Future Generation Computer Systems 165 (2025) 107550

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

The EPI framework: A data privacy by design framework to support healthcare use cases☆

Jamila Alsayed Kassem *, Tim Müller, Christopher A. Esterhuyse, Milen G. Kebede, Anwar Osseyran, Paola Grosso

University of Amsterdam, Science Park 900, Amsterdam, 1098 XH, The Netherlands

ARTICLE INFO

Keywords:
Data sharing policies
Privacy as a service
Healthcare
Workflows
Privacy risk assessment model
Data protection
Usage control

ABSTRACT

Data sharing is key to enabling data analysis and research advancement, and that is especially true in healthcare. Due to the inherited sensitivity of health data, institutions are still wary of sharing their data, especially with the increasing number of breaches in recent years and the strict privacy legislation involved (GDPR, HIPAA, etc.). Privacy and security concerns exist when making data available for use or processing.

To tackle these concerns, we initially incorporate Privacy by Design (PbD) principles. This informs our approach to constructing a data-sharing framework that aligns with said principles. Subsequently, we introduce examples of data-centric use cases requiring support, followed by the delineation of the computation events model and data properties intrinsic to a use case. Furthermore, to gain insight into the potential privacy risks associated with executing a workflow request, we expand upon the privacy threat assessment model to quantitatively evaluate the risks of data likability, identifiability, non-repudiation, detectability, unintended disclosure, indulgence, and policy & consent noncompliance. Subsequently, we construct a framework; the EPI framework; aimed at mitigating these identified risks, via adhering to PbD properties and provisioning extra services.

1. Introduction

In the era of big data and increased digitalization, the sharing of sensitive information, particularly in the medical domain, has become crucial for advancing healthcare research, treatment, and patient outcomes [1]. However, the inherent privacy risks associated with sharing such data pose significant challenges. Traditional data-sharing approaches often neglect essential privacy considerations, potentially compromising data confidentiality, integrity, and control [2]. Specific challenges include the risks of unauthorized access, data breaches, and the lack of granular control over data usage. EPI addresses these concerns by implementing stringent privacy measures, including workflow orchestration, data sharing policies reasoning, and security/privacy as a service orchestration.

Current healthcare data-sharing practices often fall short of addressing privacy risks comprehensively. Data breaches and unauthorized access to sensitive medical information have been on the rise, leading to significant financial losses and erosion of trust in digital health solutions. According to [3], healthcare data breaches cost the industry an average of $6.45 million per incident. Additionally, regulatory landscapes such as GDPR and HIPAA impose stringent requirements on data privacy and security, making it imperative for healthcare organizations to adopt robust privacy-preserving frameworks.

The EPI framework aims to establish a comprehensive and robust environment for sharing medical data whilst ensuring privacy and data utility. By incorporating workflow orchestration, data sharing policies reasoning, and privacy as a service orchestration, EPI addresses the complex privacy challenges inherent in medical data sharing. The EPI framework provides a holistic approach to data sharing by considering multiple dimensions of privacy risks. These dimensions are based on [4] and include Linkability (L), Identifiability (I), Non-repudiation (Nr), Detectability (Dt), information Disclosure (iD), data Indulgence (In), and policy Non-compliance (Nc). By thoroughly assessing these dimensions, EPI enables the impact and likelihood quantification of potential privacy risks associated with shared medical data and provides the means to mitigate this risk.

The EPI Framework focuses on maintaining data control throughout the data-sharing process, ensuring that privacy considerations are

☆ The EPI project is funded by the Dutch Science Foundation in the Commit2Data program. https://enablingpersonalizedinterventions.nl/, https://github.com/epi-project/.

* Corresponding author.
E-mail addresses: j.alsayedkassem@uva.nl (J. Alsayed Kassem), t.muller@uva.nl (T. Müller), c.a.esterhuyse@uva.nl (C.A. Esterhuyse), m.g.kebede@uva.nl (M.G. Kebede), a.osseyran@uva.nl (A. Osseyran), p.grosso@uva.nl (P. Grosso).

https://doi.org/10.1016/j.future.2024.107550
Received 2 February 2024; Received in revised form 4 August 2024; Accepted 6 October 2024
Available online 20 November 2024
0167-739X/

integrated from the outset. The framework addresses the challenge of balancing data utility with privacy protection by offering mitigations specific to the associated data risks identified during workflow submission.

This paper elaborates upon prior research discussed in Section 2, with a particular emphasis on the recurring theme of PbD. We define PbD concepts in Section 3, and, in this paper, distinguish between privacy and security properties. The use cases investigated accompanied by explicit definitions of data properties and the events computation model associated with a use case (Sections 4.1 and 4.3, respectively). We present the LINDDIN privacy assessment model in Section 5, wherein risk attributes are formulated to quantitatively assess risk, considering both data properties and the computation model. Offering an overview of the EPI framework in Section 6, we delve into more detailed descriptions of the EPI orchestration layers (Sections 7 and 8). Lastly, to evaluate the framework, we provide a detailed walk-through of a practical use case and quantify the extent to which privacy risks are alleviated before and after the implementation of the EPI framework in Section 9.

2. Related work

Several papers have contributed to the definition and development of PbD principles, frameworks, and applications. In [5], the authors outline the seven foundational principles of PbD, providing a framework for the operation of systems, processes, and technologies: proactive and preventive, default privacy, embedded in the design, full functionality of services, end-to-end security, visibility and transparency, and finally, respecting user privacy.

[6] introduces the concept of privacy-aware ubiquitous systems and presents principles for incorporating privacy by design into the development of such systems. It emphasizes the need to consider privacy throughout the design process. [7] proposes a PbD framework to evaluate the privacy gap between IoT devices and middleware platforms, such as OpenIoT, Eclipse SmartHome, etc. Minimizing data, in general, is a common practice in Literature. As an example, in [2], the framework is based on minimizing data storage (retention), raw data intake, data sources, and data knowledge discovery.

In the context of PbD in healthcare applications, [8,9] apply the 7 PbD properties introduced in [5] to evaluate current practices; namely IoT applications. In the true spirit of PbD, [10] utilizes deep learning tools to pseudo-anonymize neuroimages using defacing, skull stripping, and face masking to preserve privacy and utility.

There already exist data sharing platforms that align themselves with PbD principles in general. One of them is Mahiru [11], a decentralized data-sharing platform that emphasizes domain autonomy. A design choice that make working with the framework practical is that it broadcasts all policy information to the site that at that moment acts as an orchestrator; however, in the medical domain, this is infeasible because some policies may be private.

The EPI Framework addresses the challenges of privacy and security in data workflows by utilizing three main components: *BRANE* — the workflow orchestrator, the *policy server* — the policy reasoner, and the *Bridging Functions Chain (BFC)* orchestrator — the network function orchestrator. Through the utilization of the EPI Framework, organizations can enhance privacy protection, promote data utility, and achieve compliance with privacy regulations. The framework's holistic approach aligns with the evolving landscape of privacy and security requirements, enabling organizations to adapt to changing data privacy landscapes and emerging threats.

3. Data privacy by design

PbD is a fundamental principle that advocates for embedding privacy protections into the design and development of systems. One prominent regulation that emphasizes privacy by design is the General Data Protection Regulation (GDPR) in the European Union. GDPR fixes strict requirements for data protection and privacy, placing increased responsibility on organizations to implement PbD principles.

Integrating privacy controls and safeguards into the processes may include implementing strong access controls, encryption mechanisms, pseudonymization and anonymization techniques, and user consent mechanisms. It also entails promoting transparency by informing individuals about the purposes and scope of data processing and ensures that privacy settings are easily understandable to users. By incorporating PbD, we can establish a privacy-conscious culture, foster trust with users, and proactively mitigate privacy risks. PbD enables organizations to comply with legal requirements like GDPR, and demonstrates a commitment to protecting individuals' privacy rights.

3.1. Privacy vs. security properties

Security and privacy are closely related concepts but have distinct focuses. Security primarily deals with protecting systems, networks, and data from unauthorized access, breaches, and malicious activities. It encompasses measures such as authentication, encryption, firewalls, intrusion detection systems, and incident response mechanisms.

Privacy, on the other hand, is concerned with protecting personal information and ensuring that individuals have control over the collection, use, and disclosure of their data. It encompasses principles like data minimization, purpose limitation, consent, transparency, and individual rights. While security focuses on protecting the integrity, availability, and confidentiality of data and systems, privacy focuses on preserving individuals' autonomy, dignity, and control over their personal information. Both security and privacy are essential in today's digital landscape. Security mitigations might align with privacy needs. For example, protecting against unauthorized access ensures control over data disclosure, addressing concerns from both perspectives.

In system design, we need to recognize the importance of both security and privacy and adopt an integrated approach to protect data and individuals' privacy rights. Balancing these aspects is crucial for building data-sharing frameworks, complying with regulations, and fostering a positive user experience in an increasingly data-driven world.

4. Data properties

Medical workflows (pipelines) can involve various types of data with varying *sensitivity* and *utility*; usefulness of a data set to successfully run a specific workflow.

4.1. Data sensitivity

Medical data is inherently sensitive, as categorized by the GDPR [12]. However, this sensitivity is contingent on various factors. For instance, the state of the data – like whether it is encrypted – plays a pivotal role. Upcoming sections will revisit the specific data attributes that shed light on the associated privacy risks. We consider several data states within the EPI's data-sharing ecosystem, and are relevant to the data required for the three use cases:

- **Raw Data:** refers to data that is in its original, unprocessed format. It may include unstructured text, sensor readings, or any data that has not undergone any transformation or aggregation. Raw data often requires additional processing or anonymization to mitigate privacy risks before storage or sharing.
- **Anonymized Data:** data has undergone a process to remove or modify identifiers, making it extremely difficult or impossible to link the data back to an individual. Hence, the sensitivity of anonymized data is reduced. Anonymization techniques such as pseudonymization are applied to protect privacy.

```
Raw = { "PatientID": 1, "Age": 42, "Gender": "Male",
"Condition": "Diabetes", "Date_of_Visit": "2023-01-05",
"Cholesterol_Level": 180, "Blood_Pressure": "120/80",
"BMI": 25.5 }
```

Listing 1: Unprocessed EHR dataset

```
Psuedoanonymised = { "PatientID": "P1", "Age": 42,
"Gender": "Male", "Condition": "Diabetes",
"Date_of_Visit": "Visit1", "Cholesterol_Level": 180,
"Blood_Pressure": "120/80", "BMI": 25.5 }
```

Listing 2: Pseudo-anonymized EHR dataset

```
Anonymised = { "PatientID": "Patient1", "Age": "Age1",
"Gender": "Gender1", "Condition": "Diabetes",
"Date_of_Visit": "Visit1", "Cholesterol_Level": 180,
"Blood_Pressure": "120/80", "BMI": 25.5 }
```

Listing 3: Fully anonymized EHR dataset

Fig. 1. Three states of data processing. (a) Unprocessed data: Listing 1. (b) Pseudo-anonymized data: Listing 2. (c) Fully anonymized data: Listing 3.

