




Cloud Native Privacy Engineering through DevPrivOps

Elias Grünewald^(✉) 

Information Systems Engineering, Technische Universität Berlin, Berlin, Germany
gruenewald@tu-berlin.de

Abstract. Cloud native information systems engineering enables scalable and resilient software architectures powering major online offerings. Today, these are built following agile development practices. At the same time, a growing demand for privacy-friendly services is articulated by societal norms and policy through effective legislative frameworks. In this paper, we (i) identify conceptual dimensions of cloud native privacy engineering – that is, bringing together cloud computing fundamentals and privacy regulation – and propose an integrative approach to be addressed to overcome the shortcomings of existing privacy enhancing technologies in practice and evaluating existing system designs. Furthermore, we (ii) propose a reference software development lifecycle called DevPrivOps to enhance established agile development methods with respect to privacy. Altogether, we show that cloud native privacy engineering opens up key advances to the state of the art of privacy by design and by default using latest technologies.

Keywords: Cloud native · DevOps · Privacy · Privacy engineering · Data protection · Software engineering · Privacy enhancing technologies · DevPrivOps

1 Introduction

The enormous and unstoppable rise of digital services for people’s lives already resulted in globally interconnected digital societies. During this long-lasting process the inter- and trans-disciplinary questions on how to achieve an adequate level of privacy are still to be solved – while privacy itself is an essentially contested concept [46]. Although some seem to have accepted sheer insurmountable hurdles or are actively supporting a post-privacy age (as shown by [55]), many others, fortunately, fight for autonomy and against a “surveillance capitalism” [74]; may it be through political advocacy, privacy law, or key technological advances. In this paper, we mainly focus on the latter with respect to current trends in the field of privacy engineering.

All major digital service offerings are enabled through the extensive use of highly distributed cloud computing systems. These provision compute, storage,

and network resources, that are used to build and run scalable and dynamic infrastructure and applications [10]. Within the last decade, the service portfolio of public cloud vendors has bloomed from distributed databases over service meshes to highly-specific AI-based programming and execution platforms. However, not only the technical infrastructure has drastically changed, but also development models to create and operate distributed services. Software is crafted by diverse teams in agile programming, testing and design prototyping phases, and through iterative requirements engineering and using project management tools. Namely, agile development processes like scrum allow to develop and deploy new functionalities and complete services to production continuously (DevOps), i.e. potentially multiple times per hour [5].

Inherently, distributed services are highly complex, which is why software engineering increasingly focuses on manageability, resilience and robustness, or observability to cope with the engineering challenges and – as a secondary concern – legal obligations of privacy and cloud computing. At the same time, the still emerging field of privacy engineering [29] has to provide the most accessible conceptual methods and technical tools to achieve privacy by design and by default, as legally required by the European General Data Protection Regulation (GDPR) [20] and commonly agreed upon in privacy research. Although, fundamental privacy principles [8] such as transparency, purpose limitation, and accountability, have been long established and are more often enforced [67], so far, many developers lack a solid understanding and the concrete technologies to construct privacy-friendly cloud native systems. In short, we observe three major challenges:

- **Cloud native application architectures introduce new privacy challenges** w.r.t. distributed (personal) data management across countries, availability under immense loads, compliant information flow control, restrictive access policies et cetera.
- **Software engineers are ill-equipped with privacy-preserving methods and tools addressing *all* privacy principles**, including, among others, lawfulness, transparency, or accountability; while privacy is often misinterpreted as only subject to security-related research.
- **Agile development practices still (mostly) neglect or even contradict privacy principles** (beyond data minimization and security) as cross-cutting themes of software engineering.

Addressing these issues well aligns with related work on privacy and (early) cloud computing [72], engineering privacy by design [6,7], and, how privacy is affected by agile development practices [31]. In a similar vein, this paper aims to provide a more clear viewpoint on the term of cloud native privacy engineering through a two-fold contribution:

- A conceptual model on the **dimensions of cloud native privacy engineering** accompanied by different use case scenarios from an information systems engineering perspective, and

- Proposing a **privacy-aware *DevPrivOps* reference lifecycle** addressing the shortcomings of established agile practices explicitly tailored to cloud native environments.

