International Journal of Information Security (2024) 23:3591–3608
https://doi.org/10.1007/s10207-024-00907-y

REGULAR CONTRIBUTION

Guiding the implementation of data privacy with microservices

Pedro Antunes[1] · Nuno Guimarães[2]

Published online: 23 August 2024

Abstract
Privacy by design is nowadays recognized as essential in bringing data privacy into software systems. However, developers still face many challenges in reconciling privacy and software requirements and implementing privacy protections in software systems. One emerging trend is the adoption of microservices architectures—they bring in some qualities that can benefit privacy by design. The main goal of this study is to adapt privacy by design to the qualities brought by microservices. The main focus is at the architectural level, where the main structural decisions are made. A systematic literature review is adopted to identify a set of privacy models that underscore significant differences in software systems' protection using microservices. From the literature review, a decision framework is developed. The decision framework provides guidance and supports design decisions in implementing data privacy using microservices. The framework helps select and integrate different privacy models. An illustration of using the framework, which considers the design of an electronic voting system, is provided. This study contributes to closing the gap between regulation and implementation through design, where decisions related to data privacy are integrated with decisions on architecting systems using microservices.

Keywords Privacy by design · Microservices · Data privacy implementation · Decision framework

1 Introduction

Privacy by design is a conceptual principle that regards data privacy as critical in software development [1, 2]. The central assumption is that privacy by design is an effective way to integrate data privacy principles and regulations (e.g., GDPR [3] or CCPA [4]) into a system's design, development, and related processes, such as risk management [5]. Nevertheless, even though developers acknowledge the relevance of data privacy and privacy by design, they find the concepts difficult to translate into their practices [1]. This happens for various reasons, including lack of familiarity with privacy principles, primary focus on functionality, conflicts between privacy and functionality, and lack of guidance. This suggests that privacy by design has not yet fully fulfilled its primary purpose as the essential factor of closing the gap between regulation and implementation.

Generally speaking, design concerns the cyclic process from identifying goals and constraints to abstract problem definition, abstract solution development, and validation in concrete settings [6]. This cycle is repeated until the solution is considered satisfactory [7]. To implement data privacy, developers must integrate privacy goals and constraints into the software design process. From a socio-technical perspective, it has been suggested that enactments of privacy by design should become more material, using standard artifacts and templates [8]. This allows developers to increase confidence in their design decisions. The current research responds to this identified need with a decision framework that provides guidance and supports design decisions in implementing data privacy using microservices.

Microservices architectures have significantly changed the software design process [9, 10]. Microservices are implemented and operated as small, independent systems that give access to functionality and data through a well-defined interface. Using microservices, developers can break down complex systems into small, loosely coupled, and conceptually simple parts [11]. Each part is independently developed

✉ Pedro Antunes
padantunes@fc.ul.pt

Nuno Guimarães
Nuno.Guimaraes@iscte-iul.pt

[1] LASIGE, Faculty of Sciences, University of Lisbon, Campo Grande, Lisbon 1749-016, Portugal

[2] ISCTE - University Institute of Lisbon, Av. das Forças Armadas, Lisbon 1649-026, Portugal

and tested, usually using DevOps practices and tools, which accelerate development and support continuous deployment and redeployment [12]. Front-end (user-facing) and back-end (resources and processing) functionality can be broken down into microservices [13]. The result is a significant increase in development flexibility and agility, which explains why microservices are so popular nowadays [14, 15].

This study analyzes privacy by design through a microservices architectural lens. We consider three qualities brought by microservices that help build privacy by design: decentralization (of privacy protections), packaging (of functionality and data), and guarding (of perimeters, zones, and components). A decision framework is proposed that helps developers make critical design decisions regarding the implementation of data privacy at the architectural level.

In summary, the argument put forward by the present study articulates the following key points:

1. Setting: The primary purpose of privacy by design is bringing data privacy into software systems through design;
2. Problem: So far, privacy by design has been challenging to translate into implementation practices;
3. Viewpoint: Adapt privacy by design to the qualities brought by microservices at the architecture level;
4. Solution: A decision framework that provides guidance and supports design decisions in the implementation of data privacy using microservices;
5. Contribution: Close the gap between regulation and implementation through design.

The paper is organized in the following way. First, we provide background information regarding privacy regulation, privacy by design, and microservices. This is followed by a discussion of the adopted research approach based on a systematic literature review. Then, we synthesize the review results. A detailed discussion of the framework construction follows this. The framework is then illustrated using a case study related to electronic voting. Finally, we discuss the study contributions and provide some concluding remarks.

2 Background

2.1 Regulation

Privacy and data protection regulations are among the most fashionable types of law worldwide. EU [3], US [4, 16, 17], Canada [18], Brazil [19], China [20], and India's [21] laws are evidence enough that privacy and data protection in the digital space are universally established.

Independently of the nuances in each regulation's motivations, some originating from a personal rights perspective, some designed for consumer protection, there are common concepts. First, they share an often implicit assumption that data protection is necessary for assuring privacy. A discussion of the differences between privacy and data protection is out of the scope of this study [22, 23]. So, we will use the terminology 'data privacy.' Wherever data privacy is mentioned, we assume data protection is the key issue discussed.

Second, regulations define privacy principles. GDPR adopts the following key principles [24]:

- Lawfulness, fairness, and transparency.
- Purpose limitation.
- Data minimization.
- Accuracy.
- Storage limitation.
- Integrity and confidentiality (security).
- Accountability.

These principles limit how companies collect and process personally identifiable information (PII) [24], with direct implications on the relationships with customers and ways into which PII is collected, processed, stored, and shared. As companies seek to comply with regulations, they urge software developers to embed privacy requirements into software systems and digital services [8].

Privacy requirements have to be handled by software developers at various development stages from analysis, (e.g., requirements elicitation and analysis) to design (e.g., user journeys, software services, application interfaces, systems architectures, and information structures) and coding (e.g., algorithms, components and features) [25]. In this study, we focus specifically on the software design stage.

Third, regulations generally define the individuals' rights and relationships with the companies supplying digital services. These usually take a contractual nature in the form of "notice and consent" of the conditions stated by the supplying parties. The topic of consent is dense [26]. In this study, we limit ourselves to the primary requirements for user consent pondered at the design stage.

2.2 The gap between regulation and implementation

Prior research shows that software developers find it challenging to bring data privacy regulation into implementation [1, 5, 25, 27–29]. Identified reasons include the lack of familiarity with privacy principles, low priority given to privacy issues, being primarily focused on functionality, negative impacts of privacy on functionality, difficulties encoding privacy into software, and lack of guidance. This underlines a significant gap between more abstract or high-level principles, guidelines, protocols, and legal frameworks, and more concrete or low-level methods and technologies developers use to build software systems [30].

Furthermore, as responsibility for privacy permeates from the broader organization towards the individual software developer, there is an increasing understanding that more safeguarded, techno-regulatory, and collective approaches toward privacy implementation are necessary to increase the developers' confidence in their own decisions [8]. In this study, we take an architectural view of this techno-regulatory problem, considering different privacy models at the design/architecture level.

2.3 Privacy by design

The concept of privacy by design has emerged as a foundation for closing the gap between regulation and implementation [31, 32]. It is based on the idea that data privacy should be the default setting of every software system instead of being an afterthought or a remediation. Design becomes the center stage for integrating privacy and business needs, technical requirements, problems, and solutions.

Privacy by design brings with it a new set of principles, which developers are expected to embrace [33]: being proactive and preventative, with privacy as the default setting, embedding privacy into the design, bringing privacy and legitimate business interests together, complete lifecycle protection, visibility and transparency, and respect for the user. Adopting these principles helps avoid privacy risks, comply with legal requirements, and engage with regulators, rights advocates, users, and other stakeholders [32, 34].

However, privacy by design also has the reputation of being challenging for developers [32]. This happens because the concept is aspirational. It is framed as an organizational rather than a technical goal [8]. It centers on attitudes and beliefs, lacking direct links to real-life development practices [27, 30, 32]. In particular, privacy by design is still distant from developers' decisions at the systems architecture level [31]. Microservices raise the opportunity to address the problem at this level.

2.4 Microservices

Microservices have gained immense popularity in the last few years [11, 35]. They are recognized to increase flexibility and agility in software development, particularly when combined with new practices such as DevOps [14, 15, 36]. Microservices are characterized by dividing application functionality into many small, decentralized, and autonomous components coping with particular duties [9]. Duties involve dealing with a particular function or representing a specific resource [37]. In DevOps environments, microservices can be independently developed, deployed, tested, and continuously modified [15]. The functionality provided by a microservice may concern an application's front-end or back-end. Several front-end and back-end microservices can work together to deliver micro-frontends, which can be regarded as semi-independent, reusable mini-applications [38].

Microservice architectures have several distinct features. In particular, the communication with (and between) microservices is standardized, using a lightweight request-response protocol [39]. Another distinctive feature is that microservices can be individually deployed and scaled on demand. Focusing on data, a relevant feature is decentralized data management [9]. Each microservice can be seen as a complete vertical solution, which provides its interface, service logic, and data storage [14]. Hence, to some extent, data privacy is the responsibility of individual microservices.

2.5 Qualities brought by microservices

Microservices architectures have some qualities that can benefit data privacy and privacy by design:

2.5.1 Decentralization

Microservices allow splitting functionality and data throughout fine-grained and loosely-coupled components, affording minimal centralized management, which can benefit data privacy [40, 41]. If a vulnerability is exploited, its reach could be limited to a few components [42]. Furthermore, each affected component is easier to analyze and review. Microservices also allow further splitting of functionality and data into multiple dimensions, applying concepts such as separation of concerns and single responsibility [37]. For instance, functionality can be separated into individual front-end and back-end microservices and between message routing and processing, thus separating user interaction from data processing and storage [40].

2.5.2 Packaging

Wrapping functionality and data within individual microservices can contribute to data privacy if microservices are attentive and protective [42]. Additionally, microservices can be placed into software containers to continuously monitor and control how microservices are deployed and used, detect intrusions, and analyze suspicious behavior [43, 44].

2.5.3 Guarding

Microservices can protect perimeters, zones, and even individual components [41, 45, 46]. Various data privacy features can be built into these microservices [47, 48].

In Fig. 1, we present a diagram that relates microservices qualities to privacy and privacy by design principles [24, 32]. The diagram was built by asking the question: can quality X positively impact the implementation of principle X? The exercise highlights the significant impact of microservices on data privacy.

2.6 Related work

Several studies have previously discussed how to close the gap between regulation and implementation in the scope of privacy by design and focusing particularly on architectural issues. We organize the related works into a set of categories and briefly discuss them.