- **Encrypted Data:** refers to data that is transformed using cryptographic algorithms into an unreadable format. Encrypted storage adds a layer of security by protecting the data from unauthorized access. The encryption helps mitigate the risk of exposure if the storage is compromised, or if a transfer session is sniffed (eavesdropping).
- **Synthetic Data:** artificially generated data that mimics the statistical properties of real data while containing no identifiable or sensitive information. Synthetic data is often used for testing, development, and research purposes. Since it does not contain real patient information, the sensitivity is typically low.
- **Aggregated Data:** data that has been combined and summarized from multiple sources or individual records. Aggregation helps protect privacy by reducing the granularity of the data and removing personally identifiable information. The sensitivity of aggregated data depends on the level of detail retained and the potential for re-identification.

Furthermore, with the data state in mind, we commonly think about the sensitivity of a current dataset based on the distance compared to the original (raw) dataset(s), as shown in Eq. (1), such that $sensitivity \in \{0, 0.5, 1.0\}$. Distance and sensitivity metrics are inversely proportionate, the higher the distance of a dataset is the lower the sensitivity is:

$$sensitivity = 1 - distance(raw, current), \tag{1}$$

where 1 is an indication of the most sensitive dataset, 0 is the least sensitive and $distance \in \{0, 0.5, 1.0\}$.

Moreover, the methods used to measure the distance metric are dependent on the dataset itself. As an example, statistical measures can be used to assess the difference in statistical properties, or if the dataset contains images, metrics like structural similarity index (SSIM) or peak signal-to-noise ratio (PSNR) can be used to assess image distortion distance. The choice of a distance measurement method depends on the characteristics of the data and the specific modifications applied.

For simplicity, we assume that the distance between data states can be one of three values: 0 if the data is unprocessed $distance(raw, raw) = 0$, $distance(raw, current) = 0.5$ if the data is pseudo-anonymized or partially processed (e.g., partially encrypted or aggregated), and 1 if the data is fully processed to maximize distance.

Consider the following examples:

In Listing 1, the dataset is in its original, unprocessed form, so $distance(raw, raw) = 0$. In Listing 2, the data has been pseudo-anonymized by replacing patient IDs and visit dates with pseudonyms and generic labels. This yields a distance of 0.5. In Listing 3, further anonymization replaces age and gender with generic labels, resulting in a maximum distance of 1. Bear in mind that while distance measurement offers a quantifiable evaluation of dataset disparities, it may not fully encapsulate the effect on data utility or usefulness. Hence, a comprehensive evaluation of altered datasets should account for both sensitivity and utility aspects.

4.2. Data utility

Data utility refers to the value or usefulness of data for a specific purpose or task, such as the workflow associated with the three use cases. Defining all utility factors can be a cumbersome task, so for the sake of this paper, we focus our model on the factors that are most relevant to provided use cases. The utility of data can be influenced by various factors, including data type, data state/distance, data size, and available computing resources.

Data type: In the context of health data, a multitude of data types may be necessary for a given use case, each process to deliver a specific service. For instance, data types encompass patients' health records, medical images, clinical trial data, sensor data, wearables, and the like.

Different types of data have different utilities depending on the specific algorithm and the problem being addressed. This is showcased when a data type u_{type} is available for processing or not, when the workflow is requested:

$$u_{type} = \begin{cases} 1, & \text{type requested is available for workflow} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Computation Power: Handling and processing large, distorted, and synthetic datasets may require significant computational resources and can impact the efficiency and scalability of the ML-based workflow (CVA use case). The utility of data is influenced by the available computing resources. If the computing resources are limited, the utility of large datasets may be reduced due to the constraints on processing and model training. Conversely, with ample computing resources, the utility of large datasets can be fully realized, allowing for more comprehensive analysis and better model performance. The compute resources relate to CPU and memory resources requested r_{req} compared to that available r_{av}, and is formalized as a flag:

$$u_{compute} = \begin{cases} 1, & r_{req} < r_{av} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Data State: The first two attributes, aforementioned, are functional attributes and depends on resources' availability, while the following two attributes affect the privacy risk directly. The status of the data and the relative distance can also impact its utility. Fully Synthetic data, which is artificially generated to mimic real data, may have lower utility compared to raw or real-world data because it may not capture all the nuances and complexities present in the original data. To capture this effect, we look into the distance value and compare it to the acceptable distance submitted along the workflow, to successfully run it. This distance attribute is formalized as a flag $u_{distance}$:

$$u_{distance} = \begin{cases} 1, & distance(raw, current) \leq distance_{req} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Data size: The size of the data can affect its utility in different ways. Larger datasets generally provide more utility by enabling more accurate and robust machine learning.

On the other hand, data minimization is a sign of good privacy by design practices, and one way to do that is by sharing (or making available) the minimal dataset size while running a workflow. Other than minimizing sensitivity (increasing distance), size is an attribute that needs to be considered under privacy and utility constraints. The

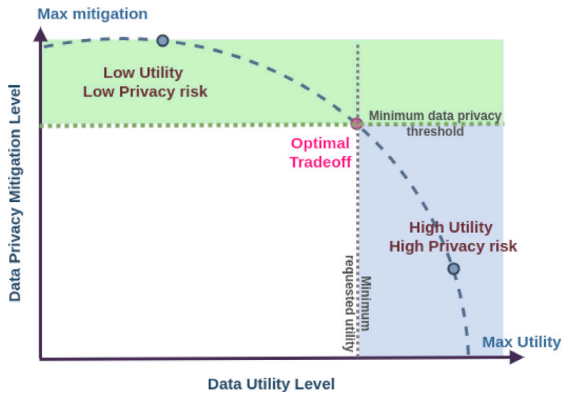


Fig. 2. A depiction of how privacy-preserving mitigations trade-off utility and sensitivity.

effect of the size attribute is formalized as flag u_{size}, where $u_{size} = 1$ means that this size utility condition is met:

$$u_{size} = \begin{cases} 1, & \text{size requested is available for workflow} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

The requested utility attributes are set by the users submitting a workflow request, and the current availability of data and computing resources are determined by the framework. As a result, the total utility (U) is calculated as the sum of these attributes, such that α, β, θ, and γ parameters are also set to determine the relevance of these attributes to the submitted workflow. Where $\alpha + \gamma + \beta + \theta \leq 1$:

$$U = \alpha u_{type} + \gamma u_{compute} + \beta u_{distance} + \theta u_{size} \quad (6)$$

As mentioned before, data privacy-preserving methods are associated with data sensitivity, but also data size. Fig. 2 showcases the effect of these methods on utility. The framework we will define aims to find the optimal trade-off point to run a workflow, to minimize privacy risk with utility as a constraint. The threshold for data privacy is set by the data sharing policies, while the threshold for minimum utility is set by the user submitting the workflow.

4.3. Data sharing events & events model

In this section, we introduce our event model, describing the building blocks of behaviour for data exchange systems. This provides a low level of abstraction of the data-sharing events, enabling our analysis of risk within the workflow event model. A single event performs a specific task, such as selecting a data entry, aggregation, and filtering. A workflow model is a composition of events occurring at possibly physically-distributed sites. Data-sharing events can refer to gaining access to data in storage, transferring data, or processing data with data in execution.

4.4. Data in storage

Data in storage events can be of different types: (**1**) data read events: reading data from a storage system or source, (**2**) data write events: writing data to a storage system or destination, (**3**) data update events: modifying existing data in a storage system, and (**4**) data delete events: removing or deleting data from a storage system. Data in storage events can be performed via different methods. Some methods are functionally equivalent, but differ in their privacy/security characteristics. A characteristic example is file system access methods. These are fundamental operations for opening, closing, reading, and writing files. These methods are generally considered less secure than their functional equivalents using Object Storage Access or Relational Database Access. This is because file system access methods rely on operating system-level permissions and access controls, which can be vulnerable to unauthorized access if improperly configured. Additionally, file system access may lack encryption mechanisms to protect data at rest.

Data storage events are expressed by specifying an event's type *storage_event*, defining the *data_value*, the data in question, the *storage_location* (ex: database location), and lastly, the method function used *storage_function*. This is expressed as follows:

```
storage_event: <data_value, storage_location
, storage_function>
```

The properties associated with data are discussed in Section 4.

4.5. Data in transfer

Data in transfer events can be of different types: (**1**) data moving events: moving data from one location to another, such as from one storage system to another, or from storage to an execution environment, (**2**) data import events: bringing external data into the pipeline for processing, and (**3**) data export events: sending processed data out of the pipeline to another system or destination.

Similarly, transfer events can utilize multiple methods to run, and methods vary in terms of privacy/security, protocols, and implementations. As an example, a requested event can run via a specific transfer function like SFTP (SSH File Transfer Protocol). SFTP is considered one of the most secure options as it provides secure file transfer capabilities over an encrypted SSH connection [13]. It offers authentication, data encryption, and integrity checks. On the other hand, a method function can also run and utilize Remote Mounting, which introduces security and privacy risks depending on the configuration and implementation. It relies on the security mechanisms provided by the underlying protocols (e.g., NFS, SMB), which may have vulnerabilities if not properly secured. Transfer events can be expressed as follows:

```
transfer_event: <source_location,
destination_location, data_value,
transfer_function>,
```

where the *transfer_event* is replaced with the event type, the *source_location* is the source site data is originating from, similarly, the *source_location* is where the data will end up, the *data_value* specifies what dataset is the object of this event, and the *transfer_function* is the method this event is utilizing to run.

4.6. Data in execution

Data execution events can be of different types: (**1**) data processing events: computing (new) output data, possibly using (existing) input data, (**2**) data transformation events: converting data from one representation (e.g., format) to another, (**3**) data aggregation events: combining multiple data elements into a single entity or summary, (**4**) data filtering events: excluding specific data based on certain criteria, e.g., to select only a particular element.

We can run a requested execution event via Secure Multi-Party Computation (SMPC). SMPC is a cryptographic technique that enables secure collaborative data processing among multiple parties. It ensures that sensitive data remains encrypted during computations, enhancing security by minimizing exposure to plain text data. We can also utilize On-Premises Processing with Strict Access Controls methods. By enforcing rigorous access controls, the risk of unauthorized access or data leakage is reduced. On the other hand, utilizing General-purpose computing environments (GPCE), such as traditional servers or cloud instances, implicates a lower level of inherent security.

The *compute_type* is defined with the event type, *input_data* and *output_data* define the input and output data of the compute event, respectively, and *processing_function* defines the method function. This is expressed as follows:

Table 1
An example events model.

Index	Workflow tasks
1	$READ\langle EHR_1, loc_a, \text{file system access}\rangle$
2	$IMPORT\langle loc_a, loc_b, EHR_1, FTP\rangle$
3	$PROCESS\langle EHR_1, algorithm(EHR_1), algorithm, GPCE\rangle$

```
compute_event: <input_data, output_data, processing_function>
```

A workflow model can be composed with a series of events, and Table 1 provides an example workflow model composed of *READ* storage event, *IMPORT* transfer event, and *PROCESS* execute event:

Employing event-based modelling for workflows not only facilitates the assessment of privacy and security risks but also provides a detailed depiction of the services essential for the management and execution of a use case. These use cases align with the format demonstrated in Table 1. In addition to this, workflow requests should comprehensively outline data values and specify utility prerequisites imperative for the operationalization of this workflow model.

