Semantic-enabled Architecture for Auditable Privacy-Preserving Data Analysis

Fajar J. Ekaputra [a,*], Andreas Ekelhart [b], Rudolf Mayer [b,a], Tomasz Miksa [b,a], Tanja Šarčević [b], Sotirios Tsepelakis [b], and Laura Waltersdorfer [a]

[a] *Information and Software Engineering Research Group, TU Wien, Vienna, Austria*
E-mails: fajar.ekaputra@tuwien.ac.at, laura.waltersdorfer@tuwien.ac.at
[b] *SBA Research, Vienna, Austria*
E-mails: rmayer@sba-research.org, tmiksa@sba-research.org, tsarcevic@sba-research.org, stsepelakis@sba-research.org

Editors: Michel Dumontier, Maastricht University, Netherlands; Sabrina Kirrane, Vienna University of Economics and Business, Austria; Oshani Seneviratne, Rensseler Polytechnic Institute, USA
Solicited reviews: Andre Dekker, Maastricht University, Netherlands; Víctor Rodríguez-Doncel, Universidad Politécnica de Madrid, Spain; Two anonymous reviewers

Abstract. Small and medium-sized organisations face challenges in acquiring, storing and analysing personal data, particularly sensitive data (e.g., data of medical nature), due to data protection regulations, such as the GDPR in the EU, which stipulates high standards in data protection. Consequently, these organisations often refrain from collecting data centrally, which means losing the potential of data analytics and learning from aggregated user data.

To enable organisations to leverage the full-potential of the collected personal data, two main technical challenges need to be addressed: (i) organisations must preserve the privacy of individual users and honour their consent, while (ii) being able to provide data and algorithmic governance, e.g., in the form of audit trails, to increase trust in the result and support reproducibility of the data analysis tasks performed on the collected data.

Such an *auditable, privacy-preserving data analysis* is currently challenging to achieve, as existing methods and tools only offer partial solutions to this problem, e.g., data representation of audit trails and user consent, automatic checking of usage policies or data anonymisation. To the best of our knowledge, there exists no approach providing an integrated architecture for auditable, privacy-preserving data analysis.

To address these gaps, as the main contribution of this paper, we propose the WellFort approach, a semantic-enabled architecture for auditable, privacy-preserving data analysis which provides secure storage for users' sensitive data with explicit consent, and delivers a trusted, auditable analysis environment for executing data analytic processes in a privacy-preserving manner. Additional contributions include the adaptation of Semantic Web technologies as an integral part of the WellFort architecture, and the demonstration of the approach through a feasibility study with a prototype supporting use cases from the medical domain. Our evaluation shows that WellFort enables privacy preserving analysis of data, and collects sufficient information in an automated way to support its auditability at the same time.

Keywords: provenance, semantic web, privacy-preserving data analysis, auditability, dpv, consent management

1. Introduction1

In Europe, the GDPR [1], in effect since May 2018, stipulates high standards in data protection and imposes substantial fines for non-compliance. Moreover,

*Corresponding author. E-mail: fajar.ekaputra@tuwien.ac.at.

security breaches have the potential to go beyond financial impact and ruin an organisation permanently, as lost reputation and trust cannot be regained easily. Hence, the organisations not only have to make sure that the data analysis techniques fulfil guarantees of privacy of the individuals representing the underlying records [2], but they also need to ensure that the use of data is compliant with the user consent. This is especially important to keep track of explicit consent for sensitive user data as defined in GDPR art. 9.

Furthermore, to enable accountability, and transparency of systems used for data analysis through auditability, we must capture and understand the broad context in which the system operates: data sources and algorithms, as well as processes and services that the deployed systems depend on [3]. In our context, we define auditability as the ability of an auditor to receive accurate results for inspection and review. Goal of an audit can be to check compliance of an organisation with pre-defined rules and/or standards and regulation. However, without a strong background in IT security and big budgets, providing a secure platform for data storage and analysis is often beyond capabilities of small and medium-sized organisations.

As a consequence, organisations often refrain from collecting data centrally and, for example, offer applications that analyse data locally, on the device that collects the data, instead. While this reduces the attack surface, it means losing on the potential of data analytics and learning from the entire collected data, and thereby hinders innovative services and research. As an example, identifying trends over cohorts of users is not possible.

Some approaches, such as Federated Learning, allow analysis of data distributed among multiple sources by computing a common result without having to centralise the inputs (data). However, most of these approaches are well explored only when data is horizontally partitioned. In our scenario, we generally have a setting with vertical partitioning, i.e. we have data records describing different aspects of the same individual(s), which still poses some challenges (cf. Section 7). Further, auditability is more difficult to achieve in the federated setting. Therefore, we focus on the centralised setup proposed in our paper.

To this end, the main research problem discussed in this paper can be formulated as follows: *How to enable auditable, privacy-preserving data analysis systems?*

Further, we divided the problem into a number of research questions as follows:

- *What are the key characteristics of auditable, privacy-preserving data analysis systems?*
- *What are the key elements of an architecture for auditable, privacy-preserving data analysis systems?*
- *How can Semantic Web technologies be used to enable auditable, privacy-preserving data analysis systems?*

Recently, research on applying Semantic Web technologies to address challenges related to auditability and user privacy has been intensified. On the one hand, the emergence of the W3C PROV-O recommendation [4] has been well-received in the community as a de-facto standard for RDF representation of provenance trails, forming a basis for auditability. As a result, the development of methods and tools around PROV-O are growing rapidly, summarised in a recent survey paper [5]. On the other hand, the research on representation and compliance checking for user consent and personal data handling also gained traction. Semantic Web technologies, with ontologies and reasoners as key resources, have been proposed to facilitate representing [6–8] and automated compliance checking [9, 10] on usage policy, which are key to enable consent-check mechanisms for privacy-preserving data analysis.

In a wider research community, privacy-preserving data analysis is an active research area. Various approaches that have been proposed for privacy-preserving data analysis can be categorised according to the data lifecycle they are applied to, for example: *privacy-preserving data publishing* (PPDP) and *privacy-preserving data mining outputs* (PPDMO) [11]. In our work we rely on the methods from PPDP and PPDMO. PPDP relies on anonymising data records before publishing. The most prominent method is k-anonymity [12], including its extensions, l-diversity and t-closeness. Approaches towards PPDMO include *query inference control*, where either the original data or the output of the query are perturbed and *query auditing* where some queries are denied because of potential privacy breach. DataSHIELD [13] is an example of such a system, which applies both of the later techniques to a number of data mining functions.

Despite research on isolated topics, to the best of our knowledge, no integrated architecture for auditable privacy-preserving data analysis exists. The lack of such an approach can be attributed in part to the contradicting goals of collecting as much data as possible

for auditability, while aiming for reduced data collection to minimise the impact of a data breach.

To fill in this gap, we propose a novel approach called WellFort, a semantic-enabled architecture for auditable, privacy-preserving data analysis which (i) provides secure storage for users' sensitive data with explicit consent, and (ii) delivers a trusted analysis environment for executing data analytic processes in a privacy-preserving manner. Our approach is novel, because organisations do not have direct access to individual data, but only access it in an aggregated or anonymised form. Organisations on the one hand can benefit from a large groups of individuals that are potentially willing to share their data for broader analytics tasks, such as those required by e.g., medical research. Users, on the other hand, benefit from a privacy-preserving and secure platform for their data, and can contribute to analytics/research projects in a secure manner. Finally, analysts (such as researchers) may obtain a detailed source of microdata, if data subjects give the corresponding consent.

The contribution of this paper is three-fold:

- We propose WellFort, a semantic-enabled architecture for auditable privacy preserving data analysis,
- We adapt Semantic Web technologies as an integral part of the architecture, including: (a) ontology-based data representation of provenance, dataset metadata, and usage policies, (b) ontology development and extension methods, as well as (c) automatic usage policy compliance checking with OWL2 reasoning, and
- We demonstrate the feasibility of our approach on real-world use cases in the medical domain.

The rest of the paper is structured as follows. Section 2 presents the requirements for the architecture. Section 3 describes the WellFort conceptual architecture and the processes it supports. Section 4 reports on the Semantic Web methods adoption to support the architecture. Section 5 presents the prototype implementation. Section 6 investigates the feasibility of a WellFort-based prototype in four use cases, and Section 7 presents related work. Finally, conclusions and an outlook are given in Section 8.

2. Requirements1

In this section, we introduce the main functional and non-functional requirements for our platform. These iteratively refined requirements are based on privacy regulations, such as GDPR [1] as well as use cases and discussions with partners from the WellFort project[1].

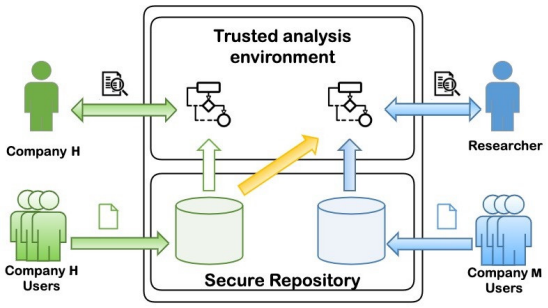

Fig. 1. WellFort overview scenario

In the following, we describe a scenario (cf. Figure 1) to exemplify and motivate the requirements for auditable privacy-preserving data analysis described in the following sub-sections. Later, in Section 6, we will derive four concrete use cases from this scenario to conduct feasibility studies of our approach.

Andrea is a user of two health apps, one from Company H, which monitors her heart rate, and another one from Company M, which records her cholesterol level. She wants to collect her data from both apps, link and share her data, but only for research purposes. She also considers sharing her data for specific commercial purposes, like getting product recommendations, but wants to decide about it later.

However, since Company H and Company M are small companies, they do not want to manage the data themselves. They plan to store their users' data in a cloud platform without having to develop and run the platform themselves.

Barbara, a researcher from TU Wien, is planning to develop a machine-learning model on the relationship between heart rate and cholesterol level for her research. She is looking for suitable data for her model, but at the same time, she wants to make sure that she is compliant with regulations.

Later on, Andrea asks Claudia, an auditor of the cloud platform, regarding the use of her data. Since Andrea's data is used as part of Barbara's research, Claudia provides Andrea with the information on the research setup.

In the scenario above, the data captured by the application providers (i.e., *Company H and M*) can be categorised in the special category of personal data in

[1] https://www.sba-research.org/research/projects/wellfort/

GDPR art. 9. Thus, the cloud platform should facilitate users like *Andrea* to define explicit consent to their data. Furthermore, due to privacy reasons, application providers and researchers like *Barbara* should not get direct access to user data repositories, but still should be able to conduct data analysis in a privacy-preserving manner without compromising user consent. *Claudia*, in the other hand, should be able to provide the audit information to *Andrea*, without having access to the content of her personal data.

2.1. Privacy-preserving Data Analysis2

Data analysis. The platform must enable data analysts to conduct privacy-preserving analysis without disclosing individual user data. To reduce the attack surface, input and intermediate data beyond the analysis must not be persisted by the platform. It should be possible to analyse data originating from the same application (organization) and also to conduct cross-application (cross-data owner) analysis (e.g., correlating heart rate with cholesterol level, even though each was collected from a different application).

Finally, all data analysis computation on raw individual data must take place inside the platform and the analysis environments can be stored only for a limited period of time, as defined by the data retention policy of the platform.

Data publishing. Analysts must not download any data in its original form, to fulfil regulatory requirements, and to minimise the surface for privacy attacks, such as user re-identification. Instead, for offline analysis, the platform should offer anonymized downloads, and the generation of synthetic data [14]. The platform must provide measures to prohibit downloading similar data multiple times by a single analyst, since this provides the channel for disclosure of sensitive information.

Usage Policy Management. Data analysis requires fine-grained consent from the involved users [1], and access control has to ensure that only authorised analysts can work with the data. To this end, usage policies, which include both user consent and data handling setup need to be represented in a machine-readable format. Further, a mechanism to conduct an automatic usage policy checking between consent and data handling is required.

2.2. Auditability2

Provenance data capture and tracing. Provenance of data must be traced in the whole data lifecycle within the platform. Automated collection of by who, when, and how data has been accessed, processed and analysed needs to be recorded, ranging from high level to precise trails. Traces must be easily accessible by auditors to check which data was used and how results were computed to identify irregularities or do checks. Finally, analysts must not be able to acquire or infer personal or otherwise sensitive data via the provenance trail.

Provenance data inspection and analysis capability. Collected provenance data streams must be accessible to users with auditing privileges through an interface, to be able to answer audit questions concerning data and its upload, usage, and analysis. Such a question could be to inspect which studies involved a user's personal data, for what purposes, and how it was analysed. Auditors should be able to access the data while preserving the privacy of users, therefore, no personal data must be acquired by the provenance feature.

Metadata-retention after deletion. Even after users revoke their consent due to their right to be forgotten, consent for past studies is still valid. In order to fulfil the requirement of providing proof of correct data capturing and data analysis, minimum necessary meta-information must be kept in the provenance module for past studies that were conducted and based on now deleted information.

3. Conceptual architecture1

In this section, we describe the conceptual architecture of the platform for auditable privacy-preserving data analysis, which addresses the requirements defined in Section 2. More on the technical details will be described on Section 4 and Section 5.

The conceptual architecture of the platform is depicted in Figure 2. There are three distinct actors:

- *User* – store their data in the platform and give consent to analyse it. They use an application provided by the organisation and do not interact with the platform directly, e.g., using a dedicated user interface.
- *Analyst* – can run experiments in the platform. They define what types of data will be used in the analysis and perform the actual analysis.
- *Auditor* – can analyse evidence collected to answer specific audit questions that depend on the purpose of the audit, e.g., a litigation case, or simple usage statistics for users wanting to know when and by whom their data was used.

