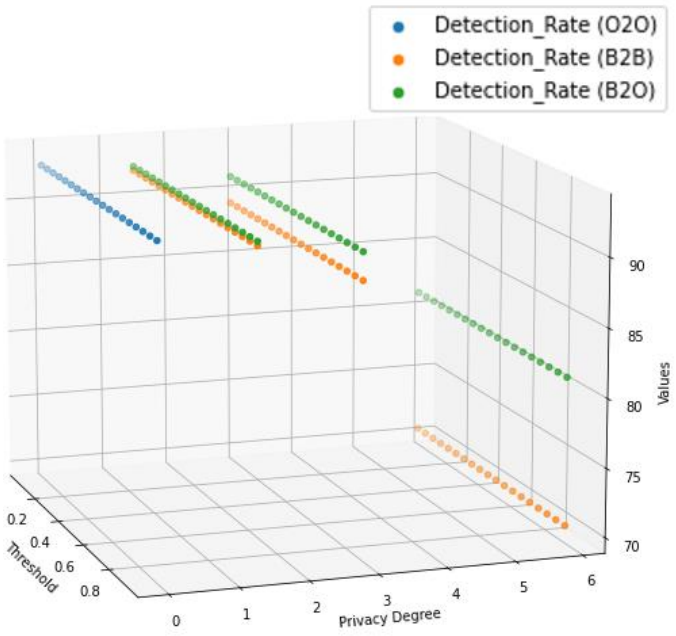

Figure 4: Face Detection Results

Face Verification Accuracy Results are reported in Figure 5, which shows the results obtained by the Face Verification component on the LFW testing Set by also considering different degrees of privacy and comparing blurred images to original images or blurred images and different similarity thresholds. The figure shows that the accuracy is greatly affected by the value of the distance threshold and by the used privacy degree.

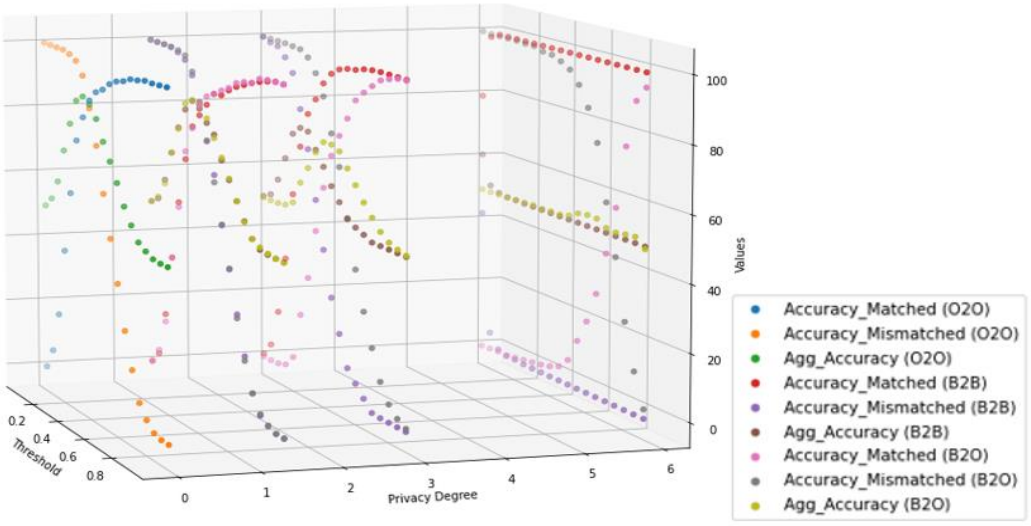

Figure 5: Face Verification Results

The Trade-off Results are reported in Figure 6, which shows that the accuracy is greatly affected by the value of the distance threshold and by the used privacy degree.

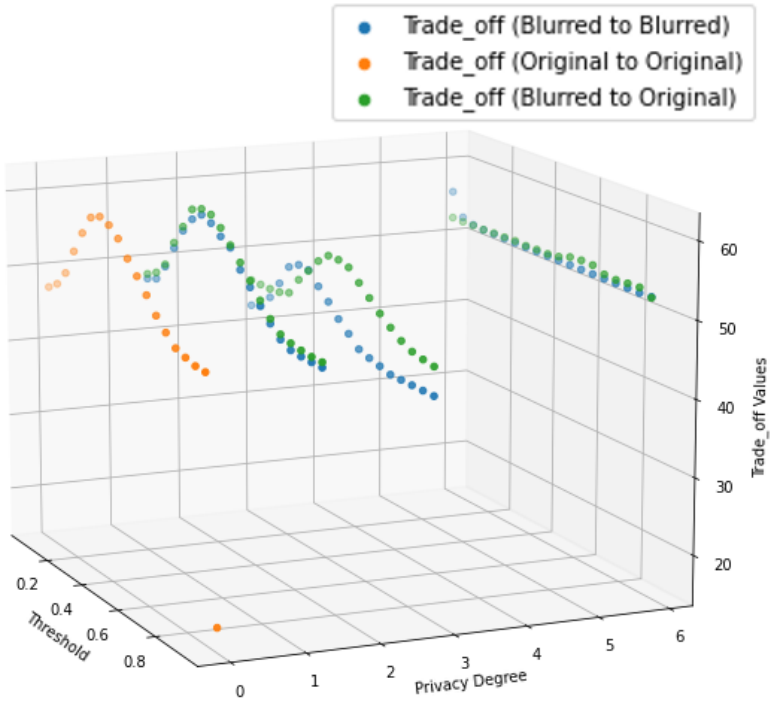

Figure 6: Trade-Off Score Results

3.1.3 Object Recognition/Detection

3.1.3.1 Aim of the Analytics

A crucial aspect of a smart home system is its ability to detect and recognize potentially dangerous objects that may not be easily identifiable by humans alone. This process involves object recognition, where suspicious objects are identified within captured images or videos and their locations within the frame are determined. These suspicious objects can range from intruders to fire incidents, elderly or vulnerable individuals in risky situations, or misplaced hazardous items like sharp tools. To achieve this, the system utilizes cameras to capture images or record videos, which are then transmitted for analysis. During the analysis phase, the system performs object recognition by identifying and classifying the objects present in the images, along with their precise locations. If any suspicious objects are detected, the user is promptly alerted, ensuring timely response and appropriate action. By integrating object recognition capabilities into the smart home system, it becomes capable of effectively identifying potential threats and alerting users to take necessary measures. This enhances the safety and security of the home environment by leveraging advanced image analysis techniques and automation.

3.1.3.2 Requirements related to the analytics

The Face Object Recognition analytic is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-17: the SIFIS-Home system shall provide means of recognition of prohibited objects inside the smart home and signal resident users and administrators.
  - F-18: the SIFIS-Home system shall provide means of recognition of allowed objects inside the smart home in unusual positions, and signal resident users and administrators.
- Non-Functional Requirements:

- PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection.
- US-26: The presence of a GPU is needed to perform DL-based analysis.
- DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.

- Security Requirements:
  - SE-21: Differential Privacy parameters are configurable for data analysis and can be changed.
  - SE-22: this analytic is able to work with anonymized data, but the accuracy of the results might be decreased.

3.1.3.3 Input Data

The object recognition algorithm employed in the system leverages images or video frames obtained from cameras positioned throughout the controlled environment. This encompasses both the camera integrated within the controlled device and external surveillance cameras placed in the smart home environment. Moreover, recorded videos and individual images can also be utilized as input data for analysis. Through the processing of this input data, the object recognition system identifies and classifies objects based on the patterns and features learned by the underlying model during training. By comparing the detected objects to the known objects, the system can recognize and categorize the objects present in the captured frames.

3.1.3.4 Analytic Design

The process of object recognition begins with acquiring images or video streams using a smart device equipped with an embedded camera. The images or video streams undergo pre-processing operations such as resizing and perturbation. The pre-processed images or video streams are then fed into our YOLOv3 model [RF18]. YOLOv3 operates by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. It uses a single convolutional neural network (CNN) to simultaneously perform object localization and classification of object class labels in a single pass. YOLOv3 is known for its speed and efficiency, making it suitable for real-time applications. The model has been trained on the COCO dataset. The pipeline of the analytic is composed of the components shown in Figure 7. It initially detects and localizes objects within the images or video frames and tracks them over time in the case of video streams. Finally, the detected objects are identified based on the COCO dataset objects that the YOLOv3 model was trained to recognize.

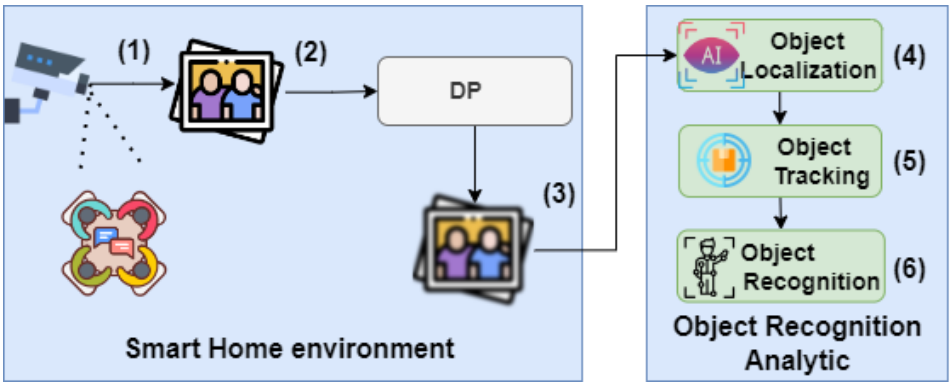

Figure 7: Privacy-Preserving Object Recognition

3.1.3.5 Training and Testing

Available Dataset
The ImageNet dataset is a widely used large-scale dataset for object recognition in computer vision research [DJS09]. It consists of millions of labeled images of thousands of object categories. The ImageNet dataset contains over 14 million images collected from various sources on the internet. There are approximately 1,000 high-level categories in the dataset, and each category contains a varying number of images. Examples of categories include animals, vehicles, household items, and natural scenes. Due to the large size and complexity of the dataset, researchers often use subsets or subsets of subsets for training and evaluation, depending on their specific needs and computational resources such as the ImageNette dataset [ImageNette].

Training strategy
The YOLOv3 model was trained on the COCO (Common Objects in Context) dataset [LMBH14]. The COCO dataset is a widely used benchmark dataset for object detection, segmentation, and captioning tasks. It contains over 200,000 images from various contexts and 80 common object categories. It covers a wide range of objects, including people, animals, vehicles, and everyday items.

3.1.3.6 Output data
The output of the object recognition analytic is the class of the detected object, including the object label. The analytic also provides information about the locations or bounding boxes of the identified objects within the image or video.

3.1.3.7 Hardware Requirements
The recommended Hardware Requirements for running this analytic are as follows:

- GPU: for better cost and performance, TX 2070 or an RTX 2080 Ti is recommended, with 16-bit models and memory >=11 GB. A blower-style fan for cooling is also recommended.
- RAM: 16 GB RAM or above is recommended, as appropriate in the presence of high-performance GPUs.
- CPU: CPU PCIe lanes and motherboard PCIe lanes must support the desired number of GPUs. A minimum of 2 threads per GPU must be supported, i.e., usually one core per GPU.

3.1.3.8 Privacy Considerations

- Input data
  In order to preserve the privacy of data, Differential privacy mechanism is used [Dwo08]. Differential Privacy is a powerful privacy-preserving technique widely used to add noise to data, including images. We use it to add random Laplacian noise to the pixel values of images and ensure that privacy is protected while still allowing valuable insights to be extracted from the data. The amount of noise added is determined by the sensitivity and privacy budget parameters, where sensitivity measures the impact of input data changes on algorithm output, and the privacy budget controls the level of noise added to protect privacy.
- Results
  The results are composed of object labels in the images or video frames. Therefore, no additional privacy mechanisms are required to be applied to the results.

3.1.3.9 Execution of the Analytics by the Analytics Toolbox
This analytic is invoked through physical analysis and multi-level analysis APIs to analyze image/video

data. The data are passed to be pre-processed for format adaption, then the privacy-preserving technique is applied to the dataset to be forwarded to the analytics engine, which performs data processing using machine learning and deep learning toolboxes. Afterwards, the analysis results are interpreted, aggregated, and presented to the user.

3.1.3.10 Implementation Details

This analytic needs the OpenCV, Keras, TensorFlow, Darknet, and Numpy libraries to work.
Object Detection: YOLOv3 employs anchor boxes of different sizes and aspect ratios to enhance the detection of objects at various scales. Also, the input image is divided into a grid of cells, and each cell is responsible for detecting objects that fall within its boundaries.
Localization: YOLOv3 performs bounding box regression to accurately localize objects.
Object Recognition: YOLOv3 predicts class probabilities for each detected bounding box. These probabilities indicate the likelihood of the object belonging to different predefined classes. The model assigns the class label with the highest probability to each detected object.

3.1.3.11 Preliminary Results

The Privacy-Aware-Object-Recognition analytic has been implemented to take images and video streams as inputs to analyze. YOLOv3 model has been trained using the COCO dataset to learn and recognize objects across various categories such as people, animals, vehicles, and everyday objects. The training process involves optimizing the model's parameters using the COCO dataset's labeled images and annotations. YOLOv3 is fast and accurate in terms of mean average precision (mAP) and intersection over union (IOU) values as well, it achieves an (AP) of around 28-33% at an (IoU) threshold of 0.5 and above. Therefore, on average, YOLOv3 correctly detects and localizes objects in the COCO dataset with an overlap of 50% or more between the predicted bounding box and the ground truth bounding box. Figure 8 shows the results obtained by the Object Detection, Localization, and Recognition components on a sample original image with the probability of the recognized object class using the YOLOv3 classifier. While Figure 9 and Figure 10 show the results obtained by applying different degrees of privacy using the epsilon and sensitivity parameters.

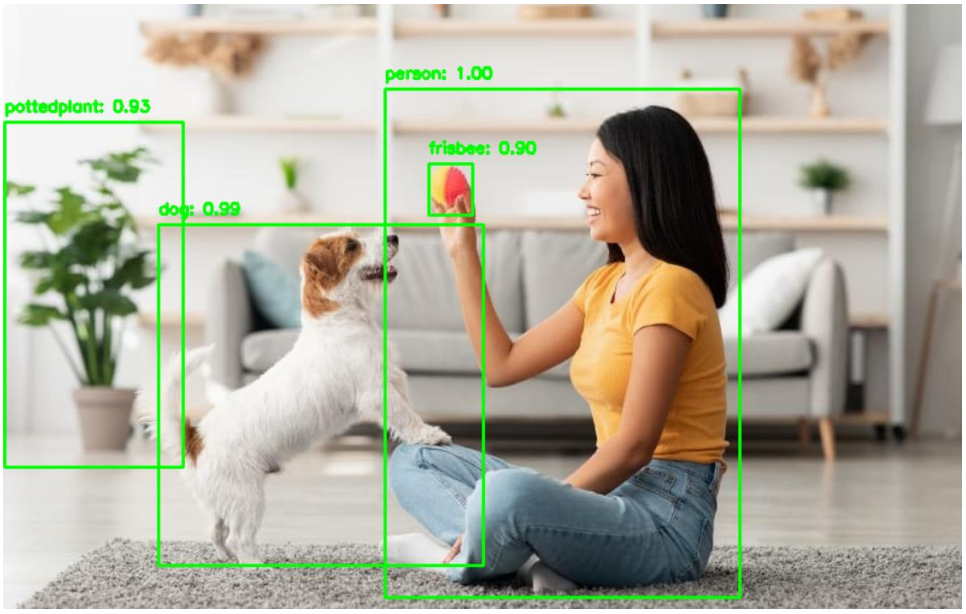

Figure 8: Privacy-Preserving Object Recognition of original image

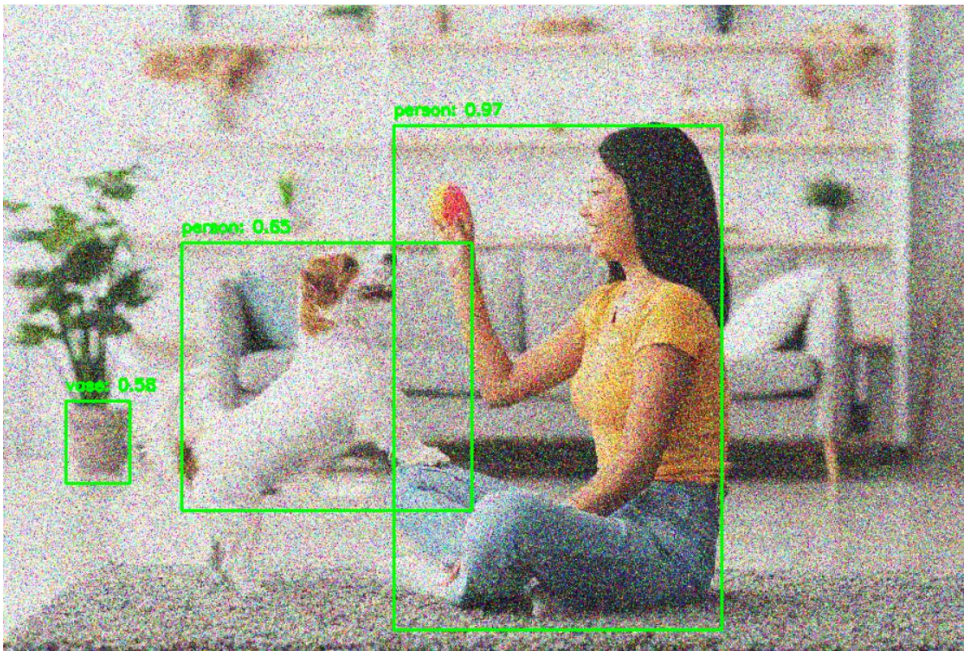

Figure 9: Privacy-Preserving Object Recognition of an anonymized image with low privacy