A workflow request has a set of requirements to successfully run with minimal acceptable utility value U_{acc}. To calculate U_{acc} utility's requirement, attributes and weights are set, as formalized in Eq. (6). Here, users will specify the data type, compute resources, data size, data distance, and data size that should be available to run the workflow. These attributes are found relevant to run the use case workflows with EPI, and we assign 0 and 1 values to $E(u_{type})$, $E(u_{compute})$, $E(u_{distance})$, and $E(u_{size})$. Note that, by default, all utility conditions are expected to be met, and hence function E sets all the utility attributes to 1. In addition to α, γ, β, and θ weights reflect the importance of these attributes.

5. Privacy risk assessment

To evaluate the workflow models we defined, privacy and security risk assessment models serve as powerful tools to evaluate and mitigate potential threats to data privacy and security. This is done by systematically identifying vulnerabilities and assessing the associated risks. These models provide a structured framework for quantifying and qualifying risks, allowing for prioritizing mitigations to the most critical areas of concern.

As we previously discussed we make a distinction between security and privacy, and we extend the LINDDIN privacy risk assessment model to assess privacy in particular, based on the model published in [4]; LINDDUN is a recognized privacy threat modelling framework, developed by privacy experts. In this paper, we primarily focus on privacy, but security concerns could align with privacy concerns and to identify these instances, several security models could be used. Namely, the Microsoft approach to threat categorization STRIDE [14].

5.1. LINDDIN privacy model

We address PbD recommendations while running different workflows, and to do that we first calculate the data privacy risk. The privacy risk is defined as the risk of losing control of data usage and disclosure. There are multiple privacy threats that we need to mitigate, as shown in Fig. 3, and they protect the corresponding features.

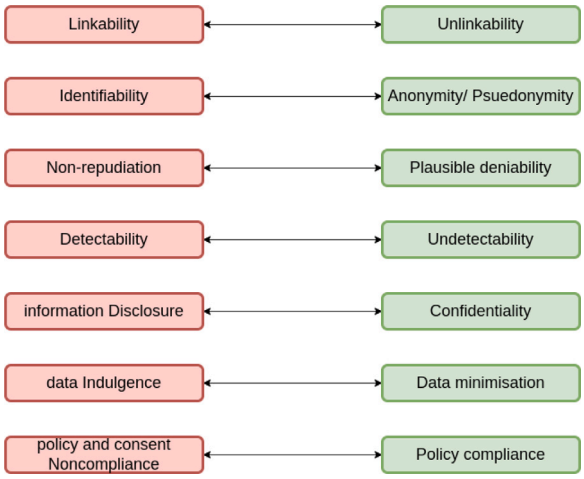


Fig. 3. The LINDDIN model's privacy features and the associated risks.

- Linkability: (L) is the threat that an attacker or unauthorized malicious users can link and relate between two or more data attributes (patient ID, age, condition, etc.). As an example, make a link that patients X and Y are related because they live in the same Postcode and have the same ethnicity. A high degree of linkability can lead to identifiability (and hence unlawful disclosure) and inference that can lead to societal harm. Figure S1 in the **supplementary document** illustrates an example threat tree where the goal is to exploit linkable data in storage. Figures S8 and S15 illustrate linkability threats relevant to different data stages. We highlight in red texts when privacy threats overlap with security threats modelled in the STRIDE threat categories.
- Identifiability: (I) is a privacy threat that can result in identifying the subject of a data object, for instance, via aggregating different data sets and interpreting correlations. This is a threat to the anonymity and pseudonymity of patients in research datasets. Figure S2 illustrated the identifiability threats of data with events in the storage stage. Further illustrations can be found in the appendix, relating transfer and execution events in Figures S9 and S16, respectively.
- Non-Repudiation: (Nr) is a privacy threat when data subjects want to have plausible deniability of events relating to their data. This relates to the threat of failing to retrieve consent for disclosing some data. Similarly, threat trees in the supplementary document are illustrated in Figures S3, S10, and S17 relating to different data stage events.
- Detectability: (Dt) is a privacy threat where an unauthorized user can determine whether an item exists despite padded data, synthetic data, noise, etc. Note that detectability does not equate to data disclosure, this item is not outright disclosed/known, but its existence can be deduced. Similarly, threat trees are illustrated in Figures S4, S11, and S18 relating to different data stage events.
- Information Disclosure: (iD) is both a privacy and security threat, and it can be quite detrimental to data usage control and privacy. Unlike previous privacy risk categories where data is intentionally but unwisely disclosed, which leads to losing control of the degree of disclosure, the iD threat focuses on unintentional disclosure resulting from weak access control, weak encryption, etc. Example iD threat trees are illustrated in Figures S5, S12, and S19 relating to different data stage events.
- Data Indulgence: (In) is the threat of failing to minimize data during workflow events. This has a direct effect on the linkability and identifiability of data subjects. Example threats are illustrated in Figures S6, S13, and S20.
- Policy Non-compliance: (Nc) is a threat that a workflow event does not adhere to data-sharing policies, and these policies are not (fully) enforced. This could be the result of different vulnerabilities like weak authorization/authentication of who can write and change a policy, bad policies that do not address edge cases and hence can be exploited, or no or weak enforcement methods. This is illustrated in Figures S7, S14, and S21 relating to different data stages events.

Table 2
The risk likelihood attributes.

Risk factors	Correlation level	Sensitivity level	Accessibility level	Skill level	Dataset size	Intrusion detectability
Low Risk	Low probable correlation with other Datasets	Data distance is maximal	Can be exploited by consortium members	Complex (multiple) tools	Small	Easily Detectable with no extra tools
Medium Risk	Medium probable correlation with other Datasets	Data is partially processed to preserve privacy	Can only be exploited by trusted third parties	Existing algorithms/malware	Medium	Can be detected with logging and monitored
High Risk	High probable correlation with other Datasets	Data is raw	Can be accessed by outsiders	Simple tools	Large	Cannot be monitored/detected

5.2. Risk evaluation

To calculate the privacy risk of a threat, we can use the formula: $risk(threat) = likelihood * impact$. The impact factor is assigned by the system engineers of the consortium. This indicates what privacy features are prioritized, and what feature has little to no concern. Like [15], in this work, we distinguish five levels of impact: critical (1), high (0.75), medium (0.5), low (0.25), and none (0).

The likelihood component considers several factors that affect the probability of a successful privacy threat. One important factor is the *Correlation Level (CL)* of the data with other datasets. Data with low probable correlation has less likelihood of being exploited, while data with medium or high correlation poses a higher likelihood of being targeted.

The *Sensitivity Level (SenL)* of the data is another factor influencing the likelihood. Raw data, which contains personally identifiable information, is more likely to be targeted compared to de-identified or pseudonymised data. Fully anonymized data, where individual identities are impossible to ascertain, has the lowest likelihood of being exploited.

The *Accessibility Level (AL)* determines who can potentially exploit the threat. If the threat can be exploited by consortium members or authorized third parties, the likelihood may be higher compared to threats that can only be carried out by external attackers.

The *Skill Level (SkL)* of the attacker required to exploit the threat is also considered in the likelihood assessment. Attacks that can be executed using simple tools or existing algorithms have a higher likelihood, while those requiring complex tools or multiple steps have a lower likelihood.

The *Dataset Size (DS)* is another influential factor. Larger datasets provide more opportunities for correlations, aggregation, and pattern recognition, increasing the likelihood of a successful privacy attack.

Lastly, the *Intrusion Detectability (ID)* plays a role in the likelihood assessment. Threats that are easily detectable and can be mitigated through monitoring systems have a lower likelihood, while threats that cannot be easily monitored or detected have a higher likelihood.

By assigning each factor an appropriate values, we can calculate the likelihood component of the privacy risk equation. The values can be set by security engineers subjective to experience, state-of-the-art adversary attacks, and statistical surveys. This comprehensive assessment allows for a more thorough understanding of the potential privacy risks associated with specific threats. Table 2 shows these attributes, and rates these values as low, medium, or high-risk values such that $CL, SenL, AL, SkL, DS, ID \in \{0, 0.5, 1\}$, respectively. The likelihood of a single threat t_m is:

$$likelihood(t_m) = \frac{CL + SenL + AL + SkL + DS + ID}{6}, \tag{7}$$

such that $t_m \in T_w$, where T_w is the set of privacy risks relevant to running the workflow according to the submitted actions. Then, the workflow risk is formulated as:

$$r(T_w) = \frac{\sum_{m=1}^{|T_w|} r(t_m)}{|T_w|} \tag{8}$$

6. The EPI framework architecture and design

After introducing the use cases, the data properties, utility attributes, the event model, and privacy risk categories, we now define the EPI Framework that aims to run a use case adhering to privacy by design. To do that, we delegate different functions into three components: BRANE workflow orchestrator, eFLINT policy reasoner, and Bridging Function Chain (BFC) orchestrator which handles reprogramming the infrastructure/request.

The EPI components are put in a collaborative architecture to address data-sharing challenges on many levels: application level where we need to run different workflows (potentially distributively) regardless of the heterogenous computing capabilities of the node, policy level where we need to manage and adhere to data-sharing agreements and laws, and reprogramming the overlay network of services to enforce some obligations and rules.

6.1. Private, dynamic policies formalization

First, the institutional policies and laws relevant to the EPI Framework need to be encoded in a way that reasoning about them can take place. As such, they need to be formalized and concretized to the level where they are left unambiguous. After all, the implementation of laws is often left for interpretation to accommodate a particular use case; and in an automated system, this concretization has to happen beforehand.

There exist multiple languages for formalizing these notions. One such language is eFLINT [16], which is designed specifically for reasoning about normative rules. Concretely, it can be used to define a *policy specification* which models and then constraints a particular system. Subsequently, it can be queried about the model's state, and its state can be updated, at which time it will detect if any of the constraints are violated. This makes it a suitable tool for implementing the constraints on a system like the EPI Framework.

6.2. Brane — distributed workflow execution

The first two levels of the EPI Framework, the application and policy levels, are implemented by Brane [17], a distributed workflow execution engine designed for use in healthcare (see Section 7). In particular, it allows the cooperative execution of tasks in a workflow on potentially privacy-sensitive data, which is owned by various organizations, in a way that allows its owners to stay in control and exert their policies. This way, it reduces risk using a PbD-approach.

6.3. Bridging function chains model

After the workflow composition with BRANE, and the policy specification, at a lower level, we enforce the conditional policy rules by introducing privacy-as-a-service (PraaS) and security-as-a-service (SecaaS) functions. These functions in the context of the EPI framework are called Bridging Functions (BFs) and are often chained into Bridging Function Chains (BFCs). The BFC orchestrator adapts the setup to promote privacy and security by creating an overlay network of extra services provisioned by the orchestrator and enforcing mitigations in accordance with the policy and workflow requirements.