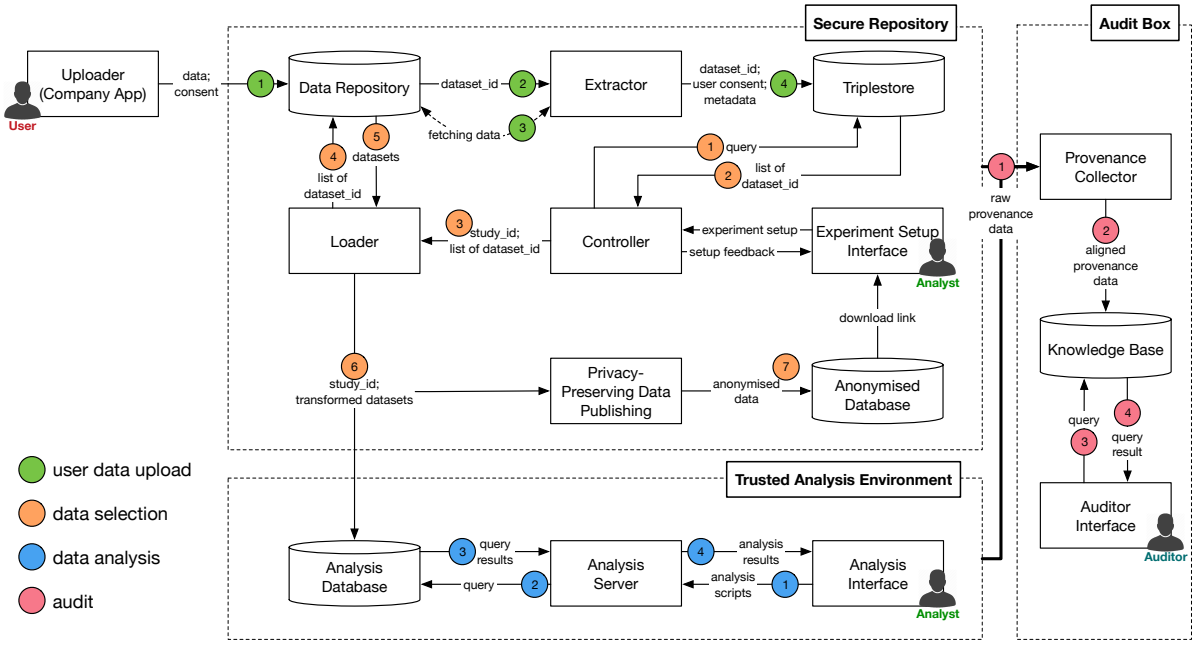

Fig. 2. WellFort conceptual architecture

The architecture consists of three component groups, each serving a different purpose:

- *Secure Repository* – stores data uploaded by a user and allows the selection of data to be used in experiments by the analyst. There are no components in this group allowing analyst to directly view or analyse raw data.
- *Trusted Analysis Environment* – selected data that fulfils experiment criteria, e.g., consent, fit for purpose, etc. is duplicated to this component for further analysis. This part of the platform provides mechanisms allowing the analysts to conduct their experiments in a privacy-preserving manner. Data selection is usually expressed via queries.
- *Audit Box* – collects and manages provenance data to support the auditability of processes within the *secure repository* and *trusted analysis environment*. The Knowledge Base can be accessed via an interface to answer audit-related questions on personal data access and usage.

In Figure 2, we use dashed lines to indicate the division into these three groups. Figure 2 further depicts four typical processes executed in the platform:

- *User data upload* – the process starts when a user's application sends data to the platform. It extracts metadata from the data, and stores it together with the data and the consent indicated during the upload in the platform. Thus, every dataset uploaded to the platform is linked to a minimal set of information that allows for its retrieval.
- *Data selection* – analysts define the search criteria for data they want to use in their experiments. If the platform has enough data fulfilling their criteria (and also consent for usage), then the process loads the actual data into the *Trusted Analysis Environment* where the analyst obtains accurate aggregate results without having access to individual datasets. The search for data relies only on high-level information provided in the metadata. For example, *BloodGlucose*, which is one of the attributes in the actual user data, has only a boolean value in the corresponding metadata to indicate whether the dataset contains information on *BloodGlucose*. Furthermore, Analysts can download anonymised data and analyse it outside (offline) of the platform. However, anonymisation can substantially reduce the quality of the data and the results of the offline data analysis thereupon.
- *Data analysis* – analysts process the data and produce results by submitting code to the platform.

The platform will ensure that the analysts will not be able to identify or infer data subjects from the analysis.
- *Audit* – provenance information obtained from each component of the platform is sent to an independent component and organised in a knowledge base. Auditors query experiment details.

We numbered the steps of each process and marked them with colours: green (user data upload), orange (data selection), blue (data analysis), and pink (audit). We describe now each of the processes in detail to explain the role of each of the components.

3.1. User data upload2

Following the terminology set by the Open Archival Information System (OAIS) reference model [15], this process is equivalent to the ingest process that transforms and enriches data uploaded by users, so that it can be later located and accessed according to the corresponding user consent.

The process starts by a user's application sending data together with the consent to the *Data Repository*. The *Data Repository* creates a record in which it stores the persistent identifier (PID), consent, and location of the received data. The PID is unique and never changes. Following the FAIR (Findable, Accessible, Interoperable, Reusable) principles [16], even if the data is deleted, the PID is kept, and the record is updated with information on why and when the data was removed.

The platform automatically creates metadata and stores it together with the consent in the *Triplestore*. The *Extractor* component receives the PID of the dataset and accesses the data. The generated metadata consists of a list of key-value pairs, where each key corresponds to an attribute located in data, e.g., heart rate, while the value is always boolean, i.e. true or false. Thus, the *Triplestore* contains PIDs of datasets, together with a high-level description of the presence of certain types of contents, and consent given. The *Triplestore* does not contain any user specific information. We provide more details on the metadata and consent representation in Section 4.1.1 and Section 4.1.2 respectively.

It would be possible to use only a *Data Repository* and omit the *Triplestore*. However, the current design has two major advantages. First, data is decoupled from its metadata, and thus analysts can query the *Triplestore*, while the actual data can be kept offline, e.g., due to security measures. Second, by using the semantics provided by the underlying ontology, we can run advanced queries to identify suitable data for an experiment that go beyond simple string matching, e.g., by using subsumption reasoning for query expansions and usage policy compliance checking.

3.2. Data selection2

Analysts use the *Experiment Setup Interface* to define their experiments. They specify the purpose of the experiment, and which data attributes they want to use in their analysis. Based on this input, the *Controller* automatically generates a query and sends it to the *Triplestore*. The *Triplestore* returns a list of dataset PIDs that fulfil the criteria (cf. Section 4.1 for details on how the result is produced). If the number of available datasets fulfils the requirements of the analyst (so that the study is meaningful from a statistical point of view), they can decide to start a new study and load the data into the *Trusted Analysis Environment*.

Each study has its own PID and links with the list of dataset PIDs, as well as metadata about the experiment. Similar to the PIDs used for datasets, the study PID allows us to refer to experiments in an unambiguous way.

The *Loader* receives the study PID and fetches the data from the *Data Repository*. Depending on the actual implementation of the *Trusted Analysis Environment*, the *Loader* transforms the data into a suitable format, e.g., database, pivot tables, etc. When the data is loaded, analysts receive information via the *Experiment Setup Interface* on how to access the *Trusted Analysis Environment*, e.g., an access URL, username and password.

Analysts can also decide to download either the k-anononymised data or synthetic data. To do so they use the *Experiment Setup Interface* to define their experiment. All the steps are the same as before, yet the data is not loaded in the the *Analysis Database* but transformed by the *Privacy Preserving Data Publishing* component and made available for download.

3.3. Data analysis2

Analysts use the *Analysis Interface* to communicate with the *Analysis Server*, which in turn has access to the data stored in the *Analysis Database*. The *Analysis Interface* allows submitting code for data analysis and viewing the processing results. It also logs every

activity of the analyst. The *Analysis Interface* can be realised as a web application running on a server side.

The *Analysis Server* inspects the code submitted by analysts and protects data from direct access. For example, the server allows only aggregation functions to be executed, and blocks requests to list properties of specific datasets (users). If the submitted scripts pass such requirements for privacy-preservation, then the results are calculated on the original data and returned to the *Analysis Interface*.

Please note that the *Analysis Interface* is part of the platform, and a direct access to the *Analysis Server* is not possible. Such a design is more secure, and allows to better track the analysis process.

A data analysis process can only be executed when the data has been loaded into the *Trusted Analysis Environment*, and when the analysts received their connection details. The *Analysis Database* receives only the data required for the specific study, keeps it for a limited time, and deletes it after the study is completed. During the analysis, there is no need to access the *Secure Repository*. Hence, thanks to loose coupling, each group of components can be instantiated by deploying them on different nodes with varying levels of security, access, as well as computational power.

3.4. Audit2

The *Secure Repository* and the *Trusted Analysis Environment* push provenance information to the *Provenance Collector* to separate this information from the components processing data, in which the provenance was captured. Such a design increases security of provenance information, because it prevents from non-authorised modification of provenance when one of the user-facing components is compromised.

Provenance capturing can be implemented in different ways and the captured information can be structured differently, hence, it is necessary to align representation of provenance information before loading it into the *Knowledge Base*.

The *Knowledge Base* stores provenance for each of the processes supported by the platform, specifically:

- *User Data Upload* – a persistent identifier of a dataset, consent, an automatically generated metadata summarising contents of the dataset, a timestamp.
- *Data Selection* – an identifier of the analyst, an identifier of the submitted query, query attributes (e.g. timestamp, study purpose, study description), query results, consent check results, and an identifier of the analysis (only if the consent check was successful and the analyst decides to analyse the data).
- *Data Analysis* – an identifier of the analysis (created in the previous step to link the query with the analysis), information on software environment and dependencies (e.g. operating system, software libraries, environment variables), script parameters, results of the analysis.

Table 1
The list of predefined prefixes and namespaces

prefix	namespace
:	http://w3id.org/wellfort/ns/dpv#
meta:	http://w3id.org/wellfort/ns/meta#
wprv:	http://w3id.org/wellfort/ns/prov#
id:	http://w3id.org/wellfort/id/
dpv:	http://www.w3.org/ns/dpv#
prov:	http://www.w3.org/ns/prov#
dcat:	http://www.w3.org/ns/dcat#
dct:	http://purl.org/dc/terms/
xsd:	http://www.w3.org/2001/XMLSchema#
rdt:	https://github.com/End-to-end-provenance/ExtendedProvJson/blob/master/JSON-format.md#
p-plan:	http://purl.org/net/p-plan#

Auditors use the *Auditor Interface* to get access to provenance information organised in the *Knowledge Base*. They define queries to retrieve relevant provenance information when performing an audit, which we will describe in the next section.

4. Semantic-Web Methods for Auditable Privacy-preserving Data Analysis1

In this section, we describe in detail how Semantic Web technologies contribute to the overall architecture presented in Section 3, in particular for consent and metadata management (Section 4.1) and auditability (Section 4.2) . Note that from here on, for consistency reasons we will use the prefixes and namespaces introduced in Table 1.

4.1. Consent and metadata management2

In this section, we describe the mechanism to manage and utilise consent and metadata with Semantic Web technologies, to ensure that experiments can be executed with a sufficient amount of data without com-

promising the user consent of the datasets. The mechanism mainly resides within the *Secure Repository*, where the analysts interact with the system to setup their experiments.

When an analyst interacts with the *Experiment Setup Interface*, she needs to describe the *Data Handling* setup, which contains information about the required category of personal data, purpose, processing, recipient, and expiry time of the experiment. This information allows the data selection process to be conducted as the following:

Step 1. Finding datasets with matching categories, where we query the *Triplestore* for datasets. As a part of the step, the RDFS reasoning is used to expand the dataset selection based on the subsumption relations between data categories.

Step 2. Usage policy compliance check. In this step, we determine whether the Data Handling setup of the analyst is compliant with the user consent of datasets collected from Step 1. Therefore, it is necessary to automate the compliance check in order to speed up the process.

From the steps above, we identify three necessary components that need to be defined in order to facilitate the consent and metadata management, namely (i) Dataset metadata representation, (ii) Usage policy representation, and (iii) Automated compliance checking mechanism.

4.1.1. Metadata Representation3

There are several suitable ontologies that can be used to represent dataset metadata, and we therefore only need to adapt and extend those where required, e.g., Data Catalog Vocabulary (DCAT) [17] or VOID [18]. In our approach, we decided to base our metadata representation[2] on Data Catalog Vocabulary (DCAT) Version 2 [17] since it is the latest W3C recommendation.

We adopt two classes from DCAT, `dcat:Catalog` to describe a catalog of datasets stored within our *Data Repository*, and `dcat:Dataset` as the superclass of our `meta:Dataset` to represent each dataset. We argue that this subclass is necessary since we add additional properties specific to `meta:Dataset` as follows:

- `meta:hasConsent` that relates a dataset and active `dpv:Consent` given by a data subject for this dataset, and
- `dpv:hasPersonalDataCategory`, where this relation indicates the categories of personal data contained within a dataset. Note that the value of this property may be an instance of several sub-classes of `dpv:PersonalDataCategory`.

These two relations imply the use of the Data Privacy Vocabulary (DPV) [6] in our architecture that we will explain in the next subsection.

```
id:WellFortCatalog a dcat:Catalog ;
  dct:title "WellFort Catalog" ;
  dct:publisher id:TUWien ;
  dcat:dataset id:dataset-1, id:dataset-2 .
id:dataset-303 a meta:Dataset ;
  meta:hasConsent id:consent-1 ;
  dpv:hasPersonalDataCategory id:category-1 ;
  dct:identifier "303" ;
  dct:issued "2011-12-05"^^xsd:date .
id:category-1 a :HeartRate, dpv:Age .
```

Listing 1: An excerpt of dataset metadata on WellFort

We provided an example metadata in Listing 1, which shows the single catalog in the platform and dataset-303, which contains personal information on heart rate and age. We deliberately removed the relation between users and datasets in our metadata.

4.1.2. Usage Policy Representation3

One of the main requirements of our approach is to ensure that the personal data can be used for analysis without compromising user consent. To this end, there is a clear need to represent usage policies, both for user consent and analyst data handling, in a clear and concise manner. In recent years, a number of ontologies has been proposed to represent usage policies, e.g., DPV [6], PrOnto [8], SPECIAL [19] and SAVE [7], among others.

For our approach, we decided to use DPV as our basis for usage policy representation, due to (i) the community-based development of the vocabularies, and (ii) DPV does not recommend a specific mechanism to use its concepts, providing adopters with a high degree of freedom to use and adapt it for their purpose. Furthermore, the DPV Community Group[3] involves researchers and practitioners with similar interests across the world and has become a strong candidate for usage policy representation.