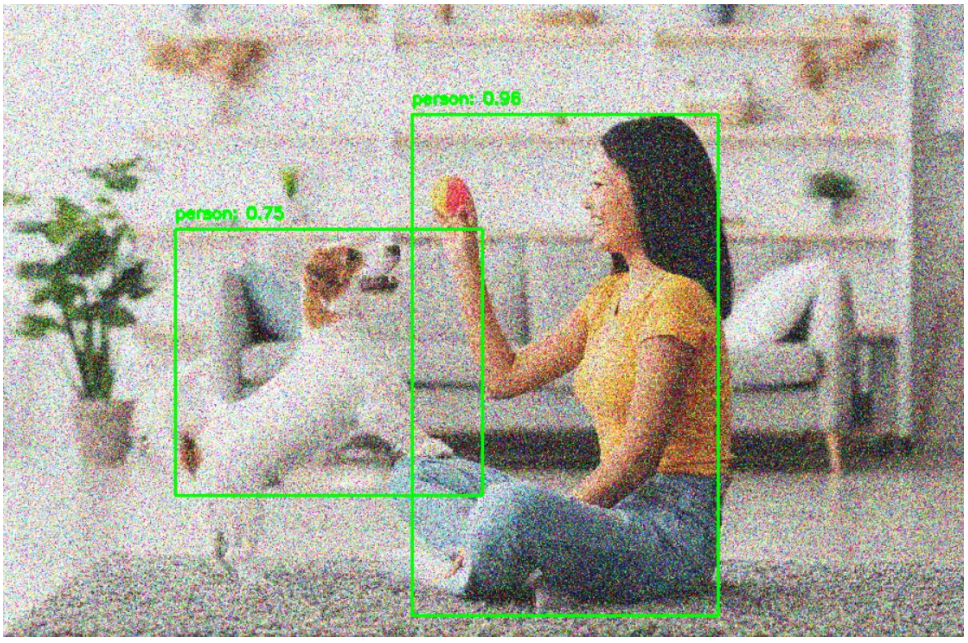

Figure 10: Privacy-Preserving Object Recognition of an anonymized image with medium privacy

3.1.4 Multilevel Anomaly Detection

3.1.4.1 Aim of the Analytics

The primary objective of the system is to identify and detect anomalies at multiple levels within an IoT infrastructure, including the kernel level, network level, application level, and DHT (Distributed Hash Table) level. By doing so, the system aims to enhance the security, reliability, and performance of IoT networks, ensuring their efficient and uninterrupted operation. By actively monitoring anomalies at

multiple levels, the system aims to strengthen the security of the networks. It detects and alerts for suspicious activities, unauthorized access attempts, and abnormal behaviors that may indicate potential security breaches or cyber-attacks. Timely detection enables the system to initiate appropriate security measures, such as isolating affected devices or blocking malicious traffic, thereby safeguarding the IoT ecosystem against threats. Overall, the multi-level anomaly detection analytics system strives to enhance the security, reliability, and performance of IoT networks. By actively identifying and detecting anomalies at the kernel, network, application, and DHT levels, the system ensures efficient and uninterrupted operation, safeguarding the integrity and functionality of the IoT infrastructure.

3.1.4.2 Requirements related to the analytics

The analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-19: The SIFIS-Home system shall detect, identify and disconnect infected devices. This analytics allows the identification of infected devices
- Non-Functional Requirements:
  - PE-10: An abnormal (suspicious) behavior caused by a malware shall be identified and notified within 60 seconds.
  - US-20: The Multi-Level Anomaly Detection system (MLADS) must monitor network traffic provided by several input sources and several locations.
  - US-21: The workload of the devices should be available to the MLADS
  - US-22: The list of applications running on each device should be available to MLADS
  - US-23: Raw sensor data must be available to be analysed by MLADS
- Security Requirements:
  - SE-17: Anomalous device behaviours should be identified and signalled in less than 60 seconds.
  - SE-26: The SIFIS-Home architecture shall be resilient to device compromising attacks.

3.1.4.3 Input Data

The input data for the analytic at each level of the multi-level anomaly detection system can be described as follows:

- Kernel Level: the input data for the analytic typically includes system calls. It includes the total number of times in a defined interval of seconds, a specific syscall has been called.
- Network Level: the input data consists of network traffic data captured from IoT devices, gateways, or network monitoring tools. This data includes packet-level information such as source and destination IP addresses, protocols used (e.g., TCP, UDP), port numbers, packet sizes, and timestamps.
- DHT Level: the input data includes information related to the distributed storage and lookup mechanisms used in IoT networks. It includes the total number of times in a defined interval of seconds, a PUSH, PUT, or DELETE has been requested on a specific topic.
- Application Level: At the application level, the input data includes logs, events, and metrics generated by IoT applications or services running on the devices or within the IoT network.

3.1.4.4 Analytic Design

Monitors are responsible for collecting and monitoring data at each level of the IoT infrastructure, including the kernel level, network level, application level, and DHT level. They gather relevant information such as system events, network traffic, application logs, and DHT-related data. Each monitor operates independently and continuously collects data from its respective level.

System Protection Manager: serves as a centralized component that acts as an intermediary between the

monitors and the analytics. Its primary role is to receive the data collected by the monitors and manage the communication between the components. The System Protection Manager is responsible for forwarding the collected data from each monitor to the analytics for further analysis.
Analytics: component performs the core anomaly detection and analysis tasks. It receives the data forwarded by the System Protection Manager and applies anomaly detection algorithms and techniques to identify abnormal patterns, behaviors, or events. The analytics component processes the data from different levels, detects anomalies, and generates alerts or notifications regarding potential security breaches or abnormal activities.

Once the analytics component detects anomalies and generates alerts or notifications, the System Protection Manager collects these results from the analytics through the DHT. It analyzes the severity and impact of the detected anomalies and decides on appropriate actions to be taken. These actions may include isolating affected devices, blocking suspicious traffic, triggering incident response protocols, or notifying system administrators for further investigation and remediation.

3.1.4.5 Training and Testing

Data Collection: The training process begins with the collection of labeled training data. For each level, data is gathered over a defined interval of seconds, capturing the total number of specific events or activities. At the kernel level, the system calls and their frequencies are recorded. At the DHT level, the requests for PUSH, PUT, or DELETE operations on specific topics are logged.

Feature Extraction: From the collected data, relevant features are extracted to represent the input for the machine learning algorithm. These features include the frequencies of specific system calls or DHT operations within the defined interval.

Labeling: The training data is labeled to indicate whether each data instance represents normal or anomalous behavior. This labeling is done manually.

Model Training: The machine learning algorithm is trained using the labeled training data and the extracted features. The algorithm learns to recognize patterns and relationships between the input features and the corresponding labels.

Model Evaluation: The trained model is evaluated using evaluation metrics such as accuracy, precision, recall, or F1-score to assess its performance. Cross-validation techniques can be applied to validate the model's generalization capabilities.

By utilizing a simple machine learning algorithm such as linear regression, the analytics can effectively recognize patterns and deviations, enhancing the anomaly detection capabilities within the IoT infrastructure.

3.1.4.6 Output data

The output of the analytic consists of the interested device or application, the type of analytic conducted, and a clear indication of whether an anomaly has been detected or not. This concise information assists in understanding the results of the analytic process and determining any necessary actions to address anomalies or ensure the normal functioning of the infrastructure.

3.1.4.7 Hardware Requirements

Devices running these analytics are required to have network connectivity and the capability of running minimal Linux distributions. Such devices can be based on either the x86 or ARM architecture.

3.1.4.8 Privacy Considerations

Only statistical values are collected and analyzed on the edge, hence no relevant privacy issues are identified.

3.1.4.9 Execution of the Analytics by the Data Analysis Toolbox

This analytics run in Docker and its integration has been executed following the approach described in Section 4. The analytics continuously run while the device is operating, hence it does not need to be invoked by the Data Analytics Toolbox.

3.1.4.10 Implementation Details

Each level presents a layer that catch the required data from the DHT and send these to the specific server which presents the data to the classifier. The response is given back to the system protection manager.

3.2 *Network Intrusion Detection*

The Network Intrusion Detection analytics exploit network flows and sensor data to detect intrusions in the smart-home network. We defined two distinct Network Intrusion Detection analytics, one performed on the edge and the other one in the cloud.

3.2.1 Netspot Network Anomaly Detection

The netspot solution is used to detect network anomalies. In the SIFIS-Home project, a new service application was created to control netspot applications and to forward data and alarm messages to appropriate applications.

3.2.1.1 Aim of the Analytics

The aim of the analytics is to detect anomalous activities in the network. For example, an exceptionally high ratio of packets conveying a certain flag (state of connection indicator), a sudden increase in the overall traffic within a time window.

3.2.1.2 Requirements related to the analytics

The analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-20: This system's purpose is to alert users if suspicious network traffic is detected, such as traffic generated by malware.
- Non-Functional Requirements:
  - PE-10: Suspicious network traffic is detected and notified within 60 seconds.
  - PE-11: Alerts are created within 5 seconds after suspicious traffic is detected.
- Security Requirements:
  - SE-22: Anonymized data can be used in this analytics.
  - SE-39: This system identifies denial-of-service attacks based on the activity within the network.

3.2.1.3 Input Data

The input data consists of a network packet exchanged between the connected devices. The packets are

captured, and relevant information parsed from them (port numbers, protocol identifiers, timestamps and IP address counts.

3.2.1.4 Analytic Design

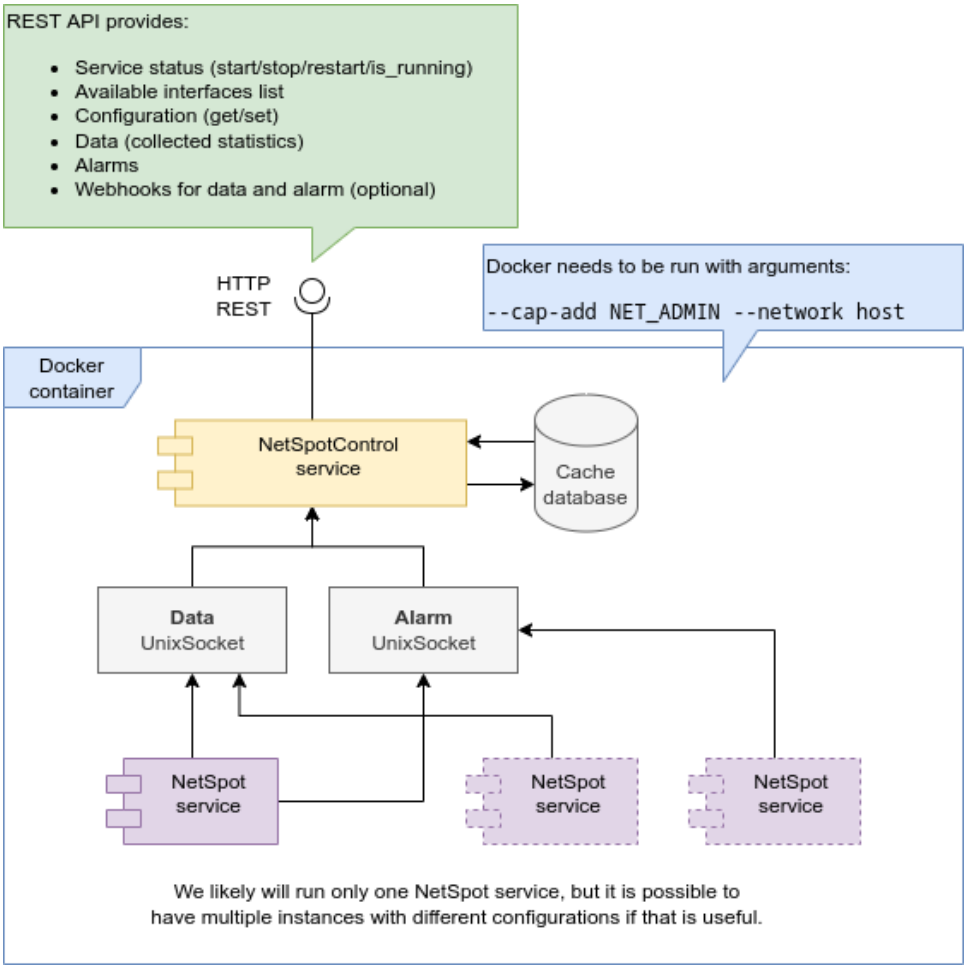

Figure 11: Netspot Docker Container

Control service and netspot instances are run in a docker container. The host system can use an argument when running the docker image to allow the container to use host system network interfaces. Using host system network interfaces is a critical part of allowing the netspot services to monitor network traffic. Docker container has a control service that provides HTTP API for configuration, controlling, and reading results.

3.2.1.5 Training and Testing

SPOT algorithm needs the initial batch of observations to form thresholds for the monitored statistics. The thresholds are dynamic, and they adapt to a subtle change in the network by updating them based on chosen number of new observations. However, if the initial thresholds are triggered, the values over or under the threshold are not used to update the thresholds.

For this test, SPOT was initialised by using 2000 observation from the normal traffic. After the thresholds were formed based on the observations, a denial-of-service attack was launched from the

Linux workstation. As a results, the statistics that monitor the overall traffic over a certain time window and the ratio of tcp packets with a set synchronize flag were triggered and alarms were raised.

3.2.1.6 Output data

Netspot outputs: raw statistics, thresholds (SPOT will compute one threshold for each monitored statistic) and the alarms (in case of triggered threshold). Netspot can also send these to an influx database

3.2.1.7 Hardware Requirements

Devices running these analytics are required to have network connectivity and the capability of running minimal Linux distributions. Such devices can be based on either the x86 or ARM architecture. Netspot and control services are lightweight and take less than 200 MiB of memory.

3.2.1.8 Privacy Considerations

Only statistical values are collected and analyzed on the edge. IP packet size and IP address, protocol and port counts are collected.

3.2.1.9 Execution of the Analytics by the Analytics Toolbox

Netspot anomaly detection solution is run in Docker and exposes the interface described in the following section. Hence, and its integration has been executed following the approach described in Section 4. To this aim, when the container is started, it is crucial to tell Docker to allow the container to use host system network interfaces with the parameter: `--network=host`

Setting this parameter is critical for the analytics to be effective in the SIFIS-Home framework, because otherwise, the solution will only monitor the container's virtual interface without proper network traffic.

3.2.1.10 Implementation Details

The control service was written in Rust language, and the netspot service is an open-source project written in Go language. Netspot version 2.1.2 was used in the docker image. In addition, Dockerfile was added to the project, which automatically compiles both required binaries. The binaries are compiled in development containers, and compiled binaries are then copied to the slim Debian docker image.
The control service provides the following endpoints:

3.2.1.10.1 Status

HTTP Method	**Endpoint**	**Parameters**	**Description**
GET	https://<ADDRESS>:<PORT>/v1/netspot/{id}/restart	Netspot configuration ID	Restart netspot configuration by ID
GET	https://<ADDRESS>:<PORT>/v1/netspot/{id}/start	Netspot configuration ID	Start netspot configuration by ID
GET	https://<ADDRESS>:<PORT>/v1/netspot/{id}/status	Netspot configuration ID	Status for the netspot configuration by ID

GET	https://<ADDRESS>:<PORT>/v1/netspot/{id}/stop	Netspot configuration ID	Stop netspot configuration by ID
GET	https://<ADDRESS>:<PORT>/v1/netspots/	-	Status of all netspot services
GET	https://<ADDRESS>:<PORT>/v1/netspots/restart	-	Restart all netspot services
GET	https://<ADDRESS>:<PORT>/v1/netspots/start	-	Start all netspot services
GET	https://<ADDRESS>:<PORT>/v1/netspots/stop	-	Stop all netspot services

3.2.1.10.2 Statistics

HTTP Method	Endpoint	Parameters	Description
GET	https://<ADDRESS>:<PORT>/v1/netspots/alarms	Optional *time* and number of *last* result parameters	Returns recorded alarms from nestpot statistics.
GET	https://<ADDRESS>:<PORT>/v1/netspots/data	Optional *time* and number of *last* result parameters	Returns recorded measurements from nestpot statistics.