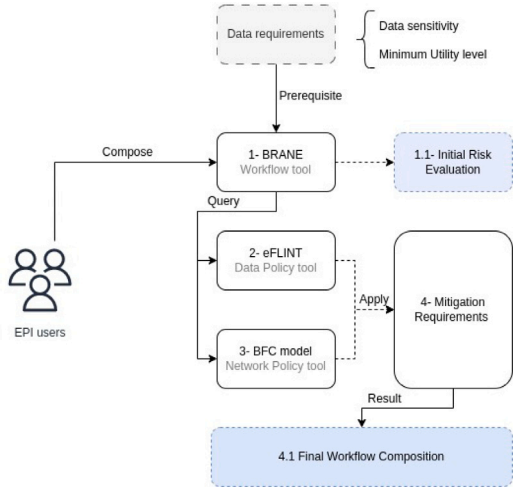


Fig. 4. A high-level view of the EPI PbD architecture.

Fig. 4 provides a high-level perspective of the EPI architecture, showcasing its three core components. Once EPI users define workflow events and input data requirements, the execution and orchestration of these workflows take place via Brane. This module interacts with data-sharing policies, and the BFC model generates setup actions accordingly to provide necessary mitigation. Consequently, the EPI framework finalizes the workflow by integrating the extra BFC services, while concurrently weighing the trade-off between risk and utility associated with this workflow.

7. Brane

This section discusses Brane, the workflow execution engine of the EPI Framework introduced in Section 6, in detail so that we can analyse its privacy risks in Section 9.

Brane operates in a federated manner. The *orchestrator* is the centralized part that orchestrates the work that decentralized parts cooperatively execute. Each decentralized part is a *domain*, representing an organization. Domains control compute resources and share them, and data, with their peers. In addition, domains are empowered to express constraints on the system by having a *policy reasoner* containing the organization's data-related policies and determining which actions are *compliant*, i.e., permitted.

The design decisions behind Brane are motivated by a few core assumptions derived from its intended use-case in the medical domain. These are based on previous work that explored the initial design of Brane [18]. The first of these is inherent to organizations in Brane's use-case:

Assumption 1. Organizations often maximize control over their data and minimize their peers' access to their data.

This is justified by considering the privacy-sensitive nature of the medical data; domains are responsible for keeping that data private. As such, we assert each domain's control over their own actions on their own resources. In turn, domains cannot directly act on the resources of their peers. This fundamental resignation of control over others' actions is an age-old idea, paramount even in the Enchiridion of Epictetus, written in 135 A.D. (e.g., translated in 1955 [19]). Somewhat more recently, it is shared by Mahiru [11], a data exchange and computation system comparable to Brane.

Assumption 2 follows from domains being autonomous:

Assumption 2. Domains cannot be forced by the orchestrator or other domains (not) to act.

Consequently, Brane has fundamentally limited control over domains that already access data. For example, Brane itself cannot compel a domain to delete data at a particular moment[1], and Brane cannot force a domain to observe the conditions on which it was given access to data. Nevertheless, domains rely on Brane to orchestrate their cooperation within domains' *constraints*. Services express these constraints in forms convenient to them, e.g., as *policies* encoding access-control rules; (see Section 9 for examples).

To simplify matters, we assume that domains control their services such that they act in their own interest, i.e.:

Assumption 3. Domain constraints are respected by their own services.

Constraints expressed in policy reasoners may themselves be privacy-sensitive. For example, as per Article 17.2 of the GDPR [20], "personal data shall, with the exception of storage, only be processed with the data subject's consent". The fact that a patient has given consent reveals the presence of their data in the dataset. This is formalized as:

Assumption 4. Constraints may be privacy-sensitive.

Because of this, Brane does not inspect the state of any policy reasoner directly. Instead, domains react to requests to permit cooperative actions at their own discretion, at their own pace, without revealing their underlying rationale.

Finally, for a practical implementation, we assume:

Assumption 5. Whether data is present on a domain, as well as data metadata, is non-sensitive information.

Concretely, the orchestrator may disseminate metadata without consulting policy reasoners. Domains may always control the dissemination of metadata *within* their domains (e.g., to their users), as well as metadata leaving their domains, but this is transparent to Brane. Intuitively, this metadata represents the *public* effects of *private* policies.

Brane's design is discussed in Section 7.1. Then, using the assumptions stated above, properties preserved by Brane are defined in Section 7.2. The preservation of these properties is elaborated on in Section 7.3.

7.1. Framework overview

In this section, we introduce the main components of Brane and how they cooperate together on a high level to implement policy-aware workflow execution.

7.1.1. Components

The Brane framework is a collection of six components fulfilling separate functions (Fig. 5). Three compose the orchestrator and direct the work performed between the domains: the *driver*, which traverses a plan and emits a series of events; the *planner*, which takes a workflow and assigns each of the steps to a domain; and the *global audit log*, which logs the behaviour of the centralized orchestrator and any information shared by local domains to allow ex-post enforcement. Then, every domain consists of the other three components: a *worker*, which takes events emitted by the driver and processes them locally; a *checker*, which encapsulates the domain's policy reasoner; and a *local audit log*, the domain-local counterpart to the global log.

Then there are two domain-local components that Brane interacts with but are typically implemented by third-party services: a backend, which executes tasks as containers; and a data source, which provides access to domain data. These components are the only ones to deal with actual data instead of only control messages. Finally, there is also the user, who interacts with the framework to execute a workflow.

[1] Note that domains can always choose to implement some cooperative scheme outside of the framework to provide this control, if necessary.

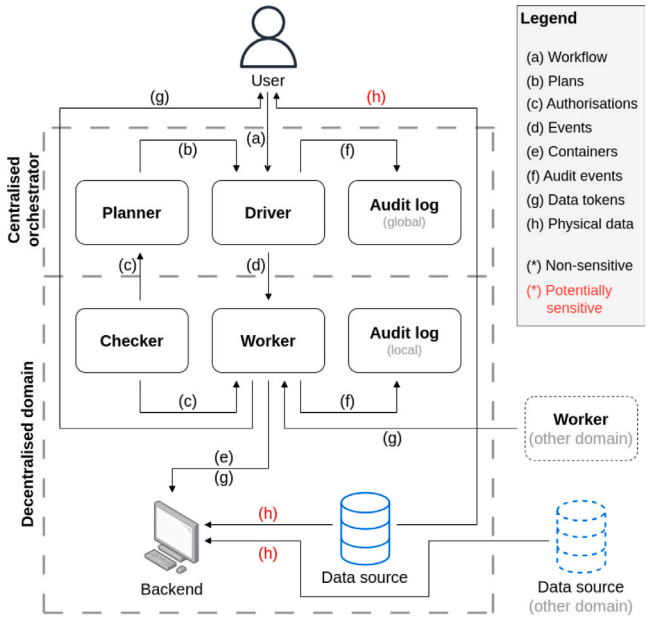


Fig. 5. Graphical overview of the BRANE framework. The rectangular nodes indicate components. Components comprising the (centralized) orchestrator occur uniquely, while the rest are replicated per domain. Third-party services are represented by nodes with suggestive symbols. Inter-node edges represent communication channels, and are directed as per the "main" information flow; e.g., audit logs mainly receive information.

To facilitate domain autonomy (Assumption 2), the implementations of the components can vary between running instances of the framework or between domains within the same instance. It also allows domains to interface with already existing systems, such as patient consent databases. Finally, it affords domains to choose their own level of scalability: *one* running service may provide the functionality of *several* components, or vice-versa.

7.1.2. Interaction between components

The components interact with each other to achieve workflow execution as discussed in [18]. In summary, execution begins when a user initiates an interaction with the driver and submits a workflow. Immediately, the driver forwards the workflow to the planner to *plan* it: in this procedure, it compiles the user-friendly, but abstract, workflow representation to an executable *plan*. Most notably, domains are assigned to tasks and to input datasets, encoded they will execute that task or provide that input, respectively. During this, the planner interacts with checkers to discover which assignments are permitted by the domains' constraints.

Once planned, the planner sends the plan back to the driver, which then starts to traverse it. The driver emits events to workers to execute the tasks encoded in the plan as it does so, including any auxiliary tasks such as transferring data. The workers process these events only if it is in accordance with the checkers relevant to that event, implementing a secondary check of domain constraints.

When all events have been processed, the driver returns the final result to the user and the interaction ends. Note that this never contains any potentially-sensitive data; instead, the user must separately request workers to get access to it, as if they were a worker themselves. This homogenizes all data accesses from the perspective of checkers.

During this process, the audit logs are updated by the other components to keep track of the system's (past) state.

7.2. Definition of safety and privacy properties

In this section, we define fundamental properties of any system using the BRANE framework.

Table 3
The *core ontology*: relations with fixed names and structures, but whose elements vary per use case and over time.

Name	Structure	Intuition
function	(compute, data)	f in $y := f(x)$
input	(compute, data)	x in $y := f(x)$
output	(compute, data)	y in $y := f(x)$
assigned	(compute, agent)	This worker agent is assigned to process this event.
involves	(compute, agent)	Processing this event requires the authorization of this agent.
labelled	(agent, X, label)	This agent gives this X (data, compute, or agent) this label.

7.2.1. Core ontology

BRANE's orchestrator and domains (together, *agents*) communicate essential information in terms of a *core ontology*, the relations in Table 3. Concretely, they create, communicate, store, and reason about members of these relations. As such, the properties defined in this subsection are formulated in terms of these relations.

The core ontology formalizes the (compute) events discussed in Section 4.3. To do so, it defines sets of identifiers that refer to BRANE entities: *compute* names all compute events that occur in the system; *agent* names all domains; and *data* abstracts over all kinds of data in the system, including implementations of processing functions. In addition, there is a fourth set called *label* that denotes arbitrary strings, although it must at least contain the string AUTHORIZED. It acts as a collection of arbitrary metadata assigned by checkers (see Section 7.2.1).

Note that the compute-set only names the events; its parameters are defined using the *input*, *output* and *function* relations. Similarly, compute events are assigned to an agent using the *assigned* relation, and which agents to query before the event may be processed are given by *involves*. Finally, *labels* are assigned to entities using the *labelled*-relation.

In the rest of this section, we denote logical variables using Greek letters or suggestive uppercase letters. For example, A denotes an arbitrary agent. We denote the membership of ϕ in relation Φ as formula $\psi \in \Phi$ or $\Phi(\psi)$, and de-structure relation elements into tuples of elements as per Table 3. For example, if $output(C, D)$ then D is data.

7.2.2. Properties defined via the core ontology

The essential properties preserved by the framework are now defined in terms of the Core Ontology.

Property 1. *Compute events using a data value whose owning agent has never given authorization are not executed.*

This property relies on the assumption that workers always adhere to their own domain's checker (Assumption 3), i.e., workers only compute, access, and transfer data as authorized by their checkers. However, due to Assumption 2, *other* agents will not necessarily follow this restriction. BRANE cannot prevent the data from being propagated further, e.g., via other communication channels, once they have access.