2.6.1 Architectural principles and guidelines

The main concern in this category is translating privacy regulations into architectural principles and design guidelines. They address what to do with application data (e.g., hiding, separating, aggregating, and reducing granularity) and supporting processes (e.g., inform, control, and enforce) [2, 49, 50]. Ontologies and models have been proposed to organize these principles and guidelines [51–53]. However, principles and guidelines suffer from the "checklist approach," directing developers more toward compliance than supporting decision-making [54].

2.6.2 Formal methods

The main focus is applying formal methods in architectural analysis. This requires creating a representation of the targeted architecture and, through formal mechanisms, analyzing if it satisfies certain privacy properties, such as data minimization and integrity [55–59]. Tools have been proposed to support the analytic process [60] and facilitate critical discussions [61]. However, there are significant limitations. One is the restricted set of properties addressed by these methods [55]. Another is that software development flexibility and agility may rapidly outdate these representations, reducing value and accountability over time [58]. Finally, applying these methods involves a significant effort, which has to be weighed against benefits [59].

2.6.3 Architectural tactics

Studies in this category propose goal-oriented architectural tactics, or design patterns, that can be contextualized to specific privacy requirements [62, 63]. Example tactics include batching data, delaying messages [63], and adding a privacy layer between the application interfaces (APIs) and the data layer [48].

2.6.4 Procedural models

This category considers procedural approaches for integrating privacy requirements into system architectures [30, 34, 62, 64–66]. They focus on how to analyze privacy requirements, perform impact assessments, and select adequate architectural tactics [62]. As such, this category complements the previous one with decision guidelines.

The present research concerns the last two categories, focusing on tactics and decisions that specifically apply to microservices architectures. This is a unique proposition,

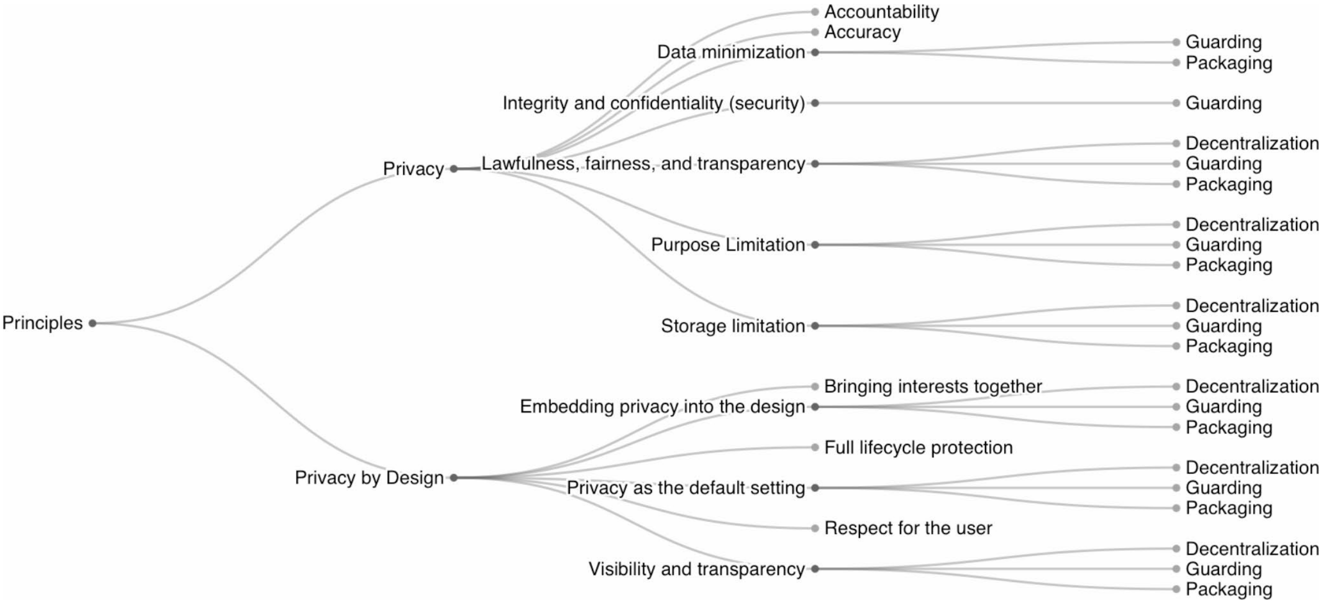

Principles
Privacy
Data minimization
Accountability
Accuracy
Integrity and confidentiality (security)
Lawfulness, fairness, and transparency
Purpose Limitation
Storage limitation
Privacy by Design
Embedding privacy into the design
Bringing interests together
Full lifecycle protection
Privacy as the default setting
Respect for the user
Visibility and transparency
Guarding
Packaging
Decentralization

Fig. 1 Diagram relating microservices qualities to data privacy and privacy by design principles

given that, as far as we could find, only one prior study has focused on microservices. The study by Mashaly et al. [48] discusses how to implement data privacy with microservices. However, it has significant limitations, proposing a one-size-fits-all privacy layer, thus overlooking the potential advantages of combining various architectural strategies. The present study addresses these limitations.

3 Research approach

A systematic literature review is the centerpiece of the selected research approach. Systematicity in literature reviews contributes to rigor in summarizing existing evidence and provides fairness and transparency regarding what is included and excluded from the review [67]. Consequently, systematicity also contributes to increasing the soundness of the design propositions extracted from the review. Next, we overview the research approach, followed by a detailed discussion of the review procedure.

3.1 Overview

The adopted research approach considers four steps (Fig. 2). The first step systematically searches microservices and data privacy literature to characterize the domain. The second step synthesizes the domain using a set of data privacy models. The third step is focused on decision guidance, for which a decision framework is developed. The fourth and final step aims to illustrate using the decision framework in a case study.

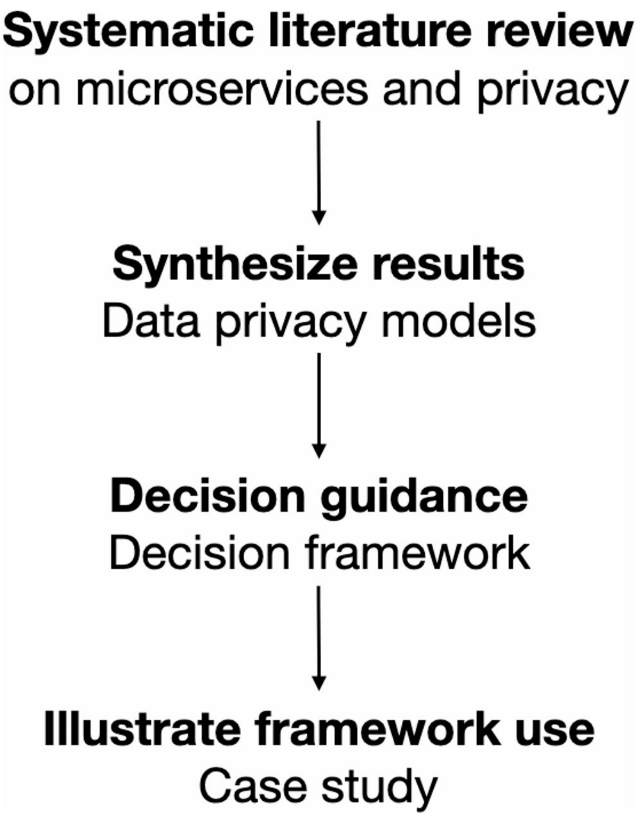


Fig. 2 Research approach

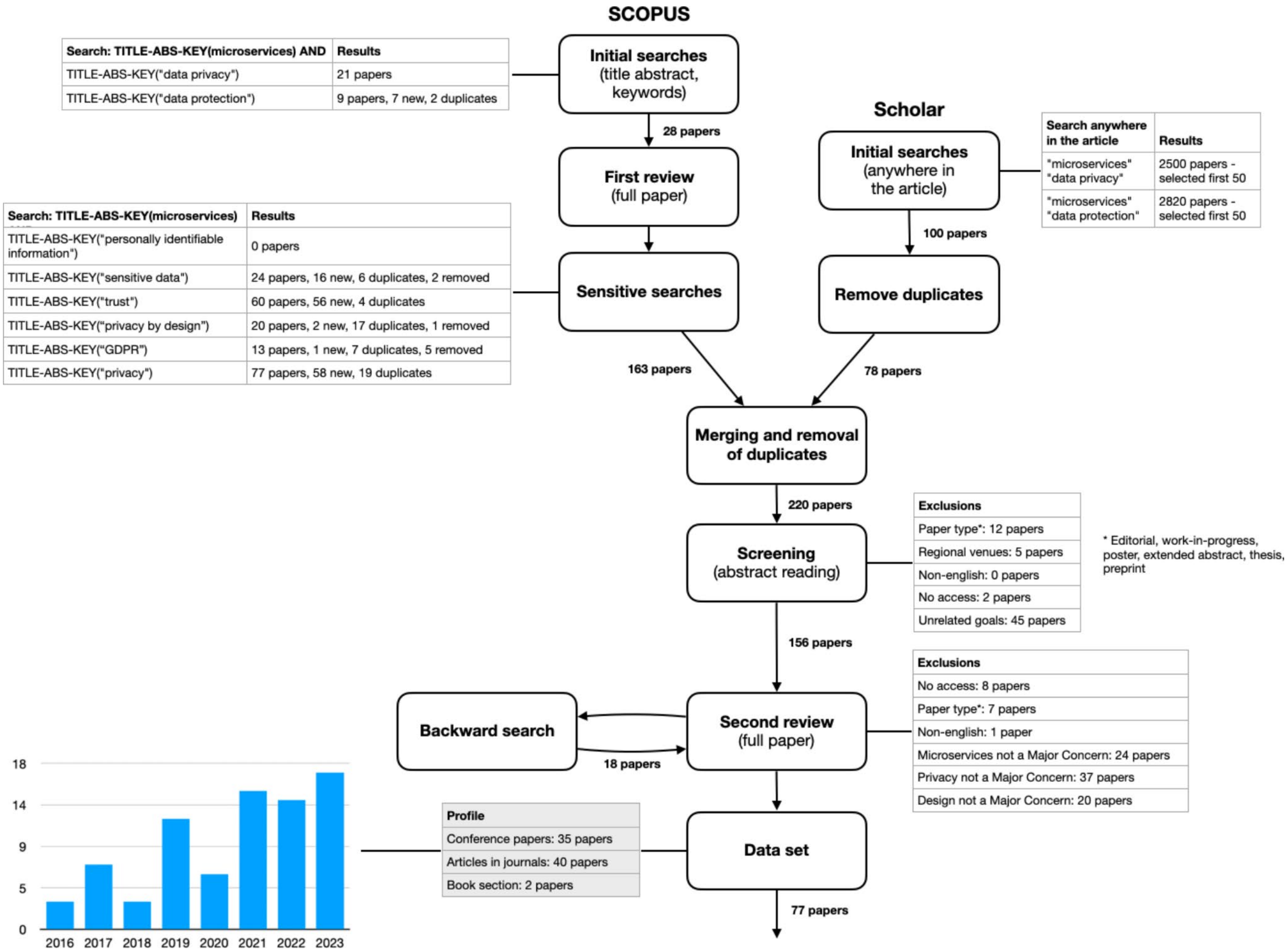
This research approach follows a familiar pattern adopted by other studies in the domain. For example, the study by Ataei and Staegemann [45] reviews the literature on the use of microservices in big-data systems, synthesizes a set of big-data requirements and design patterns, and provides guidance for the design of big-data systems. The study by Al-Doghman et al. [43] reviews the literature on the use of AI models in microservices, synthesizes a set of technical approaches from the literature, and builds a framework that provides design guidance on how to secure microservices.