The journey through this paper takes place as follows: First, we briefly introduce the established concepts of cloud native application architectures and agile software development and, further, compare to related work in Sect. 2. On this basis, we observe the dimensions of cloud native privacy engineering in Sect. 3 illustrated by several use case scenarios. Afterwards, we introduce the software development cycle called *DevPrivOps* proposed for privacy-aware information systems engineering in Sect. 4. Finally, we discuss our findings and conclude in Sect. 5.

2 Background and Related Work

This section introduces a brief background on the field of cloud native engineering and agile software development. Moreover, we summarize the latest findings in the field of privacy engineering.

2.1 Cloud Native and Agile Software Development

Within the last decade, the technical evolution of distributed service-oriented architectures has been rapid and disruptive [47]. The emergence of cloud computing, mainly characterized by on-demand access to shared compute, storage, and network resources [44, 45], has led to a diverse and powerful infrastructure, platform, and software service portfolio [41, 62]. Without doubt, the transformative power of cloud-based systems serves as an important utility across many dimensions of today’s societies [25]. Most prominently, major public cloud vendors such as Amazon Web Services, the Google Cloud Platform, Microsoft Azure, and IBM Cloud, showcase their offerings, which are adopted by a multitude of private and governmental customers. Furthermore, private and hybrid cloud approaches also enable online services. The latter are often powered by open source projects such as OpenStack¹.

To build and operate applications, which are scalable for millions of users, developers rely on so-called cloud native technologies. The Cloud Native Computing Foundation (CNCF) highlights the usage of “containers, service meshes, microservices, immutable infrastructure, and declarative APIs” [10]. In practice, modern applications may consist of hundreds of loosely-coupled microservices that communicate through well-defined programming interfaces following paradigms such as REST [16] or (g)RPC².

At the same time, we observe a transformation from a (often waterfall-like) legacy software development culture towards a more flexible, iterative and agile

¹ See <https://www.openstack.org/>.

² See, e.g., <https://developers.googleblog.com/2015/02/introducing-grpc-new-open-source-http2.html>.

organizational setup [5, 54]. DevOps is widely acknowledged as the best way to deal with the complexity of large microservice architectures [19]. In doing so, the development team should be responsible for the entire lifecycle (incl. plan, code, build, test, release, deploy, operate, and monitor phases) of a software component and their expertise may be used to make individual technology decisions [38]. Together with an adhered framework for managing tasks and responsibilities (such as scrum [56]), which integrates reasonable tool support for assisting all phases, fast-paced development with which high quality software can be achieved.

Finally, cloud native architecture, engineering, and management techniques heavily focus on the possible trade-offs between different software qualities and, moreover, ultimate technology decisions [4, 24]. Such trade-offs occur in different shapes and sizes. They vary from evidence-based benchmarking experiments for choosing a best-fit technology to multilateral discussions on, e.g., what an adequate level of fair computing practice actually is in the context of cloud-based systems [65].

2.2 Privacy

Privacy is a fundamental human right according to Art. 12 of the Universal Declaration of Human Rights [66]. Moreover, it has an even longer tradition as a societal norm and guideline for legislation and jurisdiction [69]. Consequently, it is subject to inter- and trans-disciplinary research with legal, social, economic, political, psychological, and technical discourse. Predominantly, the notion of privacy is shaped by two different western cultures [70]. Being well aware of the different interpretations of privacy and data protection (including informational self-determination), hereafter we use these terms interchangeably.

Today, privacy law (and the public discussion it is complemented by³) significantly influences business practices. Regulations, such as the GDPR or the California Consumer Privacy Act (CCPA) [11] provide strong regulatory frameworks which are accompanied by landmark case law decisions (such as “Schrems II”⁴). Eventually, the legal perspective of privacy boils down to several foundational principles (e.g., transparency, data minimization, or accountability) which have been accepted as common ground (inter alia, [8, 48]). Therefore, in Sect. 3 we extract the central privacy principles which are encoded in the GDPR to be reflected with the cloud native and agile software development trends laid out above. Before that, we briefly introduce the discipline of privacy engineering.

2.3 Privacy Engineering

Privacy engineering is the discipline of technically addressing the aforementioned privacy principles to protect data subjects and to avoid threats and vulnerabili-

³ As prominent examples may serve the Snowden, Cambridge Analytica, or lately, Pegasus revelations.