Assumption 4 Assumption 5 reveal a tension checkers must navigate during cooperation. Fundamentally, checkers express constraints on the system which must be obliged. However, checkers may not want to do so when it reveals private information. After all, an attacker may extract that constraint by observing the effects of strategic interactions, as per the algorithm in [21]. As such, it becomes important for checkers to be able to decouple their emitted authorizations from their internal constraints, in order to be able to implement countermeasures to such attacks (e.g., introduce acceptable levels of noise). To this end, BRANE generally provides that:

Property 2. *Checkers fully control their authorizations.*

However, in practice, we expect that many constraints are not privacy-sensitive. Checkers are willing, and incentivized, to share this information, as it affords others making informed decisions. For example, if checkers eagerly authorize plans, plans can be made and executed more efficiently. Generally:

Property 3. *Agents may share metadata to aid cooperation.*

Agents are free to exchange control information, even beyond the core ontology. For example, they may exchange *hints*, unreliable suggestions of plans considered desirable, serving to guide planning, but not substituting authorization.

Property 4. *Agents can prove (e.g. to an auditor) that their actions comply to the policies of their peers.*

It is important for any system that requires compliance that it should be possible to detect non-compliant behaviour. In Brane, this is done by creating a trail of *receipts* which can be used to statically determine if processing an event was allowed by all checkers involved. Moreover, it can also be used to verify whether involvement was computed correctly (see Section 7.3.1).

Property 5. *Data transfers can be delayed until the requiring function is being executed, i.e., data is transferred lazily.*

Brane attempts to maximize data privacy by minimizing data transfers; data is only transferred once it must be accessed (see Section 7.3.3). This helps to combat Data Indulgence (Section 5.1) and improve performance and compatibility, as it defers access to purpose-built containers.

To conclude, the following property of how agents should behave can be derived from the previous properties:

Property 6. *An agent can stay in control of its data if it only enables access to agents for which it has evidence they are assigned to a compute event requiring the data, and that all agents involved with that event have authorized it.*

Effectively, it establishes a norm characterizing well-behaved agents: workers should only access data in order to perform a compute authorized by all involved checkers. By Assumption 2, Brane cannot guarantee that agents will comply to this norm; so instead, it is up to the individual agents to decide which other agents can be *trusted* to comply.

In the next subsection, background will be provided for how these properties are upheld and what it means for agents to be *involved* or to *authorize* a compute.

7.3. Enforcement of properties

In this section, we explain how some key properties of Section 7.2.2 are preserved in practice.

7.3.1. Centralized annotation of computes

The core ontology fixes a set of essential concepts on whose meaning all domains agree; this meaning arises by their special usage by Brane components. Here, we discuss the relations populated by the centralized services.

Firstly, *input*, *output*, and *function* are populated to reflect the contents of workflows submitted by drivers, also reflecting the formulation of *compute (events)* in Section 4.6. Secondly, *assigned* is populated by the planner. Thirdly, *involves* is populated by the orchestrator, relating each compute to domains whose authorization is needed before execution may begin (see Section 7.3.2, to follow).

The centralized control of these relations ensures distinct computes and data are uniquely identified. Other properties make them more meaningful; concretely, (**1**) once assigned, each compute's outputs, functions, inputs, and involved agents are fixed, (**2**) each assigned compute has exactly one function and exactly one output, and (**3**) each assigned compute involves all owners of its input data. These properties let other services reason given incomplete information. For example, a checker observing $(t, x) \in function$ can infer $\forall y : y \neq x \rightarrow (t, y) \notin function$.

7.3.2. Decentralized authorization and labelling

Checkers communicate asynchronous control information by populating the relation *labelled*. Intuitively, labels establish abstractions over compute events, agents, and data. The domain of labels is unspecified, but certainly contains Authorized, which has a fixed, special meaning, arising from Property 6. Our notion of labels is inspired by *collections* and *categories* in the Mahiru system [11]. Thus, preserving Property 2 follows from the preservation of Property 7:

Property 7. *Agent A uniquely controls which elements of labelled match (A, ϕ_1, ϕ_2), i.e., it controls its own labellings.*

This stronger property is enforced by each agent independently. An agent considers $(A, X, L) \in labelled$ true only when given proof. Proof is metadata, created and disseminated by agents, for example, via gossip between domains.

We leave the details of our implementation of proof out of scope of this paper. Here, it suffices to say that our approach centres around agents creating and communicating cryptographically signing *(logical inference) rules* for inferring proof of membership of elements in *labelled*. Crucially, the framework fixes a universal well-formedness criterion for rules: rules inferring $(A, X, L) \in labelled$ must be signed by agent A, but may be *applied* by any agent. This approach strikes a desirable compromise. On one hand, this preserves Properties 2 and 7; agents ultimately control their labellings in general and authorizations in particular. On the other hand, rules let agents express authorization in terms of abstract conditions. For example, agent Amy's rule expresses "I authorize any compute whose function is labelled *pseudonymizing* by Bob"; in this example, Amy effectively delegates the authorization of particular computes to Bob.

Note that the usage of cryptographic signatures ensures *non-repudiation* and *integrity* of labellings and authorizations; they can be neither retracted nor forged.

Like the D1LP language [22] and Mahiru [11], we frame authorization as a *trust management* problem: authorization of compute is proven by attributing it the Authorized label as the result of reasoning with logical rules. However, our computes are not resources with unique owners responsible for providing authorization. Instead, involvement defines the *set* of agents whose authorizations are needed.

7.3.3. Data tokens

As stated for Property 5, it is practical if data transfers can be performed lazily instead of eagerly. However, from a checker's perspective, it is still valuable to reason about data access as explicit transfers because other domains cannot be relied upon to adhere to access restrictions (Assumption 2).

Brane solves this tension by representing data access with *access tokens*. Each is a (cryptographically-)signed value encoding where a dataset can be accessed, how it can be accessed, and with which credentials. Workers can exchange these tokens to emulate explicit data transfers and to model which worker has access to which dataset. When a worker processes an event, it can consequently pass the token to a processing function so that it can access only the required data at its own leisure.

7.3.4. The audit logs

The audit logs let Brane's various components store metadata needed to justify their actions. For example, a worker logs their receipts before beginning work.

The audit log can be seen as a large collection of facts distributed over both the global audit log and the various local audit logs. At runtime, it is unnecessary for any component to have a full overview of all logs, nor do they need to be synchronized to provide a coherent view.[2] To facilitate this, it is fundamentally unordered. Instead, required ordering can be encoded in the facts themselves (e.g., by timestamping).

When an auditor wants to perform an audit, the audit log as a whole must at least provide domains' receipts, information about which events have been started and completed and which checkers existed at the start of each event. Brane does not require this information to be available automatically; for example, an auditor may have to manually request access to a local audit log of a domain. However, domains are incentivized to ease the task of auditing, as this encourages other domains to trust them with work.

8. BFC orchestrator

In this section, we discuss the last layer of the EPI framework, as shown in Fig. 4. The BFC orchestrators mitigate any threats that were not addressed at the Brane and policy levels. This PraaS/SecaaS network orchestrator provides the means to provision mitigation methods based on the requested workflow actions, and subsequently the relevant threats. A number of these threats might already be out of concern (low-risk score) due to the current set-up of the environment, for example, if the action submitted is running data transfer with SFTP, then unauthorized access to data is of low risk. While Brane, powered by policies, guarantees a number of PbD properties, such as data minimization done by sharing data tokens instead of physical data, it does not ensure minimal risk because it is still highly subjective to the implementation of workflow and the environment setup. The BFC orchestrator deploys preventative measures, and enforces a minimal risk threshold, or otherwise rejects the workflow request. BFC offers a configurable overlay network that is not provisioned at higher layers of the EPI framework, and by that it provides an adaptive infrastructure with privacy risk in mind. It is still essential to mitigate any privacy risk that has not been mitigated by design via Brane, and to do that there is a need for an adaptive set-up offering PraaS/SecaaS. To ensure reproducibility and transparency in research using the EPI framework, it is essential to track and document the specific anonymization and pseudonymization techniques applied to the data. The framework must maintain detailed logs of all data transformations and mitigations executed during each workflow run. This logging ensures that researchers understand how data has been altered and can verify that identical functions are applied consistently across multiple executions. When the EPI framework is required to apply an anonymization function during operations like file transfers via FTP/SFTP/SSHFS, it must identify the file format and select the appropriate anonymization function capable of handling that specific format. To achieve this, the framework incorporates a comprehensive library of anonymization functions mapped to corresponding file formats. New functions, such as those for anonymization, are added to the EPI framework through a structured development process involving framework administrators with elevated privileges. Researchers may propose functions, but the actual development and integration are managed by administrators to ensure consistency and security.

In previous work [23,24], we proposed the BFC orchestrator, a dynamic orchestrator that automates the programming of the underlying networks to secure health data-sharing. Data-sharing is secured by orchestrating and managing services to add security and privacy value to each communicating node, irrespective of each node's capabilities. As mentioned earlier, the BFs are mitigation ready-to-use functions and are implemented with containerization, which allows flexible instantiation and chaining to enforce increasingly complex policies. In the context of this paper, a communicating node refers to any data flow even within the same domain; management traffic or physical data flow (edges (e), (h), and (g) in Fig. 5), relating to data in storage, execution, and transfer events.

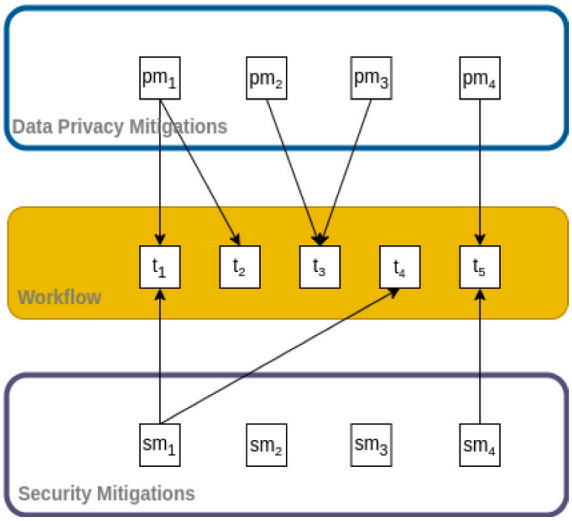


Fig. 6. Possible data privacy and security mitigations and the relation to threats.

The *adaptation* of the underlying networks is done after the workflow submission, then the BFC orchestrator evaluates the current setup, taking policy requirements and data requirements as an input, to finally translate that into setup actions. For example, data transfer from Domain A to Domain B is only permissible if the privacy risk is below a threshold, which can be done if A is able to encrypt and decrypt via a stream cypher. The automated setup of the infrastructure is also required to achieve reachability of the end-point nodes, optimal privacy and security across collaborating domains, reasonable network performance, bridging services availability, hardware selection and scalability, and ultimately, abiding by policy requirements.

The BFC orchestrator is the last layer before actually running the workflow, as part of a use case. The orchestrator operates under Assumption 6 that:

Assumption 6. There exists an exhaustive threat database.