3.2.1.10.3 Configuration

HTTP Method	Endpoint	Parameters	Description
POST	https://<ADDRESS>:<PORT>/v1/netspot	Configuration in JSON format	Create a new netspot configuration
GET	https://<ADDRESS>:<PORT>/v1/netspot/{id}	Netspot configuration ID	Get netspot configuration by ID
PUT	https://<ADDRESS>:<PORT>/v1/netspot/{id}	Netspot configuration ID and configuration in JSON format	Update an existing netspot configuration
DELETE	https://<ADDRESS>:<PORT>/v1/netspot/{id}	Netspot configuration ID	Delete netspot configuration by ID

3.2.1.10.4 Network

HTTP Method	Endpoint	Parameters	Description

GET	https://<ADDRESS>:<PORT> /v1/network/interfaces	-	Returns all available network interfaces on the host system in JSON format

3.2.1.10.5 Webhooks

HTTP Method	Endpoint	Parameters	Description
GET	https://<ADDRESS>:<PORT> /v1/netspot/ webhooks	-	List installed webhooks
POST	https://<ADDRESS>:<PORT> /v1//netspots/webhook	Webhook configuration in JSON format	Create a new webhook
GET	https://<ADDRESS>:<PORT> /v1/netspot/webhook/{id}	Webhook configuration ID	Get webhook configuration by ID
PUT	https://<ADDRESS>:<PORT> /v1/netspot/webhook/{id}	Webhook configuration ID and webhook configuration in JSON format	Update webhook configuration
DELETE	https://<ADDRESS>:<PORT> /v1/netspot/webhook/{id}	Webhook configuration ID	Delete webhook configuration by ID

3.2.1.10.6 Testing

HTTP Method	Endpoint	Parameters	Description
GET	https://<ADDRESS>:<PORT> /v1/netspots/test/alarm	Test alarm message in JSON format	This endpoint allows developers to send test alarm messages.

3.2.1.11 Preliminary Results

Preliminary tests indicate that changes in the monitored statistic ratios, e.g., high ratio of Address Resolution Protocol (ARP) packets due to Mirai and excessive traffic (DoS) in the network results in statistics that trigger alerts, according to what was monitored during the tests. The netspot control service API was tested successfully to configure and run netspot instances.

3.3 *Privacy Aware Speech Recognition*

3.3.1 Privacy Aware Speech Recognition and Voice Anonymization

3.3.1.1 Aim of the Analytics

The Privacy-Aware Speech Recognition (PSR) model is designed to accurately convert spoken language into written text while prioritizing privacy. This analytic utilizes computational linguistics to analyze audio signals and generate a verbatim and editable transcription of the spoken content. Importantly, any sensitive information within the audio is anonymized to protect privacy. Additionally, the PSR system allows for the generation of privacy-preserving versions of the original audio. By converting the anonymized text back into speech through text-to-speech translation, an audio output is

created that maintains privacy while still conveying the intended message. These privacy-preserving transcriptions and audio can then be securely shared with external services, as they do not disclose any sensitive information.

3.3.1.2 Requirements related to the analytics

The Privacy-Aware Speech Recognition analytics is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-06: The SIFIS-Home system shall provide Automatic Speech Recognition (ASR) to provide resident users and administrators the facility to control their home appliances through their speech.
  - F-07: The analytic provides an Automatic Speech Recognition that can be used to control the home appliances in the smart home;
  -
- Non-Functional Requirements:
  - PE-06: The analytic performs an audio transcription in less than 2 seconds.
  - PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection.
  - DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.
- Security Requirements:
  - SE-50: The analytic applies an anonymization of the voice timbre in order to make the identity of the speaker not identifiable;
  - SE-52: The analytic provides the possibility to anonymize the sensitive information from the audio and textual audio translation.

3.3.1.3 Input Data

The Privacy-Aware Speech Recognition system requires an audio sample containing voice as its input data for translation. The analytic is designed to process WAV audio samples with specific requirements, including a sampling rate of 16000Hz, a single channel (mono) representation, and a 16-bit format. If the audio sample is in a different format, a preprocessing step may be necessary to adjust it to meet the input requirements of the analytic.

In addition to the audio sample, the analytic also defines a list of textual entities to be anonymized from the audio. These textual entities are predefined and managed by the analytic, and you can refer to Table 1 for a comprehensive list of these entities.

Table 1: Textual Entities managed by the analytic

Entity	Description
PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FAC	Buildings, airports, highways, bridges, etc
ORG	Companies, agencies, institutions, etc
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc

WORK OF ART	Titles of books, songs, etc
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods
TIME	Times smaller than a day
PERCENT	Percentage, including "%"
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	"first", "second", etc
CARDINAL	Numerals that do not fall under another type

3.3.1.4 Analytic Design

The pipeline of the analytic is composed of three different components, as shown in Figure 12.

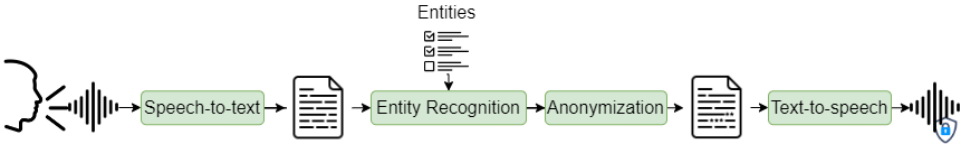

Figure 12: Pipeline of the privacy aware speech recognition analytics

Speech to Text: We employ the advanced Whisper automatic speech recognition (ASR) model [RKX22], developed and maintained by OpenAI, to convert spoken language into written text. Whisper stands out as a highly precise and efficient model that harnesses cutting-edge deep learning techniques, specifically the transformer architecture presented in Figure 13. This state-of-the-art approach has transformed numerous natural language processing (NLP) tasks, including speech recognition. By leveraging the transformer architecture, the Whisper model excels in handling the complexities of speech recognition tasks. It can capture contextual information, recognize patterns, and generate accurate transcriptions by effectively modeling the relationships between different elements of the audio sequence.

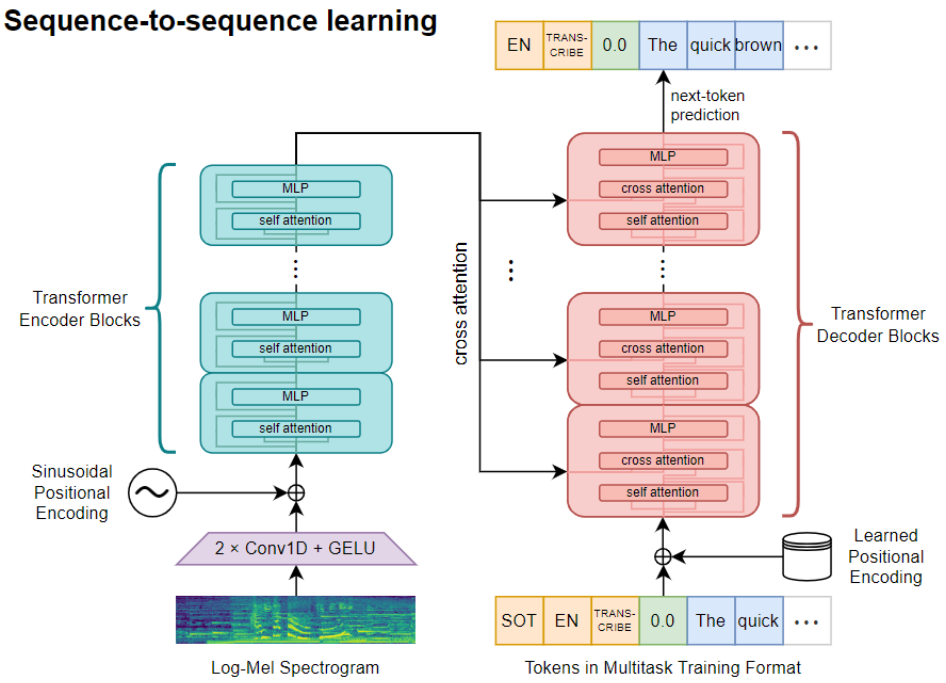

Figure 13: OpenAI Whisper Architecture: A Transformer sequence-to-sequence model [RKX22]

Named Entity Recognition: is concerned with locating key phrases and nouns in texts as entities, and these entities fall under several categories, i.e., names, locations, and addresses. The sensitivity of these entities depends on the context where the data analysis is applied. For example, names of people and locations are highly sensitive when performing data analysis and processing. However, to protect the privacy of the user, these entities can be removed from the text. Thus, still providing data valid for analysis, but without violating privacy. We used SpaCy deep learning model to perform entity recognition on the text recognized from the previous step [MHH21]. The process for Named Entity Recognition is composed of the below steps:

Sentence Segmentation: to split the text into sentences.

Tokenization: to split each sentence resulting from the previous step into tokens which are usually numbers, words, and punctuation marks.

Tokens Classification: each token is classified according to its part-of-speech (POS) as in Table 2.

Entity Detection: classifies the word entities according to their type as an address, a time, a location, a name, etc.

Table 2: Parts of Speech Description

POS	Description
NN	singular or plural noun
DT	determiner
VB	verb, base form
VBD	past tense verb
IN	preposition or subordinating conjunction
VBZ	verb, third-person singular present
NNP	singular proper noun
“TO”	Word “TO”
JJ	adjective

Text to Speech: To convert anonymized text resulting from the speech recognition and sensitive entities elimination into anonymized audio files, we use Google Text-to-Speech (gTTS) which is an interface with Google Translate's text-to-speech API. It takes a text with various and unlimited lengths as inputs and converts them into voice outputs with human-like reading and intonation, in addition to precise pronunciation corrections. gTTS uses a wide range of voices, so that the voice of the original speaker can be replaced with any of these voices to keep the identity of the speaker anonymous.

3.3.1.5 Training and Testing

Dataset

The Whisper model was trained using a large dataset consisting of 680,000 hours of multilingual and multitask supervised data sourced from the web. Among these hours of audio, 117,000 hours were dedicated to covering 96 different languages. Additionally, the dataset included 125,000 hours of translation data from various languages to English.

3.3.1.6 Output data

The Privacy Aware Speech Recognition can provide two different outputs described as following:

- **Anonymized textual translation**: It represents the audio input translation in textual format after the anonymization of the sensitive information specified in the user preferences. In the output translation, all the sensitive textual entities detected by the analytic are replaced by default keywords which preserve the essence of the word without revealing the exact word.
- **Anonymized voice information**: It represents the audio reconstruction of the anonymized textual translation.

3.3.1.7 Hardware Requirements

The recommended Hardware Requirements for running this analytic are as follows:

- GPU: Models such as NVIDIA Tesla V100 or NVIDIA GeForce RTX series GPUs are commonly used for deep learning tasks.
- CPU: Processors with multiple cores, such as Intel Core i7 or AMD Ryzen series CPUs, are typically suitable.
- RAM: A minimum of 16GB of RAM is recommended.
- Storage: Solid State Drives (SSDs) are preferred over Hard Disk Drives (HDDs) due to their faster read/write speeds, which can significantly improve overall performance.

3.3.1.8 Privacy Considerations

The Privacy-Preserving Speech analytic focuses on processing voice samples as input data, particularly in the context of the SIFIS-home scenario where the samples are captured by sensors deployed in a Smart home. It is crucial to handle such data with care as it may contain sensitive and personal information that can potentially identify the speaker. Adhering to the GDPR regulations, it is essential to ensure compliance and prevent the unauthorized transfer of transcriptions to third-party applications. Therefore, an important objective of this analytic is to protect personal data and restrict its dissemination beyond the SIFIS-Home context. To achieve this, the analytic employs techniques such as replacing sensitive information with default keywords. For example, names like "John" are replaced with the keyword "Private Data". This anonymization process safeguards the privacy of individuals by

preventing direct identification from the transcriptions. Additionally, to further enhance privacy protection, the analytic replaces the original speaker's voice with an alternative voice when generating the audio file of the anonymized text. This step ensures an additional layer of privacy for the speaker.

3.3.1.9 Execution of the Analytics by the Analytics Toolbox

This analytic is invoked through physical analysis and multi-level analysis APIs to analyze audio data. The data are passed to the Speech-to-Text component, then the privacy-preserving technique is applied to the text using the Named Entity Recognition Component, and the Google Text-to-Speech is used to generate an anonymized audio file. Finally, the analysis results are interpreted, aggregated, and presented to the user.

3.3.1.10 Implementation Details

The Python implementation of the Speech-to-Text functionality utilizes the OpenAI Whisper model [Whisper] and relies on various libraries, including NumPy, PyTorch, TorchAudio, SoundFile, Librosa, and FFmpeg. These libraries provide essential tools for audio processing, deep learning, and handling multimedia data. For Entity Recognition, the implementation leverages the SpaCy library [SpaCy], specifically using the "en_core_web_sm" pipeline. This pipeline is trained on written web text from sources such as blogs, news articles, and comments, enabling it to recognize vocabulary, syntax, and entities in English text. To enable Text-to-Speech functionality, the gTTS (Google Text-to-Speech) Python library and CLI tool is employed [gTTS]. This library serves as an interface to the Google Translate text-to-speech API, allowing the generation of synthesized speech from textual input.

3.3.1.11 Preliminary Results

The performance of the Privacy-Aware-Speech recognition analytic using is measured by the *Word Error Rate (WER)* for Whisper model. The WER uses the *Levenshtein distance* metric to measure the speech recognition accuracy by comparing the original and predicted transcriptions. For Whisper, the WER is 4.2% for English language, indicating a high level of accuracy. In general, spaCy's English NER models perform well in identifying and classifying common named entities, such as person names, organizations, locations, dates, and monetary values. They also handle other entity types, such as product names or geopolitical entities, with reasonable accuracy. It achieves more than 85% average accuracy. The preliminary results of the analytic are represented in the implementation of an API that interacts with the user recording an audio file and returning the textual translation with pre-defined sensitive information anonymized. It is composed of a preliminary activity in which the entities to be removed are defined and the second is the transcription of the audio, then SpaCy library is used to detect the entities defined in the first step and remove them form the text. As a result, the anonymized text and the anonymized audio that is created using the Google Text-to-Speech (gTTS) model are shared back with the user. A sample final result is shown in Table 3.

Table 3: Recognition and anonymization sample

Speech Data	ruth sat quite still for a time with face intent and flushed it was out now
Recognized Text	ruth sat quite still for a time with face intent and flushed it was out now
Anonymized Text	Private Data sat quite still for a time with face intent and flushed it was out now

3.3.2 Voice Recognition and Verification

3.3.2.1 Aim of the Analytics

Speaker verification is a process that involves authenticating individuals based on the unique biometric

aspect of their voice. This approach offers a non-intrusive and secure method for identity verification. Within the SIFIS-home environment, accurately identifying individuals is crucial for granting appropriate privileges based on predefined policies. To achieve this, a speaker verification system analyzes the voices of shared audio files and communicates the identified individuals to the smart-home components to grant or revoke access.

3.3.2.2 Requirements related to the analytics

The Voice Recognition and Verification is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-1: this analytic satisfies the functional requirement F-1 by identifying the resident users and administrators inside the smart home through biometrics.
  - F-3: this analytic satisfies the functional requirement F-3 by matching read biometrics against a database of stored ones.
- Non-Functional Requirements:
  - PE-2: The user verification shall happen in less than 5s.
  - PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection.
  - DE-06: the identification through biometrics should be performed correctly in more than 95% of cases.
  - DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.
- Security Requirements:
  - SE-03: Personal data stored must be encrypted.

3.3.2.3 Input Data

The Privacy-Aware Speaker Recognition and Verification system requires two audio files containing voice as input data for verification. The analytic is designed to process WAV or FLAC audio samples with specific requirements, including a sampling rate of 16 kHz or 8 kHz, a single channel (mono) audio, and the duration of the audio segment should be within a certain range, typically a few seconds. If the audio sample is in a different format, a preprocessing step may be necessary to adjust it to meet the input requirements of the analytic.