We adapt the SPECIAL Policy Language [19] to structure DPV as usage policy. In this structure, a usage policy is composed of one or more *basic usage policies* each of which is an OWL2 expression of the

[2] http://w3id.org/wellfort/ns/metadata

[3] https://www.w3.org/community/dpvcg/

```
ObjectIntersectionOf(
  ObjectSomeValuesFrom(dpv:hasPersonalDataCategory
    SomePersonalDataCategory)
  ObjectSomeValuesFrom(dpv:hasProcessing SomeProcessing)
  ObjectSomeValuesFrom(dpv:hasPurpose SomePurpose)
  ObjectSomeValuesFrom(dpv:hasRecipient SomeRecipient)
  DataSomeValuesFrom(:expiryTime
    DatatypeRestriction(xsd:dateTime xsd:maxInclusive
      SomeDateTime)))
```

Listing 2: DPV adaptation of SPECIAL Policy Language for WellFort

form shown in Listing 2. We implement a small adjustment to the original structure by replacing the storage information with an explicit expiry time of policies, which is simpler but provide the necessary time limit on the use of data. We argue that the current structure is inline with the GDPR, which defines that policies shall specify clearly (i) which data are collected, (ii) what is the purpose of the collection, (iii) what processing will be performed, (iv) whether or not the data will be shared with others, and (v) for how long the data will be stored.

Within our approach, we use the `dpv:Consent` to represent user consent for their personal data and `dpv:PersonalDataHandling` to capture the analyst data handling. Other than these two classes, similar to SPECIAL, DPV provides taxonomies that represent general categories for each type of basic usage policy, which we briefly describe in the following:

- `dpv:PersonalDataCategory` to represent the category of personal data provided by a user and requested by an analyst.
- `dpv:Processing` to describe the categories of processes that are consented by a user and processes that are planned to be executed by an analyst.
- `dpv:Purpose` to describe the allowed purpose of the personal data processing (user) and the purpose of the data handling (analyst).
- `dpv:Recipient` to describe the designated recipient of personal data processing results (analyst) and the consented recipient of the results (user).

Furthermore, there are a number of properties that we deemed necessary to be included in the consent and data handling setup:

- `dpv:hasPersonalDataCategory` to link either a consent or a data handling to the relevant personal data category.
- `dpv:hasPurpose` to link either a consent or a data handling to its purposes.

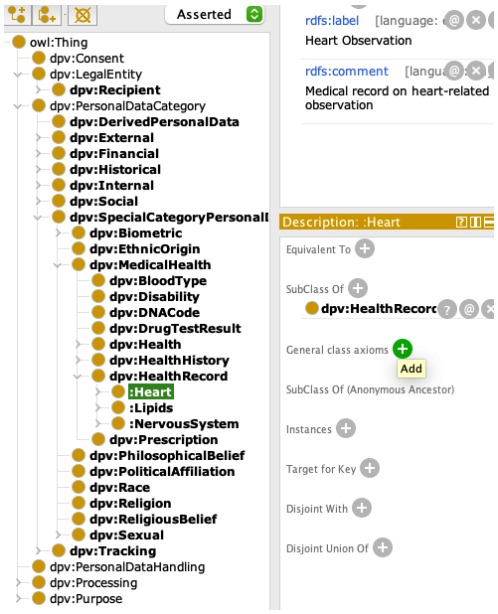

Fig. 3. An excerpt of the extended Data Privacy Vocabulary (DPV)

- `dpv:hasProcessing` to link either a consent or a data handling to the processing mechanisms.
- `dpv:hasRecipient` to link either a consent or a data handling to the target recipient.
- `:expiryTime` to store the expiry time for both user consent and data handling. We decided against using `dpv:expiry` to include `dpv:PersonalDataHandling` as a domain and to use `xsd:DateTime` as its range.
- `dpv:hasProvisionTime` to store the provision time of a user consent.
- `dpv:hasWithdrawalTime` to store the withdrawal time of a user consent.

While the generic concepts provided by DPV vocabularies cover a wide range of subjects, they might not cover the requirements for the fine-granular consent definition in domain-specific use cases. To address this issue, we adopt the vocabulary extension approach we previously developed in [20], where we adapted and extended the SPECIAL vocabularies in the smart-city domain. We reuse a similar mechanism to extend DPV concepts with use case specific concepts to allow usage policy matching on fine-granular user consent for our evaluation in Section 6. An excerpt of the extended DPV[4] is shown in Figure 3.

[4]Full version is available online: http://w3id.org/wellfort/ns/dpv

4.1.3. Automated Compliance Checking3

In SPECIAL [19], policies and log events are described in semantically unambiguous terms from the same taxonomies defining usage policies, hence it facilitates automatic compliance checking. To allow automatic compliance checking, the usage policy enforced by a data controller contains the operations that are permitted within the data controller's organisation.

In this work, we reuse the same mechanism as in SPECIAL approach [9], where the usage policy U_c attached to a *dpv:PersonalDataHandling* complies with the usage policy P_s in the data subject's *dpv:Consent* if and only if all the authorisations in U_c are also authorised by P_s, that is, U_c complies with P_s if and only if U_c is a subclass of P_s. This implies checking whether this subsumption is entailed by the combined ontology containing a subset of DPV as described in Section 4.1.2 and the aforementioned auxiliary vocabularies. This entailment is supported by general inference engines for OWL 2 (e.g. HermiT and FaCT++) and further details of the compliance checking mechanism can be found in [9, 21]. Furthermore, Bonatti et al. [22] have developed a specialised reasoning engine that allows real-time OWL2 reasoning that enhanced the reasoning performance for the automatic consent check mechanism considerably.

The reasoning behind reusing the SPECIAL usage policy compliance checking mechanism for our approach is two-fold: (i) the approach is compatible with DPV, which is mainly due to the fact that DPV is developed based on the SPECIAL vocabularies, and (ii) our successful experience in adopting the SPECIAL approach in several of our prior works [10, 20, 23].

4.2. Auditability2

In this subsection, we explain the extension made to accommodate auditability concerns and questions. First, we describe the methodology on how to capture the provenance data in our architecture (Section 4.2.1). Next, we present a list of predefined provenance-focused competency questions from our scenario and requirement analysis (Section 4.2.2). In the following, we report on our extensions to the PROV-O data model, which are tailored to these concerns (Section 4.2.3). Finally, we explain our approach to handle the mapping and query translation for provenance data captured from the system (Section 4.2.4).

4.2.1. Methodology3

Miles et al. [24] offer a methodology for developing provenance-aware applications, arguing for the identification of actors and data flows of applications. We apply and adapt a condensed form to define our auditability needs:

1. *Definition of provenance competency questions*: These questions are to validate that the correct information pieces have been gathered to answer them and will be provided in Section 4.2.2.
2. *Identification of main actors, activities and entities*: During the definition of provenance questions, we identified the main actors, activities and entities.
3. *Extension of provenance data model*: We have chosen PROV-O as the foundation for our auditability. However, adaptations are needed to answer our defined competency questions. The process from Step 2 and 3 will be described in Section 4.2.3.
4. *Mapping of system data sources to provenance data*: In this step, audit data artefacts from the system have to be analysed and mapped accordingly to the extended PROV-O.
5. *Translation of competency questions into queries on the data*: This step is concerned with the translation of questions defined in Step 1 into queries. We will briefly describe the process from Step 4 and 5 in Section 4.2.4

4.2.2. Auditability Competency Questions3

This subsection deals with the identification of audit-related questions from our scenario and requirements described in Section 2. In the process, we discussed the questions with both project members as well as our use case partners.

We group the questions according to the main actors: 1) *user*-centric that are relevant for users to gain trust in the system, e.g.: how data was used, which consent was provided,etc., 2) *analyst*-centric, focusing on search parameters, system variables and the software environment of the analyst.

User-centric questions:4

- If user data has been analysed, in which studies was it used?
- If user data has been analysed, which purposes did the study serve?
- What user consent was given for a dataset at a specific time?
- Which entities have analysed user data?

- What processing was conducted on user data?
- Which user data types were used for analysis?

Analysis-centric:4

- Which search attributes were specified for a certain study?
- Which software dependencies, deployment settings and software versions have been used when conducting a study?
- Are there differences in the results between two queries with the same search parameters but at a different time?
- Can these differences be related to user consent, additional data upload or a change in the platform/software environment?
- Which data transformation operations were conducted on data?
- Which software was used to analyse user data?
- What scripts were used to analyse user data?

4.2.3. Audit Data Model3

We decided to adopt PROV-O[5] and P-Plan[6], the former as the basis of our audit data model, due to its broad coverage of general provenance concepts, ease of use and extensibility [25], and the latter to have the capability to model plans. However, while PROV-O and P-Plan are established generic frameworks for provenance management, there is still a need to adapt to support data tracing within the scope of our platform.

One of the related works with regards to our Audit data model is GDProv [26], an effort to model provenance based on PROV-O and P-Plan to enable GDPR-related compliance queries. However, due to the auditing requirements in our use cases that are beyond consent, e.g., to model system configurations, analysis scripts and search parameters, we still need to extend GDProv to cater our needs.

During the definition of provenance questions, we identified two main processes that need to be modelled as plans (`p-plan:Plan`): (i) *Data Upload Plan*, during which user data and associated user consent is uploaded, and metadata is extracted from these uploaded inputs; and (ii) *Study Plan* associated with the querying of metadata and the analysis activities. In the following we will describe the design of our core audit data model (cf. Figure 4) to support the auditing capabilities of our platform for each plan.

[5]https://www.w3.org/TR/prov-o/

[6]http://vocab.linkeddata.es/p-plan/index.html

Data Upload Plan (`wprv:DataUploadPlan`)4

- Upload (`wprv:UploadStep`) constitutes the upload process of personal data to the platform. Inputs for this step are consent (`wdpv:Consent`) and user data coming from participating user applications (`wprv:InputDataset`). Dataset with consent (`wprv:DatasetWithConsent`) is the output from the upload step and describes the stored data, which is also input to the metadata extraction step.
- Metadata extraction (`wprv:MetadataExtractionStep`) takes as inputs the uploaded datasets to extract metadata (`wprv:Metadata`).
- UpdateUserConsent (`wprv:ConsentUpdateStep`) is an optional step in case consent for a dataset gets updated.

Study Plan (`wprv:StudyPlan`)4

- Metadata Query step (`wprv:MetadataQueryStep`) stores the information associated of querying the triplestore based on an analyst's query described in (`wprv:StudyContext`). The output of this step is a query result (`wprv:MetadataQueryResult`).
- Dataset transformation (`wprv:DatasetTransformationStep`) in case the analyst proceeds with the study after seeing how many records qualify for the search and consent, the transformation step captures the necessary information to create the (`wprv:TransformedDataset`) for further analysis.
- Data anonymisation (`wprv:DatasetAnonymisationStep`) is an optional step for cases where the Analyst wants to use the privacy preserving data publishing feature. When the step is executed, the transformed dataset is used as an input and anonymised.
- Data analysis (`wprv:AnalysisExecutionStep`) describes the processing of the data for studies within the *Trusted Analysis Environment*. Input variables are: Script (`wprv:Script`), and the transformed study dataset. The output is stored in a study result (`wprv:StudyResult`).

We do not depict the agents and entities in Figure 4 for ease of readability. Relevant agents are the analysts (`wprv:Analyst`) acting on the behalf of their organisations (`prov:Organization`) in accessing the platform. Analysts execute queries to plan their experiments and do analysis. Another relevant entity is the WellFort platform itself (`wprv:System`),

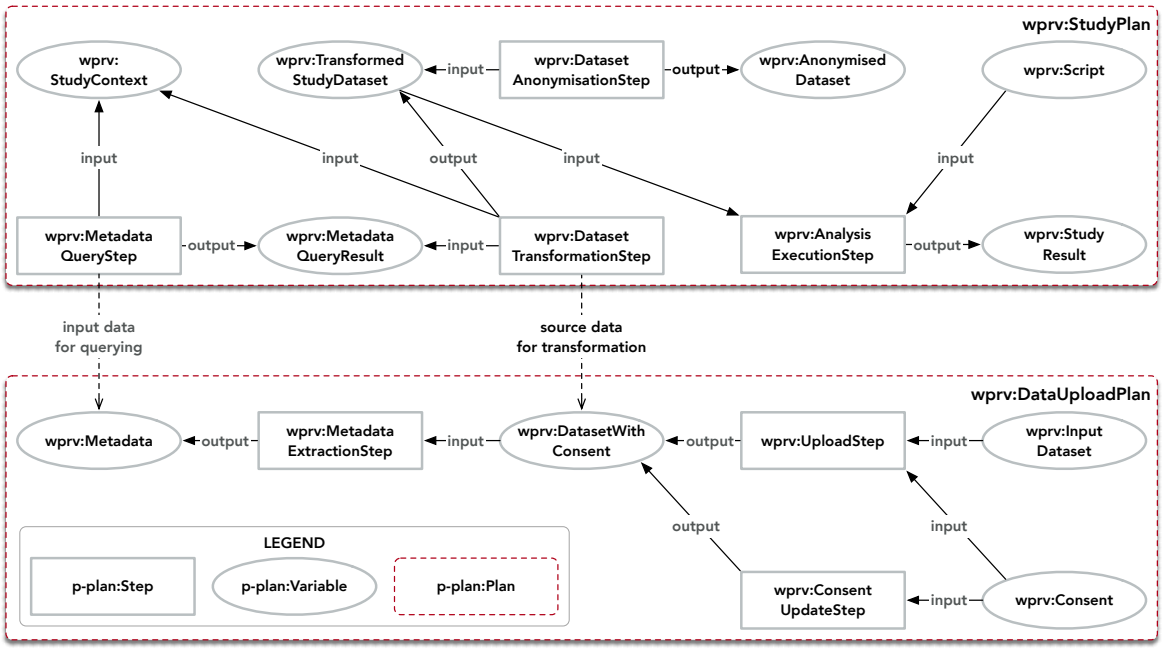

Fig. 4. Study and Data Upload plan of WellFort audit data model

which provides the capabilities of data upload, selection and analysis. Furthermore, we defined corresponding sub-classes of `p-plan:Activity` and `p-plan:Entity` to represent the instances of provenance data in our ontology. For the complete ontology, we refer interested readers to look into the ontology[7].

4.2.4. Data Source Mapping and Query Translation3

The *Audit Box* component in WellFort architecture provides a basic building block for audit data management previously explained in Section 3.4.

In the *provenance collector* of the *Audit Box*, we define a set of mappings between the raw provenance data provided by the other two components of the architecture, namely *Secure Repository* and *Trusted Analysis Environment*, and our extended PROV-O data model to ensure that we can retrieve and collect the provenance data properly. Potential mapping technologies include RML for structured data [27] or ONTOP [28].

Since we store the provenance data as RDF graphs, we naturally rely on the SPARQL query language to formulate auditor queries to the system. To this end, we support the auditors by implementing the competency questions described in Section 4.2.1 as predefined queries and provide it to the user in the *Auditor Interface*. Furthermore, it is also possible for auditors with advanced SPARQL capabilities to implement their own SPARQL queries on their audit process.