Assumption 7. There exists a many-to-many relation between threats and the BF functions.

Initially, the evaluation of the environment setup and utilized event function methods is done by cross-referencing the threat database for relevant threats. The relevance of these threats is situational and is directly mapped to the workflow by analysing the pattern formalized via the events model. On top of that, Assumption 7 express a many-to-many relation between said threats and mitigation measures. Meaning that many threats can be mitigated in many ways, by utilizing different BF implementations. The choice of which BF to instantiate, and chain, is dependent on the policy workflow requirements. One thing to note is that these mitigations can affect utility in varying accordance. This is further illustrated in Fig. 6, such that these measures can be privacy or security mitigations.

Fig. 6 illustrates the many-to-many relation between threats relevant to a workflow, and makes a distinction between data privacy mitigation (such as anonymization, aggregation, etc.) and security mitigations (such as network ad-hocs, VPNs, and access control mechanisms).

[2] In fact, this is discouraged, because incoherence between logs likely indicates non-compliant behaviour.

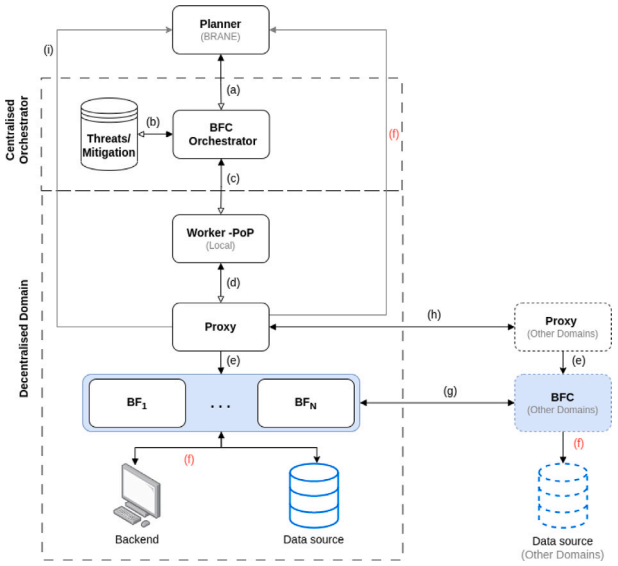


Fig. 7. Overview of the BFC Framework. There are two main layers to deploy the BFC framework, the centralized orchestration layer (the central place) and the decentralized domain(s) (services in this layer are duplicated/domain in the system). The edges illustrated between services are: (a) Event plans, (b) threat queries, (c) Service management, (d) route configuration, (e) rerouted dataflow, (f) Physical data, (g) distributed service chaining, (h) domain-to-domain dataflow, and (i) Data tokens.

At this stage, Assumptions 8 and 9 at play, as explained in Section 7, the workflow submitted already have checked against the policy. Then the policy will give us an inkling of what is an acceptable risk, and what requirements to enforce. Moreover, as previously explained, and as per assumption Assumption 10, the utility attributes in Section 4.2 are set by the user submitting the workflow.

Assumption 8. Workflows submitted are already checked against policies

Assumption 9. Acceptable risk thresholds are set by policy.

Assumption 10. The utility attributes are set by the user whilst submitting the workflow request.

8.1. BFC framework overview

In this section, we introduce the main components of the BFC framework, the design decisions, and an overview of policy requirements enforcement.

8.1.1. Components

After submitting the workflow request to BRANE, the request goes through the first two layers to plan and ensure compliance. The BFC Framework is deployed to enforce minimal privacy risk by placing extra PraaS and SecaaS; BFCaaS. The framework consists of 4 components spanning over two hierarchical layers control plane and a decentralized plane, as illustrated in Fig. 7: (**1**) BFC orchestrator, which is where service instantiation, placement, and provisioning decisions are made. (**2**) Worker nodes, which are Point of Placement (PoP) where BF services are hosted to be instantiated. (**3**) Proxy, which holds routing information to enforce dataflow via BF or BFC. (**4**) BF; containerized instances of these services.

For the implementation of the BFC orchestrator, we need to be able to instantiate on-the-fly services and redirect all actions to go through these services. The implementation is further detailed in previous works [25], where we utilize cilium[3] and deep reinforcement learning to optimally provision and place BFCaaS.

After giving a high-level overview of the BFC framework components and implementation, we will now explain how to reason about the risk and enforce policies.

8.1.2. BFCaaS & privacy requirements

To evaluate the current setup, we need to refer back to, Assumption 6 and consider, a set of possible privacy threats $T_{workflow}$ that are relevant to running the workflow. BF is the set of possible mitigation functions, such that $bf_j \in BF$, and BF is:

$$BF = \{sm_k | k = 1, \dots, d\} \cup \{pm_l | l = 1, \dots, v\}, \tag{9}$$

sm_k and pm_l are members of the data privacy and security measures sets, respectively. Each workflow in accordance with the requested actions has a set of expected known threats, and these threats are more or less relevant ($risk \approx 0$) according to the set-up in place relating to the type of function utilized to run a transfer, storage, or execution events.

As per Assumption 7, a single threat can be mitigated via a single bf_i or chain of bf; $BFC_j \in BFC$, such that:

$$BFC_j \in \mathcal{P}(BF), \tag{10}$$

A chain is a set of bf combinations, where you have one or more bf, and $\mathcal{P}(BF)$ is the power set of BF. Moreover, Property 8 showcases the degree of mitigation these measures have relating to a threat.

Property 8. *A bridging function or a chain of bridging functions can mitigate a threat to a varying degree.*

Assumption 11. A bridging function or a chain of bridging functions are implemented securely, hence added inherent risk is negligible.

In relation to Property 8, we measure the mitigation ratio by defining the mitigation factor f, such that $t_m \in T_w$:

$$f(t_m, bf_i) = \begin{cases} 1, & \text{fully mitigated} \\ \frac{\delta r(t_m)}{r_0(t_m)}, & \text{otherwise} \end{cases} \tag{11}$$

where $\delta r(t_m) = r_0(t_m) - r_1(t_m)$, such that r_0 and r_1 are the risk of threat t_m before (initial) and after applying the mitigation measure bf_j. $f(t_m, bf_i) \leq 1$ and $f(t_m, bf_i) > 0$ because if $r_0(t_m) \approx 0$ then $t_m \notin T_w$. $\delta r(t_m) \geq 0$ building on Assumption 11, where BF functions implemented by security experts. Moreover, to calculate the cumulative effect of applying the chain BFC to mitigate threat t_m:

$$f(t_m, BFC_j) = \begin{cases} \sum_{p=1}^{|BFC_j|} f(t_m, bf_p), & \text{If} < 1 \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

The conditional function ensures that the mitigation factor $f \leq 1$, meaning it is already fully mitigated if $f = 1$. Moreover, the remaining risk of a threat after applying BFCaaS is:

$$r_r(t_m) = r_0(t_m).(1 - f(t_m, BFC_j)), \tag{13}$$

where $0 \leq r(t_m)_r \leq 1$.

Moreover, to calculate the remaining risk of the total workflow $r_r(T_w)$, then we look at all the risks of all the threats, and the different mitigations applied. The remaining risk of a workflow is calculated such that:

$$r_r(T_w) = \frac{\sum_{m=1}^{|T_w|} \left[r_0(t_m) \cdot (1 - \sum_{j=1}^{j=N} f(t_m, BFC_j)) \right]}{T_w}, \tag{14}$$

where N is the total number of BFCaaS mitigation deployed.

[3] https://cilium.io/use-cases/service-mesh/

8.2. Privacy & security vs utility

Other than the mitigation factor, utility requirements also factor into the decision of BFCaaS provisioning. Some BFs might affect the utility depending on the type of function, so as an example anonymization function can increase the sensitivity, hence decreasing utility. This effect is formalized as utility factor g, and $g : U_{bf_{(i-1)}} \times bf_i \rightarrow U_{bf_i}$, such that U_{bf_i} is the new utility of $U_{bf_{(i-1)}}$ after applying bf_i. If $U_{bf_i} = U_0$; the initial utility, then bf_i has no effect on utility.

The resulting utility after applying the BFCaaS is formalized as U_r, such that:

$$U_r = g(U_0, BFC) = g(U_{BFC_{j-1}}, BFC_j) | 1 < j < N, \tag{15}$$

in Eq. (15) U_r the calculated in terms of initial utility U_0, and the set of all mitigation chains BFC.

$$g(U_0, \phi) = U_0, \tag{16}$$

where in Eq. (16) no mitigations were applied, hence $BFC = \phi$, and no effect on utility occurred.

Property 9. *BFC mitigation decision is a constrained optimizing problem.*

Property 10. *BFC are deployed to minimize privacy risk with utility in mind.*

Property 11. *A workflow request can fail if utility or privacy requirements are not met.*

With that in mind, searching for the optimal BFC set of mitigation chains is dependent on the utility and privacy risks acceptable U_{acc} and r_{acc}, as discussed in Section 4.2 and Section 5.2. This search problem is a constrained optimizing problem, in reference to Property 9, where the BFC orchestrator aims to minimize privacy risk and maximize utility according to the following constraints:

$$U_r \geq U_{acc} \tag{17}$$

$$r_r(T_w) \leq r_{acc} \tag{18}$$

This problem is addressed with a heuristic search algorithm, as in Algorithm 1. The problem is simplified to find the best effort BFC mitigation chains with Property 11 in mind. We start by assigning BFC, U_r, and $r_r(T_w)$ to initial values in lines 1–3. We set λ to 1 in line 4, to prioritize minimizing privacy risk (privacy risk has maximum priority in the first iteration of the search). Loop over all threats in T_w, and try to find the best mitigation (chain) for this threat by finding the set combination $BFC_j \in \mathcal{P}(BF)$ that minimizes risk and maximizes utility in line 8–9. Then check the resulting effect of this BFC_j in lines 11–12, and compare it against the acceptable values. If this fails, then remove BFC_j from the BFC sets that will be deployed, and tune the λ parameter to change the search weight and prioritize the utility more in lines 18–19. Try again until $\lambda < 0$, then exit the while loop, and look into other threats. After going over all threats and adding mitigation (chain) sets to BFC, check the effect of the whole set as in Eqs. (14) and (15), and reject or accept the workflow according to the constraints.