3.2 Review procedure

An initial search was conducted in SCOPUS (Fig. 3, left). This database was selected because it offers comprehensive coverage in computer science and provides quality control over peer-reviewed literature. The initial searches used the strings [‘microservices’ and ‘data privacy’] and [‘microservices’ and ‘data protection’]. These searches considered title, abstract, and keywords. This initial search identified 28 papers.

The 28 papers were analyzed in detail to identify new search keywords and conduct sensitive searches on the topic [68], turning the search process more inclusive. The selected new keywords were: personal identifiable information, sensitive data, trust, privacy by design, and GDPR. Further, an additional search was conducted considering the string [‘microservices’ and ‘privacy’] to avoid false negatives caused by a restrictive use of ‘data privacy.’ Fig. 3 (left) summarizes the results from these sensitive searches, considering the number of items returned from each search, new and duplicate items, and removed items. All removed items were editorials in proceedings, as they did not have research content. The set of sensitive searches increased the dataset from 28 to 163 papers.

As searching one single database may be limiting, additional searches were conducted in Google Scholar (Fig. 3, right). These searches used the strings [‘microservices’ and ‘data privacy’] and [‘microservices’ and ‘data protection’]. However, they contemplated anywhere in the article. Considering this more expansive scope and the broader coverage of Scholar, the results were much higher (2.500 and 2.820 papers, respectively) than those from SCOPUS. A cutoff line was defined, and only the first 50 results from each search were selected (ordered by relevance). These choices reflect a compromise between practicality (manually processing more than 5.000 items would be unfeasible)


SCOPUS
Initial searches (title abstract, keywords)
28 papers
First review (full paper)
Sensitive searches
163 papers
Scholar
Initial searches (anywhere in the article)
100 papers
Remove duplicates
78 papers
Merging and removal of duplicates
220 papers
Screening (abstract reading)
156 papers
Second review (full paper)
Backward search
18 papers
Data set
77 papers
18
14
9
5
0
2016
2017
2018
2019
2020
2021
2022
2023

Search: TITLE-ABS-KEY(microservices) AND	Results
TITLE-ABS-KEY("data privacy")	21 papers
TITLE-ABS-KEY("data protection")	9 papers, 7 new, 2 duplicates

Search: TITLE-ABS-KEY(microservices)	Results
TITLE-ABS-KEY("personally identifiable information")	0 papers
TITLE-ABS-KEY("sensitive data")	24 papers, 16 new, 6 duplicates, 2 removed
TITLE-ABS-KEY("trust")	60 papers, 56 new, 4 duplicates
TITLE-ABS-KEY("privacy by design")	20 papers, 2 new, 17 duplicates, 1 removed
TITLE-ABS-KEY("GDPR")	13 papers, 1 new, 7 duplicates, 5 removed
TITLE-ABS-KEY("privacy")	77 papers, 58 new, 19 duplicates

Search anywhere in the article	Results
"microservices" "data privacy"	2500 papers - selected first 50
"microservices" "data protection"	2820 papers - selected first 50

Exclusions
Paper type*: 12 papers
Regional venues: 5 papers
Non-english: 0 papers
No access: 2 papers
Unrelated goals: 45 papers

* Editorial, work-in-progress, poster, extended abstract, thesis, preprint

Exclusions
No access: 8 papers
Paper type*: 7 papers
Non-english: 1 paper
Microservices not a Major Concern: 24 papers
Privacy not a Major Concern: 37 papers
Design not a Major Concern: 20 papers

Profile
Conference papers: 35 papers
Articles in journals: 40 papers
Book section: 2 papers

Fig. 3 Review procedure

and the chances of finding relevant papers missing from the SCOPUS results.

The two Scholar searches produced several duplicate items, which were removed to identify 78 unique papers. The 78 papers from the Scholar dataset were merged with the 163 papers from the SCOPUS dataset. The identified duplicates were removed, resulting in a combined dataset with 220 papers. To avoid dealing with too many false positives, no sensitive searches were done in Scholar.

This procedure was supported by the Zotero reference management tool. The search results from SCOPUS were exported in the RIS format and imported into Zotero. The search results from Scholar were imported into Zotero using a browser plugin provided by Zotero.

The 220 papers were screened using a set of exclusion criteria. Only conference papers and journal articles were considered (excluding editorials, work-in-progress, posters, extended abstracts, theses, and non-peer-reviewed preprints). Papers in regional/local conferences and non-English papers were also excluded. Two references to unreachable abstracts were removed from the dataset. Finally, after analyzing the abstracts imported into Zotero, 45 papers were excluded for being unrelated to this research. Papers were excluded if they explicitly did not address the design of systems. For example, papers focused on education, visualization, hardware security, wireless networks, workflow management, high-performance computing, and cryptographic algorithms were excluded. The resulting dataset contained 156 papers.

The 156 papers were reviewed in more detail, considering the full text. Once again, a set of exclusion criteria was applied. Unreachable papers (8 items), papers of inadequate types (7 items, see Fig. 3 for details), and papers not in English (1 item) were removed. Furthermore, three additional exclusion criteria were considered:

- Papers where microservices were not a significant concern. This included papers where microservices were adopted for implementation but not conceptualization (e.g., intelligent agents and edge resources) and cases where microservices were regarded as subsidiary to another concept (e.g., Internet-of-Things and containerization).

Table 1 Identified data privacy models

Models	Nr. of references
Gatekeeper model	24
Service model	13
Enclave model	9
Endpoint model	5
Mesh model	4
Federation model	3
Sidecar model	2

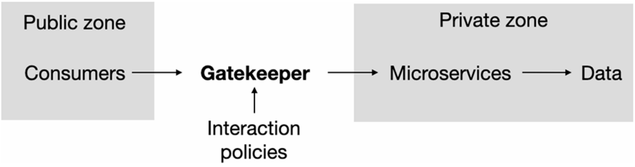


Fig. 4 Gatekeeper model

- Papers where privacy was not a significant concern. For example, cases where the main focus was security, cloud computing, and AI.
- Papers where design was not a significant concern. This included, in particular, research focused on algorithms, protocols, and computing platforms with a significant focus on implementation, testing, and performance.

While doing the second review, particular attention was committed to doing backward searches, where the references cited by papers in the dataset were checked for relevance [69]. Through this procedure, 18 additional papers were added to the dataset and reviewed in detail.

The resultant dataset comprised 77 papers, relatively equally divided between conference papers and journal articles (35 and 40 papers, respectively), plus two book chapters (Fig. 3, bottom-left). The distribution according to publication date (Fig. 3, bottom-left) indicates that the interest in the topic addressed by this study is recent (starting in 2016) and has been increasing. Next, we synthesize the results.

4 Synthesis from the literature

We identified seven data privacy models in the searched literature, listed in Table 1 according to their relative standing in the analyzed dataset (Annex A links the identified models to specific dataset items). Next, we characterize the distinctive features of the identified models.

4.1 Gatekeeper model

This model defines private and public zones separated by a gatekeeper (Fig. 4). If a consumer in the public zone requires data or functionality from the private zone, it has to go through the gatekeeper. The 'gatekeeper' term reflects a generic purpose. Depending on the specific functionality, gatekeepers are often designated as gateways and other times as proxies, protection layers/components, and data filters. The gatekeeper can be a microservice, a collection of microservices, or a software layer/component.

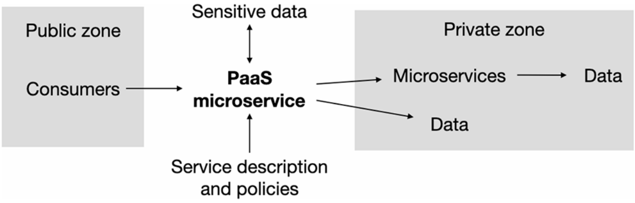


Fig. 5 Service model

Gatekeepers manage policies related to the interactions between public and private zones. For instance, Bugshan et al. [47] developed a gatekeeper that perturbs health-related data gathered by smart applications in the public zone from Internet-of-Things devices in the private zone.

4.2 Service model

The service model adopts a paradigm known as X-as-a-service, where a dedicated service provider implements functionality X. Privacy-as-a-service (PaaS) microservices oversee the exchanges between consumers in public zones and microservices and data in private zones (Fig. 5). PaaS microservices manage service descriptions and policies. They also manage local data stores with sensitive data in case the private zone is not trusted or data has to be filtered or perturbed [47].

Alic et al. [70] developed two PaaS microservices providing authentication-as-a-service and privacy-as-a-service. The former deals with authentication requests such as sign-in/out, token verification, password management, and authorizations. The latter protects sensitive data by applying anonymization policies to data exchanges, integrations, and analytics.

4.3 Enclave model

The enclave model relies on trusted software enclaves (e.g., Docker Containers) to isolate and secure the boundary of specific microservices (Fig. 6). Enclaves black-box the internal activities of isolated microservices. Enclaves rely on event-based policies to protect microservices. Enclaves can be deployed on untrusted software, e.g., cloud storage and software developed by third parties.