⁴ See <https://curia.europa.eu/juris/document/document.jsf?text=&docid=228677&doclang=en>.

ties (inducing risks) while meeting all functional and non-functional requirements of data controllers and processors. Clearly, this does not only include the operationalization of producing source code, but also encompasses the holistic view on software architecture, business organization and culture including all stakeholders. This perspective led to the umbrella term *Privacy by Design and By Default* [8, 30, 32, 59]. From a legal perspective, privacy engineering is motivated through said motto in Art. 25 GDPR. Controllers, therefore, have to take into account the “state of the art, the cost of implementation and the nature, scope, context and purposes of processing as well as the risks of varying likelihood and severity” of processing personal data. Further, “appropriate technical and organisational measures” need to be implemented. As a consequence, there is a steady and momentous incentive for building applicable technical components. Since they may advance the state of the art, they then have to be used by data controllers in practice to protect data subjects. Naturally, when exactly the state of the art might be significantly advanced is questionable from case to case. However, the GDPR, for instance, enables certification procedures in Art. 42, which also take into consideration the differences between dominant economic players and small and medium-sized enterprises. Additionally, among others, the European Data Protection Board, constantly publishes guidelines and recommendations which are clear indicators on compliant technical and organizations measures. Likewise, other civic or research institutions provide their expertise to the public.

Focusing on the implementation, Privacy Enhancing Technologies (PETs) are subject to the core of privacy engineering research. With each generation of new technologies, the conceptual frameworks further matured: From early visions [26], over elaborated strategies for software architecture in practice [32, 34, 35] and related privacy patterns⁵, to topical challenges of software engineering and service architectures [37].

Reputed early projects such as Cranor’s P3P [13] or the European PRIME [33] and PrimeLife [52] catalysed the discourse around PETs further. More recent projects then focused on privacy and especially transparency, also in distributed contexts (e.g., Privacy & Us⁶, PRISMACloud⁷, SPECIAL⁸, or DaSKITA⁹). While many approaches focus on (not less important) data subject facing technologies (such as privacy dashboards), key advances that keep pace with the rising complexity of distributed cloud native systems are hard to identify.

Still, product managers and software engineers are ill-equipped with the right tools to put privacy by design in practice. Studies show, that there is a fundamental responsibility issue among engineers [61].

Although the majority of them is aware of the threats of non-compliant software systems and the potential harm they could produce to data subjects, they lack the means to proactively implement countermeasures against attack

⁵ See <https://privacypatterns.org/>.

⁶ See <https://privacyus.eu/>.

⁷ See <https://prismacloud.eu/>.

⁸ See <https://specialprivacy.ercim.eu/>.

⁹ See <https://daskita.github.io/>.

vectors or the ethical design of IT infrastructures [61]. Further, extensive literature review reveals that there are (i) a lack of viable tools and practices for the complete software development cycle, and (ii) misconceptions when such implementations achieve their goals [2].

As introduced above, the way software is developed has fundamentally changed (“The Agile Turn”). Traditional shrink-wrap products are to be replaced by interconnected online service offerings powered by cloud native architectures [31]. This, in turn, makes it inevitable to rethink both, the complexity of inter-relations of data processors and the functional and non-functional requirements the future generation of PETs needs to address. The same is true for the resulting automation potentials, e.g., within data protection impact assessments [73]. Furthermore, cloud native engineering is constantly in flux and will be extended through IoT and fog computing scenarios [50]. Therefore, we continue examining which dimensions cloud native privacy engineering is subject to in the following section.

3 Dimensions of Cloud Native Privacy Engineering

In the following section, we propose a cloud native privacy engineering matrix, that illustrates conceptual dimensions, which will be exemplified by subsequent use case scenarios.

First, we reiterate the importance of regulatory frameworks such as the GDPR [20] or the CCPA [11] in the context of privacy-aware cloud systems – we refer to **legislation**. Through further legislative proposals such as the European ePrivacy Regulation¹⁰, Data Governance Act¹¹, and the Digital Services Act¹² the future guidelines will be complemented. Together with evolving social norms and expectations or professional privacy threat analysis frameworks, such as LINDDUN [14], these will and already are highly influencing the compliance strategies of enterprises. Therefore, the discipline of privacy engineering has to keep track of all these legal requirements to be implemented in their software products.