[7] http://w3id.org/wellfort/ns/prov

5. Prototype1

Our overall guidelines when implementing the prototype of our platform were as follows:

- For data handling, we re-use proven data formats, standards, and ontologies as much as possible, to avoid common pitfalls when designing from scratch, and to foster re-use and interoperability with other systems.
- For software components, we re-use existing, well proven and tested components, e.g., for the basic repository system, database servers or analysis platform, and focus our effort on orchestration and integration of these components into a functional platform serving the intended purpose of providing auditable, privacy-preserving data analysis.

In the following, we provide details on our prototype and the technologies utilised.

5.1. Secure Data Repository2

We implemented the *Data Repository* using the *Invenio framework*[8], which is used by institutions to build data repositories that enable publication of data according to FAIR principles. The invenio framework allows us to store and organise datasets uploaded from users' applications, assign PIDs to datasets, and manage access control. Invenio framework is a modular web application that is implemented in python. We added custom-built modules to achieve the needed functionality.

We implemented the *Extractor* and *Controller* modules as Flask[9] services inside the web application. They are responsible for automated extraction of metadata and consent, as well as managing the experiment. The *Experiment Setup Interface*, also built with Flask, is the only component of the Secure Data Repository that the Analyst can interact with. It performs validation of the user input, provides pages for search queries, and functionality to setup the Analysis Environment. The *Loader* component was implemented with watchdog[10], and is capable to supervise file system events, such as file creation. Its purpose is to *listen* for specific file patterns created (i.e., of personal data), that in turn invoke the import of the data to the analysis database.

The core part of the *Triplestore* component is built using SparkJava[11] and RDF4J[12], with GraphDB[13] as its storage component. The *Triplestore* is responsible for retrieving a set of dataset ids based on the evaluation setup provided by an Analyst, taking into account the availability of the requested data, as well as the compliance between user consents and the setup.

To this end, we developed a *Usage Policy Compliance Check* engine to complement the *Triplestore* with a compliance check mechanism, which is built using the OWL-API[14] and Hermit reasoner [29]. We explained the compliance check mechanism in Section 4.1.3, where user consent and experiment setup data is structured into OWL class restrictions and checked for their compliance using OWL2 reasoning, ensuring that only dataset ids with compliant consent are returned to the Analyst. We describe the data structure, which is based on SPECIAL Policy Language and DPV, on Section 4.1.2

The *Privacy-Preserving Data Publishing* module is implemented using open-source tools for data anonymisation. There are two methods enabling privacy-preserving download: (i) k-anonymisation [12] and (ii) synthetic data generation. For *k*-anonymisation we utilise ARX Data Anonymisation Tool[15]. The tool allows for extending the privacy guarantee of the downloaded data by supporting multiple additional data privacy models, such as *l*-diversity, *t*-closeness, etc. The choice of identifiers and quasi-identifiers is done inside the platform and is based on DPV vocabularies described in Section 4.2.3. Direct identifiers are described by `dpv:Identifying` category and are suppressed for the data download. Quasi-identifying attributes are those falling under a subset of class `dpv:PersonalDataCategory`, such as `dpv:Ethnicity`, `dpv:Geographic`, e.g. home address and `dpv:PhisicalCharacteristics`. The Synthetic Data Vault (SDV)[16] is used for generating synthetic data in the *Privacy-Preserving Data Publishing* module. It is a Python library based on probabilistic graphical modelling and deep learning for generating synthetic data that has the same format and statistical properties as the original data.

5.2. Trusted Analysis Environment2

The *Trusted Analysis Environment* contains the mechanisms for secure and privacy-preserving data analysis. The mechanism is provided by the analysis server, the analysis database, where the study data are temporarily stored, and the analysis interface, which allows the Analyst to interact with the data.

Analysis Server4 For this component, we rely on Opal[17], which is a data management system that provides tools for importing, transforming, describing and exporting data. Data managers can securely import a variety of data types, e.g., text, numerical data, images, geo-localisation, etc. and formats, e.g., from SPSS or CSV.

[8] https://inveniosoftware.org/
[9] https://flask.palletsprojects.com
[10] https://github.com/gorakhargosh/watchdog
[11] https://sparkjava.com/
[12] https://rdf4j.org/
[13] https://graphdb.ontotext.com/
[14] https://owlcs.github.io/owlapi/

[15] https://arx.deidentifier.org/
[16] https://sdv.dev/
[17] Opal is open-source and freely available at www.obiba.org under a General Public License (GPL) version 3, and the metadata models and taxonomies that accompany them are available under a Creative Commons licence.

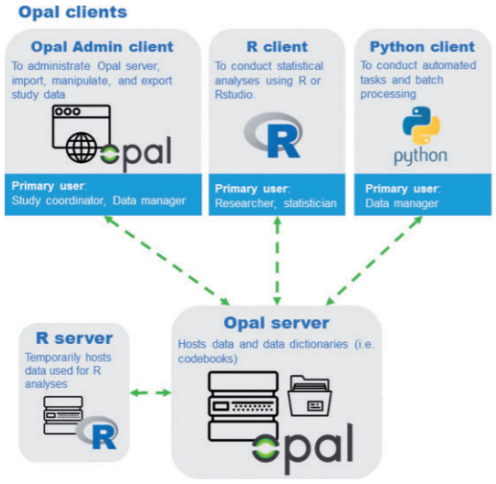

Fig. 5. Connection between Opal and clients [30]

Opal is built on REST web services, which allows connections from different software web clients using the HTTPS protocol, as shown in Figure 5. These clients are designed to ensure a broad range of specialised tasks to be achieved by different users. For example, a Python client allows data managers (in this case our platform administrator) to automate repetitive tasks such as data imports and access permission management. An R client, e.g., RStudio, allows conducting statistical analysis of the data stored on Opal. Opal provides the access to the *Analysis Database* to the authenticated users. The *Loader* sets the user permissions for the data tables loaded into *Analysis Database* so that the Analyst can access them thought the *Analysis Interface*. This allows the Analysts to operate only on the data that has been retrieved for studies they initiated. Client authentication is performed using either certificates, username/password, or token mechanisms.

Analysis Database4 In the *Trusted Analysis Environment*, Opal is connected to the *Analysis Database*, a PostgreSQL[18] database that temporarily stores the data relevant for the analyses. The tables from all studies conducted on our platform are imported as CSV and JSON files from the *Loader* and queried only through the *Analysis Interface*.

The details about the data formats within the *Analysis Database* are provided in Section 5.4.2.

Analysis Interface4 In our prototype platform, we utilise the R client of Opal as an interface where the Analyst conducts her studies on selected data using R DataSHIELD packages. DataSHIELD [13] is an open-source software that provides a modified R statistical environment linked to an Opal database. It was originally developed for remote, non-disclosive analysis of biomedical, healthcare and social-science data, although it can be used in any setting where micro-data (individual-level data) must be analysed but cannot or should not physically be shared with the research users for various ethico-legal reasons, or when the underlying data objects are too large to be shared. DataSHIELD implements >100 statistical methods[19] that embed privacy-protection traps, which prevent the Analyst from identifying individual data subjects or inferring the value of particular variables in given subjects based on analytic results. For instance, subsetting is functionally restricted by not generating any subset data set containing, by default, only 1-4 observations since results based on these subsets might be disclosive.

RStudio Server[20] is employed as a user interface of our platform, which an Analyst uses to connect to Opal and perform privacy-preserving analysis via DataSHIELD functions.

5.3. Audit Box2

The *Audit Box* collects and integrates provenance data from the entire platform, and provides an interface for Auditors to audit processes within the platform. The *Audit Box* consists of three components of which the implementation will be described in the following.

Provenance Collector. The task of this component is retrieving and transforming raw provenance data into an integrated RDF graph. In our prototype, we implement this component as a set of REST services. The services accept raw provenance data in JSON format from various components in the *Secure Repository* and *Trusted Analysis Environment* and transform it into PROV-O compliant RDF graphs.

Specifically for *Trusted Analysis Environment*, we rely on an existing tool to collect provenance from R-Scripts[21]. RDT format its provenance data using an extended PROV-DM[22] model and allows to collect all relevant information from the analysis environment

[18] https://www.postgresql.org/

[19] https://rdrr.io/github/datashield/dsBaseClient/man/

[20] https://rstudio.com/

[21] https://github.com/End-to-end-provenance/rdtLite

[22] https://www.w3.org/TR/prov-dm/

```
{
  "description": "...",
  "consent": [ {
      "data_type": "Demographic,HealthRecord",
      "processing": "PseudoAnonymise",
      "purpose": "Security",
      "recipient": "CompanyH",
      "until": "2022-02-10"
    } ],
  "execution": {
    "invenio_ts": "2021-01-20T21:23:53",
    "end": "2021-01-20T21:23:53",
    "start": "2021-01-20T21:23:52"
  },
  "triggered_by": "api.py:40",
  "invenio_id": "378",
  "patient_id": "243",
  "upload_id": "0a5c1d64"
}
```

Listing 3: An example of raw provenance data

such as operating system, language version, and script directory. We use RML mappings [27] and the caRML engine[23] to execute the transformation from the raw data into RDF graphs.

We provide an example of raw provenance data produced by the *Secure Repository* in Listing 3 and the respective RDF graph result in Listing 4. These examples capture the provenance of the *upload* process, where a user finished uploading her personal data with a consent to the *Secure Repository*. Both the upload activity and the uploaded dataset are captured as instances of `prov:Activity` and `prov:Entity` respectively. We show in the resulted provenance graph that the upload process is associated with a python script `api.py` and executed in one second.

Knowledge Base. We collected the provenance graph from the Provenance Collector into the Knowledge Base component. This component stores and manages the provenance data, which is necessary to support Auditors in conducting their tasks. Similar to the metadata store in the *Secure Repository*, we utilise GraphDB as our Knowledge Base for *Audit Box*. The communication between GraphDB and the Provenance Collector is handled using Eclipse RDF4J[24] that allows seamless I/O communication between them.

Auditor Interface. In our current prototype, the Auditor Interface relies on the SPARQL user interface of GraphDB, both for storing the pre-defined queries for the audit-related competency questions described in Section 4.2.1 and to allow auditors to formulate their own queries related to their tasks.

[23]https://github.com/carml/carml
[24]https://rdf4j.org/

```
# upload activity
id:upload-0a5c1d64 a prov:Activity, wprv:Upload ;
  prov:startedAtTime
    "2021-01-20T21:23:52"^^xsd:dateTime ;
  prov:endedAtTime
    "2021-01-20T21:23:53"^^xsd:dateTime ;
  prov:wasAssociatedWith id:script-api-py-40 .

# uploaded dataset
id:dataset-303 a prov:Entity, dcat:Dataset ;
  prov:wasGeneratedBy id:upload-0a5c1d64;
  prov:generatedAtTime
    "2021-01-20T21:23:53"^^xsd:dateTime ;
  prov:wasAttributedTo id:patient-367 ;
  :hasConsent id:consent-0a5c1d64.

# consent
id:consent-0a5c1d64 a dpv:Consent, owl:Class ;
  owl:equivalentClass [ owl:intersectionOf (
    [ a                    owl:Restriction ;
      owl:onProperty       dpv:hasPersonalDataCategory ;
      owl:someValuesFrom [ owl:unionOf
            ( dpv:Demographic dpv:HealthRecord ) ] ]
    [ a                    owl:Restriction ;
      owl:onProperty       dpv:hasProcessing ;
      owl:someValuesFrom dpv:PseudoAnonymise ]
    [ a                    owl:Restriction ;
      owl:onProperty       dpv:hasPurpose ;
      owl:someValuesFrom dpv:Security ]
    [ a                    owl:Restriction ;
      owl:onProperty       dpv:hasRecipient ;
      owl:someValuesFrom :CompanyH ]
    [ a                    owl:Restriction ;
      owl:onProperty       dpv:expiryTime ;
      owl:someValuesFrom [
        a                      rdfs:Datatype ;
        owl:onDatatype         xsd:dateTime ;
        owl:withRestrictions ( [ xsd:maxInclusive
            "2022-02-10T23:59:59"^^xsd:dateTime ] )
    ] ] ) ] .
```

Listing 4: The resulted RDF graph from Listing 3

5.4. Data Model2

This section explains the data model used in our prototype implementation: (i) a domain-specific vocabulary on health record and (ii) the analysis data model for our *Analysis Database*.

5.4.1. Domain Specific Vocabularies: Health Record3

The proposed conceptual architecture does not impose limitations on the formats of the data used in the Data Repository and therefore, the data model depends on the specific use case. For the prototype presented in this paper, we consider the following characteristics for our health record data model:

1. **Based on open-source standards** that guarantees stable specifications for platform users.
2. **Designed for interoperability and information exchange**, which is critical in the healthcare domain due to the heterogeneity of data and stakeholders [31]; and

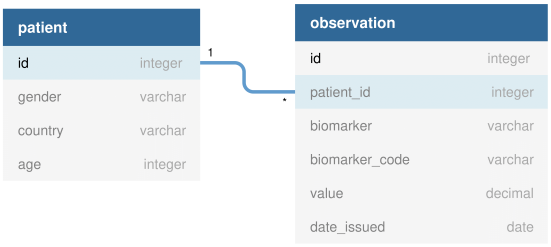

Fig. 6. *Analysis Database* schema

3. **Widely used in the healthcare communities** to allow easier adoption for stakeholders that are already familiar with the data model.

Based on these criteria, we decided to use the Fast Healthcare Interoperability Resources (FHIR) developed by Health Level 7 International (HL7) as the data format in our Data Repository for our prototype. HL7-FHIR is the latest standard by HL7 for electronically exchanging healthcare information based on emerging industry approaches [32]. FHIR offers a number of advantages for prototype development, such as its solid foundation in Web Standards and supports for RESTful architectures.

In our use case, we used FHIR to represent Patient, Diagnostic Report, and Observation. While FHIR recommends the use of LOINC[25] for observation data, we decided to use the custom classification codes provided by our use case partners for our prototype for practical reasons.

5.4.2. Analysis Database3

The privacy-preserving nature of the *Trusted Analysis Environment* comes with the price of reducing the data utility and flexibility of an Analysis compared to full data access. Another challenge is that the data originating from different sources needs to be transformed to the same common format, while keeping the data intuitive for Analysts to work with. Therefore, in the *Analysis Database* we aim to keep the most relevant data for the analysis in a rather concise but comprehensive format.