3.3.2.4 Analytic Design

For speaker verification, we use ECAPA-TDN model with the architecture shown in Figure 14. This model was proposed in [DTD20] and developed as part of the SpeechBrain AI toolkit [RPP21]. ECAPA-TDNN model employs ECAPA Time Delay Neural Networks (TDNNs) derived embeddings, and it consists of an input layer, followed by a convolutional block with ReLU activation and batch normalization. Then, a sequence of three Squeeze-and-Excitation and residual blocks. Next, a convolutional block with ReLU activation. Followed by a layer that applies statistics pooling to project variable-length utterances into fixed-length speaker characterizing embeddings with batch normalization. Then a fully connected dense layer with batch normalization, and an Additive Angular Margin (AAM) Softmax layer. Finally, an output layer to classify the inputs as yes or no for verification results.

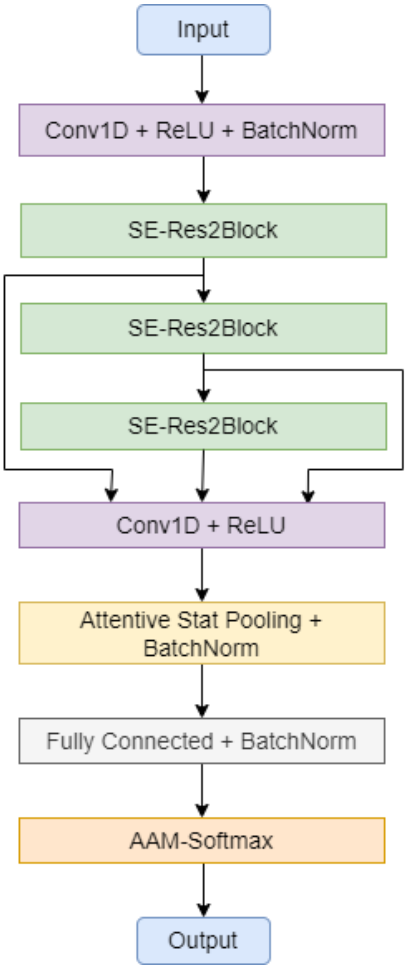

Figure 14: ECAPA-TDNN Model for Speaker verification

3.3.2.5 Training and Testing

Dataset

The ECAPA-TDN model was trained on the VoxCeleb2 dataset, which is a comprehensive and extensive dataset specifically curated for speaker recognition and diarization tasks. VoxCeleb2 serves as an expansion of the original VoxCeleb dataset, offering a significantly larger and more diverse collection of speaker data with over 1 million utterances from more than 6,000 speakers. These recordings are sourced from a wide array of contexts, including interviews, speeches, YouTube videos, and public appearances. This diverse collection ensures that the dataset captures a wide range of speaker variations and real-world scenarios, making it an ideal resource for training and evaluating speaker recognition models.

3.3.2.6 Output data

The output of this analytic is a binary decision indicating whether the two input audio samples belong to the same speaker or not. It evaluates the similarity or dissimilarity between the input sample and the enrolled speaker's reference data in the other sample. The output is represented as a similarity metric using cosine similarity.

3.3.2.7 Hardware Requirements

The hardware requirements are as below:

- CPU: A multi-core processor, such as an Intel Core i5 or i7, or an equivalent AMD processor, is sufficient for running the analytic.
- GPU: For faster training and inference, a dedicated GPU is recommended. Models like ECAPA-TDNN can leverage the parallel processing capabilities of GPUs. NVIDIA GPUs, such as the GeForce RTX series or the NVIDIA Titan series, are commonly used for deep learning tasks.
- RAM: At least 8 GB of RAM is recommended, but larger models or datasets may require more RAM.

3.3.2.8 Privacy Considerations

Since applying privacy mechanisms would alter the speaker's voice in the audio files, the protection mechanisms that can be applied with this analytic include file encryption and employing secure protocols for transmitting voice data.

3.3.2.9 Execution of the Analytics by the Analytics Toolbox

This analytic is invoked through physical analysis and multi-level analysis APIs to analyze audio data. The data are passed to the analytics engine, which performs data processing using machine learning and deep learning toolboxes. Afterward, the analysis results are interpreted, aggregated, and presented to the user.

3.3.2.10 Implementation Details

The Python implementation of the Speaker Verification functionality utilizes the SpeechBrain ECAPA-TDNN model [ECAPA-TDNN] and relies on various libraries, including SpeechBrain, Torch, NumPy, SciPy, Matplotlib, and Pandas.

3.3.2.11 Preliminary Results

For this analytic, the weights of the pre-trained ECAPA-TDNN model, that has been trained on the VoxCeleb2 standard dataset and evaluated on the VoxCeleb1 test sets. The performance of the model is measured by the *Equal Error Rate (EER)*, which corresponds to the error rate value when the *False Acceptance Error Rate* is equal to the *False Rejection Error Rate*. The *False Acceptance Error Rate* is the rate of incorrectly accepted speaker speech segments to the total number of speech segments, while the *False Rejection Error Rate*. is the rate of incorrectly rejected speaker speech segments to the total number of speech segments. For ECAPA-TDNN model, the *EER* value is equal to 0.80% which represents a higher accuracy, since the error rate is very low. The model has been validated with the use of privacy mechanisms on 10,000 audio files of 5,000 pairs for matched speakers, and 10,000 audio files of 5,000 pairs for mismatched speakers. The results of matched speaker verification show that the model has predicted 99.3% of the pairs correctly with a total number of 4,964 pairs. For mismatched speakers verification, the model has predicted 96.4% of the pairs correctly as different speakers with a total number of 4,820 pairs.

3.3.3 Anomaly Detection in Audio Signal Analysis

3.3.3.1 Aim of the Analytics

One valuable source of information for detecting anomalies is the audio signals captured within the smart home. Audio anomaly detection in smart homes adds an extra layer of protection, enabling early

detection and response to potential threats, improving safety, and enhancing the overall quality of life for the occupants. Performing audio anomaly detection in smart homes enhances the overall safety and security of the occupants by identifying unusual or potentially dangerous events that may occur within the home environment. This includes detecting anomalies such as breaking glass, loud and sudden noises, unusual patterns of speech or conversation, or other signs of potential threats or emergencies. By continuously monitoring audio signals in the smart home, the system can quickly identify and raise an alert for any abnormal activities.

3.3.3.2 Requirements related to the analytics

The Anomaly Detection in Audio Signal Analytic is relevant for satisfying the following requirements defined in [D1.2]:

- Functional Requirements:
  - F-1: this analytic satisfies the functional requirement F-1 by identifying the resident users and administrators inside the smart home through biometrics.
  - F-3: this analytic satisfies the functional requirement F-3 by matching read biometrics against a database of stored ones.
  - F-15: this analytic satisfies the functional requirement F-15 by identifying anomalous behaviours inside the smart home, as it detects anomalous temperature or humidity readings of home devices.
- Non-Functional Requirements:
  - PE-24: the analytics results are presented in less than 30 seconds. The fulfillment of this requirement depends on the device computational power and on the time required for data collection.
  - US-26: The presence of a GPU is needed to perform DL-based analysis.
  - DE-06: this analytic can be installed (i.e., replicated) on distinct devices to avoid a single point of failure.
- Security Requirements:
  - SE-21: Differential Privacy and Audio Scrambling parameters are configurable for data analysis and can be changed.
  - SE-22: this analytic is able to work with anonymized data, but the accuracy of the results might be decreased.

3.3.3.3 Input Data

The input data for audio anomaly detection in smart homes is the audio signals captured within the smart home environment. These audio signals can be obtained from various sources such as microphones or audio sensors deployed throughout the home. This analytic is designed to process WAV or FLAC audio samples with specific requirements, including a sampling rate of 16 kHz, a single channel (mono) audio represented in 16 bit. There is no specific duration requirement for the input audio files, but longer audio files may take more time to process, and extremely short audio snippets might not provide sufficient information for accurate classification. If the audio sample is in a different format, a preprocessing step may be necessary to adjust it to meet the input requirements of the analytic.

3.3.3.4 Analytic Design

We employ the IBM MAX audio classification model developed and maintained by IBM [IBM-MAX]. This model is a multi-attention classifier designed to analyze and categorize audio data into various predefined classes or labels. The classifier leverages the power of deep neural networks to learn patterns and features from large-labeled audio datasets, allowing it to make predictions on new, unseen audio inputs. The core component of the IBM Audio Classifier is a deep neural network model. The architecture consists of multiple layers, such as convolutional layers, pooling, and fully connected

layers. These layers are designed to learn hierarchical representations and capture relevant audio features for classification. The model output includes the top 5 class predictions along with their corresponding probabilities. The model is designed to support 527 classes, defined within the Audioset Ontology.

3.3.3.5 Training and Testing

Dataset

The IBM MAX Audio Classifier model was trained on the "AudioSet" dataset. AudioSet is a large-scale dataset created by Google Research that consists of a collection of labelled audio clips from a wide variety of sources, including YouTube videos. The dataset covers a diverse range of audio categories, such as musical instruments, human sounds, animal sounds, and environmental sounds. AudioSet contains over 2 million 10-second audio clips that are labelled with one or more of 632 audio event classes. These event classes cover a broad range of sounds and provide a rich source of training data for audio classification models.

3.3.3.6 Output data

The analytic generates the top 5 predictions or classifications, providing insights into the most probable classes or categories present in the input audio. For the purpose of audio anomaly detection, we have curated a specific set of classes that are indicative of abnormal behavior, including events like glass breaking, screaming, and fire. By leveraging these classifications, we can identify potential anomalies and ass ess whether the audio exhibits any concerning patterns or events.

3.3.3.7 Hardware Requirements

The hardware requirements are as below:

- CPU: A multi-core processor, such as an Intel Core i5 or i7, or an equivalent AMD processor, is sufficient for running the analytic.
- GPU: For faster training and inference, a dedicated GPU is recommended. Models like ECAPA-TDNN can leverage the parallel processing capabilities of GPUs. NVIDIA GPUs, such as the GeForce RTX series or the NVIDIA Titan series, are commonly used for deep learning tasks.
- RAM: At least 8 GB of RAM is recommended, but larger models or datasets may require more RAM.

3.3.3.8 Privacy Considerations

Input data

In order to preserve the privacy of data, two data privacy mechanisms are applied. The first mechanism is Differential privacy [Dwo08], a powerful privacy-preserving technique widely used to add noise to data, including audio. We use it to add random Laplacian noise to the audio and ensure that privacy is protected while still allowing valuable insights to be extracted from the data. The amount of noise added is determined by the sensitivity and privacy budget parameters, where sensitivity measures the impact of input data changes on algorithm output, and the privacy budget controls the level of noise added to protect privacy. The second method is Audio Scrambling, in which the audio file is segmented into smaller segments of a fixed duration, shuffles their order, and then concatenates them back together to create the scrambled audio. This scrambling technique alters the original audio's temporal arrangement, making it difficult to understand the content

Results

The output of this analytic is the five top labels of sound classifications included in the audio files. These labels do not infer identifying information about the tenants.

3.3.3.9 Execution of the Analytics by the Analytics Toolbox

This analytic is invoked through physical analysis and multi-level analysis APIs to analyze audio data. The data are passed to the analytics engine, which performs data processing using machine learning and deep learning toolboxes. Afterward, the analysis results are interpreted, aggregated, and presented to the user.

3.3.3.10 Implementation Details

The Python implementation of the audio anomaly detection functionality utilizes the IBM MAX Audio Classifier model [IBM-MAX] and relies on various libraries, including soundfile, pydub, NumPy, Matplotlib, and Pandas.

3.3.3.11 Preliminary Results

The preliminary outcomes of the Privacy-Aware Anomaly Detection in Audio Signal Analytic are demonstrated through a Python application. This application takes an audio file as input and incorporates privacy mechanisms to introduce noise or scramble the audio. It then identifies the top five sound classes along with their corresponding probabilities. The IBM MAX Audio classifier is capable of recognizing 527 classes, including anomalous classes such as fire and glass breaking sounds. While the model exhibits higher accuracy in speech and music sound due to the training set bias, it also performs well across other classes. The classifier is trained on data from Audioset, which originates from YouTube videos. However, the model's utility is not limited to music and speech domains, as it can be effectively applied to a wide range of audio files. The accompanying test assets encompass diverse audio samples, ensuring the model's adaptability and effectiveness across various contexts.

4 Integration of Analytics within the Data Analysis Toolbox

An important activity that has been carried out in the second period of the project is the integration of all the analytics designed and developed in WP4 within the Data Analysis Toolbox component of the SIFIS-Home framework. Please refer to Deliverable D1.3 for a detailed description of the components building up the SIFIS-Home framework. This integration allows the other components of the SIFIS-Home framework to invoke the available analytics, and the get the results they produced through the DHT. For instance, The System Protection Manager component (part of the Secure LifeCycle Module) needs to receive the results computed by the analytics to take proper countermeasures to protect the Smart Home from intrusions.

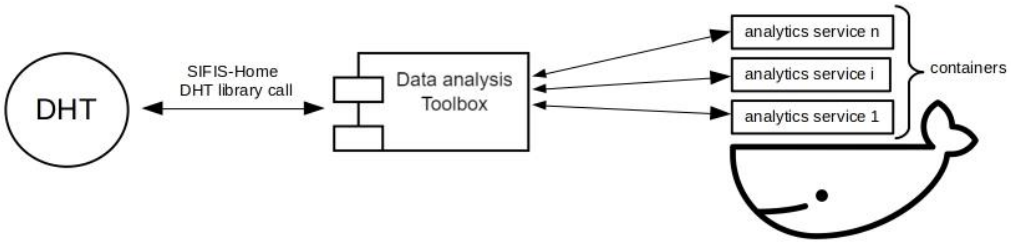

Figure 15: Analytics Integration within Data Analysis Toolbox

In particular, as shown in Figure 15: Analytics Integration within Data Analysis ToolboxFigure 15, the Data Analysis Toolbox is the component that directly interacts with the DHT, which is the main communication mean of the SIFIS-Home framework. This interaction occurs by exploiting the SIFIS-Home DHT library, which defines (among the other functionalities) the functionality to get messages from the DHT and publish messages on the DHT. The Figure 15: Analytics Integration within Data Analysis ToolboxData Analysis Toolbox component is activated by the other components of the framework, which request the execution of analytic functions on given sets of data. Figure 17 shows the transaction used to publish the image in Figure 16 to invoke the Privacy-Aware Object Recognition Analytic through the DHT, and Figure 18 shows the message published to the DHT as an object recognition analytic request including all the required parameters.

Figure 1616: Image published for Privacy-Aware Object Recognition Analysis

```
C:\Users\wisam\Desktop\Final_Integration\publish_video_data_object_recognition>docker run -it -v /var/run/docker.sock:/var/run/docker.sock
 --net=host publish_video_data_object_recognition python -m publish_video_data_object_recognition --file_name dog.jpg --file_path C:\Users
\wisam\Desktop\Final_Integration\my_data\dog.jpg --requestor_type NSSD --epsilon 0.03 --sensitivity 1.0
### Connection established ###
```

Figure 1717: Transaction to publish image to the DHT

```
Received command {"RequestPostTopicUUID": {"topic_name": "SIFIS:Privacy_Aware_Object_Recognition", "topic_uuid": "Object_Recognition", "va
lue": {"description": "Object Recognition", "requestor_id": "docker-desktop", "requestor_type": "NSSD", "request_id": "dockerdesktop202306
06113902.685272dd9d799b11aa621215b46ee152c2009d01064324cc3fa4cf473b7e5f78633433", "Type": "Video_file", "file_name": "dog.jpg", "file_path
": "C:\\Users\\wisam\\Desktop\\Final_Integration\\my_data\\dog.jpg", "epsilon": 0.03, "sensitivity": 1.0}}}
WebSocket RequestPostTopicUUID
Persistent message received SIFIS:Privacy_Aware_Object_Recognition Object_Recognition
Domo Event received
Persistent
```

Figure 1818: Message published to the DHT as an object recognition analytic request

For each analytics that has been defined in the project (see the list in Section 1), two DHT topics has been defined: one used by the other components of the SIFIS-Home framework for invoking the

analytics, and the other for publishing the results returned by the analytics. Each analytics runs in a dedicated docker container, and it is implemented as an HTTPS REST service. Hence, the Data Analysis Toolbox invokes the analytics service exploiting the interfaces it exposes.
The Data Analysis Toolbox receives the requests from the DHT, having subscribed to the related topics, and the parameters are included in the messages paired to the requests. Once a published message has been received by the DHT, the data analysis module related to the specified topic is called and executed on the data. For instance, For the Privacy-Aware Object Recognition Analytic, the first DHT topic is "SIFIS:Privacy_Aware_Object_Recognition" for invoking the analytic as depicted in Figure 10 and the second DHT topic is "SIFIS:Object_Recognition_Results" as depicted in Figure 13. The process starts by publishing the analytic request to the DHT on the first topic, the analytic toolbox continuously checks all published messages to the DHT as shown in Figure 19, and once there is a new message on the first topic "SIFIS:Privacy_Aware_Object_Recognition", it invokes the running analytic responsible for this topic through an HTTPS REST, which is the Privacy-Aware Object Recognition as shown in Figure 20, and finally the analysis results "bicycle, dog, truck" are published to the DHT as depicted in Figure 21.