Second, enterprises are changing their **organization** through more innovative workforce structures. On the one hand, many firms are no longer just supported by software, but software development is at the core of their business activity. With these changes come shifts in personnel and governance structures, roles and responsibilities, and more flexible methods of operation. This is why, from a business perspective, established models to integrate privacy need to be reviewed. These concerns are of utmost importance for decision-makers and strategists within companies to align with the aforementioned regulatory requirements (i.e. in order to avoid penalties), but also to keep being competitive. In

¹⁰ See <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52017PC0010>.

¹¹ See <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020PC0767>.

¹² See <https://eur-lex.europa.eu/legal-content/en/TXT/?uri=COM:2020:825:FIN>.

the cloud native context, this includes, for instance, make-or-buy or vendor lock-in decisions with regard to (multi-/hybrid-) cloud computing infrastructure or (external) privacy consulting.

Third, we emphasize a **process**-related dimension. Closely related to the organizational questions are the handling of effective communication and clear privacy by policy [60] responsibilities. From a computer science and engineering perspective, technical components are aspired to automate as many things as possible. As we will see later on, the smart implementation of privacy-related tools into the continuous integration and deployment (CI/CD) workflows can greatly heighten the level of data protection. However, “purely technical approaches might prove insufficient for aligning nuanced legal policies with engineering artifacts” [29]. As a consequence, engineers need to be engaged and cherished for their individual contributions to all cloud native privacy engineering efforts. This can be done through a supportive and efficient culture, incentive schemes and, most importantly, developer-centric privacy engineering solutions. These are primarily characterized through developer-friendliness (including intuitive usage, appropriate documentation etc.) and low implementation overhead [27, 51]. At the same time, already established cloud native tooling provides tremendous potential to be unlocked for *(i)* aligning with privacy law, *(ii)* supporting organizational efficacy, and *(iii)* automating many steps of the process of dealing with hundreds of services. All of these reflect the highly-specific perspectives driven by the business model and implementation of fulfillment processes of a data controller.

Furthermore, cloud native engineering is heavily focused on the specifics of (at least) three different layers. Usually, these layers are denoted as Infrastructure, Platform, and Software as a Service (XaaS). These terms emphasize the share between self-managed and fully provided solutions by the cloud provider. Since we are discussing software development in general, we rename “Software” to “Application” layer to avoid confusion. Thus, all major cloud vendors offer¹³ three layers:

- **Infrastructure** that consists of compute, storage, and network resources (virtual and/or pooled)
 - *Examples: Virtual machines, Storage buckets, Software Defined Networks*
- **Platform** for building, testing, deploying, running, and scaling services on managed infrastructure
 - *Examples: Container orchestrators, Serverless/Functions as a Service, Pre-trained machine learning environments, Managed databases, Elastic load balancers*
- **Application** that is handling the business logic and may contain several user or application programming interfaces.

¹³ Note that some of these example attributions may differ in details depending on their concrete system design. Some of the abstraction levels also increasingly blur together.

- *Examples: Depending on the business scenario, any application written in any programming language incl. interface and communication specifications.*

All of the latter are building blocks for large-scale data processing. From this follows, privacy engineering needs a bouquet of solutions to cope with the different deployment models and configurations of cloud native architecture, since personal data is processed in many different ways. We have now identified the first six dimensions of cloud native privacy engineering. Three of them (Legislation, Organization, and Process) are addressing mainly the external factors privacy engineers are influenced by.

Oriented orthogonally to the dimensions already mentioned, we will therefore now add 10 more to complete the proposed view of cloud native privacy engineering. All of the following ones are distilled from both literature and the GDPR, who we denote as essential privacy principles. Note that none of the following principles is new per se, however, it is of utmost importance to see them in conjunction with the aforementioned cloud engineering layers of abstraction. For an in-depth study, we refer to extensive related work [8, 39, 68]. We only list them very briefly for the sake of simplicity:

- **Lawfulness** (Art. 5(1a), 6–11 GDPR) comprises the prohibition of all personal data processing activities *unless* there is one of the well-defined permission options present (e.g. consent).
- **Fairness** (Art. 5(1a) GDPR) refers to proportionality between interests and necessities of both data controllers and data subjects. Moreover, it can be interpreted as procedural fairness which includes timeliness or burden of care [9]. Fairness is also an umbrella term for multiple concepts as defined by the OCED guidelines [48] and the Fair Information Practices [21].¹⁴
- **Transparency** (Art. 5(1a), 12, 13, 14, 30 GDPR) includes transparent information, communication and modalities for the exercise of data subjects and the respective obligations for data controllers or processors which allows independent verification and enables trust [8].
- **Accountability** (Art. 5(2), 24 GDPR) entails the responsibility and ability for demonstration of compliance with all the other principles. Therefore, it is closely related to enforcement and audit strategies of supervisory authorities.
- **Purpose limitation** (Art. 5(1b) GDPR) requires specific, explicit, and legitimate purpose specifications. This prohibits overly broad statements and data processing upon retrospective amendments or further incompatible processing with the initially stated purpose.
- **Data minimization** (Art. 5(1c) GDPR) limits the collection of personal data for further processing. Frequent tactics are excluding, selecting, stripping, perturbing, and deleting personal data as much as possible [35]. Possible safeguards include anonymization and (to a limited degree) pseudonymization.

¹⁴ Note, although fairness “remains under-defined from a legal perspective”, it still has to be considered in explicit design trade-offs; see also [23].

- **Accuracy** (Art. 5(1d) GDPR) determines that all personal data are to be kept up-to-date and correct. Therefore, data subjects have the right to rectification (Art. 16), which is important to reduce possible algorithmic discrimination because of false assumptions.
- **Storage limitation** (Art. 5(1e) GDPR) specifies period for which personal data can be processed. This period is strongly coupled to the lawfulness and the specific purpose for which the processing is permitted.
- **Security** (Art. 5(1f), 32 GDPR) safeguards against unauthorized and unlawful data processing. The technical and organizational measures need to ensure confidentiality, integrity, and availability (CIA triad) [1].
- **Access & Data portability** (Art. 15, 20 GDPR) refer to all data subjects' right to get a copy of all personal data relating to them. Closely related, the GDPR guarantees the freedom - where technically feasible - to transmit their personal data from one controller to another. The latter also enables a (in theory) effective mean against dominant market positions [15].

Privacy by design needs to target a positive-sum, not zero-sum to unfold its real societal impact [8]. Although within systems engineering trade-offs need to be discussed during the development process, the ultimate goal has to be to align with *all* the privacy principles best. In this context, we also acknowledge the classifications of privacy engineering *by architecture* [60], *policy* [60], and *interaction* [29] which clear the mist for evaluating proposed systems. In addition, we can contextualize (again) the privacy design strategies [35] that can be directly mapped to many of the resulting matrix elements which are depicted in Fig. 1.

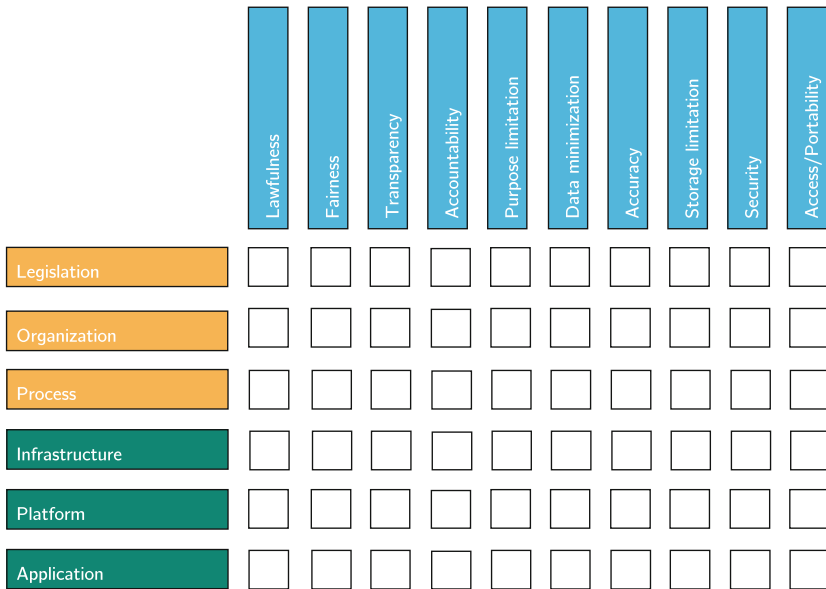
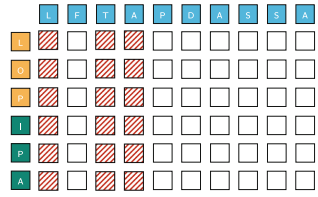