The database consists of two default tables defined by the types of FHIR resources *Patient* and *Observation*:

- *Patient*: provides information about the users included in the study, such as sex, age and country.

- *Observation*: contains information about the specific measurements for a user. A biomarker can be any attribute contained in the source databases, for instance, blood sugar, activity level, etc. Each biomarker is represented with its code (*biomarkerCode*) and full name (*biomarker*). A patient can have multiple measurements of the same biomarker measured at a different moment in time, so the attribute *issued* provides the information of when the specific measurement has been taken.

The two tables are connected by a foreign key *Observation:patientId - Patient:id*. The database schema can be seen in Figure 6.

As specified by Opal, each table is represented by (i) one CSV file called *data table* and (ii) one JSON file called *data dictionary*.

- *Data table*: the columns of this table correspond to the attributes in the actual table *Patient*/*Observation*. Each row is one data entry. An example data table file is shown in Figure 7.

- *Data dictionary*: this file is used to describe one data table. These tables must follow the scheme defined by Opal which imposes a set of fields such as *entityType*, *index*, *name*, *unit*, *valueType*, etc. for describing each of the attributes form the *data table*.

```
id,patientId,biomarker,value,biomarkerCode,issued
0,0,Diet,normal-diet,diet,2019-06-11T12:32:24Z
1,1,Diet,normal-diet,diet,2019-02-07T09:13:07Z
2,2,Diet,low-carb-diet,diet,2019-02-01T14:56:31Z
3,1,Vitamin B12,157.13,VIT-B12,2019-02-07T12:32:24Z
```

Fig. 7. An excerpt of data table CSV file for table *Observation*

6. Evaluation1

In this section, we demonstrate the platform through representative use cases. In the summary we assess how well the demonstrated scenarios answer to the requirements posed in Section 2.

6.1. Data2

For the evaluation we use data originating from two sources, both from SMEs that provide a health-care or lifestyle related application to end-users. For confidentiality reasons, we denote these sources as Company (or Datasource) M and Company H. A subset of the attributes provided by these two data sources are shown in Table 2. We generated 1100 synthetic datasets for 600 users based on the original data from both companies with varying consent configurations. Each dataset

[25] https://www.hl7.org/fhir/valueset-observation-codes.html

Table 2
Attributes of evaluation data

Datasource M		Datasource H	
attribute	type	attribute	type
Age	int	Birthday	date
Sex	text	Sex	text
Country	text	Sympathicus activity	dec
Blood Glucose	dec	Parasympathicus activity	dec
Lipids	dec	Heart rate	dec
Liver	dec	Pulse/Respiration Quotient	dec
Kidney	dec	Vitality index	dec

contains a single record for patient data (e.g., ID, age, and country) and health observation attributes (cf. Table 2) that we observe in the use cases for a single day. The synthetic datasets are generated using the Synthetic Data Vault (SDV)[26] model trained with original datasets provided by companies M and H. Further details on the evaluation users, datasets and the associated user consent is given in Table 3, where **processing** and **expiry** refer to the dataset consent for processing and consent expiry time, respectively. We set the consent of all user data to the default values: `dpv:hasPurpose` is set to `dpv:AcademicResearch` and `dpv:hasRecipient` is set to `:TUWien`.

Table 3
Synthetic users and datasets for the evaluation

users	dataset	processing	expiry
u0-49	ds0-49 (H) ds550-599(M)	dpv:Processing	2021-01-05
u50-199	ds50-199 (H) ds600-749 (M)	dpv:Processing	2021-01-10
u200-399	ds200-399 (H) ds750-949 (M)	dpv:Adapt, dpv:Combine, dpv:Analyse	2021-01-10
u400-499	ds400-499 (H) ds8=950-1049 (M)	dpv:Analyse	2021-01-10
u500-549	ds500-549 (H)	dpv:Processing	2021-01-10
u550-599	ds1050-1099 (M)	dpv:Processing	2021-01-10

6.2. Use cases2

We define four use cases which will be processed in our platform:

- UC1: *Descriptive analysis of data from a single data source*
- UC2: *Descriptive analysis of data from multiple data sources*
- UC3: *Predictive modelling*
- UC4: *Data publishing*

UC1, UC2 and UC3 describe scenarios of using *Trusted Analysis Environment*, while UC4 describes the purpose of *Privacy-Preserving Data Publishing* component. UC1 and UC3 deal with data originating from a single data source, while presenting the spectrum of functionalities provided to the Analyst. UC2 focuses on a study utilising data from multiple sources. Within each use case, we provide a discussion on privacy concerns of the respective analyses.

We simulate the following scenario to introduce dynamic aspects:

- Initial state: all user datasets are uploaded and collected in the *Secure Repository*.
- Day 1 (05-01-2021): UC1 and UC2 are executed. UC1 analysis setup (expiry: 05-01-2021; processing: `dpv:Adapt` and `dpv:Analyse`) yields 450 datasets in total from Datasource H. UC2 analysis setup (expiry: 07-01-2021; processing: `dpv:Adapt`, `dpv:Combine`, and `dpv:Analyse`) yields 800 datasets in total, 400 each from Datasource H and Datasource M.
- Day 2 (06-01-2021): User consent from 50 users expired (u0-u49). Furthermore, 50 users (u200-u249) removed `dpv:Adapt` from their consent
- Day 3 (07-01-2021): UC3 and UC4 are executed. Due to the changes and expiration of some user consents, UC3 analysis setup (expiry: 07-01-2021; processing: `dpv:Adapt` and `dpv:Analyse`) only yields 350 datasets from Datasource H, which is less than UC1 with an identical evaluation setup. UC4 analysis setup (expiry: 07-01-2021; processing: `dpv:Share`, `dpv:Anonymise`, and `dpv:Derive`) yields 200 datasets from Datasource M.

The flow for each of our use cases shares some common steps:

1. The **Analyst** initiates their study by selecting relevant attributes, specifying the study purpose and study expiry date.
2. the **platform** yields the available datasets according to:
   - the user consent that needs to match with the Analyst's study purpose
   - the expiry date of data consent that needs to be later than the study expiry date.

[26] https://sdv.dev/

The number of the available datasets needs to exceed the predefined *threshold*, otherwise the analysis is considered to be potentially disclosive and the platform terminates the study setup. For our evaluation, we set the *threshold* to 10.

3. For each study, after initialisation, the **platform** transforms the available datasets into the files describing two relational tables, *Patient* and *Observation*, as discussed in Section 5.4.2.
4. For each study the **platform** defines the user permission in Opal for the Analyst that allows using data via DataSHIELD - this way each Analyst only sees the studies they initiated.
5. When the study is initiated, the **Analyst** uses DataSHIELD and their credentials to connect to Opal and perform the analysis in case they use the *Trusted Analysis Environment*, or download their data if they use the *Privacy-Preserving Data Publishing* component.

6.2.1. *UC1: Descriptive analysis of data from a single source3*

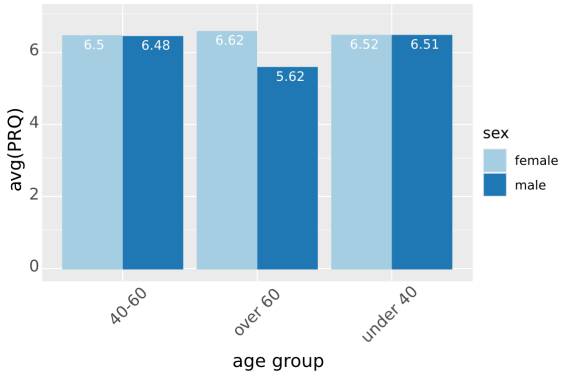

Fig. 8. UC1 (Single source use case example): Average PRQ of patients

In our first use case, we assume the Analyst wants to conduct a set of descriptive studies to visualise selected attribute values across the population of platform users. Let this attribute be *Pulse/Respiration-Quotient (PRQ)* originating from the database of *Datasource H*. The Analyst then wants to study how the *PRQ* values differ between age groups and sex.

In our evaluation, the platform finds 450 participants with a set of records of the selected attributes, therefore the *Patient* table contains 450 records of users' age and sex. and *Observation* table 450 records of *PRQ*.

Figure 8 shows the results of the analysis, i.e. the average *PRQ* for different age groups (under 40, 40-60 and over 60) for male and female participants separately. The averages of *PRQ* are obtained by DataSHIELD functions and then plotted using the native R plotting package `ggplot`. The Analyst uses a set of data manipulation and statistical functions from DataSHIELD, for instance:

- Joining tables via `ds.cbind`: the Analyst needs to join the tables Patient and Observation by *patient_id*.
- Subsetting via `ds.dataFrameSubset`: this function is used to subset the table joint in the previous step by the values of the attribute age and sex.
- Mean function via `ds.mean`: the Analyst uses this function to calculate the mean of each subset.

The listed DataSHIELD functions for subsetting and calculating the mean value are potentially disclosive functions. For example, applying subsetting repeatedly might lead to revealing the actual values from the table. DataSHIELD achieves privacy-preserving versions of these functions by prohibiting subsetting and calculating simple statistics, such as mean, on less than 4 data samples. Functionality for joining tables is implemented such that the method does not return any value to the Analyst, except for potential error messages. The joined table is persisted on the server in a variable, which the Analyst can use in other functions only by name, therefore no values can be disclosed in the process.

6.2.2. *UC2: Descriptive analysis of data from multiple sources3*

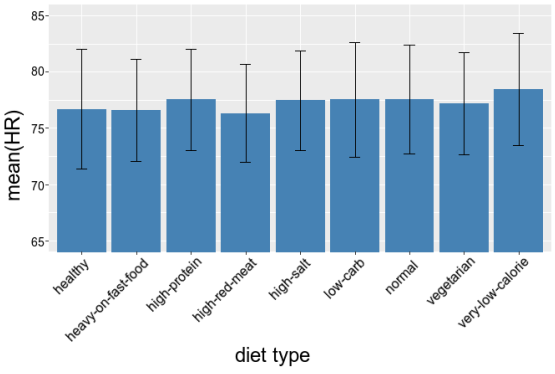

Fig. 9. UC2 (Multiple source use case example): Mean and standard deviation of heart rate of users with different dieting profiles.

In this use case, the Analyst wants to conduct a study on attributes that are originating from separate data

sources. For example, the Analyst wants to compare the resting heart rates of people with different dieting profiles, such as vegetarian diet, low-carbohydrate diet, etc. For that, the platform needs to collect the data from two applications. *Datasource M* provides the attribute *diet* and *Datasource H* provides the attribute *heart rate*. This analysis is only possible on users which use both applications. Users control the linking of their data and have to enable linking between those applications upon data submission (in their applications).

In our evaluation, the *Patient* table contains 400 user records and *Observation* table contains 800 records for these 400 users; 400 records of heart rate and 400 of diet type. The Analyst groups the patients by their diet type, and then for each dieting group calculates the mean and standard deviation of the heart rate values. To highlight a few steps, let us list the specific DataSHIELD functions that allow these operations:

- Subsetting via `ds.dataFrameSubset`: splitting the Observation table depending on the attribute 'biomarker' such that each subtable contains one type of biomarker, i.e. one table for diet, another table for HR.
- Joining tables via: `ds.merge`: joining the subtables from the previous step on attribute *patient_id* and in that way pivot the *Observation* table into a table with the new set of columns: [patient_id, diet_value, HR_value, ...]
- Aggregation via `ds.meanSdGp`: given that the Analyst subsets the pivoted table from the previous step by *diet_value*, this function obtains the mean and standard deviation of heart rate values for each dieting group.

The resulting analysis is shown in Figure 9. It shows the mean and standard deviation of the heart rate value for individuals in different dieting groups.

From the privacy-preserving perspective, the function `ds.merge` is achieved in the same manner as previously mentioned `ds.cbind`, by not returning any outputs to the Analyst. The implementations of aggregation functions are achieved similarly as subsetting, by prohibiting operations on less than 4 data samples.

6.2.3. UC3:Predictive modelling3

The Analyst wants to conduct a more sophisticated study to predict the value of the Vitality Index (a measure of the resources that are available to the organism for activity and health maintenance). To that end,

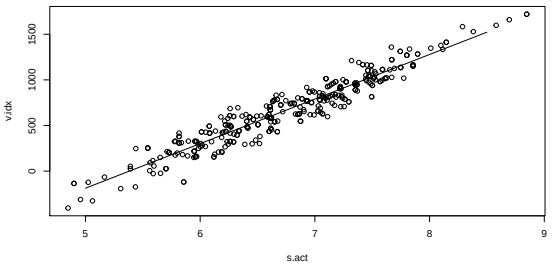

Fig. 10. UC3: Univariate GLM for target attribute Vitality Index and predictive attribute Sympathicus Activity

```
$family

Family: gaussian
Link function: identity

$formula
[1] "v.idx ~ s.act + ps.act + sex"

$coefficients
               Estimate Std. Error    z-value
(Intercept) -2332.955688   61.37485 -38.011594
s.act         359.214501   15.10462  23.781766
ps.act        114.199106   11.34462  10.066368
sexmale        -8.615435   19.75788  -0.436046

               p-value   low0.95CI  high0.95CI
          0.000000e+00 -2453.24818 -2212.66320
         5.157566e-125   329.60999   388.81901
          7.779775e-24    91.96406   136.43415
          6.628034e-01   -47.34008    30.10939
```

Fig. 11. UC3: Output of multivariate GLM for target attribute Vitality Index and predictive attributes Sympathicus Activity, Parasympathicus Activity and Sex

the Analyst wants to learn and apply a generalised linear model (GLM) that describes the linear combination of the attributes *sex*, *parasympathicus activity* and *sympathicus activity* and enables predicting the expected value of the attribute *Vitality Index* of a person. Such a predictive model functionality is provided as a DataSHIELD function, and thus readily available.

Datasource H provides the data about vitality index, sympathicus and parasympathicus activity and sex of its patients, thus this analysis task is a single source setting. The Patient table contains 350 records and Observation table 1050 records of vitality index, parasympathicus activity and sympathicus activity for the same set of patients.

The Analyst is now free to use any type and number of DataSHIELD and native R functionalities to obtain privacy-preserving results. We single out some specific DataSHILED methods used in this study:

- Generalised Linear Model via `ds.glm`: this function is used for fitting GLM. The implemen-

tation supports binomial, gaussian and poisson error distribution functions.
- Visualisation via `ds.scatterPlot`: one of the non-disclosive plotting functions in DataSHIELD. This function may be used in this study to plot data points, taking into account two attributes, e.g., Vitality Index and Sympathicus Activity.