Algorithm 1 Search for Optimal Mitigations

Require: T_w, BF, r_{acc}, U_{acc}
Ensure: BFC, $r_r(T_w)$, U_r
1: $BFC \leftarrow \phi$
2: $U_r = U_0$
3: $r_r(T_w) = r_0(T_w)$
4: $\lambda = 1$
5: ordered list T_w
6: **for all** $t_m \in T_w$ **do**
7: **while** $\lambda \geq 0$ **do**
8: $BFC_j \in \mathcal{P}(BF)$
9: $BFC_j = argmin_{BFC_j}(\lambda \cdot f(t_m, BFC_j) - (1 - \lambda) \cdot g(BFC_j))$
10: $BFC \leftarrow BFC_j$
11: $r_r(T_w) \leftarrow$ new $r_r(T_w)$
12: $U_r \leftarrow g(U_0, BFC)$
13: **if** $r_r(T_w) \leq r_{acc}$ **and** $U_1 > U_{acc}$ **then**
14: Success!
15: **else if** $r_r(T_w) > r_{acc}$ **and** $U_r > U_{acc}$ **then**
16: Continue to the next threat
17: **else**
18: $BFC - \{BFC_j\}$
19: $\lambda \leftarrow \lambda - 0.1$
20: **end if**
21: **end while**
22: **end for**
23: **if** $r_r(T_w) > r_{acc}$ **or** $U_r < U_{acc}$ **then**
24: Reject workflow
25: **else**
26: **return** BFC, $r_r(T_w)$, U_r
27: **end if**

9. DIPG USe CASE: A walk-through

Previous sections laid out all the use case concepts, the risk assessment method, and the EPI Framework components. This section expands on the first use case; the DIPG; and provides a walk-through that applies all said concepts to showcase the EPI framework's capabilities in privacy risk minimization. Since not all the details of this use case are known, we make assumptions to concretely reason about it, and give example setups of the data exchange environment and the associated EPI framework decisions.

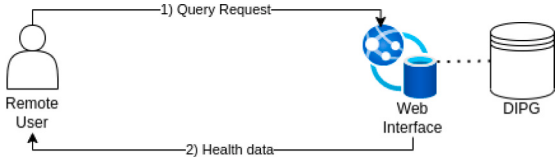


Fig. 8. The workflow illustration of the DIPG use case.

9.1. An EPI use case: DIPG-registry

One of the use cases under the EPI project is the Diffuse Intrinsic Pontine Glioma (DIPG) registry, founded by the SIOPE DIPG/DMG network. The registry was established in 2011 with the goal of advancing DIPG research. The DIPG is a rare paediatric brain cancer for which there is no curative treatment. The registry holds information on DIPG patients across Europe and a partner registry in North America, the International DIPG/DMG Registry, with patient data from the USA, Canada and Australia [26]. The registry serves the goal of advancing DIPG research by granting researchers conditional access to selected datasets to perform analysis with more data points and encouraging members to contribute datasets to the registry.

The registry serves to improve DIPG research by granting members (conditional) access to selected datasets in order to perform analyses with more data points and thus higher efficacy. Members submit clinical data from their institution to the DIPG Registry via a secure online CRF-structured web application and database.[4] Fig. 8 provides an illustration of the data flow required to run this use case. This is a simple workflow of query requests of remote users, and getting back the health data queried.

[4] https://dipgregistry.eu

Table 4
The DIPG event model.

Index	DIPG workflow task
1	$READ\langle EHR_1, loc_{DIPG}, SQL\rangle$
2	$MOVE\langle loc_{DIPG}, loc_{hosp}, EHR_1, FTP\rangle$

9.2. Data properties & event model

Table 4 specifies an example DIPG workflow request, where an EPI user with a data scientist role wants to read and transport a small DIPG dataset EHR_1 from the DIPG repository location loc_{DIPG} to the hospital as a destination loc_{hosp}.

The data properties related to this EHR_1 record are sensitivity and utility. First, in this use case, the data maintained in the DIPG repositories is pseudo-anonymized, hence the data sensitivity is medium and therefore can be set to 0.5 ($distance = 0.5$); thus is not completely anonymized.

Second, the required data utility attributes for this workflow are set by the user. The requested data type for this workflow is EHR, where EHR_1 is expected to be made available. No compute power is needed (no processing required whilst executing this workflow), meaning $r_{req} = 0$. Moreover, a higher distance is acceptable for this workflow (> 0.5), and the distance required can be $distance_{req} = 1$. Subsequently, the EPI framework is allowed to further anonymize the data without compromising the workflow.

With $E(u_{type}) = E(u_{compute}) = E(u_{distance}) = E(u_{size}) = 1$, further assume that type, distance, and size requirements are equally important, and compute resources requirement is not. Hence, the utility parameters are set to $\alpha = \beta = \theta = \frac{1}{4}$. As a result, acceptable utility is a weighted sum of the expected attribute values $U_{acc} = \frac{3}{4}$.

The workflow events are modelled in Table 4, where there are two events: storage event of type read with SQL as the read function method, and transfer event of type move with FTP as the transfer function method.

9.3. Initial risk

According to the utilized function methods specified in the events model, the workflow might be vulnerable to a number of threats. Table 5 showcases an example threats list relevant to the DIPG workflow. In the table, we also set the risk attributes according to Section 5.2. Relevant threats are primarily related to the data in storage and data in transit stages events. These threats concern one or many privacy risk categories.

To calculate the initial risk, we go over the 14 threats' risk attributes, and average the total risk as in Eq. (14). In this example, the initial risk is:

$$r(T_w) = \frac{\sum_{m=1}^{14} r(t_m)}{14} = \frac{\sum_{m=1}^{14} likelihood(t_m) \cdot impact(t_m)}{14} \tag{19}$$

The exact answer is subjective to the set impact for each threat, and this is set at a higher consortium level. For this example, assume that the impact of these threats is the maximum value ($impact = 1$), hence $risk = likelihood$ and the initial risk is then only dependent on the set risk attributes, such that:

$$r(T_w) \approx 0.54 \tag{20}$$

9.4. The EPI framework's mitigations & residual risk

This section focuses on demonstrating how certain privacy risks can be mitigated with the EPI Framework. Starting with BRANE PbD mitigations, then the formalization of policies stipulated in privacy regulations and contracts, and lastly BFCaaS setups.

9.4.1. BRANE mitigations

Because of its Privacy-by-Design principles, Brane allows for the mitigation of certain risks. Mostly, this is done by empowering the policy reasoners on a domain to dynamically apply the mitigations. For example, Brane has the option to lower the risk of an attacker getting access to Linkable Data Attributes, but only if the policy reasoners implement the correct constraints on Brane. As such, most of these mitigations are discussed as particular policies in the next section.

However, there are a few mitigations that Brane always provides regardless of policy. Mainly, by making use of its audit logs, Brane improves the detectability of certain threats. In particular, any threat relating to data access for which the involved checkers have to give authorization can be detected as a discrepancy in the logs' proof trees. However, the attacks can only be detected if the violation occurs within the framework itself: for example, it can be detected if a domain processes a workflow on a dataset to which it should not have access, but not if a domain is hacked and the data is copied manually without leaving a trace. As such, the detectability risks of such threats are reduced to medium risk (M).

Further, the skill level required to perform attacks is lowered in a lot of cases by the fact that Brane only needs very limited access to a domain through well-defined interfaces. This makes it harder for attackers to perform Policy Modification on a particular domain, which further reduces the Skill risk of other threads. However, because the policies are residing on domain infrastructure and not Brane-managed infrastructure, the Skill required to attack the policy is also dependent on how well-secured the domain is.

9.4.2. eFLINT mitigation

In this section, a brief explanation of policies that potentially mitigate some of the risks listed in Table 5 is provided. All examples in eFLINT are shortened for brevity, a detailed explanation of the code can be found in previous work [27].

Granting access to authorized users only: An access control mechanism is one way to mitigate linkability. Access control allows organizations to restrict access to data values, for example based on the sensitivity of data and/or based on the roles of users. For the DIPG use case, access to the data values is granted only to users who are affiliated with a member organization and have a project proposal approved by SIOPE DIPG/DMG. The following eFLINT fragment formalizes that:

```
1 Fact approved-project Identified by project *
      member.
2 Extend Act read Holds when (Exists project,
      member:
3   selected(asset,project) && approved(project,
      member)
4   && affiliated-with(user,member)).
5 ?Enabled(read(<user>,DCOG,<EHR1>)).
```

Listing 4 Grant access to researcher with an approved project

Listing 4 specifies that the action "read" is to be understood as accessing a file, and it is enabled when there exists a data value(asset) that is selected for an approved project and the user that is requesting access is affiliated with a member organization. The query is evaluated to be true if user is affiliated with a member organization and the user has a project approved for which EHR1 has been selected. Such a policy ensures that unauthorized access does not take place and reduces the high risk associated with unauthorized access, in Table 5, to a minimum.

Granting access to authorized purposes: The GDPR dictates that data collected for a certain purpose be used only for said purpose unless a different legal basis applies. In addition to authorizing users based on approved projects and affiliations, access is granted based on authorized purposes specified by data subjects or controllers. By doing so, the probability of unlawful data sharing is minimized. Furthermore, restricting access based on authorized purposes restricts organizations

Table 5
The DIPG workflow's relevant threats and the example risk attributes. After applying the EPI Framework, some threats are mitigated, translating to lower risk estimations. The new values are colour-coded, where grey values are mitigated by Brane 's design, orange values are mitigated by the chosen eFLINT policies, and the blue values are mitigated by the BFC orchestrator. L: **L**inkability, I: **I**dentifiability, Nr: **N**on-**r**epudiation, Dt: **Dt**ectability, iD: **i**nformation **D**isclosure, In: data **In**dulegnce, Nc: policy **N**on-**c**ompliance.

Threat name	Stage	Category	Correlation	Sensitivity	Accessibility	Skill	size	Detectability
Unauthorized Access	Storage	iD	M →L	M → L	H →L	H →M	L	H →M
Linkable Data Attributes		L	M →L	M → L	L →L	L	L	H →M
Identifiable Data Access		I	M →L	M → L	L	L	L	H →M
Profiling and Tracking		Nr	M →L	M → L	H	M	L	H →M
Long Retention		Nr	M →L	M → L	M	H →M	L	H →M
Unlawful Retention		Nr	M →L	M → L	H	H →M	L	H →M
Policy Modification	Transit	Nc	M →L	M → L	M	H →M	L	M
Eavesdropping/Sniffing		iD	M →L	M → L	H →L	H →L	L	H
MitM		iD	M →L	M → L	H →L	H →L	L	M
Traffic Analysis		L	M →L	M → L	H →L	L	L	H
IP Tracking		L, I	M →L	M → L	H →L	L	L	H
Metadata Leakage		iD, Dt	M →L	M → L	H →L	H	L	H
Oversharing of Data		Nc, In	M →L	M → L	M →L	H →M	L	M
Unlawful Sharing		Nc	M →L	M → L	M →L	H →M	L	M

from re-purposing data values, since controllers cannot re-purpose data values unless authorized by data-subjects or other legal basis.

The policy in Listing 5 specifies the action of collecting personal data.

```
Fact accurate-for-purpose Identified by data *
    purpose .
Extend Act read Holds when (Exists project,
    member:
  selected(asset,project) && approved(project,
    member)
  && affiliated-with(user,member))
  && accurate-for-purpose()).
?Enabled(read(<user>,DIPG-purpose,<EHR1>)).
```

Finally, restricting access to data values for a specific purpose minimizes linkability by restricting access to the amount of contextual information one can gather from certain data values, making it harder to link data values to a specific individual.

9.5. BFCaaS

The last step of mitigation setup is to go through the BFC orchestrator and instantiate BFCaaS in compliance with r_{acc} set by the policy, and u_{acc} submitted by the EPI user. At this stage, $r_r(T_w)$ is calculated again, such that:

$$r_r(T_w) \approx 0.18, \tag{21}$$

where this indicates the remaining risk after deploying BRANE and formalizing the policy with eFlint.