Fetzer et al. [71] developed two enclaves, one that forces microservices to use a secure message bus for communication

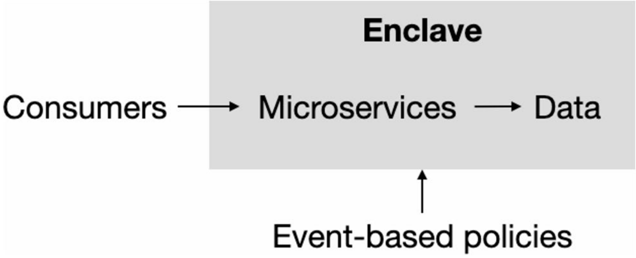


Fig. 6 Enclave model

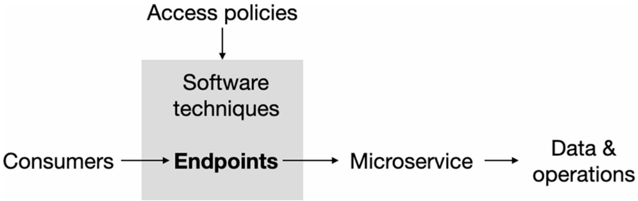


Fig. 7 Endpoint model

(with message encryption) and another that secures the storage of sensitive data in untrusted environments.

4.4 Endpoint model

This model is focused on the endpoints provided by microservices to consumers (Fig. 7). Microservices have APIs with multiple endpoints. Each endpoint interacts with the microservice's internal resources through CRUD (create, read, update, and delete) operations. Since CRUD operations can expose sensitive data, access to specific endpoints may have to be protected using software techniques. These software techniques grant or deny endpoint access based on defined access policies. Policies can use attributes related to the types of data, operations, use cases, and consumers [72].

Preuvenners and Joosen [73] developed a feature-toggle approach that restricts or allows endpoint access based on policies and dependencies that can be dynamically determined. The dependencies between consumers, requested resources, and operations are checked before access to the endpoint is granted.

4.5 Mesh model

The mesh model fully embraces the idea that microservices should be fine-grained and loosely coupled, where each microservice performs a small, clearly defined, and manageable function [43]. According to this model, data privacy is implemented by a mesh of privacy-preserving microservices standing between application microservices operating as requestors and responders (Fig. 8).

As privacy-preserving microservices stand between application microservices, they can protect specific interconnections and operational workflows. Furthermore, given that privacy-preserving microservices are single-purpose, they can protect specific functions or data.

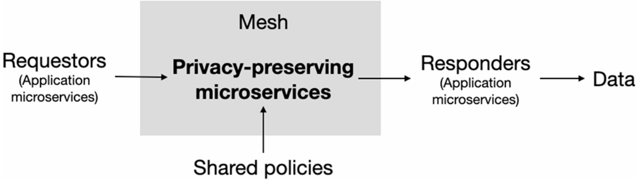

Mesh
Requestors
(Application microservices)
Privacy-preserving microservices
Responders
(Application microservices)
Data
Shared policies

Fig. 8 Mesh model

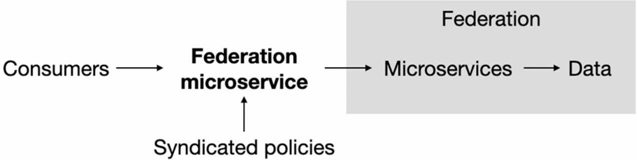


Fig. 9 Federation model

Privacy-preserving microservices share policies. Policies can contemplate a variety of attributes, including contextual information such as reasons for requesting data, allowed usage, and circumstances [74]. For instance, a particular case could be the location of an application microservice, either locally, in a server, or in the cloud.

Alanezi and Mishra [75] demonstrate the use of this privacy model in Internet-of-Things applications, where a set of system microservices (privacy-preserving) administrate the interactions between application microservices and sensor and actuator microservices. The system microservices may require information about the currently running application microservices to determine what policies to apply.

4.6 Federation model

The federation model adopts the concepts of federation and syndication, where a set of independent entities are integrated, represented, and controlled by a single entity [76]. Federated microservices communicate and share information freely to accomplish their mutually agreed goals [77]. However, their interactions with the outside world depend on the federation microservice, which syndicates the applied policies (Fig. 9).

Bhargava et al. [78] developed a system adopting the federation model. The system implements a federation microservice (central monitor) that manages a set of microservices according to domain characteristics, such as running on a mobile device. The central monitor intercepts service requests to federated microservices and performs other unified roles, like anomaly detection.

4.7 Sidecar model

The sidecar model has similarities with the enclave model, as both seek to protect single microservices. However, while the enclave model wraps around microservices through a secure software component, the sidecar model puts that component alongside the microservice.

Sidecars protect microservices from the inside rather than the outside (Fig. 10). Consumers send their requests to the microservice, which communicates with the sidecar for policy checking. By operating alongside the microservice, the sidecar can closely interact with the microservice, therefore applying highly localized and contextualized policies, which can consider the consumers' requests, the requested operations, and the sensitivity of operations and data. Sidecars operate independently alongside microservices; they can assume that microservices are not trustworthy [79].

Meadows et al. [79] developed a system that uses two types of sidecars. One type checks the incoming requests (ingress sidecar), while the other type (egress sidecar) checks the post-processing activities of a microservice.

4.8 Discussion

The literature review highlights the wide variety of data privacy models brought by microservices. This has been made possible by decentralization (of privacy protections), packaging (of functionality and data), and guarding (of perimeters, zones, and components). Together, they give developers plenty of options. More so if we consider that developers can combine multiple models in their implementations, this allows developers to implement hybrid solutions. Since microservices operate independently and use common communication standards, they do not pose constraints to hybridization [80]. The "one size fits all" adage does not apply in this area.

Even though developers have many options, a comprehensive decision framework helping developers adopt microservices for privacy protection is missing. Next, we develop such a framework.

5 Decision framework for the implementation of data privacy

The decision framework is derived from the literature review using a set of logical considerations detailed below.

5.1 The granularity of privacy protections

According to the Cambridge Dictionary, granularity is the quality of including many small details. The set of privacy models synthesized from the literature review shows that data privacy can be defined and implemented at different granularity levels. For instance, the gatekeeper model has low granularity because it uniformly protects an undefined, potentially sizable number of microservices. As such, no details related to individual microservices are considered. On the other hand, the endpoint model has high granularity because it targets the attributes of specific microservices, endpoints, and consumers. Therefore, granularity is crucial when deciding what privacy approaches to adopt for a system.

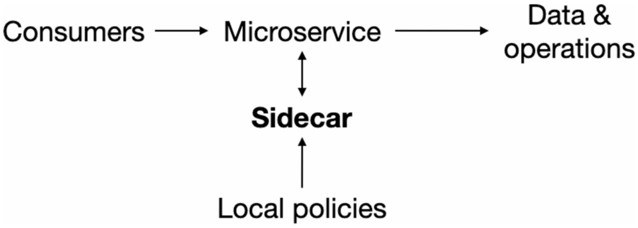


Fig. 10 Sidecar model

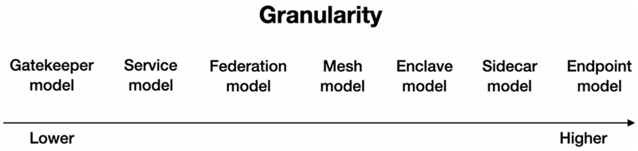


Fig. 11 Granularity dimension

In Fig. 11, we align the identified models according to granularity. The gatekeeper model is regarded as the least granular because protections are defined to manage a sizable number of private microservices uniformly. Next, we find the service model. It is more granular because each PaaS microservice can be specified to address more specific details of managed microservices. The federation model comes next. Its granularity increases because the federation microservice targets a small set of microservices, albeit not considering the details of what happens inside the federation.

The mesh model appears in the middle of the scale. Its granularity increases because the mesh targets specific microservices and their interactions. The enclave model follows this. It has higher granularity because it targets the external details of individual microservices. The sidecar model has increased granularity because it targets individual microservices' external and internal details. Finally, we have the endpoint model, which has the most granularity. This occurs because it targets individual microservices, specific endpoints, and external and internal details.

5.2 Versatility of privacy protection

According to the Cambridge Dictionary, versatility is the ability to change easily or to be used for different purposes.

We can also regard the synthesized privacy models from this perspective. To do this, we contrast comparable models (Fig. 12). The gatekeeper model is similar to the service model, as both are intended to manage a large number of microservices. However, the service model offers more versatility, as it can break down data privacy into independent services. That is, there is more separation of concerns.

The federation and mesh models are comparable approaches that seek to manage fewer microservices. However, the mesh model is more versatile than the federation model because a single entry point does not constrain it and allows mesh reconfigurations without impacting the privacy-preserving or application microservices.

The enclave and sidecar models have similarities, which can be compared in terms of versatility. Both rely on a software component that protects individual microservices. However, the sidecar model white-boxes those protections, while the enclave model black-boxes them. The white-box approach is more versatile, as it provides more options.

Finally, we do not compare the endpoint model to other models since the focus on endpoints is an exclusive characteristic of this model.

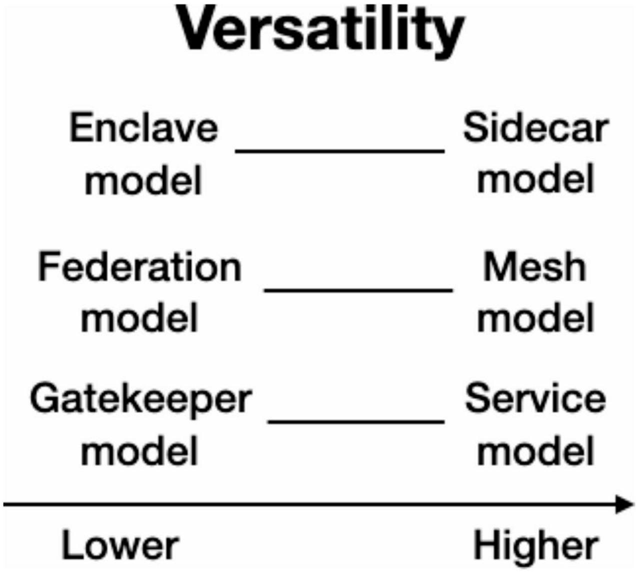


Fig. 12 Versatility dimension

5.3 Putting together the granularity and versatility dimensions

We now bring together the two discussed dimensions (Table 2). We categorize the identified privacy models using three granularity categories: low, medium, and high. Using categories rather than a continuous classification of models (as in Fig. 10) simplifies the decision-making process. Fine-grained decisions do not have to be made about the granularity dimension. They can be made considering the combination of granularity and versatility dimensions.

We consider the two dimensions of versatility already discussed, high and low. This arrangement highlights that gatekeepers and PaaS microservices provide low granularity. The choice for one or the other depends on versatility: implementations requiring more versatility should adopt PaaS microservices; if requiring less, then gatekeepers should be adopted.