One option for the Analyst is to examine the relationship between Vitality Index and one of the aforementioned attributes by fitting a univariate GLM. The Analyst chooses sympathicus activity as a predictor attribute for Vitality index and Gaussian error distribution. Figure 10 shows the distribution of data points and the estimated linear model for target attribute Vitality Index, using Sympathicus Activity as a predictive attribute.

Furthermore, to fit a more complex model, i.e. multivariate GLM, the Analyst in addition includes more predictive attributes: sex and parasympathicus activity. Figure 11 shows the output of the DataSHIELD function `ds.glm`. The Analyst sees the coefficient estimates, standard error, p-value indicating significance of a predictor for the model, etc.

For both univariate and multivariate analysis, the output of `ds.glm` are the learnt estimates of regression coefficients and their standard errors, both of which intrinsically do not disclose any personal information. The Implementation of the privacy-preserving GLM model for the usage with the data schema of our platform does not differ from its original form in native R. However, a scatter plot generated from the original data is disclosive, therefore not permitted in DataSHIELD in its original form. There are rather two implementations of the privacy-preserving scatter plot available: (i) based on k-Nearest Neighbours (kNN) and (ii) based on adding random noise. In the first implementation, instead of the actual data point, kNN for each data point is calculated and the average value is plotted. k is chosen by the Analyst, but cannot be set to a value smaller than 3. In the second implementation, a random normal noise is added to the x and y coordinate of each data point. In this analysis the kNN method is used with $k = 3$.

6.2.4. UC4: Data publishing3

In this use case, the Analyst wants to download anonymised data so that they can perform an offline analysis on the given data, using other data analysis tools rather than the *Trusted Analysis Environment*. Reasons for this might be of technical nature, such as programming language preference, familiarity and reliance on another analysis framework, required functionalities that are not covered by DataSHIELD, etc. Furthermore, the Analyst might be interested in the structure of the data and particular data values rather than statistical analysis, or wishes to avoid being time-constrained for conducting their studies.

Table 4
Original dataset

id	sex	age	country	Total Cholesterol
382	female	32	PT	3.38
24	female	32	BG	4.58
239	male	25	NL	4.86
253	female	38	AT	4.42
293	female	25	BG	5.08
77	male	63	PT	4.6
212	male	53	HR	5.45

The Analyst chooses attributes *sex, age, country* and *Total Cholesterol* provided by the Company M. In the k-anonymised dataset the direct identifiers such as user full name or any kind of IDs are not displayed at all. Other indirectly identifying attributes with personal information such as user country, age and sex, i.e. quasi-identifiers, are generalised according to the hierarchies predefined by the platform for quasi-identifying attributes as discussed in Section 5.1:

- *sex*
  - Level 1: * (i.e.fully generalised)
- *age*
  - Level 1: [20,40[, [40,60[, [60, 81[
  - Level 2: [20, 81[
- *country*
  - Level 1: SE (South Europe), SEE (South-east Europe), CE (Central Europe), ...
  - Level 2: *

Furthermore, the exact values of non-identifying attributes such as *Total Cholesterol* remain in their original form, since the possibility of re-identify people from the database based on this information is minute. Tables 4 and 5 show a subset of data before and after applying k-anonymisation, respectively, where only the later is the output of this analysis.

6.3. Auditability2

In this subsection, we focus on the evaluation of consent and provenance mechanisms that enable au-

Table 5
Anonymised dataset

id	sex	age	country	Total Cholesterol
*	female	32	*	3.38
*	female	[20,81[	*	4.58
*	male	25	*	4.86
*	female	38	AT	4.42
*	female	[20,81[	BG	5.08
*	male	63	PT	4.6
*	male	[40,60[	SEE	5.45

ditability of the platform within the context of the previously described use cases.

Auditor Queries4 We stated in Section 2.2 that provenance data must be automatically captured from each platform component to cover the whole data lifecycle within the platform. Here, we demonstrate the capability of the system in using the captured provenance data to answer Auditor queries.

We selected four questions that arose from the context of the use cases to verify our designed provenance and consent capabilities. Several questions, i.e., Q1-Q3, are parts of the common auditability competency questions described in Section 4.2.1. However, in some cases, Auditor may require more complex queries that are not in the pre-defined list. On these cases, the auditor can formulate her own SPARQL queries, such as in the case of Q4.

- Q1: *In which experiments was a dataset used?*
- Q2: *Which software has been used to analyse data in a specific study?*
- Q3:*Which query parameters were used for the selection of data in experiments?*
- Q4: *What is the ratio between analysed datasets in a study and the total number of datasets in the triplestore?*

Q1. Query Q1 addresses concerns regarding consent and data usage for study purposes (cf. Listing 5). In case of suspicions regarding the correctness of given consent and analysed data, auditors can verify which data was used for which purposes stated in the study description.

Q1 demonstrates the capability of an Auditor to establish such a connection between a study, user id, user dataset metadata, and consent attached to the dataset. Table 6 shows the answer to Q1, linking a dataset used in a specific study to a user. Based on this information, an Auditor can inform concerned users on which studies their data was used and under which purposes, similar to the situation described in the scenario provided in Section 2.1.

```
select distinct ?studyId ?datasetId
where {
        ?QueryResult prov:wasGeneratedBy ?query .
        ?QueryResult prov:hadMember ?ds .
        ?query rdf:type wprv:MetadataQueryActivity .
        ?query prov:used ?study .
        ?study rdf:type wprv:StudyContextEntity .
        ?study wprv:studyId ?studyId .
        ?ds rdf:type wprv:DatasetWithConsentEntity .
        ?ds wprv:datasetId ?datasetId .
    FILTER(?datasetId = "_DATASET_1050")
}
```

Listing 5: Q1- Experiments in which a specific dataset was used

Table 6
Results Q1- In which studies was a dataset used

studyId	datasetId
_EXPERIMENT_1	_DATASET_1050
_EXPERIMENT_2	_DATASET_1050
_EXPERIMENT_3	_DATASET_1050

```
select ?libraryName where {
    ?script rdf:type wprv:ScriptEntity .
    ?script prov:hadMember ?library .
    ?library rdf:type wprv:ScriptLibraryEntity .
    ?library rdt:name ?libraryName .
    ?exe rdf:type wprv:AnalysisExecutionActivity .
    ?exe prov:used ?script .
    ?result rdf:type wprv:StudyResultEntity .
    ?result prov:wasGeneratedBy ?exe .
    ?result wprv:studyId ?studyId .
    Filter(?studyId = "_EXPERIMENT_1")
}
```

Listing 6: Q2- Used libraries in a study

Table 7
Results Q2- An excerpt of used libraries in a study

libraryName
"base"
"datasets"
"devtools"
"dsBaseClient"
"..."

Q2. Query Q2 (cf. Listing 6) is concerned with the usage of software libraries applied in the analysis as described in *UC-2 descriptive analysis of data from multiple sources*.

In our prototype, the provenance trails of the libraries, environment, and scripts executed on the R-server are collected using the RDT-extension and stored in the Audit Box. Q2 provides an answer to the question of which software libraries are used in a spe-

```
SELECT ?studyId ?patientAttrs ?observationAttrs
WHERE { ?study a wprv:StudyContextEntity ;
  :attributesObservation ?observationAttrs ;
  :studyId ?studyId . OPTIONAL
  { ?study :attributesPatient ?patientAttrs }
}
```

Listing 7: Q3- Retrieving search parameters used in an Analyst query

Table 8
Results Q3- Search parameters Analyst queries

studyId	patientAttrs	observationAttrs
_EXPERIMENT_1	gender	*PRQ*
_EXPERIMENT_2		*diet, HR*
_EXPERIMENT_3	gender	*PSA, SA*
_EXPERIMENT_4	gender,age,country	*TCH*

cific study, with example results shown in Table 7. The purpose of retrieving such information is to increase the study reproduceability and also to ensure the legitimate existence of the analysis results.

Q3. Query Q3 refers to the search parameters for personal data categories from UC3 predictive modelling (cf. Listing 7).

Before being able to conduct an analysis, an Analyst has to provide details on which data categories of personal data are required for her analysis (e.g. Gender, Age, heart rate etc.; we also called them as *search parameters*) to check the availability of the data. This parameter, together with rest of the data handling setup, which includes the purpose of the analysis and expiry time, among others, are captured and stored in the *Audit Box*. Using this provenance information, we can retrieve specific search parameters used for a Query (see Table 8 for answers to Q3). This information can be used by an Auditor to check whether there are any suspicious activities conducted by certain analysts, e.g., if there are too many queries with varying search parameters by one analyst during a short amount of time.

Q4. Query Q4 (cf. Listing 8) shows the ratio of utilised datasets in studies compared to the actual available datasets in the triple store. This ratio (see Table 9) can be regarded as an indicator on how representative the study sample is in relation to the overall population and could serve as a robustness factor within the platform. To this end, there are two possible reasons why datasets were not available for analysis: (i) it could be that the consent was not applicable to the purpose given by the Analyst, or (ii) the dataset dose not contain the personal data categories required by the Analyst.

```
select ?result (?sumdata/?usdata as ?ratio)
where {
  { select ?result (count(?data) as ?sumdata)
    where {
      ?data rdf:type wprv:DatasetWithConsentEntity;
        prov:wasAttributedTo ?patient .
        ?data ^prov:hadMember ?result .
    } group by ?result } . {
    select (count(?sum) as ?usdata)
    where {
      ?sum rdf:type wprv:DatasetWithConsentEntity.
    } group by ?result
  }  ?result rdf:type wprv:MetadataQueryResultEntity
}
```

Listing 8: Results Q4- Retrieving the ratio between data sample and general user population

Table 9
Ratio of datasets analysed in a study compared to total datasets

queryResult	ratio
QueryResult-_EXPERIMENT_1	*0.409*
QueryResult-_EXPERIMENT_2	*0.727*
QueryResult-_EXPERIMENT_3	*0.363*
QueryResult-_EXPERIMENT_4	*0.181*

6.4. Summary2

In this section we described the behaviour of our platform throughout the concrete scenario that encompasses the dynamic aspects of the data and consent management, and 4 different data analysis use cases. The coherent user story allowed us to explore the application of our platform in a holistic manner, i.e. covering the behaviour of all components of our platform through one process and showing the feasibility of the proposed auditable privacy-preserving data analysis platform.

To summarise the evaluation, we now assess our platform according to the requirements defined in Section 2.2, targeting particular components of the platform. In Table 10, we explicitly list the requirements and indicate to which extent they are satisfied by the described scenario: fully, partially or unsatisfied. The listed requirements follow the structure from Section 2.2, therefore **R1.x** denotes requirements regarding privacy-preserving data analysis, divided into 3 subcategories, (i) data analysis, (ii) data publishing and (iii) usage policy management, and **R2.x** denotes requirements regarding auditability: (i) provenance data capture and tracing, (ii) provenance data inspection and analysis capability and (iii) metadata-retention after deletion.

Requirements R1.1, R1.2 and R1.5 are fully satisfied in the platform, which is shown by each use case

Table 10
Evaluation on the requirements

Requirement	full	partial	unsatisfied
R1.1: Input and intermediate data are not persisted by the platform	•		
R1.2: Analysis from a single data source	•		
R1.3: Analysis from multiple data sources		•	
R1.4: All computation on original data is inside the platform	•		
R1.5: Platform limits the period of time for storing analysis environment	•		
R1.6: No data download in original form	•		
R1.7: Anonymised data download	•		
R1.8: Synthetic data download	•		
R1.9: Measures for prohibiting the download of similar data			•
R1.10: Fine-grained consent	•		
R1.11: User access control	•		
R1.12: Automatic usage policy compliance checking	•		
R2.1: Data provenance traced in the whole data lifecycle	•		
R2.2: Automated trails of data access, processing and analysis.	•		
R2.3: Provenance of software components		•	
R2.4: Traces are available for each result		•	
R2.5: Trails do not contain private user data	•		
R2.6: Provenance data streams are accessible for users with auditing privileges	•		
R2.7: No personal data can be acquired by auditors	•		
R2.8: Users' consent for studies is still valid after revoking their consent	•		
R2.9: Minimum meta-information is kept for the past studies	•		

from our evaluation. The data for analysis is stored in the transformed format inside the analysis database only, and the outputs of the analysis are shown to the Analyst and not stored in the platform at all. Both of these intermediate products of data analysis are part of the analysis environment and are deleted together with the environment, which is defined by the data expiry. UC1 and UC3 directly support the requirement R1.2. R1.3, however, faces the limitations regarding user linkage across different platforms that we already discussed in the scope of UC2. Our platform offers the infrastructure to combine data with different schemes of data originating from different sources. However, record linkage methods impose some privacy and accuracy concerns [33], therefore we give the user full control over linking their data. As our solution depends on user engagement, we evaluate it as partially satisfied. Furthermore, requirements R1.6, R1.7 and R1.8 are described via UC4. R1.9 aims to address the potential surface for privacy attacks, when a malicious user has the access to multiple datasets, either similar anonymised or synthesised. We will address this issue as part of the future work by employing a query-auditing method. R1.10 and R1.11 are discussed in Sections 4.1 and 5.2, respectively. The fulfilment of R1.12 was demonstrated in the data selection scenario in Section 6.2, where the number of retrieved dataset for each UC depends on the data handling setup and the dynamics of user consent.

The second part of requirements concerns the auditability and provenance of the system: R2.1, R2.2 and R.2.5 are fully satisfied, shown through the automated capturing of provenance trails and the provenance management and analysis capabilities of the audit box. The Secure Repository and Trusted Analysis Environment both send log files in a predefined format, which is then mapped on our provenance model. R2.3 is currently enabled through RDT-Lite, a library capturing provenance in R-Scripts. R2.4 is also enabled through RDT-Lite, capturing the script as a directed graph. In the future, we are planning to extend its support to enable improved analysis capabilities, especially when heterogeneous data traces from other analysis environments, e.g. python scripts are integrated. Regarding R2.6, the audit box is a separate component of our architecture, where only qualified users have access to. Furthermore, it only captures metadata, hence no personal data can be acquired by auditors through

the audit box, which addresses R2.7. R2.8 and R2.9 are concerned with past studies: only data with valid consent is analysed through the platform. However, it may occur that users whose data has been analysed revoke their consent after a study has been conducted. In this case, the applicable data will be removed, and only pointers will remain for past studies.

7. Related Work1

This section discusses related work from three perspectives: privacy-preserving data analysis (Section 7.1, auditability (Section 7.2), and consent management (Section 7.3).