To further minimize this risk probability, the orchestrator deploys two example services: anonymization and secure file transfer protocol $\in pm$ and $\in sm$, respectively. With these services, risk attributes like sensitivity decrease to L, and subsequently, this decreases correlation as well where the anonymization function replaces correlatable data values with randomized new values. This is also compliant with the acceptable requested utility. Moreover, the secure transfer protocol decreases the probability of accessibility relating to packet sniffing, MitM, and similar attacks. As a consequence of the traffic being encrypted over the network, it takes more complex tools to exploit (some skill attributes are lowered to L).

10. Conclusion and future work

Medical data sharing is essential for research advancement, but PbD principles are as important as ever, especially in dealing with medical data. Implementing this framework involves an initial setup that integrates BRANE for workflow execution and the BFC orchestrator for security management. Initially, an EPI/BRANE/eFLINT expert is needed for configuration and policy definition. However, with a user-friendly interface, predefined policy templates, and automated policy suggestions, non-technical project managers can handle routine configurations. This approach reduces dependency on technical experts, facilitating broader adoption and smoother integration into healthcare workflows. The EPI Framework facilitates data sharing while ensuring compliance with PbD principles. This framework effectively manages privacy risks, maintains control over data, and governs disclosure. It is constructed with the utilization of BRANE, a workflow execution engine, and the BFC orchestrator, a BFCaaS network orchestrator. We introduce a privacy risk model to quantify the level of privacy risk associated with data sharing. We evaluate the mitigated risk in reference to data utility and the predefined acceptable risk as per relevant policies. The EPI framework, despite its robust privacy-preserving capabilities, has several limitations. The initial setup and integration are resource-intensive and require technical expertise. Developing new anonymization functions for specialized file formats is challenging and poses data exposure risks during development and debugging. Ensuring comprehensive regulatory compliance across diverse regions and maintaining the framework with continuous updates and security audits add to its complexity and resource demands. In our future research endeavours, we intend to explore additional security risks by employing risk models such as DREAD/STRIDE, evaluate the completeness of data-sharing policies, and the effect of policy qualities on risk.

CRediT authorship contribution statement

Jamila Alsayed Kassem: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Tim Müller:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Christopher A. Esterhuyse:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Milen G. Kebede:** Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Anwar Osseyran:** Writing – review & editing, Supervision. **Paola Grosso:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The EPI project is funded by the Dutch Science Foundation in the Commit2Data program (grant no: 628.011.028).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.future.2024.107550.

Data availability

No data was used for the research described in the article.

References

[1] T. Hulsen, Sharing is caring—Data sharing initiatives in healthcare, Int. J. Environ. Res. Public Health 17 (9) (2020) http://dx.doi.org/10.3390/ijerph17093046, URL https://www.mdpi.com/1660-4601/17/9/3046.

[2] A. Gkoulalas-Divanis, G. Loukides, Medical Data Privacy Handbook, Springer, 2015.

[3] IBM Security, Cost of a Data Breach Report 2019, Tech. Rep., IBM Corporation, 2019, URL https://www.ibm.com/downloads/cas/RDEQK07R. (Accessed 03 August 2024).

[4] K. Wuyts, W. Joosen, LINDDUN privacy threat modeling: a tutorial, CW Rep. (2015).

[5] A. Cavoukian, et al., Privacy by Design: The 7 Foundational Principles, vol. 5, Information and privacy commissioner of Ontario, Canada, 2009, p. 12.

[6] M. Langheinrich, Privacy by design—principles of privacy-aware ubiquitous systems, in: International Conference on Ubiquitous Computing, Springer, 2001, pp. 273–291.

[7] C. Perera, C. McCormick, A.K. Bandara, B.A. Price, B. Nuseibeh, Privacy-by-design framework for assessing internet of things applications and platforms, in: Proceedings of the 6th International Conference on the Internet of Things, IoT '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 83–92, http://dx.doi.org/10.1145/2991561.2991566.

[8] A. Cavoukian, A. Fisher, S. Killen, D.A. Hoffman, Remote home health care technologies: How to ensure privacy? Build it in: Privacy by design, Identity Inf. Soc. 3 (2010) 363–378.

[9] F.H. Semantha, S. Azam, K.C. Yeo, B. Shanmugam, A systematic literature review on privacy by design in the healthcare sector, Electronics 9 (3) (2020) 452.

[10] D. Eke, I.E. Aasebø, S. Akintoye, W. Knight, A. Karakasidis, E. Mikulan, P. Ochang, G. Ogoh, R. Oostenveld, A. Pigorini, B.C. Stahl, T. White, L. Zehl, Pseudonymisation of neuroimages and data protection: Increasing access to data while retaining scientific utility, Neuroimage: Reports 1 (4) (2021) 100053, http://dx.doi.org/10.1016/j.ynirp.2021.100053, URL https://www.sciencedirect.com/science/article/pii/S2666956021000519.

[11] L.E. Veen, S. Shakeri, P. Grosso, Mahiru: a federated, policy-driven data processing and exchange system, 2022, arXiv preprint arXiv:2210.17155.

[12] General Data Protection Regulation (GDPR): Recital 35 health data, 2019, URL https://gdpr-info.eu/recitals/no-35/.

[13] Cisco, What is SFTP? Secure file transfer protocol explained, 2023, URL https://www.cisco.com/c/en/us/products/collateral/security/what-is-sftp.html. (Accessed 03 August 2024-03).

[14] STRIDE/DREAD, The DREAD approach to threat assessment, 2020, URL https://learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers.

[15] L. Zhang, A. Taal, R. Cushing, C. Laat, P. Grosso, A risk-level assessment system based on the STRIDE/DREAD model for digital data marketplaces, Int. J. Inf. Secur. 21 (2022) http://dx.doi.org/10.1007/s10207-021-00566-3.

[16] L.T. van Binsbergen, L.-C. Liu, R. van Doesburg, T. van Engers, EFLINT: A domain-specific language for executable norm specifications, in: Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, in: GPCE 2020, Association for Computing Machinery, New York, NY, USA, 2020, pp. 124–136, http://dx.doi.org/10.1145/3425898.3426958.

[17] O. Valkering, R. Cushing, A. Belloum, Brane: A framework for programmable orchestration of multi-site applications, in: 17th IEEE International Conference on EScience, EScience 2021, Innsbruck, Austria, September 20-23, 2021, IEEE, 2021, pp. 277–282, http://dx.doi.org/10.1109/eScience51609.2021.00056.

[18] C.A. Esterhuyse, T. Müller, L.T. van Binsbergen, A.S.Z. Belloum, Exploring the enforcement of private, dynamic policies on medical workflow execution, in: 18th IEEE International Conference on E-Science, E-Science 2022, Salt Lake City, UT, USA, October 11-14, 2022, IEEE, 2022, pp. 481–486, http://dx.doi.org/10.1109/eScience55777.2022.00086.

[19] Epictetus, T.W. Higginson, The Enchiridion, Bobbs-Merrill, 1955.

[20] European Commission, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016, URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[21] F.W. Vaandrager, B. Garhewal, J. Rot, T. Wißmann, A new approach for active automata learning based on apartness, in: D. Fisman, G. Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held As Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I, in: Lecture Notes in Computer Science, vol. 13243, Springer, 2022, pp. 223–243, http://dx.doi.org/10.1007/978-3-030-99524-9_12.

[22] N. Li, Delegation Logic: A Logic-based Approach to Distrbuted Authorization (Ph.D. thesis), New York University, USA, 2000, URL https://cs.nyu.edu/media/publications/li_ninghui.pdf.

[23] J.A. Kassem, C. De Laat, A. Taal, P. Grosso, The EPI framework: A dynamic data sharing framework for healthcare use cases, IEEE Access 8 (2020) 179909–179920, http://dx.doi.org/10.1109/ACCESS.2020.3028051.

[24] J.A. Kassem, O. Valkering, A. Belloum, P. Grosso, EPI framework: Approach for traffic redirection through containerised network functions, in: 2021 IEEE 17th International Conference on EScience, EScience, 2021, pp. 80–89, http://dx.doi.org/10.1109/eScience51609.2021.00018.

[25] J.A. Kassem, L. Zhong, A. Taal, P. Grosso, Adaptive services function chain orchestration for digital health twin use cases: Heuristic-boosted Q-learning approach, 2023, arXiv:2304.12853.

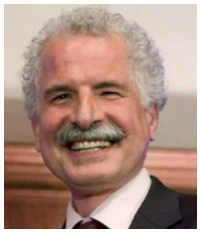
[26] J. Baugh, U. Bartels, J. Leach, B. Jones, B. Chaney, K.E. Warren, J. Kirkendall, R. Doughman, C. Hawkins, L. Miles, et al., The international diffuse intrinsic pontine glioma registry: an infrastructure to accelerate collaborative research for an orphan disease, J. Neuro-Oncol. 132 (2) (2017) 323–331.

[27] L.T. van Binsbergen, M.G. Kebede, J. Baugh, T. Van Engers, D.G. van Vuurden, Dynamic generation of access control policies from social policies, Procedia Comput. Sci. 198 (2022) 140–147.

Jamila Alsayed Kassem received her BEng in computer engineering in 2017 from Beirut Arab University, Debbieh campus. She, then, received her M.Sc. in information and network security in 2018 from the University of the West of Scotland, United Kingdom. She is currently working towards her Ph.D. in MultiScale Networked Systems, the University of Amsterdam. Her research interests include novel network infrastructure, programmable infrastructure, health data sharing, and information security.

Tim Müller is a scientific programmer at the University of Amsterdam focussing on data sharing, infrastructures for federated machine learning, normative reasoning, and cloud-based systems. Previously, he was employed for the EPI project to build a workflow execution engine that securely leverages data hosted by various medical institutions. Now he is working on the upcoming DMI project to generalize the work in the context of AMdEX.

Christopher A. Esterhuyse is a Ph.D. student at the University of Amsterdam, studying various formal methods in application to the development and control of various federated systems. These methods include logic programming, coordination languages, and program refinement. These systems include data exchange systems, where consortia of autonomous entities support their cooperation by creating, communicating, and enforcing formal consortium agreements.



Anwar Osseyran is since 2014 professor of Business Analytics and Computer Science at the Business School of the University of Amsterdam. His research focuses on smart energy and smart health systems using High-Performance Computing, Big Data and Artificial Intelligence to improve technology and business models. Professor Osseyran was a member of the Executive Board of the Dutch National SURF Foundation (2015-2020), CEO of the national supercomputing centre (2001-2020) and CEO of Vancis BV (2008-2012), an HPC spin-off of SARA.

Paola Grosso is a Full Professor at the University of Amsterdam where she leads the Multiscale Networked Systems research group (mns-research.nl). Her work focuses on the creation of sustainable and secure e-infrastructures and relies on the provisioning and design of programmable networks. She has an extensive list of publications on the topic and contributes to several national and international projects.