Overall, the arrangement of models in these dimensions highlights that to implement data privacy, developers should make two sequential decisions: (1) decide what level of granularity is required; and (2) decide what level of versatility is desired. With these two decisions, the most adequate privacy model to implement is identified.

Table 2 Arranging privacy models according to granularity and versatility dimensions

		Granularity			
		Low	Medium	High	
Versatility	Low	Gatekeepers	Federation microservices	Enclaves	Endpoints
	High	PaaS microservices	Privacy-preserving mesh	Sidecars	

5.4 Implementing multiple defensive lines

There is no particular reason for implementing one single privacy model. Various models can be integrated to reinforce data privacy. For instance, the gatekeeper model can be adopted to define a private zone, and then, within that zone, multiple federation microservices can be used to protect specific groups of microservices. Indeed, the adoption of multiple privacy models increases data privacy. The reason is supported by the widely acknowledged Swiss Cheese Model (SCM) developed in the safety domain [81]. SCM posits that systems with fewer barriers are more vulnerable than systems with more. SCM explains metaphorically that accidents happen when unsafe events find a trajectory through a series of holes in these barriers (cheese slices). Similarly, systems with fewer privacy defensive lines are more vulnerable than systems with more defensive lines [82, 83].

From Table 3, three defensive lines emerge. The first line operates at low granularity, for which gatekeepers or PaaS microservices can be used. The second defensive line operates at medium granularity, based on federation microservices or privacy-preserving meshes. Finally, the third defensive line operates at high granularity, based on enclaves, sidecars, or endpoints. Table 3, in combination with the models synthesized from the literature review,

Table 3 Template for using privacy models as "nutrition labels"

Managed data	Granularity		
	Low	Medium	High
A			
B			
…			

Table 4 Template for defining privacy policies

Model	Entity	User consent policies
X	A	
Y	B	
…	…	

provides the conceptual infrastructure for integrating multiple defensive lines.

5.5 Considering transparency in the implementation of privacy protections

Transparency is a widely accepted principle of data privacy. It gives individuals the right to understand how their data is managed [1, 30, 84]. The categorization of privacy models in Table 2 provides a foundation for communicating critical privacy by design features in a simple way. In particular, Table 2 can be used to convey how data is handled by a system at the architectural level using the designations adopted for the different privacy models as "nutrition labels" [84].

Table 3 provides such a template. It identifies data sets relevant to the users (which can differ from how they are structured using microservices) and the privacy models adopted to protect them. The granularity categories are presented to the users, indicating the level of detail adopted by privacy protections. Furthermore, having multiple categories also helps communicate whether various defensive lines have been used. The versatility dimension is not considered, as it concerns an internal design decision.

5.6 User consent and policy management

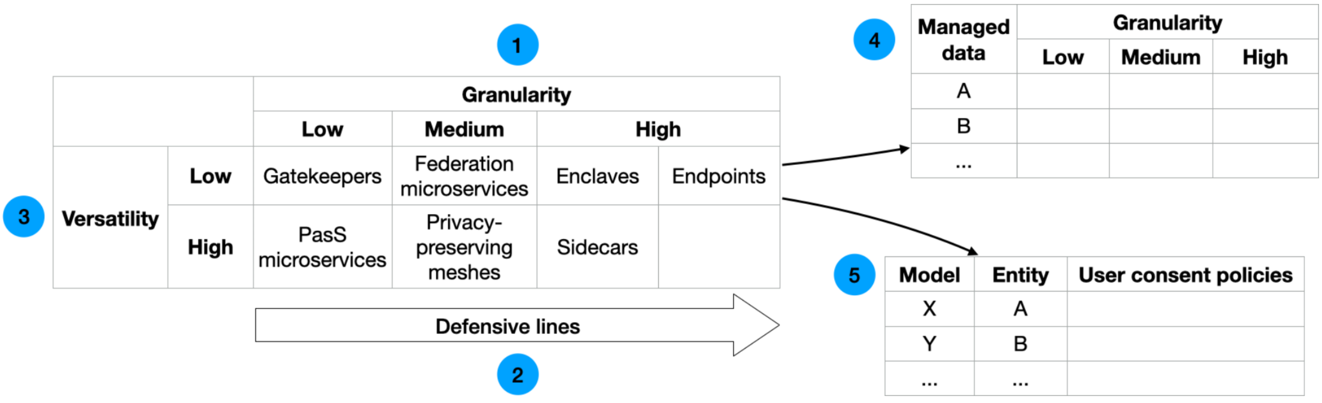
Consent is another widely accepted principle of data privacy. It provides individuals with a right to manage privacy [84]. Policies are the centerpiece for defining user consent. They are applied by different components, as specified by the privacy models (Figs. 4, 5, 6, 7, 8, 9 and 10). They also address a variety of attributes, which depend on the model's granularity. Policies can be applied at a general level, as defined by the gatekeeper and service models. They can also be used at a collective level, as specified by the federation and mesh models. Finally, they can be applied at a component level, as in the enclave, endpoint, and sidecar models. The template shown in Table 4 can be used to delineate how users can intervene at these different levels to provide consent and manage policies.

5.7 Decision framework

Finally, we synthesize a decision framework integrating the abovementioned elements. The framework defines several decision steps (Fig. 13).

First, developers must make a critical decision about the granularity of privacy protection. Low granularity is easier to manage; however, failures in low-granularity elements may lead to significant privacy breaches. On the other hand, high-granularity elements afford more contextualized and contained protections at the cost of additional complexity. The second decision concerns the number of defensive lines to implement. Having more defensive lines decreases the overall risk of the system at the cost of additional complexity.

The third decision concerns versatility. Developers must decide whether to implement high or low-versatile privacy models for each selected granularity level. This depends on the expected frequency of changes to the system. Even though microservices architectures afford frequent changes, an associated privacy protection cost must be considered. Continuous changes in the system require continuous



		Granularity			
		Low	Medium	High	
Versatility	Low	Gatekeepers	Federation microservices	Enclaves	Endpoints
	High	PasS microservices	Privacy-preserving meshes	Sidecars	

Managed data	Granularity		
	Low	Medium	High
A			
B			
…			

Model	Entity	User consent policies
X	A	
Y	B	
…	…	

Fig. 13 Decision framework

privacy assessment and adaptation. Therefore, a decision has to be made on whether it is better to adopt more versatile privacy models immediately or later. Since microservices are decentralized and autonomous, changes can be made continuously. Therefore, selecting a particular privacy model to protect specific data or functionality should not be considered fixed. An increased frequency of changes may lead to a shift from low to high versatile models.

After selecting the privacy models, developers must consider how to communicate their choices to the users. This involves identifying the managed data sets and selected privacy models, including indicating adopted granularity levels and defensive lines. Finally, the consent and user-defined policies must be specified for entities providing privacy protection. According to the defined privacy models, entities include individual microservices, collections of microservices, or software components and techniques.

6 Example

We illustrate the use of the framework with an example. The example considers an electronic voting system (Fig. 14). The system comprehends several application microservices divided between front-end and back-end. Each microservice is responsible for specific data, including the electoral registry, list of candidates, choices, ballots, and voting results.

The voting process starts with voter registration. The registration page interacts with the voter while the registrar checks whether the voter can participate in the poll and records whether the voter has recorded a choice. Then, the process proceeds with the actual voting. The poll manager delivers the list of candidates (e.g., local, regional, and national) to the voting page. The voting page displays the list and allows the voter to choose. The recorder saves the choice while the controller adds the choice to the ballot box. The counter tallies the votes. Finally, the reporter delivers the results.

Figure 14 only shows one instance of each microservice. However, the system is expected to deploy multiple instances. For example, multiple registration and voting pages are needed for parallel voting; the controller and counter could be instantiated locally or nationally to increase performance and resilience.

Regarding this system, we consider the following data privacy requirements:

- R1: The integrity of the vote must be assured, and the voting process should be resilient.
- R2: Only registered voters can vote; the voter is irreplaceable and can only vote once.
- R3: The vote must be kept secret, i.e., the link between registry and choice must be kept anonymous.
- R4: Any direct or indirect links (e.g., time stamps) between the registrar, recorder, and controller should be removed to avoid breaking anonymity.
- R5: The voting page should not display the voter's final choice (to avoid coercing users to take photos).

Developers must make several design decisions regarding these requirements. The first decision concerns granularity. Given the societal sensitivity associated with electronic voting, granularity has to be high. To assure the integrity of the vote (R1), all application microservices can be protected using enclaves like Docker Containers. Enclaves can provide general access control tailored to the voting process: the registrar can only communicate with the registration and voting pages; the poll manager can only communicate with the voting page and recorder; and so on.

Another decision concerns the adoption or not of multiple defensive lines. Given the risks associated with electronic voting systems, implementing all three defensive lines considering low, medium, and high granularity is a sensible choice. The system can implement a gatekeeper, which only gives access to users with unique, valid, and single-use access keys to the election (R2). This provides a first line of defense for the voting process.

A second defensive line may be implemented using federation services and privacy-preserving meshes. A federation service can hide the real information displayed by the

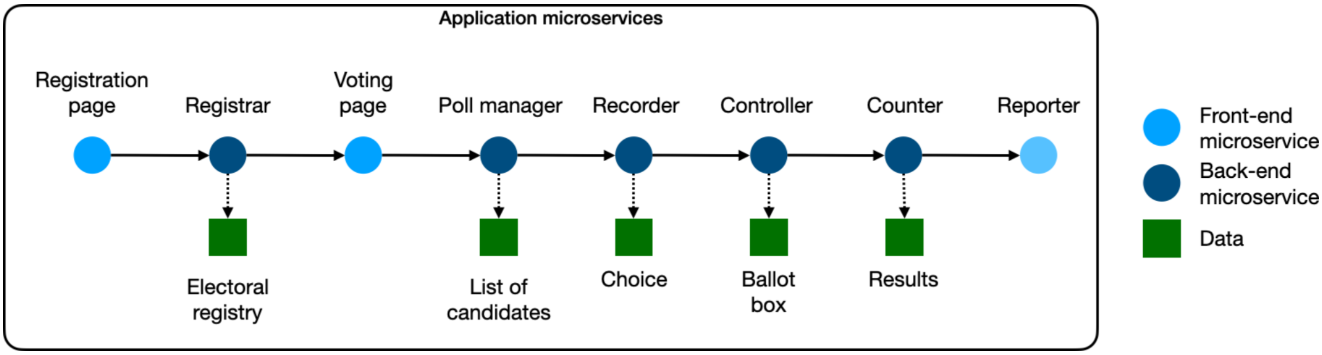


Fig. 14 Example: electronic voting system

registration and voting pages, e.g., through data masking or obfuscation [45] (R5). A privacy-preserving mesh between the registrar and the recorder is necessary to anonymize the link between the voter's registry and choice (R3). Another privacy-preserving mesh is required to avoid traceability between the registrar, recorder, and controller, e.g., through segmentation or tokenization [85] (R4).