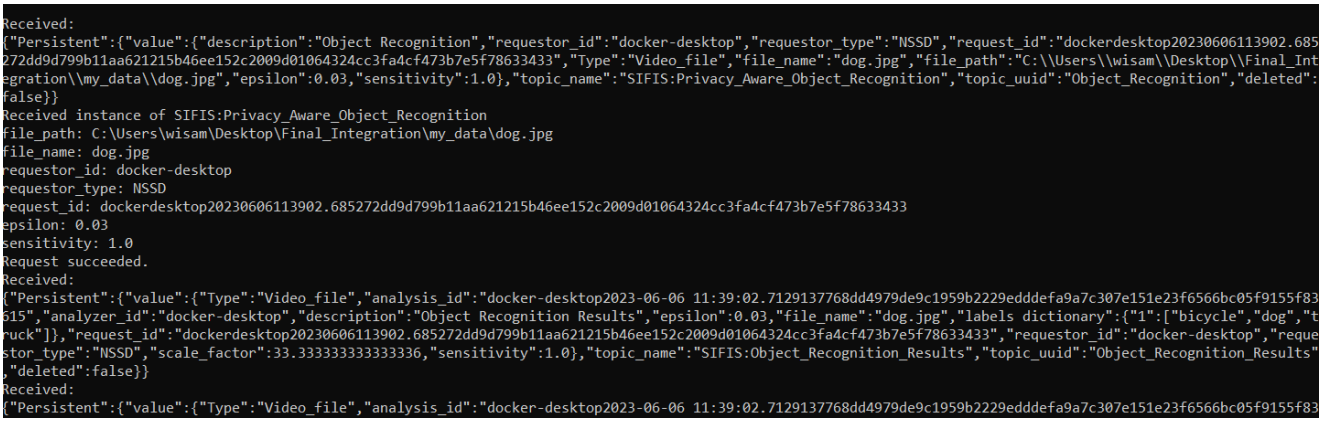

Figure 1919: Data Analysis Toolbox DHT messages check

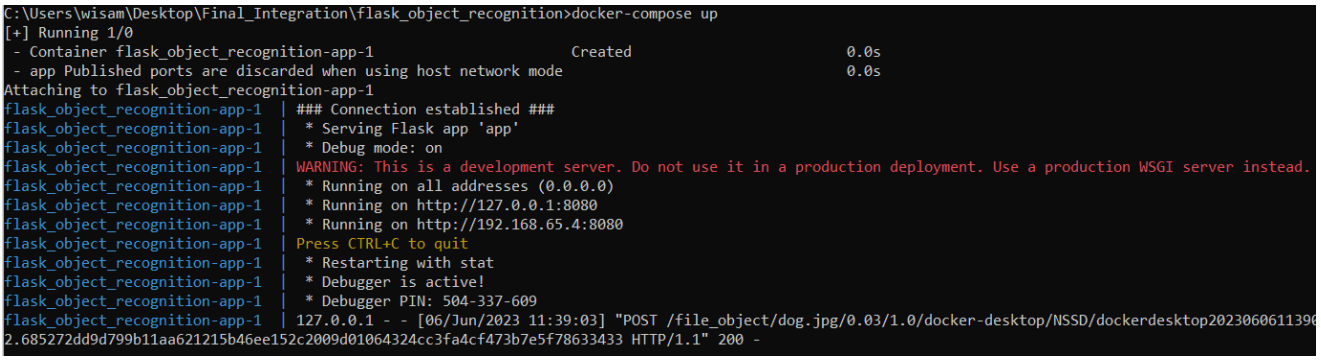

Figure 2020: Privacy-Aware Object Recognition Component running

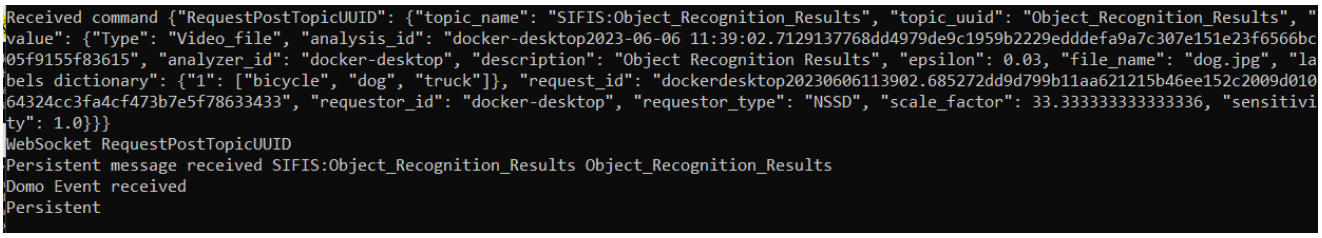

Figure 21: Message published to the DHT with the object recognition analysis results

5 Preserving Privacy of Data Exploited in Analytics

AI-based data analysis entail some concerns related to data privacy and also to transparency of the analysis process [Xu19,Stro19], and such concerns are relevant in particular when data analysis is performed by a third party with respect to the devices deployed in the smart home. In particular, data collected in smart homes could bring personal and privacy sensitive information, and their exposure to a third party might tamper the reputation of the residents in the smart home. To address this issue, Privacy-Preserving Machine Learning (PPML) [AC19] techniques are used to protect such data from being disclosed, while still allowing the execution of meaningful data analysis.

Recently, there has been also a new growing need for AI-based analytics: decision explainability. In fact, by following also recent directives, such as the EU proposal for the Artificial Intelligence Act[1], understanding decision-making criteria is important for both technical and ethical reasons.

As privacy requirements and their enforcement normally affects the effectiveness accuracy of a decision system, considering explainability adds another dimension, which might be in contrast with both accuracy of the decision and ensured privacy. Still, all these elements are of crucial importance as they pose the basis for the Trustworthy AI paradigm [Hag19].

Moreover, when analytics are executed by third parties, data produced by distinct smart homes could be elaborated together, in order to get better or earlier results. We call the analytics that consider data coming from several source collaborative analytics.

In light of the previous considerations, we defined a methodology for collaborative data analysis, to define the optimal trade-off between data utility, privacy, and explainability, applied it to face analysis. In particular, we exploited two common techniques for preserving data privacy, Differential Privacy and Atuoencoders (already described in D4.2) and one technique for ensuring model explainability (Saliency Mas enhanced with Smoothgrad) which can be tuned through input parameters, and we apply them to a face analysis problem. In this context, the proposed approach defines a measure for evaluating the trade-off among the data **Privacy Gain**, **Data Utility Loss**, and model **Explainability Gain** obtained by applying such techniques. This measure is meant to allow the tuning of the techniques and parameters for preserving data privacy and ensuring model explainability for a specific data analysis problem, in order to define the configuration which maximizes the Privacy Gain and the model explainability while minimizing the Data Utility Loss.

The methodology defines thus a general trade-off measure and exploits novelly designed tri-dimensional compatibility matrices to find the configuration yielding the best trade-off value.

Furthermore, given a set of requirements on minimum Privacy Degree and explainability level, which are provided by the participants to collaborative analysis, the proposed framework will calculate the optimal solution satisfying these requirements.

We apply or methodology to facial expression recognition model. The model classifies individual faces into different emotion categories, such as happy, sad, or angry, based on the expression detected in an image or a video frame. To preserve privacy, for example, a face can be anonymized so that the identity of this person is kept private but the facial expression can still be predicted, so the image attributes that are responsible to reveal the identity are hidden.

Both privacy and explainability are extremely relevant in this problem, as privacy of classified faces might be a requirement, and it is interesting to understand which physical features are actually relevant for the classification decision. For our experiments, we will refer in the following to the Facial Expression Recognition (FER) dataset[2]

[1] Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence: https://bit.ly/3y5wf6e

[2] https://www.kaggle.com/msambare/fer2013

5.1 *Privacy-preserving Techniques*

Data privacy-preserving techniques are aimed at protecting dataset attributes that are considered sensitive or may lead to the re-identification of the person the data refers to. There are two main approaches for data privacy protection: the first one is related to hiding sensitive attributes to prevent person re-identification, and the second is about delivering models that perform responsible data analysis by learning general patterns instead of memorizing specific sensitive attributes and data instances. To protect data privacy when being shared or analyzed by third parties, anonymization-based techniques are exploited. The most popular ones are k-anonymity [Swee02], l-diversity with all its variants [MKG07], t-closeness [LT07,RTG00], (ϵ, δ)- differential privacy [Dwo08] and autoencoders [PVG20]. In this work, we will mainly use the last two techniques, which are powerful and effective especially when working with image datasets. We detail them in the following.

5.1.1 (ε, δ)-differential privacy

(ϵ, δ)-differential privacy (DP) is considered one of the most powerful privacy-preserving techniques. DP technique differs from the traditional methods in the mechanism it uses to add noise to the data either before or during analysis, which makes it invulnerable to re-identification or data reconstruction attacks. As a matter of fact, traditional methods add noise to individual records, while the DP technique adds Laplace or Gaussian distributions noise during the analysis phase and to the learning model used. When the DP technique is applied to two neighboring datasets (i.e., datasets differing by one data instance), the outputs of the same data analysis on the two datasets are indistinguishable, thus don't disclose whether the given data instance was included in the original dataset or not. The degree to which these outcomes are indistinguishable depends on the values of the privacy budget parameter and the sensitivity parameter, which measures the algorithm sensitivity to the insertion or removal of an individual item from the dataset and is explained in detail in [Dwo08]. (ϵ, δ)-differential privacy equation is presented in Equation 1 [Dwo08].

$$Pr[M(D_1) \in S] \leq Pr[M(D_2) \in S] \times exp(\epsilon) + \delta$$

Equation 1

Where Pr is the probability of the event in square brackets, ϵ is the privacy budget that is used in our approach (as will be explained in section 4.1), δ is the failure probability, M is the randomized algorithm that is said to provide (ϵ, δ)- differential privacy, and D1 and D2 are two datasets that differ in at least one data instance, and $S \subseteq Range(M)$.

For privacy analysis, in this work, we use the moments accountant privacy budget tracking method proposed by Abadi et al. [ACG16], which uses a Differential Private Stochastic Gradient Descent (DP-SGD) algorithm with an additive Sampled Gaussian Mechanism (SGM). SGM is an additive Gaussian noise [DKM06] and Sampling [RSL08,Smi09] mechanism used in differential privacy as defined in Equation 2 for a real-valued function f mapping subsets of D to R^d:

$$M(D) \triangleq f(D) + N(0, S_f \sigma^2)$$

Equation 2

where D is the dataset and a subset of its elements are sampled randomly and independently from each other with sampling rate $0 < q \leq 1$ to be used by the algorithm f. $N(0, \sigma^2)$ is the Gaussian distribution of the noise added with a mean equals to 0, and σ is the noise added with $S_f \sigma^2$ standard deviation of the noise bounded to ℓ_2 sensitivity.

The moments accountant method exploits the Composability and Group privacy attributes of DP for several applications of the Gaussian mechanism on random samples of the dataset, and accumulates the overall privacy budget for these executions using the privacy accountant concept introduced in [Mcs09]

by implementing the accountant procedure at each execution of the Gaussian mechanism on a sample data according to the sampling ratio and performs privacy budget accumulation at the end of all procedure executions.
Using moments accountant, the accounting procedure allows proving that an algorithm is (ϵ, δ)-differentially private for appropriately selected configurations of the parameters for any $\varepsilon < c_1 q^2 T$ and for any $\delta > 0$ if the noise multiplier σ was defined to be as in Equation 3 proposed in [ACG16]:

$$\sigma \geq c_2 \frac{q\sqrt{Tlog(1/\delta)}}{\epsilon}$$

Equation 3

where c_1 and c_2 are constants so that given the sampling probability $q = L/n$, L is the sampling ratio of each Lot, n is the size of the dataset, and T is the number of training steps and $T = E$ and E is the number of Epochs.
The relationship between the noise multiplier and the privacy budget ϵ is negative, which implies better privacy protection when increasing the value of the noise multiplier. TensorFlow Privacy provides an implementation of the DP privacy accountant method for SGM[5] and a documentation for the privacy framework.
There are three ways to add noise to machine learning models to satisfy the differential privacy property, the first one is to add the noise to the objective function, the second is to add noise to the gradients at each iteration of the training phase, and the third is to add noise to the output of the training phase [JE19].

5.1.2 Autoencoders

This technique is a type of neural network that is trained to compress data and represent their most important features as a latent representation code. The technique involves two components: an encoder and a decoder. The encoder takes an input dataset and encodes it in a latent representation (code), to get its important features represented as a code. Then, the decoder takes the latent representation and decodes it to get the reconstructed image. A typical application is the dimensionality reduction of the feature space, but it is widely used for learning data generative models as well. The Utility Loss in autoencoders is represented as the distance between the original image and the reconstructed one. The anonymization degree is controlled by the code size (i.e., the latent representation: the lower the code size, the higher anonymization we get). The underlying mechanism can be considered similar to the compression one [HAK20,MCC19]. Autoencoders are used as a privacy-preserving technique themselves [DJD18,LLL19], but they may be fine-tuned or combined with other privacy-preserving methods [GAS22].

5.1.3 Explainable Artificial Intelligence

Explainable/Explicable Artificial Intelligence (XAI) is a recent term that has been proposed to enforce safety, fairness, trust and transparency in AI models. AI algorithms and processes have their own drawbacks of being complex to trace or understand in terms of the way they process data entries to produce results. Sometimes they may be charged with causing discrimination or inaccurate results, or not being transparent in the decision-making process, requiring the user to have blind faith in their predictions. XAI provides insights into how the predictive model works and the correlations among data sources and features, and is mainly used for extracting knowledge about the model and the data.
Hence, XAI can be exploited for discrimination avoidance, and for granting users the right to get an explanation of why certain decisions have been made by AI models. On the other hand, XAI represents a threat to data privacy and gives the opportunity for exploration and exploitation of the model by

possible malicious users. For instance, when the users understand how the model works, they can introduce designed adversarial inputs to produce specific outputs for manipulation and deception. In our work, we used Saliency Maps enhanced with the SmoothGrad technique for addressing explainability.

5.2 *Saliency Maps*

The Saliency Maps concept was first proposed in \cite{simonyan2013deep} as a gradient-based method to explain deep neural networks. The generated maps using this technique make the pixels of the input image that have the highest gradient, i.e., the most influence on the classification of the image, more visible in the image. These gradients are computed using two alternative methods which differ in the scope (local vs global):

Image-specific class saliency visualization method (local scope): an approximation class score is being calculated in this technique using Equation 4 where $S_l(d)$ is the class scoring function, d represents the input image, l is the label class, w_l is the weight vector, and b_l is the model bias.

$$S_l(d) = w_l^T d + b_l$$

Equation 4

In the case of a Convolutional Neural Network (CNN) model, the equation is updated to be as in Equation 5, where w is the derivative of S_l with respect to the image d at the point d_0 computed using Equation 6:

$$S_l(d) \approx w^T d + b$$

Equation 5

$$w = \frac{\partial S_l}{\partial d}|_{d_0}$$

Equation 6

Class model visualizations (global scope): generate numerically computed images representing the appearance of classes for a given CNN model and a learned classification by this model. It uses a regularisation parameter Ω to generate an L_2-regularised image for a specific class l with respect to a specific ConvNet layer for an image d, given a high score value for the class scoring function S_c by Equation 7:

$$arg\,max_d S_l(d) - \Omega\|d\|_2^2$$

Equation 7

5.3 *SmoothGrad*

This technique smooths the saliency maps produced by gradient-based models through the elimination of noise in these maps. Its main concept is about reducing the noise in generated explanations by adding Gaussian noise, which is the standard deviation of the Gaussian perturbations responsible for saliency maps sharpening in order to generate better and more accurate explanations of the model predictions. The explanation quality can be controlled by regulating the introduced Gaussian noise, as illustrated in Figure 22 [STK17], unlike other explainability mechanism. The higher the value of Gaussian noise, the better the explanation, as it appears in the rightmost column of the image where a 75% noise value is added and resulted in a better explanations than the first column, where 0% noise is added. According to [STK17], the ideal noise level depends on the input.