7.1. Privacy-preserving Data Analysis2

For achieving privacy-preserving data analysis, often two complementary approaches are considered - privacy-preserving data publishing (PPDP) [34], and privacy-preserving computation, such as secure-multiparty computation (SMPC) or homomorphic encryption.

Regarding PPDP, early approaches include e.g., k-anonymity [12], where data is transformed so that individual records can not be distinguished from $k-1$ other records, and thus re-identification becomes difficult. Several implementations perform these transformations, e.g., ARX [35].

Differential privacy [36] aims at maximising the accuracy of responses to queries to databases, while minimising the likelihood of being able to identify the records used to answer them – whether a specific instance is included in the dataset should not increase the privacy of the individual represented. Systems implementing differential privacy include e.g., GUPT [37], PINQ [38], Airavat [39].

Recently, synthetisation of data, which creates an artificial dataset that however still preserves global properties of the original data, is emerging as an alternative method [14]. One approach is e.g., the Synthetic Data Vault (SDV) [40]. SDV builds a model of the data based on estimates for the distributions of each column. In order to preserve the correlation between attributes, the synthesizer applies a multivariate version of the Gaussian copula and, subsequently, computes the covariance matrix. This model is then used to generate new, synthetic samples.

While these solutions provide building blocks of privacy-preserving data analysis, they do not provide a complete system comprising data collection, consent management, data transformation and the actual privacy-preserving data analysis. We will review related work that addresses these aspects to some degree. A recent report on Trusted Research Environments [41] elaborates on requirements for such environments in the health domain.

A system utilising block-chain to trace the usage of federated data sources in an analysis process is presented in [42], also providing a mechanism to check for consent.

The DEXHELPP project [43] focuses on safe-haven like settings, e.g., allowing the user a relatively unrestricted access to data inside a specific remote computing environment, but does not address issues of data integration, or consent management.

DataSHIELD [13] is a system allowing analysis of data from federated sources, enabling privacy by restricting the analyst to only executing aggregate queries. It does not address aspects of data and consent management. In our prototypical implementation, we utilise DataSHIELD to provide the functionality of the *Trusted Analysis Environment*. A similar system is ViPAR [44], a secure analysis platform for federated data. The system offers a central server host where the data is virtually pooled via the Secure Shell protocol from remote research datasets hosted by research institutions. The data can then be used for analysis through a secure analysis portal, and it is deleted once the analysis is done. The central pooling component is the major difference between DataSHIELD and ViPAR. DataSHIELD applies the analysis to each of the remote sources separately and then combine the analysis results, while ViPAR combines the data itself.

BioSHaRE project [45] (Biobank Standardisation and Harmonisation for Research Excellence in the European Union) is a collaborative European project focused on developing tools for data harmonisation, database integration and federated analysis to enable large-scale studies across multiple institutions. BioSHaRE includes DataSHIELD as a privacy-preserving analysis tool. Others systems used in the project are DataHSaPER for database federation and harmonisation and OBiBa as information technology tools.

If data is distributed among multiple sources, computing a common result without having to share the inputs (data) can solve privacy requirements. For example, secure-multiparty computation provides a cryptographic protocol to compute the output without the need of a third party. Other more light-weight approaches such as Federated Learning [46] have been

proposed as a potential solution. Federated Learning generally aims to learn a global model by aggregating locally learned models; this way, raw data does not need to leave the device where it is stored.

However, most approaches, such as Federated Learning, are mostly explored when data is horizontally partitioned, i.e. when there are multiple sites that hold data that is described by the same attributes, but contains different individual records. In our scenario, when combining data from multiple origins, we generally have a setting of vertical partitioning, i.e. we have data records describing different aspects of the same individual(s). Such an integration and linkage of multiple data sources, considering also consent checking, still poses challenges, such as keeping membership secret [47]. Further, auditability is more difficult to achieve in the federated setting. Therefore, we focus on the centralised setup proposed in our paper.

7.2. Auditability2

Traditionally, system logs have been collected and analysed to conduct system audits and hence, making the underlying systems more transparent. However, the analysis of such logs is cumbersome and tedious. Moreover, the right to explain, as postulated by the European Union General Data Protection Regulation (GDPR) [1], demands better suited methods to provide context and reasoning. Therefore, semantic technologies seem to be a promising method to capture and manage data lineage in a structured way and to provide support in addressing auditability questions.

Various frameworks and solutions to capture workflows, such as Taverna [48] or data models to collect provenance data such as PROV-DM [49] enable improved analysis and management. Provenance data can also support the reproduceability of research results by capturing the experiment context, for example by using ontologies [50–52].

Language-specific solutions, such as noworkflow [53] for python scripts, or rdt for R scripts among others have been developed and offer richer support such as visualisation [54]. Prov-O serves as a fundamental building block for capturing provenance-related data. Its generic nature however, calls for extension and adaption to specific contexts to be able to benefit from contextual information.

ProvStore, developed to be the first public repository of provenance documents showcases the need for such improved data management [55]. Use cases related to auditing and provenance include the possibility to prove adherence to legislative regulation (e.g. GDPR) as suggested by [56] to visualisation of dependencies or debugging capabilities [57]. However, so far, most tools and approaches are very specific to certain environments or setups without integrating heterogeneous provenance trails, applicable to our use case.

7.3. Consent Management2

The European General Data Protection Regulation (GDPR) [1] defines a set of obligations for controllers of personal data. Among other requirements, GDPR requires data controllers to obtain explicit consent for the processing of personal data from data subjects. Furthermore, organisations must be transparent about their processing of personal data and demonstrate that their systems comply with usage constraints specified by data subjects.

The traditional representation of user consent in the form of human-readable description does not allow for automatic processing. Formal policy languages are designed to unambiguously represent usage policies, which makes it possible to automatically verify whether data processing is covered by consent given by data subjects. Here, we will briefly review the current alternatives for (i) usage policy representation, and (ii) GDPR compliance tools.

Usage Policy Representation. There are several potential candidates for the formal representation of usage policies. KAoS [58] is a general policy language which adopts a pure ontological approach, whereas Rei [59] and Protune [60] use ontologies to represent concepts, the relationships between these concepts and the evidence needed to prove their truth, and rules to represent policies. PrOnto [8] is another ontology developed based on GDPR to represent a legal knowledge modelling of GDPR conceptual cores, i.e., privacy agents, data types, types of processing operations, rights and obligations. SPECIAL[27], a European H2020 project, have developed the SPECIAL Policy Language [19] around the recent OWL2 standard with the goal to adequately trade off expressiveness and computational complexity for usage policy checking. Finally, there is the SAVE ontology [7], which integrates and aligns to DPV [6], ODRL [61], and ORCP [62] to provide a fine-grained representation of privacy policies. While in essence any of these vocabularies can be used for usage policy representation in our use case, none of

[27] https://www.specialprivacy.eu/

them is community-based and geared towards open standards, unlike DPV [6]. Furthermore, it is important that the representation would allow for automatic compliance check, which will be discussed next.

GDPR compliance tools. [26] extends Prov-O and P-Plan with GDPR-specific concepts aiming to describing the provenance of data and using SPARQL to formulate compliance related queries over the data lifecycle. The Information Commissioner's Office (ICO) in the UK [63], Microsoft [64], and TrustArc[28] have developed compliance checking tools that enable companies to assess the compliance of their applications and business processes by completing a predefined questionnaire. These approaches are manual with regards to the compliance checking, which are not suitable for our use cases, where compliance checking happens during the use of the platform and must be automated. GConsent[29] is an ontology to model consent based on GDPR in a structured way, which is under development. Another notable project is Business Process Re-engineering and functional toolkit for GDPR compliance (BPR4GDPR)[30], a EU Horizon 2020 research project until April 2021. The project focused on process-based re-engineering to check security and compliance policies, one result being the Compliance ontology[31].

For automatic usage policy compliance checking, other than the SPECIAL compliance check mechanism [9] that we use in our approach, MIREL[32] and DAPRECO[33] projects extend PrOnto with LegalRuleML [65] to allow legal reasoning through Defeasible Logic theory. The goal for this approach is to conduct complex legal reasoning and compliance, which is not the goal in our use cases. While not explicitly designed for GDPR, PrivOnto [66] provides a framework that could facilitate analysis of privacy policies, including usage policy compliance.

8. Conclusion and Future Work1

In this paper we discussed the problem of *enabling auditable privacy-preserving data analysis systems*, inspired by the challenges faced by many small and medium-sized organisations in acquiring, storing and analysing personal data due to data protection regulations. We detailed the problem into a number of research questions that we addressed in this paper. We discuss each of the research questions and our approach to address it in the following.

What are the key characteristics of auditable privacy-preserving data analysis systems?4 We described a scenario to exemplify the requirements for auditable privacy-preserving data analysis and motivate a number of functional and non-functional requirements for our approach listed in Table 10. These are the results of our iterative process of identification and refining the requirements based on privacy standards as well as use cases and discussions with partners from the WellFort project.

What are the key elements of an architecture for auditable privacy-preserving data analysis systems?4 We proposed a novel approach called WellFort, a semantic-enabled architecture for auditable privacy-preserving data analysis which provides secure storage for users' sensitive data with explicit consent as well as a trusted analysis environment for executing data analytics processes in a privacy-preserving manner.

How can Semantic Web technologies be used to enable auditable privacy-preserving data analysis systems?4 We detailed how we select and utilise a set of Semantic Web technologies to support our conceptual architecture for auditable privacy-preserving data analysis systems. We found that for some tasks, e.g., the automatic usage policy compliance checking, the technology already has a high-level of maturity. For others, some adaptations are still required. Nevertheless, it is encouraging to see that research results can be adapted for real-world use cases.

In our evaluation, we demonstrated the feasibility of our approach on four real-world use cases in the medical domain. We show that the platform prototype is able to track the entire data lifecyle, starting from the deposit of data, to producing results in a scientific study. Thus, the architecture provides means to inspect which data was used, whether the consent was granted, or which specific libraries and code were used to produce the results. This, in turn, increases reproducibility and trust in the results, and can be used as evidence in litigation cases to discharge claims.

The future work will focus on adoption of the proposed platform to application domains in which auditability and privacy are the key concerns. This will

[28] http://bit.ly/3tTplPq
[29] http://openscience.adaptcentre.ie/ontologies/GConsent/docs/ontology
[30] https://www.bpr4gdpr.eu
[31] https://www.bpr4gdpr.eu/wp-content/uploads/2019/06/D3.1-Compliance-Ontology-1.0.pdf
[32] https://www.mirelproject.eu/
[33] https://www.fnr.lu/projects/data-protection-regulation-compliance/

require changes in the data model used in the repository and development of methods to automate knowledge graph generation from new types of data.

Another branch of future work is to investigate the possibility of adopting Semantic Web technologies for the *Trusted Analysis Environment*. Recent works in this direction have emerged in the Semantic Web community, e.g., on privacy-aware query processing [67, 68]. Additionally, we will focus on improving the quality of the *Trusted Analysis Environment*. This includes embedding privacy-preserving record linkage to improve the analysis of data originating from different companies, and extending the repertoire of DataSHIELD functionalities.

Furthermore, we plan to continue our work on the Audit Box to develop a multi-purpose RDF-based audit toolbox to support auditability of data analysis systems. Such toolbox facilitates capturing and auditing provenance trail data from heterogeneous sources, e.g., NoWorkflow [53] for python scripts. To this end, we also consider the possibility of linking the toolbox with our prior work on RDF data generation from security log data [69].

Acknowledgements. This work was sponsored by the Austrian Research Promotion Agency FFG under grant 871267 (WellFort) and 877389 (OBARIS). SBA Research (SBA-K1) is a COMET Centre within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG. Furthermore, this work was sponsored by the Austrian Science Fund (FWF) and netidee SCIENCE under grant P30437-N31 and the Vienna Business Agency (VasQua Project). The authors thank the funders for their generous support.

References

[1] E. Parliament and C. of European Union (2016), Regulation (EU) 2016/679, *Official Journal of the European Union* **59** (2016), 156.

[2] F.D. McSherry, Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis, in *SIGMOD '09*, Association for Computing Machinery, New York, NY, USA, 2009, pp. 19–30–. ISBN 9781605585512. doi:10.1145/1559845.1559850.

[3] A.U.P.P. Council, Statement on algorithmic transparency and accountability, *Communication of ACM* (2017).

[4] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik and J. Zhao, Prov-O: The Prov Ontology, *W3C Recommendation* (2013).

[5] L.F. Sikos and D. Philp, Provenance-Aware Knowledge Representation: A Survey of Data Models and Contextualized Knowledge Graphs, *Data Science and Engineering* **5**(3) (2020), 293–316. doi:10.1007/s41019-020-00118-0.

[6] H.J. Pandit, A. Polleres, B. Bos, R. Brennan, B. Bruegger, F.J. Ekaputra, J.D. Fernández, R.G. Hamed, E. Kiesling, M. Lizar et al., Creating a vocabulary for data privacy, in: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, 2019, pp. 714–730. doi:10.1007/978-3-030-33246-4_44.

[7] K. Krasnashchok, M. Mustapha, A. Al Bassit and S. Skhiri, Towards Privacy Policy Conceptual Modeling, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2020), 429–438. doi:10.1007/978-3-030-62522-1_32.

[8] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini and L. Robaldo, Pronto: Privacy Ontology for Legal Compliance, in: *Proc. 18th European Conference on Digital Government (ECDG)*, 2018, pp. 142–151.

[9] P.A. Bonatti, Fast Compliance Checking in an OWL2 Fragment, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Vol. 2018-July, International Joint Conferences on Artificial Intelligence Organization, California, 2018, pp. 1746–1752. ISSN 10450823. ISBN 9780999241127. doi:10.24963/ijcai.2018/241.

[10] J.D. Fernández, F.J. Ekaputra, P.R. Aryan, A. Azzam and E. Kiesling, Privacy-aware Linked Widgets, in: *Companion Proceedings of The 2019 World Wide Web Conference - WWW '19*, ACM Press, New York, USA, 2019, pp. 508–514. doi:10.1145/3308560.3317591.

[11] R. Mendes and J.P. Vilela, Privacy-Preserving Data Mining: Methods, Metrics, and Applications, *IEEE Access* **5** (2017), 10562–10582. doi:10.1109/ACCESS.2017.2706947.

[12] P. Samarati, Protecting Respondents Identities in Microdata Release, *IEEE transactions on Knowledge and Data Engineering* **13**(6) (2001), 1010–1027. doi:10.1109/69.971193.