Developers also need to consider the versatility of the voting system. Voting processes do not change frequently. Therefore, the preferred choice is for low-versatility models. This explains the preference for using a gatekeeper instead of PaaS microservices to implement R2. It also explains the preference for a federation microservice to implement R5. However, regarding R3 and R4, privacy-preserving meshes are preferred to federation microservices. This is because the mesh model is more versatile than the federation model in providing the required protections. While the federation microservice would offer a single interface to communicate with the registrar, recorder, and controller, mesh microservices can combine a variety of protections.

The final decisions concern transparency, user consent, and policy management. Regarding transparency, voters should be aware of four data sets: all process data related to voting; private data displayed on pages; the individual votes, including links to registry and choice; and the vote workflows, considering the interactions between registrar, recorder, and controller. The adopted granularity and selected protections should be disclosed for each data set. Considering user consent and policy management, we note that voting processes are peculiar because not much control is usually given to users. The capacity to revoke a vote after it has been cast is impossible in paper-based elections and is not adopted in electronic voting. In this example, users can only revoke unused credentials.

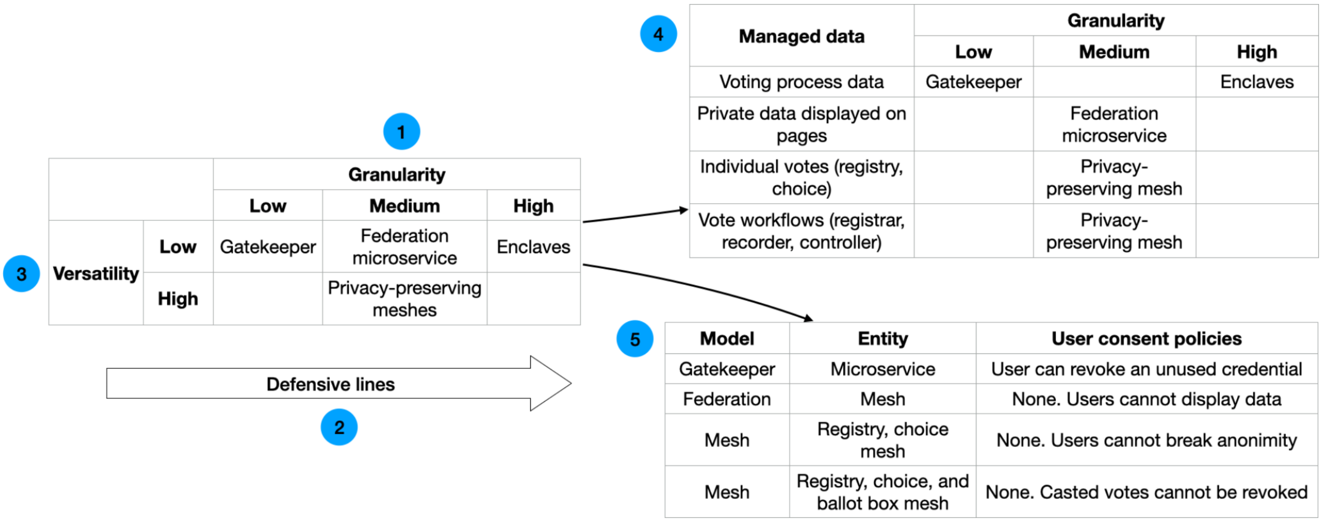
Figure 15 overviews the various decisions to implement data privacy in the voting system. This simplified example illustrates how developers can break down an application into a set of microservices and then add multiple layers of privacy-preserving microservices to ensure privacy requirements.

7 Discussion

7.1 Privacy models

One key discussion point concerns the most direct outcome from the literature review, i.e., the privacy models. The identified models define different ways privacy protection can be built into a system using microservices. Each model highlights what is protected, how microservices can provide protection, and how to manage privacy policies. Even though these models have individual traits, they are defined from a common viewpoint. Therefore, the value brought by the privacy models extends beyond the obvious, i.e., a list of options. What they provide is more substantive: a comprehensive blueprint for architecting systems based on microservices.

A system designed using this set of models can have higher specialization/division of labor in satisfying privacy requirements and express a signature of compounded functionalities that can be systematically mapped to technical requirements and principles defined by privacy regulations. These models are not contingent on specific regulations, as privacy principles are addressed from an architectural perspective using three qualities associated with microservices: decentralization, packaging, and guarding. These models are also not contingent on technical requirements and



		Granularity		
		Low	Medium	High
Versatility	Low	Gatekeeper	Federation microservice	Enclaves
	High		Privacy-preserving meshes	

Managed data	Granularity		
	Low	Medium	High
Voting process data	Gatekeeper		Enclaves
Private data displayed on pages		Federation microservice	
Individual votes (registry, choice)		Privacy-preserving mesh	
Vote workflows (registrar, recorder, controller)		Privacy-preserving mesh	

Model	Entity	User consent policies
Gatekeeper	Microservice	User can revoke an unused credential
Federation	Mesh	None. Users cannot display data
Mesh	Registry, choice mesh	None. Users cannot break anonimity
Mesh	Registry, choice, and ballot box mesh	None. Casted votes cannot be revoked

Fig. 15 Decisions made for the voting system

technologies, as these concerns are built into individualized microservices.

7.2 Framework

Another critical discussion point concerns the framework. The framework organizes the identified privacy models using two criteria: granularity and versatility. These two criteria bring together systems architecture and data privacy. They help consider data privacy an integral aspect of systems architecture, which fulfills the privacy by design paradigm. They also help assess the variety of choices offered by privacy models closer to software development, as privacy protections are built into specific system components.

Interestingly, the notions of granularity and versatility concern both the privacy and software viewpoints. From a privacy viewpoint, granularity helps consider which defensive lines a particular system requires, and versatility allows one to assess what will happen if the technical and privacy requirements change. From a software viewpoint, granularity helps to break down data and functionality, and versatility allows one to manage changes.

The framework also helps developers deal with transparency and consent, which are present in most privacy regulations [30]. Developers should communicate design choices impacting privacy, indicating what specific data is managed, which privacy models have been adopted to protect that data, the granularity of selected protections, distinct entities providing those protections, and adopted user consent policies. This level of detail is not necessarily overwhelming or obfuscating for the users. The advantage offered by the framework is taking a design perspective over transparency and consent, where problems and solutions are abstracted at an intermediate level of detail. The framework can be tailored to address communication and visualization requirements [84].

A greater awareness by the users about the developers' architectural choices may reinforce trust and consent. The trust required for users to consent in data processing emerges from the evidence requirements imposed on developers (see, for example, GDPR [3] Article 24). The proposed framework can help increase users' awareness of architecting and protecting systems. From this point of view, we may say that microservices are a society of software components, some trustworthy while others not. The proposed framework, along with privacy models, highlights these relationships. In this society, some (if not all) microservices play a role in protecting privacy. Trustworthiness depends on how this society operates as a whole.

7.3 Architecture-level regulation

Regulations on data privacy have evolved from the pronunciation of principles to the privacy by design provision [8, 28]. As the concept of privacy by design develops, regulations are expected to evolve accordingly. This study explores the next step in privacy by design, which goes beyond the enunciation of principles toward architecture-level decisions. Therefore, it may be argued that regulation should evolve towards pronouncing architecture-level rules. This could be accomplished using the framework developed in this study.

To illustrate this point, let us consider GDPR [3]. Article 28 asserts that "the controller shall use only processors providing sufficient guarantees to implement appropriate technical and organizational measures [to] ensure the protection of the rights of the data subject" and also that "[p]rocessing by a processor shall be governed by a contract or other legal act […] that is binding on the processor with regard to the controller […]." These enunciations do not exploit microservices' architectural characteristics, particularly decentralization, packaging, and guarding. A microservices-savvy enunciation would state that sensitive microservices should be packaged and guarded at low, medium, and high granularity; each sensitive microservice should provide mechanisms for data subjects to manage privacy policies. The pronunciation should also indicate that sensitive data should be decentralized and packed into single-purpose microservices. This pronunciation is closer to development and embraces some design-level concerns revealed by microservices, such as developing systems that use microservices from multiple independent sources and protecting the interactions between microservices (which is imprinted in the guarding concept). In summary, the framework developed in this study can be seen as a foundation for architecture-level regulation. Even though there are risks associated with infrastructure standardization [86], the framework is not closed or prescriptive.

7.4 Research outputs, benefits, and contributions

Table 5 summarizes the abovementioned topics and highlights the study's main outputs, benefits, and contributions. Overall, the study helps researchers and practitioners engage with the problem of privacy by design through microservices.

7.5 Limitations, future work, and future research directions

The present study adopts a research methodology that utilizes the systematic literature review to identify privacy

Table 5 Research outputs, benefits, and contributions

Topics	Research outputs	Benefits	Contributions
Systematic literature review on data privacy and microservices	Set of privacy models using microservices	Blueprint for architecting systems based on microservices	Abstract, architecture-level approach, which can be mapped to both privacy and technical requirements
Logic considerations from the literature review	Decision framework organizing privacy models using two design criteria, granularity and versatility	Address data privacy as an integral aspect of systems architecture	Turn architectural choices related to privacy more transparent, with potential to increase users' awareness
Mapping regulations to systems	Proposal of a microservices framework for a regulatory specification	Meet high level (societal, legal) requirements with discrete systems components	Demonstrate a path from often broad/undetermined concepts (as privacy) to a system structure
Managing privacy implementation	Support architecture-level design decisions related to privacy	Contribute to a baseline for architecture-level privacy support	A next step in privacy by design, supporting the development of privacy-related architectural rules

models and a framework that provides guidance and supports design decisions in implementing data privacy using microservices. However, a different research methodology could be adopted based on empirical research, where the framework would be experimentally evaluated, e.g., through field tests with developers. Such an approach would provide empirical evidence about the framework, complementing the logical propositions in this study. We will consider such empirical studies in future work.

The research is centered on the positive characteristics of microservices (decentralization, packaging, and guarding). It does not reflect on the potentially negative impacts of microservices. For instance, microservices increase system complexity, which in turn increases risk. Threats may come from multiple parties, either inside or outside [87]. Future work should also reflect on the negative characteristics of microservices.