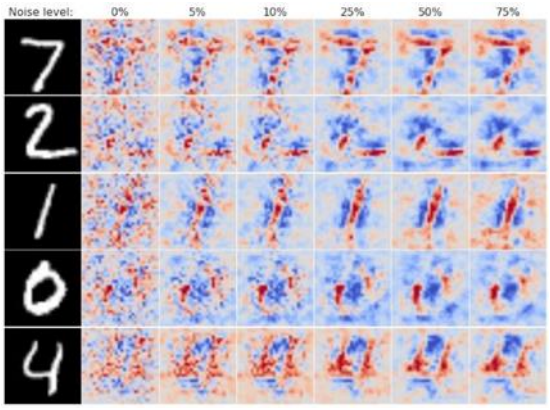

Figure 22 Effect of Gaussian noise [STK17]

5.4 *Reference Scenario and Problem Statement*

This section presents a sample reference scenario where we implemented the proposed methodology. The reference scenario we are considering includes a number of stakeholders (i.e., the smart homes) that produce image data, the faces taken from surveillance cameras, and they are looking to perform collaborative analysis to increase the accuracy of a facial expression detector. To this aim, such stakeholders share their data with a centralized *honest-but-curious* server that will perform the analysis (as shown in Figure 23).

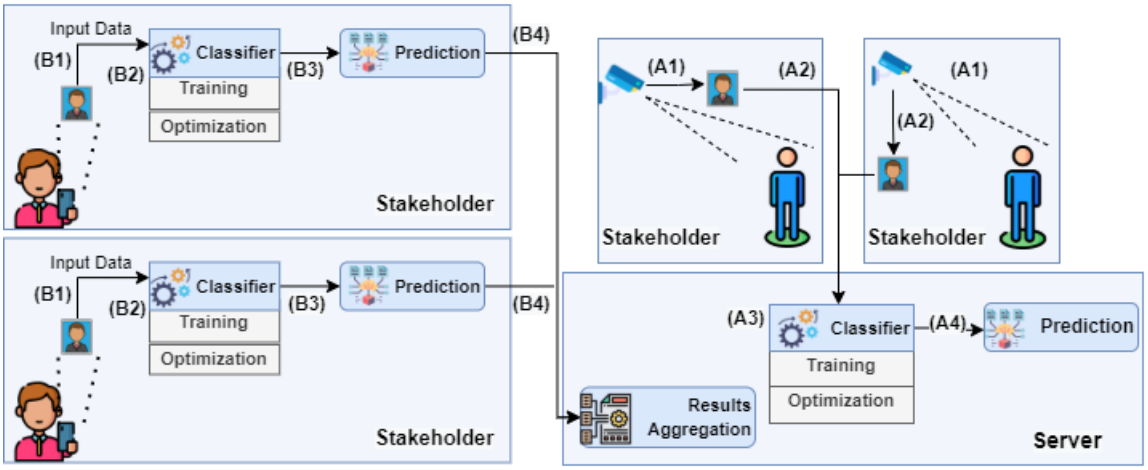

Figure 23: Scenario

However, sharing data with a third party (the central server) or among the stakeholders carries privacy concerns [SSS17]. Hence a kind of anonymization process might be required before sharing, and the techniques previously described can be used to this aim. Figure 23shows an example where the stakeholders on the left adopt the Differential Privacy technique for model anonymization, and they share the predictions obtained using these differential private models and/or locally trained model weights with the central server, while the stakeholders on the top right adopt the Autoencoders technique for data anonymization, and they share the anonymized datasets with the central server.
Explainability is another key requirement for the image recognition model since the stakeholders must be able to explain the criteria on which decisions were made. For instance, a stakeholder should be able

to understand why a given image submitted by his smart device has been classified as a “happy face” [Act21].

Moreover, the adoption of privacy and explainability techniques might impact the overall accuracy and vice-versa. In fact, the adoption of a technique to fulfill one of the requirements could negatively affect the satisfaction of the other two requirements. For example, if the technique used to preserve the privacy of the people captured by the smart cameras introduces too much noise in the images, the explainability of the decision process could be negatively affected, as well as the accuracy of the results. Our methodology addresses this problem by proposing an approach for properly configuring the technique to preserve data privacy and the technique to provide explainability in order to obtain the best trade-off among Privacy Gain, Data Utility Loss, and Explainability Gain (which are formally defined in the following) related to the process for the classification analysis of image datasets.
Moreover, the problem could also include constraints on the values of the parameter passed to the privacy and explainability techniques, i.e., the Privacy Degree and the Explainability Degree, as requirements imposed by the stakeholders for sharing their data. For instance, a stakeholder could authorize the usage of their dataset in a given analytics only if the privacy of such dataset is preserved by applying the autoencoders technique with a code size, i.e., a Privacy Degree, less than 55% of the image size.

5.5 *Concepts and Model*

This section formally discusses the three key measures on which the proposed approach is based, namely: the **Privacy Gain**, the **Data Utility Loss**, and the **Explainability Gain**.

5.5.1 Anonymization and Privacy Gain

Data privacy in image datasets can be preserved using the two techniques previously presented: Autoencoders and Differential Privacy. Autoencoders technique is used to disturb the original images before being fed to the model, i.e., before being sent to the server, while the differential privacy technique in this proposal is used during the analysis phase, i.e., on the stakeholder side to restrict model memorization of specific attributes and dataset entries and to let the model only learn general patterns about the dataset. Referring to the facial expression recognition problem, applying the previous two privacy-preserving mechanisms on the face images would alter either the original images to produce anonymized faces that are unrecognizable, or the gradients computed during the data processing phase. However, in both cases, this anonymization could affect the accuracy of the facial expression recognition model causing some facial expressions to be not correctly recognized anymore. Moreover, the explainability of the decisions in terms of saliency maps could be affected since the dataset images are less visible.
The first privacy-preserving technique, autoencoder deep neural networks, first encodes the original image into a latent representation and then decodes the latent representation to reconstruct the anonymized image. This phase is executed on the smart device which captures the image. The anonymized image is then fed to the classifier, which runs on the server. This technique enhances data privacy because performing encoding and decoding procedures on the original image and the latent space representation, respectively, alters the original image in a noisy manner. The integration of the autoencoders technique within the image classification system is illustrated in Figure 24.

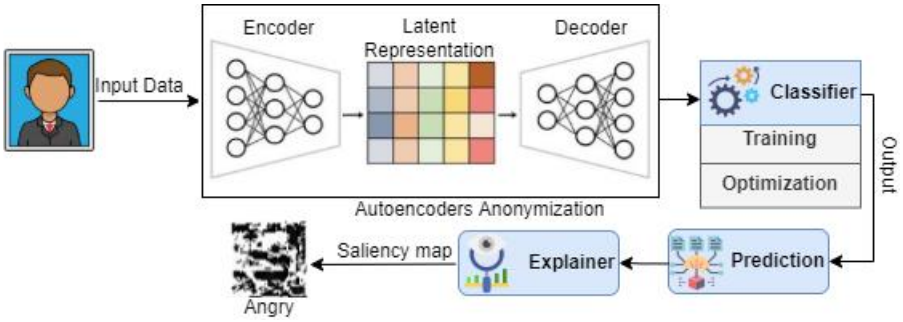

Figure 24: Autoencoders for dataset anonymization with multi-class classifier architecture

The second privacy-preserving technique, differential privacy, instead of operating on the images of the dataset, add noise to the gradients during the training process using a differential private optimizer. Hence, the classification model memorization of data instances is limited, as the used model only learns general patterns of the dataset. In fact, the trained differential private model can't distinguish whether specific dataset instances were used for training, thus providing protection against reconstruction and membership inference attacks [CWS20,PSM22]. The integration of a differential privacy mechanism within our image classification system is depicted in Figure 25.

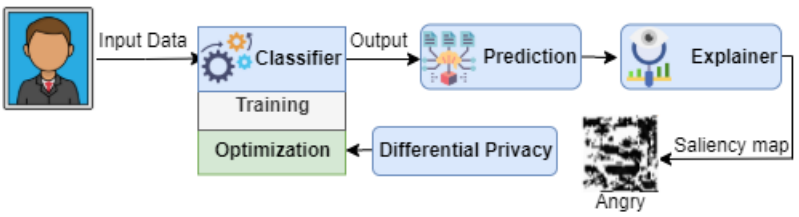

Figure 25: Differential Privacy with multi-class classifier architecture

Both the previous anonymization techniques accept an input parameter, the **Privacy Degree**. In particular, when adopting the differential privacy technique, such a parameter is represented by the noise introduced in the model, as explained in Section 5.1.1.
Instead, when adopting the autoencoders technique, such a parameter is represented by the size of the code, as explained in Section 5.1.2 as well. Figure 26 illustrates an example of the application of the autoencoders technique on FER dataset with different code sizes.

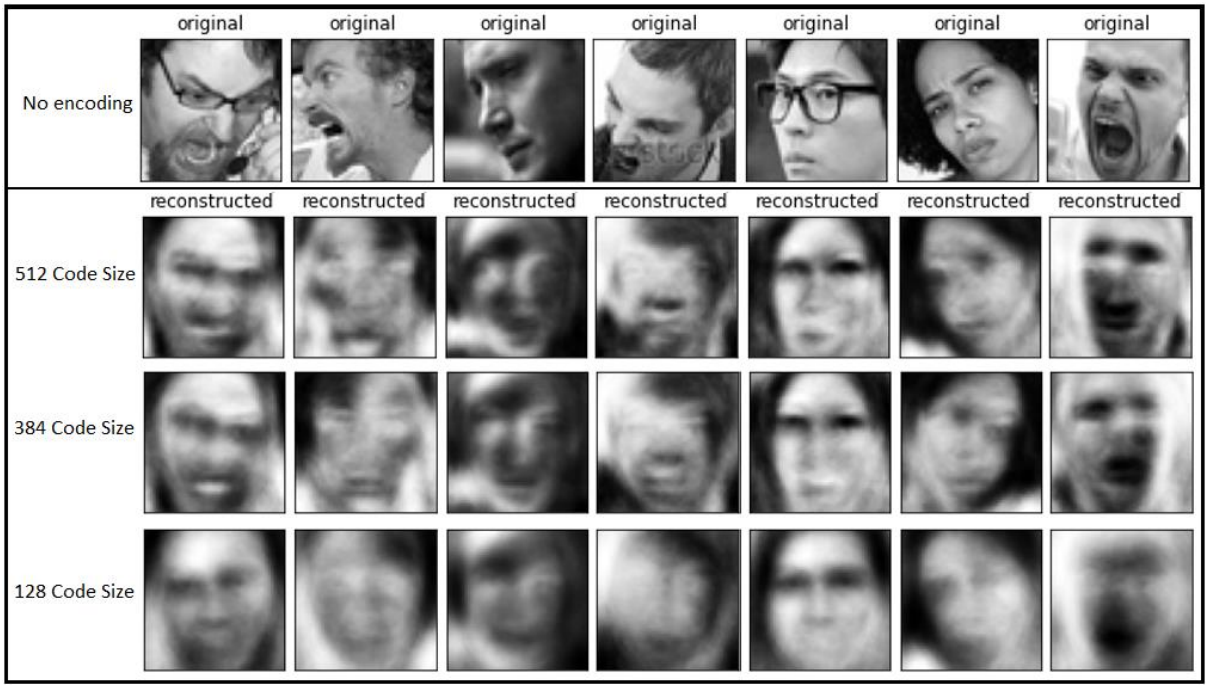

Figure 26: Autoencoders technique applied to FER dataset with distinct code size

Applying such techniques in the image classification process results in a Privacy Gain, i.e., the modifications made by these techniques introduce an uncertainty degree in the resulting model. This uncertainty is due to the lack of knowledge of all dataset attributes caused by the application of the anonymization techniques, and it reduces the confidence in the predictions performed using this model [NSH19].

The Privacy Gain obtained by applying the differential privacy technique to the dataset *D* and the classifier λ, denoted by $PG_{DP}(D, \lambda)$, is measured according to the returned privacy budget, i.e., the ε value we got from Equation 1). **Errore. L'origine riferimento non è stata trovata.** The lower the ε value, the higher the Privacy Gain returned, as shown in Equation 8.

When the autoencoders method is applied, the Privacy Gain is measured as the percentage of the code size *C* compared to the original image size *I*, as expressed in Equation 9. Considering that the code size *C* is always not greater than the size of the images in the dataset *D* (denoted by *I(D)*), we observe that $PG_{Auto}(C,D)$ is normalized in the interval (0,1]. The smaller the code size defined, the higher the Privacy Gain returned, since the code size represents the compression degree and the smaller the code size is, the more compression degree is applied and the more privacy gained.

$$\mathrm{PG_{DP}}(\mathrm{D},\lambda) = \frac{1}{\epsilon(\mathrm{D},\lambda)}$$

Equation 8

$$\mathrm{PG_{Auto}}(\mathrm{C},\mathrm{D}) = 1 - \frac{\mathrm{C}}{\mathrm{I(D)}}$$

Equation 9

5.5.2 Classifier and Data Utility Loss

The enforcement of the privacy-preserving techniques discussed above may reduce data utility and affect model performance. Data Utility Loss computation depends on the privacy technique used. In particular, when adopting the autoencoders technique, the Utility Loss U_{Auto} can be measured as shown

in Equation 10 based on the divergence concept, which represents the change between two distributions (or two datasets in this case):

$$U_{Auto}(D, D') = GD(D, D')$$

Equation 10

where *GD* is the divergence between the original dataset distribution *D* and the anonymized dataset distribution *D '*.

Divergence computation is done using several methods such as: (i) Earth Mover's distance (EMD, Wasserstein distance) where a ground distance is defined between any pair of values [RTG00], (ii) Total Variation Distance (TVD), (iii) Hellinger Distance (HD), (iv) Kullback-Leibler Divergence (KLD) [KL51], (v) X^2 divergence.

In our approach, we adopt the Earth Mover's distance (EMD, Wasserstein distance) as a divergence computation method because it defines the ground distance between any pair of values [RTG00]. EMD is simply the minimum cost or work in Equation 11 needed to match two distributions normalized by the weight of one of these distributions *(D, D')*, as shown in Equation 12:

$$Work(D, D', F) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}$$

Equation 11

$$EMD(D, D') = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}$$

Equation 12

where *D* is the first dataset distribution with *m* elements, *D'* is the second dataset distribution with *n* elements, $[d_{ij}]$ is the ground distance matrix where d_{ij} is the ground distance between clusters D_i and D'_j, and *F* represents a flow matrix $[f_{ij}]$ among the two dataset distributions *D* and *D'* with f_{ij} the flow between D_i and *D'j*, that minimizes the overall cost to get the optimal flow *F*. We recall that in our scenario *m=n* because the dataset *D'* has been produced by anonymizing each element of the dataset *D* applying the autoencoder privacy technique.
Other popular divergence computation methods used in literature are the Kullback-Leibler Divergence (KLD) [KL51] and the Total Variation Distance (TVD). However, these methods do not reflect the semantic distance among values of the clusters, since they compare the ratios of the probability density function of the two distributions, and even if the values of the dataset attributes are diverse, some values of one attribute are related to one category and semantically close. Therefore, these measurements fail to reflect this semantic distance among values. Instead, the EMD method measures the ground distance between the two distributions, taking into consideration all values even if they were semantically close.
To compute the Data Utility Loss when the differential privacy technique is exploited, instead, we cannot adopt the same method and compute the divergence between the original and the anonymized datasets. As a matter of fact, in our approach the differential privacy technique does not anonymize the original dataset before the training process, but, instead, it adds the noise to the gradients during gradients computation. Thus, the method we use in order to compute the Utility Loss compares the accuracy of the classification results of the original data analysis model against the accuracy of the classification results obtained from the differential private model and converts this percentage reference into a decimal number, following the approach proposed in [JE19].
The resulting Utility Loss measure is shown in Equation 13 and it falls in the range 0-1:

$$U_{DP}(D, \lambda, \lambda') = Acc(D, \lambda) - Acc(D, \lambda')$$

Equation 13

where *D* is the original dataset, λ is the analysis model, λ' is the differential private analysis model that adds noise to the gradients during training, and *Acc* is the accuracy of the results of the analysis model applied to the dataset.