[13] A. Gaye, Y. Marcon, J. Isaeva, P. LaFlamme, A. Turner, E.M. Jones, J. Minion, A.W. Boyd, C.J. Newby, M.-L. Nuotio et al., DataSHIELD: Taking the Analysis to the Data, not the Data to the Analysis, *International Journal of Epidemiology* **43**(6) (2014), 1929–1944. doi:10.1093/ije/dyu188.

[14] S.M. Bellovin, P.K. Dutta and N. Reitinger, Privacy and Synthetic Datasets, *Stan. Tech. L. Rev.* **22** (2019), 1.

[15] ISO, *ISO 14721: International Standard: Space Data and Information Transfer Systems, Open Archival Information System (OAIS), Reference Model*, ISO, 2012.

[16] M. Wilkinson, M. Dumontier, I. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. Boiten, L. da Silva Santos, P. Bourne et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific Data* **3** (2016), 160018. doi:10.1038/sdata.2016.18.

[17] W.W.W. Consortium, Data Catalog Vocabulary (DCAT) - Version 2, 2020. http://www.w3.org/TR/vocab-dcat/.

[18] R. Cyganiak, J. Zhao, A. Keith and M. Hausenblas, Vocabulary of Interlinked Datasets (voiD), 2011. http://vocab.deri.ie/void.

[19] P.A. Bonatti, S. Kirrane, I. Petrova, L. Sauro and E. Schlehahn, D2.5 - Policy Language V2, Zenodo, 2018. doi:10.5281/zenodo.2545177.

[20] J.D. Fernández, M. Sabou, S. Kirrane, E. Kiesling, F.J. Ekaputra, A. Azzam and R. Wenning, User consent modeling for ensuring transparency and compliance in smart cities,

Personal and Ubiquitous Computing **24**(4) (2020), 465–486. doi:10.1007/s00779-019-01330-0.

[21] S. Kirrane, P. Bonatti, J.D. Fernández, C. Galdi, L. Sauro, D. Dell'Erba, I. Petrova and I. Siahaan, D2.8 - Transparency and Compliance Algorithms V2, Zenodo, 2018. doi:10.5281/zenodo.2543622.

[22] P.A. Bonatti, L. Ioffredo, I.M. Petrova, L. Sauro and I.R. Siahaan, Real-Time Reasoning in OWL2 for GDPR Compliance, *Artificial Intelligence* **289** (2020), 103389. doi:10.1016/j.artint.2020.103389.

[23] F.J. Ekaputra, P.R. Aryan, E. Kiesling, C. Fabianek and E. Gringinger, Semantic Containers for Data Mobility: A Seismic Activity Use Case., in: *SEMANTICS Posters&Demos*, 2019. http://ceur-ws.org/Vol-2451/paper-11.pdf.

[24] S. Miles, P. Groth, S. Munroe and L. Moreau, Prime: A Methodology for Developing Provenance-Aware Applications, *ACM Transactions on Software Engineering and Methodology (TOSEM)* **20**(3) (2011), 1–42. doi:10.1145/2000791.2000792.

[25] L. Moreau, P. Groth, J. Cheney, T. Lebo and S. Miles, The Rationale of PROV, *Journal of Web Semantics* **35** (2015), 235–257. doi:10.1016/j.websem.2015.04.001.

[26] H.J. Pandit and D. Lewis, Modelling Provenance for GDPR Compliance using Linked Open Data Vocabularies., in: *PrivOn@ Workshop co-located with ISWC 2017*, 2017, pp. 1–15. http://ceur-ws.org/Vol-1951/PrivOn2017_paper_6.pdf.

[27] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: *LDOW Workshop*, 2014. http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[28] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro and G. Xiao, Ontop: Answering SPARQL Queries over Relational Databases, *Semantic Web* **8**(3) (2017), 471–487. doi:10.3233/SW-160217.

[29] B. Glimm, I. Horrocks, B. Motik, G. Stoilos and Z. Wang, HermiT: An OWL 2 Reasoner, *Journal of Automated Reasoning* **53**(3) (2014), 245–269. doi:10.1007/s10817-014-9305-1.

[30] D. Doiron, Y. Marcon, I. Fortier, P. Burton and V. Ferretti, Software application profile: Opal and mica: Open-source software solutions for epidemiological data management, harmonization and dissemination, *International Journal of Epidemiology* **46**(5) (2017), 1372–1378. doi:10.1093/IJE/DYX180.

[31] T. Benson and G. Grieve, *Principles of Health Interoperability: SNOMED CT, HL7 and FHIR*, Springer, 2016. doi:10.1007/978-3-319-30370-3.

[32] FHIR: Fast Healthcare Interoperability Resources. http://hl7.org/implement/standards/fhir/.

[33] C. Clifton, M. Kantarcioğlu, A. Doan, G. Schadow, J. Vaidya, A. Elmagarmid and D. Suciu, Privacy-preserving data integration and sharing, in: *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '04*, ACM Press, Paris, France, 2004, p. 19. ISBN 978-1-58113-908-2. doi:10.1145/1008694.1008698.

[34] B.C.M. Fung, K. Wang, R. Chen and P.S. Yu, Privacy-Preserving Data Publishing: A Survey of Recent Developments, *ACM Computing Surveys* **42**(4) (2010). doi:10.1145/1749603.1749605.

[35] F. Prasser and F. Kohlmayer, Putting Statistical Disclosure Control into Practice: The ARX Data Anonymization Tool, in: *Medical Data Privacy Handbook*, Springer International Publishing, Cham, 2015, pp. 111–148. doi:10.1007/978-3-319-23633-9_6.

[36] C. Dwork, Differential Privacy: A Survey of Results, in: *International Cnference on Theory and Applications of Models of Computation*, Springer, 2008, pp. 1–19. doi:10.1007/978-3-540-79228-4_1.

[37] P. Mohan, A. Thakurta, E. Shi, D. Song and D. Culler, GUPT: Privacy Preserving Data Analysis Made Easy, in: *Proceedings of the 2012 International Conference on Management of Data - SIGMOD '12*, ACM Press, Scottsdale, Arizona, USA, 2012, p. 349. doi:10.1145/2213836.2213876.

[38] F.D. McSherry, Privacy integrated queries: an extensible platform for privacy-preserving data analysis, in: *Proceedings of the 35th SIGMOD international conference on Management of data - SIGMOD '09*, ACM Press, Providence, Rhode Island, USA, 2009, p. 19. ISBN 978-1-60558-551-2. doi:10.1145/1559845.1559850.

[39] I. Roy, S.T.V. Setty, A. Kilzer, V. Shmatikov and E. Witchel, Airavat: Security and Privacy for MapReduce, in: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, USENIX Association, USA, 2010.

[40] N. Patki, R. Wedge and K. Veeramachaneni, The Synthetic Data Vault, in: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, Montreal, QC, Canada, 2016, pp. 399–410. ISBN 978-1-5090-5206-6. doi:10.1109/DSAA.2016.49.

[41] U.H.D. Alliance, Trusted Research Environments – A Strategy to Build Public Trust and Meet Changing Health Data Science Needs, 2020. https://ukhealthdata.org/wp-content/uploads/2020/07/200723-Alliance-Board_Paper-E_TRE-Green-Paper.pdf.

[42] M. Koscina, D. Manset, C. Negri and O. Perez, Enabling Trust in Healthcare Data Exchange with a Federated Blockchain-Based Architecture, in: *IEEE/WIC/ACM International Conference on Web Intelligence on - WI '19 Companion*, ACM Press, Thessaloniki, Greece, 2019, pp. 231–237. doi:10.1145/3358695.3360897.

[43] N. Popper, F. Endel, R. Mayer, M. Bicher and B. Glock, Planning Future Health: Developing Big Data and System Modelling Pipelines for Health System Research, *SNE Simulation Notes Europe* **27**(4) (2017), 203–208. doi:10.11128/sne.27.tn.10396.

[44] K.W. Carter, R.W. Francis, K. Carter, R. Francis, M. Bresnahan, M. Gissler, T. Grønborg, R. Gross, N. Gunnes, G. Hammond, M. Hornig, C. Hultman, J. Huttunen, A. Langridge, H. Leonard, S. Newman, E. Parner, G. Petersson, A. Reichenberg, S. Sandin, D. Schendel, L. Schalkwyk, A. Sourander, C. Steadman, C. Stoltenberg, A. Suominen, P. Surén, E. Susser, A. Sylvester Vethanayagam and t.I.C.f.A.R.E. Yusof Z, ViPAR: a software platform for the Virtual Pooling and Analysis of Research Data, *International Journal of Epidemiology* **45**(2) (2015), 408–416. doi:10.1093/ije/dyv193.

[45] D. Doiron, P. Burton, Y. Marcon, A. Gaye, B.H.R. Wolffenbuttel, M. Perola, R.P. Stolk, L. Foco, C. Minelli, M. Waldenberger, R. Holle, K. Kvaløy, H.L. Hillege, A.-M. Tassé, V. Ferretti and I. Fortier, Data Harmonization and Federated Analysis of Population-Based Studies: the BioSHaRE Project, *Emerging Themes in Epidemiology* **10**(1) (2013), 12. doi:10.1186/1742-7622-10-12.

[46] B. McMahan, E. Moore, D. Ramage, S. Hampson and B.A.y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, in: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, eds, Proceedings of Machine Learning Research, Vol. 54, PMLR, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[47] J. Sun, X. Yang, Y. Yao, A. Zhang, W. Gao, J. Xie and C. Wang, Vertical Federated Learning without Revealing Intersection Membership, 2021. https://arxiv.org/abs/2106.05508.

[48] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat et al., Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows, *Bioinformatics* **20**(17) (2004), 3045–3054. doi:10.1093/bioinformatics/bth361.

[49] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker et al., PROV-DM: The prov data model, *W3C Recommendation* (2013).

[50] T. Miksa and A. Rauber, Using ontologies for verification and validation of workflow-based experiments, *Web Semantics: Science, Services and Agents on the World Wide Web* **43** (2017), 25–45. doi:10.1016/j.websem.2017.01.002.

[51] B. Goesswein, T. Miksa, A. Rauber and W. Wagner, Data Identification and Process Monitoring for Reproducible Earth Observation Research, in: *IEEE eScience 2019*, IEEE, 2019, pp. 28–38. doi:10.1109/eScience.2019.00011.

[52] R. Mayer, G. Antunes, A. Caetano, M. Bakhshandeh, A. Rauber and J. Borbinha, Using Ontologies to Capture the Semantics of a (Business) Process for Digital Preservation, *International Journal of Digital Libraries (IJDL)* **15** (2015), 129–152. doi:10.1007/s00799-015-0141-7.

[53] J.F. Pimentel, L. Murta, V. Braganholo and J. Freire, noWorkflow: A Tool for Collecting, Analyzing, and Managing Provenance from Python Scripts, *Proceedings of the VLDB Endowment* **10**(12) (2017). doi:10.1016/j.cosrev.2019.05.002.

[54] A.M. Ellison, E.R. Boose, B.S. Lerner, E. Fong and M. Seltzer, The End-to-End Provenance Project, *Patterns* **1**(2) (2020), 100016. doi:10.1016/j.patter.2020.100016.

[55] T.D. Huynh and L. Moreau, ProvStore: A Public Provenance Repository, in: *International Provenance and Annotation Workshop*, Springer, 2014, pp. 275–277. doi:10.1007/978-3-319-16462-5_32.

[56] B.E. Ujcich, A. Bates and W.H. Sanders, A Provenance Model for the European Union General Data Protection Regulation, in: *International Provenance and Annotation Workshop*, Springer, 2018, pp. 45–57. doi:10.1007/978-3-319-98379-0_4.

[57] J.F. Pimentel, J. Freire, L. Murta and V. Braganholo, A Survey on collecting, Managing, and Analyzing Provenance from Scripts, *ACM Computing Surveys (CSUR)* **52**(3) (2019), 1–38. doi:10.1145/3311955.

[58] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni and J. Lott, KAoS policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement, in: *Proceedings - POLICY 2003: IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 93–96. doi:10.1109/POLICY.2003.1206963.

[59] L. Kagal, T. Finin and A. Joshi, A Policy Language for a Pervasive Computing Environment, in: *Proceedings - POLICY 2003: IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 63–74. doi:10.1109/POLICY.2003.1206958.

[60] P. Bonatti, J.L. De Coi, D. Olmedilla and L. Sauro, A Rule-Based Trust Negotiation System, *IEEE Transactions on Knowledge and Data Engineering* **22**(11) (2010), 1507–1520. doi:10.1109/TKDE.2010.83.

[61] R. Iannella and S. Villata, ODRL Information Model 2.2, *W3C recommendation* (2018). https://www.w3.org/TR/odrl-model/.

[62] S. Kirrane, M. De Vos and J. Padget, ODRL Regulatory Compliance Profile 0.2, Technical Report, Vienna University of Economics and Business, 2020.

[63] I.C. Office, Getting Ready for the GDPR, 2020. https://ico.org.uk/for-organisations/data-protection-self-assessment/.

[64] M.T. Center, Detailed GDPR Assessment, 2017. http://aka.ms/gdprdetailedassessment.

[65] T. Athan, G. Governatori, M. Palmirani, A. Paschke and A. Wyner, LegalRuleML: Design Principles and Foundations, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **9203** (2015), 151–188. doi:10.1007/978-3-319-21768-0_6.

[66] A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T.B. Norton, N.C. Russell, P. Story, J. Reidenberg and N. Sadeh, PrivOnto: A Semantic Framework for the Analysis of Privacy Policies, *Semantic Web* **9**(2) (2018), 185–203. doi:10.3233/SW-170283.

[67] K.M. Endris, Z. Almhithawi, I. Lytra, M.-E. Vidal and S. Auer, BOUNCER: Privacy-Aware Query Processing over Federations of RDF Datasets, in: *International Conference on Database and Expert Systems Applications*, Springer, 2018, pp. 69–84. doi:10.1007/978-3-319-98809-2_5.

[68] R. Taelman, S. Steyskal and S. Kirrane, Towards Querying in Decentralized Environments with Privacy-Preserving Aggregation, *CoRR* **abs/2008.06265** (2020). https://arxiv.org/abs/2008.06265.

[69] A. Ekelhart, F.J. Ekaputra and E. Kiesling, The SLOGERT Framework for Automated Log Knowledge Graph Construction, in: *European Semantic Web Conference*, Springer, 2021, pp. 631–646. doi:10.1007/978-3-030-77385-4_38.