Regarding future research directions, the set of privacy models provides a foundation for building a toolkit with standardized and reusable privacy-preserving microservices. Having such a toolkit would promote the framework's adoption. The framework could be further developed to consider a dynamic view of privacy models. The decision on which models to use does not have to be static; it could be made in runtime based on continuous risk assessment. For instance, a system could rely on gatekeepers if the risk is assessed as low, but federation microservices and privacy-preserving meshes could be added as the risk increases. The collaboration between privacy-preserving microservices should also be considered in this dynamic setting. Finally, future work could also consider using the framework for visualizing how data is handled by a system at the architectural level, e.g., using "nutrition labels." Empirical validation would indicate the understandability and utility of the templates in Tables 4 and 5.

8 Conclusions

Mapping high-level requirements usually defined in regulatory frameworks onto system components is hard as it requires a conceptual jump. On the other hand, the abundance of tech-related regulations and laws imposes specific system behaviors and functionalities, such as transparency, auditability, fairness, and risk assessment possibilities (see the recent European Union AI Act [88]). This paper adopts privacy as the problem domain, a domain addressed by legislation worldwide [89], and uses microservices to enact privacy by design.

Microservices architectures have brought several advantages to software systems and software development. Developers widely recognize them. This study explores bringing those advantages to the data privacy domain. Microservices support a design-oriented data privacy analysis focused on systems architecture. The present study analyses how to architect systems using microservices, where privacy is built into microservices and in between microservices. Multiple defensive lines can be made using microservices to protect systems.

Microservices are regarded as a society of software components, some trustworthy while others not. Privacy-protecting microservices take the responsibility for protecting the whole system. A variety of privacy models has been extracted from the literature. These models have been consolidated using a common understanding of data privacy, common language, and methodical consideration for three qualities of microservices: decentralization, packaging, and guarding.

The analysis of different ways of organizing microservices supports the decision framework. The framework helps developers make decisions related to privacy protection using two criteria: granularity and versatility. Decisions are made at the architectural level. On the one hand, they are logically separated from technical requirements and implementations. On the other hand, they are also logically separated from broader considerations about privacy principles, regulations, and compliance. Indeed, the primary purpose of the framework is to function as a middle ground, or a bridge, between regulation and implementation. As such, the framework realizes the main goals and desires of privacy by design.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10207-024-00907-y.

Author contributions P.A. conducted the review and wrote the main manuscript. N.G. made substantial contributions to the interpretation of data.

Funding Open access funding provided by FCT|FCCN (b-on).

Data availability Data from the systematic review collected from SCOPUS and Google Scholar has been collected in zotero files and is available upon request.

Declarations

Ethical approval This article does not contain any studies with human participants performed by any authors.

Competing interests The authors declare no competing interests.

References

1. Alhazmi, A., Arachchilage, N.: I'm all ears! Listening to software developers on putting GDPR principles into software development practice. Personal. Uniquit. Comput. **25**, 879–892 (2021)
2. Saltarella, M., Desolda, G., Lanzilotti, R., Barletta, V.: Translating privacy design principles into human-centered Software Lifecycle: A literature review. Int. J. Human–Computer Interact. 1–19. (2023)
3. EU, General Data Protection Regulation (GDPR): Official J. Eur. Union L. **119**, 1 (2016)
4. State of California, California Consumer Privacy Act (CCPA), State of California - Department of Justice - Office of the Attorney General: (2024). https://oag.ca.gov/privacy/ccpa (accessed November 16, 2023)
5. Spiekermann, S.: The challenges of privacy by design. Commun. ACM. **55**, 38–40 (2012). https://doi.org/10.1145/2209249.2209263
6. Wynn, D., Clarkson, P.: Process models in design and development. Res. Eng. Design. **29**, 161–202 (2018). https://doi.org/10.1007/s00163-017-0262-7
7. Simon, H.: The Sciences of the Artificial, Third Edition, The MIT Press, Cambridge, USA, (1996)
8. Rommetveit, K., Van Dijk, N.: Privacy engineering and the techno-regulatory imaginary. Soc. Stud. Sci. **52**, 853–877 (2022)
9. Baškarada, S., Nguyen, V., Koronios, A.: Architecting Microservices: Practical opportunities and challenges. J. Comput. Inform. Syst. **60**, 428–436 (2020). https://doi.org/10.1080/08874417.2018.1520056
10. Ünlü, H., Kennouche, D., Soylu, G., Demirörs, O.: Microservice-based projects in agile world: A structured interview. Inf. Softw. Technol. **165**, 107334 (2024). https://doi.org/10.1016/j.infsof.2023.107334
11. Waseem, M., Liang, P., Shahin, M., Di Salle, A., Márquez, G.: Design, monitoring, and testing of microservices systems: The practitioners' perspective. J. Syst. Softw. **182**, 111061 (2021). https://doi.org/10.1016/j.jss.2021.111061
12. Leite, L., Rocha, C., Kon, F., Milojicic, D., Meirelles, P.: A survey of DevOps concepts and challenges. ACM Comput. Surveys. **52**, 1–35 (2019)
13. Pavlenko, A., Askarbekuly, N., Megha, S., Mazzara, M.: Micro-frontends: Application of microservices to web front-ends. J. Internet Serv. Inform. Secur. **10**, 49–66 (2020)
14. Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D.: From monolithic systems to Microservices: An assessment framework. Inf. Softw. Technol. **137**, 106600 (2021)
15. Larrucea, X., Santamaria, I., Colomo-Palacios, R., Ebert, C.: Microservices IEEE Softw. **35**, 96–100 (2018)
16. California, The California Online Privacy Protection Act (CalOPPA):, Consumer Federation of California (2015). https://consumercal.org/about-cfc/cfc-education-foundation/california-online-privacy-protection-act-caloppa-3/ (accessed November 16, 2023)
17. US: Fair Information Practice Principles (FIPPs), (n.d.). (2023). https://www.fpc.gov/resources/fipps/
18. Canada, The Personal Information Protection and Electronic Documents Act (PIPEDA):, (2021). https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/ (accessed November 16, 2023)
19. Brazil: Law 13.709 (LGPD), (2018). https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm (accessed November 16, 2023)
20. China, P.I.P.L.: (2021). https://personalinformationprotectionlaw.com/ (accessed November 16, 2023)
21. India: Digital Personal Data Protection Act (DPDP), (2023)
22. Solove, D.: Understanding Privacy. Harvard University Press, Cambridge, Massachusetts (2008)
23. Solove, D.: The limitations of privacy rights. Notre Dame Law Rev. **98**, 975 (2022)
24. Zaeem, R., Barber, K.: The effect of the GDPR on privacy policies: Recent progress and future promise. ACM Trans. Manage. Inform. Syst. **12**, 1–20 (2020)
25. Iwaya, L., Babar, M., Rashid, A., Privacy Engineering in the Wild: Understanding the Practitioners' Mindset, Organisational Aspects, and, Practices, C.: IEEE Transactions on Software Engineering 49 (2023)
26. Kim, N.: Consentability: Consent and its Limits. Cambridge University Press (2019)
27. Bednar, K., Spiekermann, S., Langheinrich, M.: Engineering privacy by design: Are engineers ready to live up to the challenge? Inform. Soc. **35**, 122–142 (2019)
28. Koops, B.-J., Leenes, R.: Privacy regulation cannot be hardcoded. A critical comment on the 'privacy by design'provision