5.5.3 Explainer and Explainability Gain

A relevant requirement of the reference scenario concerns the understanding of why the model has produced a given prediction, e.g., the classification of a face image as a "happy" facial expression.
For instance, the prediction could have been made due to edges at certain positions of the image, brightness in some areas, or some other aspects. XAI provides the possibility to understand this by producing explanations. In other words, XAI provides an explanation of why a specific prediction has been made.
In our approach, we produce these explanations in the form of Saliency Maps, using the SmoothGrad explainability method described in Section 5.3. SmoothGrad method takes an input image, and generates *n* sample images by adding Gaussian noise, computes a Saliency Map for each of these sample images, and computes a single Saliency Map as an average of the previous ones. Despite adding the same amount of Gaussian noise to the same image, the resulting *n* samples are different due to the random nature of the noise addition process. This process is controlled acting on the number of samples used during the averaging process and on the Gaussian noise level parameter used for map sharpening. Summarizing, the computation of a saliency map for each data instance of any image dataset used by a classification model λ is shown Equation 14:

$$\widehat{\alpha_l}(d) = \frac{1}{n}\sum_{1}^{n} \alpha_l\left(d + N(0, \sigma^2)\right)$$

Equation 14

where *d* is a data instance representing an input image, *l* is the class predicted for an input image among a set of classes *L*, *n* is the number of sample images used to produce the final saliency map, *α(d + N (0 , σ2))* is a Saliency Map for a sample image generated by adding Gaussian noise (N (0, σ2)) to the original one. The saliency map of a sample image is computed by differentiating α_l with respect to the input image *d* based on the analysis function λ, as shown in Equation 15:

$$\alpha_l(d) = \frac{\partial S_l(d)}{\partial d}$$

Equation 15

where S_l is the class scoring function. Smoothgrad method uses Equation 14 to enhance originally generated saliency maps by basing a visualization on a smoothing of gradient $\partial S_l(d)$ with a Gaussian kernel instead of basing it directly on $\partial S_l(d)$. Detailed equations explanation is presented in [STK17].

SmoothGrad mechanism improves the quality of the saliency maps generated by increasing the degree of Gaussian noise used to smooth the map compared to the original non-smoothed map as indicated in [STK17]. Thus, to compute the Explainability Gain *EG(d)* for the smoothed saliency map, we use Equation 16:

$$EG(d) = \frac{n}{\max(n)} \times N(0, \sigma^2)_d \times 100$$

Equation 16

where *d* is a data instance (an input image), *n* is the number of sample images used for averaging saliency maps to produce one saliency map, *max(n)* is the maximum number of used samples, and *N(0, σ2)* is the Gaussian noise.
However, this equation needs to be modified in case a privacy mechanism has been used, since the privacy mechanisms applied in our methodology either modifies the analysis model in the way it computes gradients when using differential privacy, or alters the input dataset in case of autoencoders. Therefore, to quantify Explainability Gain EG_{DP} when applying differential privacy mechanism, we need to take into consideration the model performance change represented by model accuracy, because explainability mechanisms do explain model decisions. Thus, when the model's accuracy decreases due to the use of a differential private model, the Explainability Gain needs to be bounded to this change represented by U_{DP} in Equation 13. Explainability Gain when using differential privacy mechanism is reported in Equation 17, where *d* is a data instance (an input image), *D* is the Dataset, λ is the analysis function, and λ' is the differential private analysis function obtained from λ.

$$EG_{DP}(d, \gamma) = \left(\left(\frac{n \times N(0, \sigma^2)_d}{\max(n)} \right) - U_{DP}(\gamma) \left(\frac{n \times N(0, \sigma^2)_d}{\max(n)} \right) \right)$$

Equation 17

For Autoencoders privacy-preserving mechanism, Equation 18, where $\theta = \{D, D'\}$ is used to compute the Explainability Gain, where $\gamma = \{D, \lambda, \lambda'\}$, by taking into consideration the change percentage between images of the original dataset and anonymized dataset, since saliency maps are generated based on these input image and any change to the input image affects the output saliency map. This the *EG(d)* is also bounded to the change of input images. This change percentage is measured by the geometric divergence using the EMD method.

$$EG_{Auto}(d, \theta) = \left(\left(\frac{n \times N(0, \sigma^2)_d}{\max(n)} \right) - GD(\theta) \left(\frac{n \times N(0, \sigma^2)_d}{\max(n)} \right) \right)$$

Equation 18

Providing decision explanations for classifications made on image datasets means highlighting the pixels that have the most influence on the model prediction in the generated saliency maps.
For instance, in the facial expression recognition case, the pixels representing eyes and mouth are highlighted, as shown in the maps of **Errore. L'origine riferimento non è stata trovata.** for FER dataset.
As a matter of fact, controlling the noise parameter affects saliency map explanations. The higher value we define for the Gaussian noise, the more explainable the analysis result is. Therefore, the Explainability Gain is quantified using the noise value, and this noise value ranges between *0* and *1*.

5.6 *Proposed Methodology*

This section describes the data format and the methodology we defined to properly set up the data analysis process, i.e., to find the values for the Privacy Degree and for the Explainability Degree parameters (see Section 4.1), in order to balance preserved data privacy, explainability of the model and classification analysis accuracy. As discussed in Section 3, we are considering a scenario where several stakeholders produce image datasets consisting of faces taken from surveillance cameras
and they want to exploit them to perform collaborative analysis, by training a common analysis model which would give more accurate results than a model built from the dataset of a single user only.

In particular, each stakeholder defines their own privacy and explainability requirements in terms of constraints on the values of Privacy Degree and Explainability Degree parameters, which must be respected for the stakeholder to consent to the use of their dataset.
In this scenario, the proposed methodology is aimed at finding a proper configuration of the data analysis process which allows for obtaining an analysis model which respects the stakeholders' requirements while balancing preserved data privacy, explainability level, and classification accuracy.

5.6.1 Data Format

We are focusing on image datasets, i.e., datasets where each element is a picture. Typical features of the elements of such datasets are thus pixel-related, such as: color, brightness, edges. A further feature is the image resolution, *n*, which is equal to $x \times y$ (where x and y are the dimensions of the image) in the case of gray-scale images and to $x \times y \times 3$ in the case of colored images. In particular, we are considering two labeled gray-scale image datasets, FER and MNIST, in addition to a labeled colored image dataset, CIFAR-10. The FER dataset is relevant for the reference scenario because the images in the dataset are people's faces, and the labels paired with the images represent the facial expression category, they are classified into seven classes (e.g., angry, disgust, fear, happy). In the MNIST dataset, instead, the images are handwritten digits and the label paired with each image corresponds to the digit represented by the image. Hence MNIST images are classified into ten classes, with labels ranging from 0 to 9. This dataset has been chosen because it is a well-known dataset typically used as a benchmark to validate proposed approaches. In the CIFAR-10 dataset, instead, the images are of various objects and the label paired with each image corresponds to the object represented by the image. Hence, the images are classified into ten classes. This dataset has been chosen because it is a more complex well-known dataset and is used usually as a benchmark for validation. As usual, all datasets are divided into a training set and a testing set. Each dataset is thus processed by the multi-class classifier λ, one image at a time to be divided into pixels based on the image resolution as an input layer in the neural network.

5.6.2 Optimal Trade-Off computation through Compatibility Matrix

As discussed, two possible techniques can be used for performing the analysis in a privacy-preserving way, namely autoencoders and differential privacy. When the autoencoders technique is used, our methodology determines the optimal values of the Privacy Degree and Explainability Degree parameters as follows. First of all, the stakeholders apply the autoencoder technique to their datasets. In particular, each stakeholder computes a set of anonymized datasets varying the value of the Privacy Degree parameter according to the requirements they defined, and computes locally both the Privacy Gain and the Utility Loss for each of these datasets. Then, each stakeholder shares requirements, the set of anonymized datasets, and the related Privacy Gains and Utility Losses with the server. The server executes the data analysis process on each of the received datasets, applies the Smoothgrad technique for a set of values of the Explainability Degree (chosen according to the requirements defined by the stakeholder originating the dataset), and for each of them computes the related Explainability Gain. Finally, using the values of the Privacy Gain, Utility Loss, and Explainability Gain computed varying the values of the Privacy Degree and Explainability Degree parameters, the server computes the trade-off scores to fill the compatibility matrix.
If the Differential Privacy technique is selected, then the analysis process is performed on the stakeholder side and only analysis results will be shared with the server. The values of Privacy Gain, Explainability Gain, Utility Loss, and trade-off score are computed on the stakeholder’s side for each configuration of Privacy Degree and Explainability Degree satisfying the stakeholder’s requirements. Afterward, all these values, along with the stakeholder’s requirements, are shared with the server to find the optimal trade-off score.
Hence, as a first step of our methodology, we define the formula to calculate the trade-off among the

aforementioned three measures, i.e., T_{DP} (D, λ, λ′) (shown in Equation 19) when the differential privacy technique is applied or T_{Auto}(D, D′, λ) (shown in Equation 20) when the autoencoders technique is applied. The trade-off formulas enable us to balance the conflicting objectives of privacy, explainability, and utility loss by aggregating them into a single score. The numerator of the formula captures the desirable goals of privacy and explainability, while the denominator represents the undesirable objective of utility loss. Dividing the desirable objectives by the undesirable objective yields a trade-off score that reflects the optimal balance between these objectives.

$$T_{DP}(D, \lambda, \lambda') = \frac{PG_{DP}(D, \lambda) + EG_{DP}(d, D, \lambda, \lambda')}{2 + U_{DP}(D, \lambda, \lambda')}$$

Equation 19

where *D* is the original dataset, λ is the analysis function, λ' is the differential private analysis function, $PG_{DP}\ (D, \lambda)$ is the Privacy Gain defined by Equation 8, $EG_{DP}\ (d, D, \lambda, \lambda')$ is the Explainability Gain defined by Equation 17, and $U_{DP}\ (D, \lambda, \lambda')$ is the Utility Loss defined by Equation 13;

$$T_{Auto}(D, D', \lambda) = \frac{PG_{Auto}(C, D) + EG_{Auto}(d, D, D')}{2 + U_{Auto}(D, D')}$$

Equation 20

where *D* is the original dataset, D' is the anonymized dataset, λ is the analysis function, *C* is the Code size used for the autoencoders technique, $PG_{Auto}(C, D)$ is the Privacy Gain defined by Equation 9, $EG_{Auto}(d, D, D')$ is the Explainability Gain defined by Equation 18, and $U_{Auto}(D, D')$ is the Utility Loss defined by Equation 10.
By first performing discretization on the intervals of Privacy Degree and Explainability Degree, we transform these continuous intervals into two sets of discrete values across the original interval range. Thus, for any specific dataset, privacy parameter value, and explainability value in the discretized ranges, the related trade-off is computed. Considering the available datasets, the computed trade-off values can be represented in a schematic manner by means of a tri-dimensional compatibility matrix.

5.6.3 Compatibility Matrix

The compatibility matrix has been defined in [SSM21] as a mechanism to compute the best trade-off between data utility and privacy in a multi-stakeholder collaborative analysis problem. Each element of the compatibility matrix represents, in fact, the trade-off value computed on a specific dataset (row) for a given privacy configuration (column). In our work, we are considering a further dimension, i.e., explainability. Hence, we define a tri-dimensional compatibility matrix which on the *x* dimension it reports the values obtained from the discretization of the Privacy Degree (ϕ_k) interval, on the *y* dimension it reports the available datasets (D_i, generally one per stakeholder), and on the x dimension the values obtained from the discretization of the Explainability Degree (ω_j) interval. Each element $CM_{i,k,j}$ represents the trade-off value computed with the specified combination of the three parameters, i.e., the trade-off computed on dataset D_i, with Privacy Degree ϕ_k according to the selected privacy-preserving mechanism, and with the Explainability Degree ω_j.

In the image analysis problem that we are considering, we are using two different privacy mechanisms, whose trade-off values are not comparable, since they are computed in different ways. To this end, two compatibility matrices will be defined respectively for the Differential Privacy-based analysis, and for the Autoencoders-based one. Thus, the compatibility matrix representing the differential privacy mechanism uses Equation 19 to compute the trade-off, while the compatibility matrix representing the autoencoders mechanism uses Equation 20. Figure 27 reports the structure of a compatibility matrix.

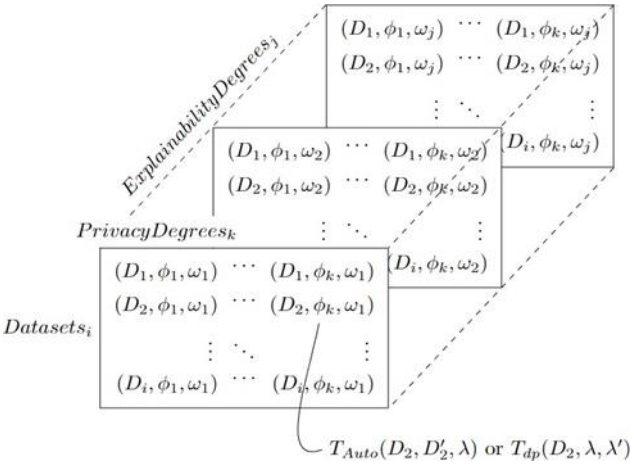

Figure 27 Tri-dimensional compatibility matrix

If the values of either ϕ_k or ω_j do not respect specific requirements expressed on the dataset D_i by the owning stakeholder, the corresponding element $CM_{i,k,j}$ is set to 0, representing thus the impossibility to compute a trade-off for that specific configuration. Once the compatibility matrix has been computed, it is reduced to bi-dimensional by aggregating the dataset dimension. In particular, the trade-offs obtained for distinct datasets with the same Privacy Degree and Explainability Degree are averaged to compute the optimal trade-off, according to Equation 21 for differential privacy mechanism and Equation 22 for autoencoders mechanism:

$$\overline{T_{DP}}(\overline{D}, \lambda, \lambda') = \sum_{i=1}^{m} w_i T(D_i, \lambda, \lambda')$$

Equation 21

$$\overline{T_{Auto}}(\overline{D}, \overline{D'}, \lambda) = \sum_{i=1}^{m} w_i T(D_i, D_i', \lambda)$$

Equation 22

where w_i is the inverse of the number of datasets for which the trade-off value is not 0.
Finally, after obtaining the best trade-off score using the concept of linear objective optimization, which chooses the maximum score among all trade-off scores, the corresponding privacy and Explainability Degrees are used for performing the analysis process in a trustworthy manner satisfying all stakeholders' requirements.

5.6.4 Applicative Example

Let us consider a case where three stakeholders, S_1, S_2, and S_3 are willing to perform collaborative analysis. The stakeholders request an analysis function on their own datasets, D_1, D_2, and D_3, respectively, from a central server using the architecture defined in Section 5.4, where $D_1 \cup D_2 \cup D_3$ represent the whole dataset D.

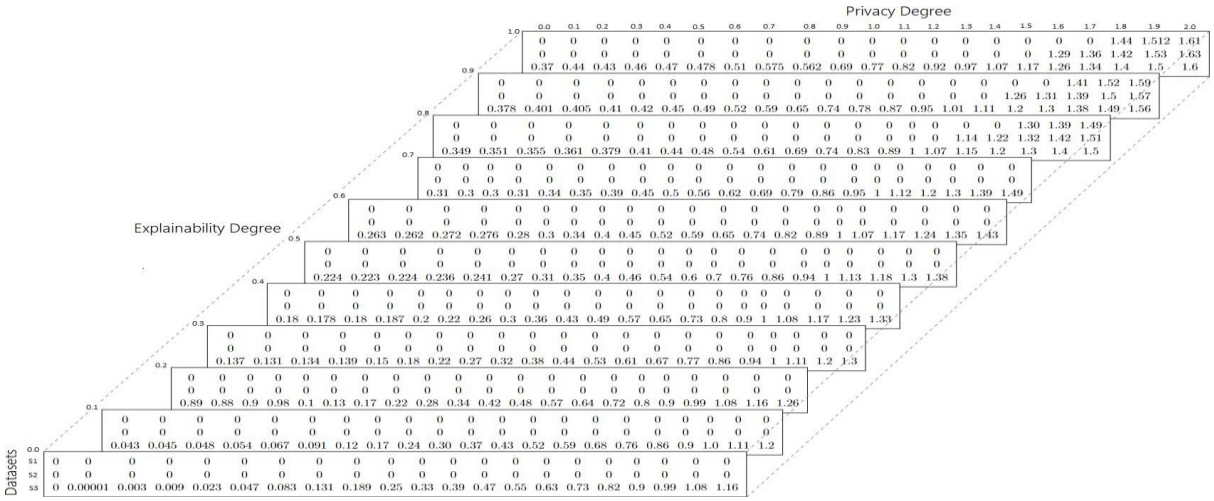

Figure 28 Tri-dimensional compatibility matrix for three stakeholders using DP mechanism example

Each stakeholder shares its privacy and explainability requirements R_1, R_2, and R_3, where each requirement defines the privacy mechanism, and constraints on the Privacy Degree to be used. The Privacy Degree is defined by each stakeholder to be higher than a specific threshold or to obtain the maximum achievable degree. Also, the Explainability Degree is defined by each stakeholder to be higher than a specific threshold or to obtain the maximum achievable degree. Hence, the requirement items provided by stakeholder S_i are as the following:

- Pm_i: Privacy mechanism selected, which is either autoencoders or differential privacy in this work.
- ϕ_i: Minimum Privacy Degree, which is the code size for autoencoders mechanism and Privacy Degree parameter for differential privacy.
- ω_i: Minimum Explainability Degree.

To find the best trade-off score, all returned configurations trade-off scores are investigated for the datasets using Equation 21 or Equation 22 which refer, respectively to the differential privacy and the autoencoders privacy-preserving techniques. Then the trade-off score that has the maximum value representing the optimal solution is returned. An optimal solution is a configuration that provides the maximum achievable degrees of privacy and explainability with the least possible effect on data utility. As an example, taking into account the reference scenario described in Section 5.4, in the following we consider 3 stakeholders having an image dataset of 20, 000 images each. These stakeholders want to analyze these datasets with the help of a central server. In order to do so, each stakeholder selects the privacy mechanism of interest and defines his own privacy and explainability requirements with specific degrees for each of them. For instance, we assume that all stakeholders have chosen the same privacy technique, e.g., differential privacy, and that they defined their privacy and explainability requirements as shown in Table 1. For instance, Stakeholder1 wants that the Privacy Degree, ϕ, is greater than 1.7, and that the Explainability Degree, ω, is greater than 0.6. Instead, Stakeholder3 does not specify any requirement, meaning that he is interested in getting the optimal value among all.

Table 1 Stakeholder's Requirements

Stakeholders	ϕ	ω
Stakeholder1	> 1.7	> 0.6
Stakeholder2	> 1.5	> 0.6
Stakeholder3	Empty	Empty

From the above-mentioned details, the compatibility matrix shown in Figure 28 is created. The *x* dimension of the matrix consists of 21 values for Privacy Degree in the interval [0.0 − 2.0] with step 0.1, the *y* dimension of the matrix consists of 3 rows representing the 3 stakeholders' datasets, while the *z* dimension consists of 11 possible degrees of explainability ω of the SmoothGrad mechanism in the interval [0.0 − 1.0] with step 0.1. Hence, the size of the compatibility matrix is 21 × 3 × 11 and is computed following the below steps:

Step 1: For Stakeholder1, Trade-off score is calculated for all possible combinations of Privacy Degrees $1.8 \leq \varphi \leq 2.0$ and Explainability Degrees $0.7 \leq \omega \leq 1.0$ and reported in the related cells of the compatibility matrix;

Step 2: For Stakeholder2, Trade-off score is calculated for all possible combinations of Privacy Degrees $1.6 \leq \varphi \leq 2.0$ and Explainability Degrees $0.7 \leq \omega \leq 1.0$ and reported in the related cells of the compatibility matrix;

Step 3: For Stakeholder3, Trade-off score is calculated for all possible combinations of Privacy Degrees φ and Explainability Degrees ω and reported in the related cells of the compatibility matrix;

Step 4: The remaining empty cells of the compatibility matrix are filled with 0, because the corresponding combination of Privacy Degrees and Explainability Degrees do not satisfy the requirements defined by stakeholders. The resulting tri-dimensional matrix is represented in Figure 28;

Step 5: The averaged trade-off score is calculated on the y dimension by aggregating the trade-off values obtained for the 3 datasets using Equation 21 with their respective weights. For instance, the trade-off score cells for $\varphi = 1.0$ and $\omega = 0.5$ would have the value 0 for Stakeholder1 and Stakeholder2, while it would be greater than 0 for Stakeholder3. Therefore, the trade-offs aggregation equation will have the following weights: w1 = 0, w2 = 0, w3 = 1. On the other hand, the trade-off score cells for $\varphi = 1.8$ and $\omega = 0.7$ would have values greater than 0 for Stakeholder1, Stakeholder2, and Stakeholder3 since they have all defined requirements having these values for privacy and Explainability Degrees. Therefore, the trade-off aggregation equation will have the following weights: the $w_1 = 1/3$, $w_2 = 1/3$, $w_3 = 1/3$. The averaged compatibility matrix is represented in Figure 29;

Step 6: Finally, the combination of privacy and Explainability Degrees yielding the best-averaged trade-off score is considered the optimal solution, which is at $\varphi = 2.0$ and $\omega = 1.0$ in our example as it returns the maximum trade-off score highlighted in Figure 29.

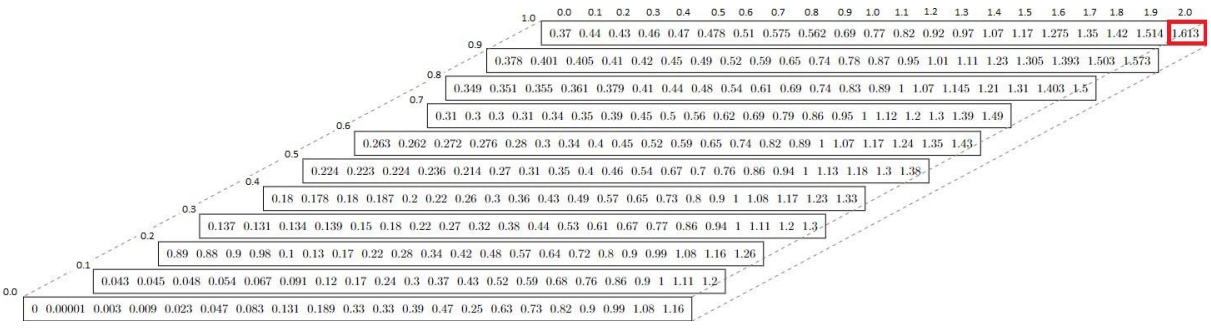

Figure 29 Averaged Tri-dimensional compatibility matrix for three stakeholders using DP mechanism example

6 Conclusion

This document is a part of the final deliverable from Work Package 4 “Privacy Aware Analytics for Security and Services”, and it described the outcome of the WP4 activities carried out during the second half of the project. In particular, this deliverable provided a detailed description of the analytics designed and developed in the second part of the SIFIS-Home project, namely: xAnomaly, Face Recognition and Person Recognition, Object Detection, Multilevel Anomaly Detection, Voice Recognition and Verification, and Anomaly Detection in Audio Signal Analysis. The Netspot Network Anomaly Detection and the Privacy Aware Speech Recognition Analytics have been significantly modified with respect to the version presented in D4.2, and hence an updated description has been provided in this document.

Moreover, this deliverable describes how all the SIFIS-Home analytics have been integrated within the Data Analytics Toolbox, which is a component of the SIFIS-Home framework embedded in the Application Toolboxes module, as shown in Figure 1. Consequently, the other components of the SIFIS-Home framework can benefit of the results of the SIFIS-Home analytics for carrying out the smart home protection tasks. For instance, the System Protection Manager can exploit the results produced by the Network Intrusion Detection analytics to decide that a device of the smart home must be removed from the SIFIS-Home devices because it has been corrupted.

Finally, since some analytics could be executed on the Cloud, and the data collected in smart homes could bring personal and privacy sensitive information, their exposure to a third party might tamper the reputation of the residents in the smart home. Hence, smart home data must be anonymized before being sent to the Cloud to be processed. We proposed a methodology to decide the anonymization level of such data in order to obtain, at the same time, a good privacy protection as well as a good accuracy of the analytics’ results.

7 References

[Dwo08] C. Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer Berlin Heidelberg, 2008.

[HRB08] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition.

[RTV08] Rothe, R., Timofte, R., & Van Gool, L. (2015). Dex: Deep expectation of apparent age from a single image. In Proceedings of the IEEE international conference on computer vision workshops (pp. 10-15).

[EEH14] Eidinger, E., Enbar, R., & Hassner, T. (2014). Age and gender estimation of unfiltered faces. IEEE Transactions on information forensics and security, 9(12), 2170-2179.

[HCC14] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

[ABD19] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., ... & Weber, G. (2019). Common voice: A massively-multilingual speech corpus. arXiv preprint arXiv:1912.06670.

[MHH21] Montani, I., Honnibal, M., Honnibal, M., Landeghem, S. V., Boyd, A., & Peters, H. (2021). spaCy: Industrial-strength natural language processing in Python.

[Mozilla] Mozilla. Project deepspeech. https://github.com/mozilla/DeepSpeech,2017.

[OWLAPI] The OWL API, http://owlapi.sourceforge.net/

[HERMIT] HermiT OWL Reasoner, http://www.hermit-reasoner.com/

[JGRAPHT] JGraphT - a Java library of graph theory data structures and algorithms, https://jgrapht.org/

[FOAF] FOAF Vocabulary Specification, http://xmlns.com/foaf/spec/

[AC15] Jinwon An and Sungzoon Cho. “Variational autoencoder based anomaly detection using reconstruction probability”. In: Special lecture on IE 2.1 (2015), pp. 1–18. url: http://dm.snu.ac.kr/ static/docs/TR/SNUDM-TR-2015-03.pdf.

[Xu+18] Haowen Xu et al. “Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications”. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ’18. ACM Press, 2018. doi: 10.1145/3178876.3185996. url: https://doi.org/10.1145%2F3178876.3185996.

[Pan+21] Guansong Pang et al. “Deep Learning for Anomaly Detection: A Review”. In: ACM Computing Surveys 54.2 (Mar. 2021), pp. 1– 38. doi: 10.1145/3439950. url: https://doi.org/10.1145% 2F3439950.

[Ruf+21] Lukas Ruff et al. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: Proceedings of the IEEE 109.5 (2021), pp. 756–795. doi: 10 . 1109 / JPROC . 2021 . 3052449. url: https : / /ieeexplore . ieee . org / stamp / stamp . jsp ? tp = &arnumber =9347460.

[HRB08] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in 'Real-Life' Images: detection, alignment, and recognition.

[SER20] Serengil, S. I., & Ozpinar, A. (2020, October). Lightface: A hybrid deep face recognition framework. In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-5). IEEE.

[Facebook] Facebook. Project deepface. https://github.com/serengil/deepface.

[AMS22] Abbasi, W., Mori, P., Saracino, A., & Frascolla, V. (2022, June). Privacy vs accuracy trade-off in privacy aware face recognition in smart systems. In 2022 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-8). IEEE.

[MAS22] Müller, A., Abbasi, W., & Saracino, A. (2022, June). Usage Control using Controlled Privacy Aware Face Recognition. In 2022 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-3). IEEE.

[PUL19] Pulfer, E. M. (2019). Different approaches to blurring digital images and their effect on facial detection.

[DJS09] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). I.

[LMBH14] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 740-755). Springer International Publishing.

[RF18] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[ImageNette] ImageNette Dataset, https://github.com/fastai/imagenette.

[RKX22] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision. arXiv preprint arXiv:2212.04356.

[MHH21] Montani, I., Honnibal, M., Honnibal, M., Landeghem, S. V., Boyd, A., & Peters, H. (2021). spaCy: Industrial-strength natural language processing in Python.

[SpaCy] SpaCy, https://spacy.io/.

[gTTS] gTTS, https://github.com/pndurette/gTTS.

[Whisper] Whisper, https://github.com/openai/whisper.

[DTD20] Desplanques, B., Thienpondt, J., & Demuynck, K. (2020). Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. arXiv preprint arXiv:2005.07143.

[RPP21] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., ... & Bengio, Y. (2021). SpeechBrain: A general-purpose speech toolkit. arXiv preprint arXiv:2106.04624.

[ECAPA-TDNN] ECAPA-TDNN, https://github.com/speechbrain/speechbrain.

[IBM-MAX] IBM-MAX, https://github.com/IBM/MAX-Audio-Classifier.

[Xu19] Z. Xu, "An empirical study of patients' privacy concerns for health informatics as a service," Technological Forecasting and Social Change, vol. 143, pp. 297–306, 2019.

[Stro19] M. Strobel, "Aspects of transparency in machine learning," in Proceedings of the 18th International Conference on Au- tonomous Agents and MultiAgent Systems, 2019, pp. 2449–2451.

[SSM21] M. Sheikhalishahi, A. Saracino, F. Martinelli, and A. L. Marra, "Privacy preserving data sharing and analysis for edge-based architectures," Int. Journal of Information Security, Mar. 2021.

[AC19] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," IEEE Security & Privacy, vol. 17, no. 2, pp. 49–58, 2019.

[Hag19] A. Hleg, "Ethics guidelines for trustworthy ai," B-1049 Brussels, 2019.

[Swee02] L. Sweeney, "k-anonymity: A model for protecting privacy," International Journal of Uncertainty, Fuzziness and Knowledge- Based Systems, vol. 10, no. 05, pp. 557–570, 2002.

[MKG07] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 1, no. 1, pp. 3–es, 2007.

[LT07] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in 2007 IEEE 23rd Int. Conference on Data Engineering. IEEE, 2007, pp. 106–115.

[RTG00] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," International journal of computer vision, vol. 40, no. 2, pp. 99–121, 2000.

[PVG20] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in Machine learning. Elsevier, 2020, pp. 193– 208.

[ACG16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.

[DKM06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in Annual int. conference on the theory and applications of cryptographic techniques. Springer, 2006, pp. 486–503.

[RSL08] S. Raskhodnikova, A. Smith, H. K. Lee, K. Nissim, and S. P. Kasiviswanathan, “What can we learn privately,” in Proceedings of the 54th Annual Symposium on Foundations of Computer Science, 2008, pp. 531–540.

[Smi09] A. Smith, “Differential privacy and the secrecy of the sample,” Blog: Oddly Shaped Pegs, 2009.

[Mcs09] F. D. McSherry, “Privacy integrated queries: an extensible plat- form for privacy-preserving data analysis,” in Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009, pp. 19–30.

[JE19] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 1895–1912.

[HAK20] O. Hajihassani, O. Ardakanian, and H. Khazaei, “Latent representation learning and manipulation for privacy-preserving sen- sor data analytics,” in 2020 IEEE Second Workshop on Machine Learning on Edge in Sensor Systems (SenSys-ML). IEEE, 2020, pp. 7–12.

[MCC19] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, “Mobile sensor data anonymization,” in Proceedings of the international conference on internet of things design and implementation, 2019, pp. 49–58.

[DJD18] M. D’Souza, M. Johnson, J. Dorn, C. Van Munster, M. Diederich, C. Kamm, S. Steinheimer, K. Kravalis, J. Boisvert, I. Ormesher et al., “Autoencoder-a new method for keeping data privacy when analyzing videos of patients with motor dysfunction (p4. 001),” 2018.

[LLL19] J. Liu, J. Liu, P. Li, and Z. Kuang, “Embedded autoencoders: A novel framework for face de-identification,” in International Cognitive Cities Conference. Springer, 2019, pp. 154–163.

[GAS22] G. Giorgi, W. Abbasi, and A. Saracino, “Privacy-preserving analysis for remote video anomaly detection in real life environments.” J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl., vol. 13, no. 1, pp. 112–136, 2022.

[STK17] D. Smilkov, N. Thorat, B. Kim, F. Vi´egas, and M. Wattenberg, “Smoothgrad: removing noise by adding noise,” arXiv preprint arXiv:1706.03825, 2017.

[SSS17] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Member- ship inference attacks against machine learning models,” in 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017, pp. 3–18.

[ACT21] A. I. Act, “Proposal for a regulation of the european parliament and the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts,” EUR-Lex-52021PC0206, 2021.

[CWS20] J. Chen, W. H. Wang, and X. Shi, “Differential privacy protec- tion against membership inference attack on machine learning for genomic data,” in BIOCOMPUTING 2021: Proceedings of the Pacific Symposium. World Scientific, 2020, pp. 26–37.

[PSM22] P. Stock, I. Shilov, I. Mironov, and A. Sablayrolles, “Defending against reconstruction attacks with r ’enyi differential privacy,” arXiv preprint arXiv:2202.07623, 2022.

[NSH19] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive pri- vacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in 2019 IEEE symposium on security and privacy (SP). IEEE, 2019, pp. 739–753.

[KL51] S. Kullback and R. Leibler, “On information and sufficiency,”Annals of mathematical statistics, vol. 22, no. 1, pp. 79–86, 1951.

Glossary

Acronym	Definition
ACL	Age Classification Layer
CPN	Colored Petri Net
DFT	Discrete Fourier Transform
DHT	Distributed Hash Table
EVT	Extreme Value Theory
FEL	Face Extraction Layer
FR	Functional Requirements
IDFT	Inverse Discrete Fourier Transform
IQR	InterQuartile Range
JSON	Javascript Object Notation
MUD	Manufacturer Usage Description
NER	Named Entity Extraction
NFR	Non-functional requirement
NSSD	Not So Smart Device
OS	Operative System
OWL	Web Ontology Language
P2P	Peer to Peer
PAP	Policy Administration Point
PSR	Privacy-Aware Speech Recognition
PTP	Policy Translation Point
RNN	Recurrent Neural Networks
SCPN	Semantic Colored Petri Net
SD	Smart Device
SIFIS-Home	Secure Interoperable Full Stack Internet of Things for Smart Home
STT	Speech To Text
UC	Use case
US	User story
XACML	eXtensible Access Control Markup Language