in data-protection law. Int. Rev. Law Computers Technol. **28**, 159–171 (2014)
29. Hadar, I., Hasson, T., Ayalon, O., Toch, E., Birnhack, M., Sherman, S., Balissa, A.: Privacy by designers: Software developers' privacy mindset. Empir. Softw. Eng. **23**, 259–289 (2018)
30. Aljeraisy, A., Barati, M., Rana, O., Perera, C.: Privacy laws and privacy by design schemes for the internet of things: A developer's perspective. ACM Comput. Surveys. **54**, 1–38 (2021)
31. Bu, F., Wang, N., Jiang, B., Liang, H.: Privacy by design implementation: Information system engineers' perspective. Int. J. Inf. Manag. **53**, 102124 (2020)
32. Cavoukian, A.: Understanding how to implement privacy by design, one step at a time. IEEE Consum. Electron. Mag. **9**, 78–82 (2020)
33. Cavoukian, A.: Operationalizing Privacy by Design: A Guide to Implementing Strong Privacy Practices, Information and Privacy Commissioner, Ontario, Canada, (2012)
34. Drev, M., Delak, B.: Conceptual model of privacy by design. J. Comput. Inform. Syst. **62**, 888–895 (2022)
35. Blinowski, G., Ojdowska, A., Przybyłek, A.: Monolithic vs. microservice architecture: A performance and scalability evaluation. IEEE Access. **10**, 20357–20374 (2022)
36. Dörnenburg, E.: The path to devops. IEEE Softw. **35**, 71–75 (2018)
37. Esposito, C., Castiglione, A., Choo, K.: Challenges in delivering Software in the Cloud as Microservices. IEEE Cloud Comput. **3**, 10–14 (2016). https://doi.org/10.1109/MCC.2016.105
38. Peltonen, S., Mezzalira, L., Taibi, D.: Motivations, benefits, and issues for adopting Micro-frontends: A Multivocal Literature Review. |Inf. Softw. Technol. **136**, 106571 (2021). https://doi.org/10.1016/j.infsof.2021.106571
39. Jamshidi, P., Pahl, C., Mendonça, N., Lewis, J., Tilkov, S.: Microservices: The Journey so far and challenges ahead. IEEE Softw. **35**, 24–35 (2018). https://doi.org/10.1109/MS.2018.2141039
40. Nikouei, S., Xu, R., Chen, Y., Aved, A., Blasch, E.: Decentralized smart surveillance through microservices platform. In: Sensors and Systems for Space Applications XII, SPIE, pp. 160–175. (2019). https://doi.org/10.1117/12.2518999
41. Qu, Q., Xu, R., Nikouei, S., Chen, Y.: An Experimental Study on Microservices based Edge Computing Platforms. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, pp. 836–841. (2020). https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9163068
42. Esposito, C., Castiglione, A., Tudorica, C., Pop, F.: Security and privacy for cloud-based data management in the health network service chain: A microservice approach. IEEE Commun. Mag. **55**, 102–108 (2017). https://doi.org/10.1109/MCOM.2017.1700089
43. Al-Doghman, F., Moustafa, N., Khalil, I., Sohrabi, N., Tari, Z., Zomaya, A.: AI-Enabled Secure Microservices in Edge Computing: Opportunities and challenges. IEEE Trans. Serv. Comput. **16**, 1485–1504 (2023). https://doi.org/10.1109/TSC.2022.3155447
44. Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., Babar, M.: Understanding and addressing quality attributes of microservices architecture: A systematic literature review. |Inf. Softw. Technol. **131**, 106449 (2021). https://doi.org/10.1016/j.infsof.2020.106449
45. Ataei, P., Staegemann, D.: Application of microservices patterns to big data systems. J. Big Data. **10**, 56 (2023). https://doi.org/10.1186/s40537-023-00733-4
46. Torkura, K., Sukmana, M., Meinel, C.: Integrating Continuous Security Assessments in Microservices and Cloud Native Applications. In: Proceedings of The10th International Conference on Utility and Cloud Computing, Association for Computing Machinery, New York, NY, USA, pp. 171–180. (2017). https://doi.org/10.1145/3147213.3147229
47. Bugshan, N., Khalil, I., Moustafa, N., Rahman, M.: Privacy-preserving Microservices in Industrial Internet-of-things-Driven Smart Applications. IEEE Internet Things J. **10**, 2821–2831 (2023). https://doi.org/10.1109/JIOT.2021.3098980
48. Mashaly, B., Selim, S., Yousef, A., Fouad, K.: Privacy by Design: A Microservices-Based Software Architecture Approach. In: 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference, pp. 357–364. (2022). https://doi.org/10.1109/MIUCC55081.2022.9781685
49. Hoepman, J.: Privacy design strategies. In: IFIP International Information Security Conference, Springer, pp. 446–459. (2014)
50. Perera, C., McCormick, C., Bandara, A., Price, B., Nuseibeh, B.: Privacy-by-design framework for assessing internet of things applications and platforms. In: Proceedings of the 6th International Conference on the Internet of Things, pp. 83–92. (2016)
51. Allison, D., El Yamany, H., Capretz, M.: Metamodel for privacy policies within SOA. In: 2009 ICSE Workshop on Software Engineering for Secure Systems, pp. 40–46. (2009). https://doi.org/10.1109/IWSESS.2009.5068457
52. Diamantopoulou, V., Kalloniatis, C., Gritzalis, S., Mouratidis, H.: Supporting privacy by design using privacy process patterns. In: ICT Systems Security and Privacy Protection: 32nd IFIP TC 11 International Conference, Sect. 2017, Rome, Italy, May 29–31, Proceedings 32, Springer, 2017: pp. 491–505. (2017)
53. Roubtsova, E., Bosua, R.: Privacy as a Service (PraaS): A Conceptual Model of GDPR to Construct Privacy Services. In: Business Modeling and Software Design: 11th International Symposium, BMSD 2021, Sofia, Bulgaria, July 5–7, 2021, Springer, 2021: pp. 170–189
54. Kühtreiber, P., Pak, V., Reinhardt, D.: A survey on solutions to support developers in privacy-preserving IoT development. Pervasive Mob. Comput. **85**, 101656 (2022)
55. Antignac, T., Le, D., Métayer: Privacy architectures: Reasoning about data minimisation and integrity. In: International Workshop on Security and Trust Management, Springer, pp. 17–32. (2014)
56. Ta, V.-T., Antignac, T.: Privacy by design: On the conformance between protocols and architectures. In: International Symposium on Foundations and Practice of Security, Springer, pp. 65–81. (2014)
57. Antignac, T., Le, D., Métayer: Trust driven strategies for privacy by design. In: 9th IFIP International Conference on Trust Management, Springer, Hamburg, Germany, pp. 60–75. (2015). https://doi.org/10.1007/978-3-319-18491-3_5
58. Le Métayer, D., Privacy by design: a formal framework for the analysis of architectural choices. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy, ACM, Texas: USA, pp. 95–104. (2013)
59. Kunz, I., Xu, S.: Privacy as an Architectural Quality: A Definition and an Architectural View. In: 2023 IEEE European Symposium on Security and Privacy Workshops, IEEE, pp. 125–132. (2023). https://doi.org/10.1109/EuroSPW59978.2023.00019
60. Alkhariji, L., De, S., Rana, O., Perera, C.: Semantics-based privacy by design for internet of things applications. Future Generation Comput. Syst. **138**, 280–295 (2023)
61. Perera, C., Barhamgi, M., Vecchio, M.: Envisioning tool support for designing privacy-aware internet of thing applications. IEEE Internet Things Magazine. **4**, 78–83 (2021)
62. Alshammari, M., Simpson, A.: Privacy architectural strategies: an approach for achieving various levels of privacy protection. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, Toronto Canada, pp. 143–154. (2018)
63. Hafiz, M.: A collection of privacy design patterns. In: Proceedings of the 2006 Conference on Pattern Languages of Programs, ACM, Portland, USA, pp. 1–13. (2006)
64. Burmeister, F., Kurtz, C., Vogel, P., Drews, P., Schirmer, I.: Unraveling Privacy Concerns in Complex Data Ecosystems with

Architectural Thinking.. In: Forty-Second International Conference on Information Systems, Austin, USA, (2021)

65. Kung, A.: PReparing Industry to Privacy-by-design by supporting its Application in REsearch, (2016)
66. Notario, N., Crespo, A., Martín, Y.-S., Del Alamo, J., Le Métayer, D., Antignac, T., Kung, A., Kroener, I., Wright, D.: PRIPARE: integrating privacy best practices into a privacy engineering methodology. In: 2015 IEEE Security and Privacy Workshops, IEEE, pp. 151–158. (2015)
67. Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. University of Durham, UK (2007)
68. Booth, A.: Searching for qualitative research for inclusion in systematic reviews: A structured methodological review. Syst. Reviews. **5**, 74 (2016). https://doi.org/10.1186/s13643-016-0249-x
69. Webster, J., Watson, R.: Analyzing the past to prepare for the future: Writing a literature review. MIS Q. **26**, xiii–xxiii (2002)
70. Alic, A., Almeida, J., Aloisio, G., Andrade, N., Antunes, N., Ardagna, D., Badia, R., Basso, T., Blanquer, I., Braz, T., Brito, A., Elia, D., Fiore, S., Guedes, D., Lattuada, M., Lezzi, D., Maciel, M., Meira, W., Mestre, D., Moraes, R., Morais, F., Pires, C., Kozievitch, N., Santos, W., Silva, P., Vieira, M.: BIGSEA: A Big Data analytics platform for public transportation information. Future Generation Comput. Syst. **96**, 243–269 (2019). https://doi.org/10.1016/j.future.2019.02.011
71. Fetzer, C., Mazzeo, G., Romano, L., Oliver, J., Verburg, M.: Integrating reactive cloud applications in SERECA. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, pp. 1–8. (2017)
72. Genfer, P., Zdun, U.: Avoiding Excessive Data exposure through microservice APIs. In: Gerostathopoulos, I., Lewis, G., Batista, T., Bureš, T. (eds.) Software Architecture, pp. 3–18. Springer International Publishing, Cham (2022). https://link.springer.com/https://doi.org/10.1007/978-3-031-16697-6_1
73. Preuveneers, D., Joosen, W.: Access control with delegated authorization policy evaluation for data-driven microservice workflows. Future Internet. **9** (2017). https://doi.org/10.3390/fi9040058
74. Abidi, S., Essafi, M., Guegan, C., Fakhri, M., Witti, H., Ghezala, H.: A web Service Security Governance Approach based on dedicated micro-services. Procedia Comput. Sci. **159**, 372–386 (2019). https://doi.org/10.1016/j.procs.2019.09.192
75. Alanezi, K., Mishra, S.: Incorporating individual and group privacy preferences in the internet of things. J. Ambient Intell. Humaniz. Comput. **13**, 1969–1984 (2022). https://doi.org/10.1007/s12652-021-02959-7
76. Fowley, F., Pahl, C., Jamshidi, P., Fang, D., Liu, X.: A classification and comparison Framework for Cloud Service Brokerage architectures. IEEE Trans. Cloud Comput. **6**, 358–371 (2018). https://doi.org/10.1109/TCC.2016.2537333
77. Atitallah, S., Driss, M., Ghezala, H.: FedMicro-IDA: A federated learning and microservices-based framework for IoT data analytics. Internet Things. **23** (2023). https://doi.org/10.1016/j.iot.2023.100845
78. Bhargava, B., Angin, P., Ranchal, R.: Privacy-preserving data sharing and adaptable service compositions in mission-critical clouds. In: CEUR Workshop Proceedings, pp. 60–66. (2021)
79. Meadows, C., Hounsinou, S., Wood, T., Bloom, G.: Sidecar-based Path-aware Security for Microservices. In: Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 157–162. (2023). https://doi.org/10.1145/3589608.3594742
80. Yu, D., Jin, Y., Zhang, Y., Zheng, X.: A survey on security issues in services communication of Microservices-enabled fog applications. Concurrency Computation: Pract. Experience. **31**, e4436 (2019). https://doi.org/10.1002/cpe.4436
81. Larouzee, J., Coze, J.L.: Good and bad reasons: The Swiss cheese model and its critics. Saf. Sci. **126**, 104660 (2020). https://doi.org/10.1016/j.ssci.2020.104660
82. Kamoun, F., Nicho, M.: Int. J. Healthc. Inform. Syst. Inf. **9**, 42–60 (2014). https://doi.org/10.4018/ijhisi.2014010103 Human and Organizational Factors of Healthcare Data Breaches: The Swiss Cheese Model of Data Breach Causation And Prevention
83. Schlackl, F., Link, N., Hoehle, H.: Antecedents and consequences of data breaches: A systematic review. Inf. Manag. **59**, 103638 (2022). https://doi.org/10.1016/j.im.2022.103638
84. Barth, S., Ionita, D., Hartel, P.: Understanding online privacy—a systematic review of privacy visualizations and privacy by design guidelines. ACM Comput. Surveys. **55**, 1–37 (2022)
85. Pathak, G., Singh, M.: A Review of Cloud Microservices Architecture for Modern Applications. In: (2023). https://doi.org/10.1109/WCONF58270.2023.10235199
86. Van Dijk, N., Tanas, A., Rommetveit, K., Raab, C.: Right engineering? The redesign of privacy and personal data protection. Int. Rev. Law Computers Technol. **32**, 230–256 (2018)
87. Hannousse, A., Yahiouche, S.: Securing microservices and microservice architectures: A systematic mapping study. Comput. Sci. Rev. **41**, 100415 (2021). https://doi.org/10.1016/j.cosrev.2021.100415
88. European Parliament, Topics European Parliament: EU AI Act: first regulation on artificial intelligence, (2023). https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence (accessed June 18, 2024)
89. UNCTAD, Data Protection and Privacy Legislation Worldwide, Trade, U.N., Development: (2024). https://unctad.org/page/data-protection-and-privacy-legislation-worldwide (accessed June 18, 2024)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.