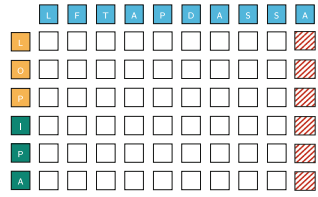
Fig. 1. Dimensions of cloud native privacy engineering

We now examine two different use cases, illustrating the range of different privacy engineering mechanisms:

Use Case 1. Transparency in large service-based cloud architectures is key to strengthen data subjects’ level of informedness. Traditionally, written privacy policies try to convey transparency information in legalese language. However, they are not only hard to understand for users, but also incompatible with agile development practices, as they – by design – cannot be changed multiple times per day. Cloud native architectures, in turn, need a machine-readable representation and additional tooling for processing said transparency information in order to describe the multitude of services in real-time. TILT [27] and TIRA [28], as technical mechanisms, address this issue being explicitly tailored to large-scale cloud native systems, agile development practices, and the legal requirements. Consequently, the proposed policy language and programming toolkit of TILT, and the OpenAPI extension and dashboard of TIRA address transparency, accountability, and lawfulness on many different levels.



Use Case 2. The Right to Data Portability (RtDP) is still uncharted territory in real-world systems. At least, many data controllers provide so-called takeouts¹⁵ for semi-automatically fulfilling the right to access according to Art. 15 GDPR. Also the CCPA clearly specifies in Sec. III that data controllers “shall promptly take steps to disclose and deliver, free of charge to the consumer, the personal information required”. Closely related, Art. 20 GDPR states the right to data portability. As a consequence, the data also has to be provided in a machine-readable format. However, the automatic transfer of all personal data from controller A to B is still somewhat disregarded. At least, one major PET has been proposed by a consortium of big technology companies, namely the Data Transfer Project (DTP)¹⁶. The DTP addresses the RtDP through three main components. First, there are several data models that can be extended by the community, and are to be used for describing the personal data to be transferred. Next, they propose company-specific adapters for authentication and how to communicate with the provider’s core infrastructure (preferably through well-defined APIs). Third, they connect these components through various middleware components enabling in-transit encryption or failure handling. The project is in an experimental state, however, it is a serious attempt to enable the RtDP. Notably, the tool is built using common cloud native techniques such



¹⁵ E.g. Google’s takeout under <https://takeout.google.com/settings/takeout>.
¹⁶ See <https://datatransferproject.dev/>.

as containerization and well-defined web APIs for developer-friendly integration at application level.

As we can infer from these two examples, cloud native privacy engineering is still a difficult endeavour. On the one hand, there is no such option as free lunch, since not a single or two tools can possibly cover the complete range of the dimensions at hand. Secondly, a remaining question is as to whether a PET is considered as “appropriate” measure. Calculating the security-related risk of a password brute-force attack is fairly easy, while, in comparison, measuring an adequate level of *fair* or *transparent* data processing is an unsolved problem. Especially in these cases, as in other compliance contexts, we need to put the organizational and process-related dimensions into the center of attention. Notwithstanding, by the help of the proposed model, we can now compare different architectures by checking how sparse or dense the matrix is filled. As a rule of thumb, the more privacy principles at different levels are met (indicated by a colored matrix element), the better is the overall rating. Salient privacy engineering solutions then cover complete columns or even span rows. In contrast, a system described by a sparse matrix faces a substantial need for remedial action. In a second step, case-specific data protection impact assessments (DPIAs) following a risk-based approach should be carried out. By its very nature, the level of ensured privacy cannot be put into a single evaluation model. However, evidence-based experiments and research shall complement the discussion: On the one hand, we need to consider the cost of implementation efforts according to Art. 25(1) in relation to the (risks associated with the) processing. On the other hand, we argue that all phases of development and operations need to be taken into account. Consequently, we need empirical studies for various kinds of PETs relating to all dimensions of cloud native privacy engineering. Having these, we can better compare and evaluate complete systems w.r.t. to architecture, engineering, and management.

After having discussed the dimensions of cloud native privacy engineering, we head over towards the software development cycle to demonstrate the implementation in practice.

4 DevPrivOps: Privacy Engineering in Practice

In this section, we suggest an enhanced *DevPrivOps* lifecycle complementing the model of [57], that illustrates how privacy can be ensured in cloud native architectures and through which tools the privacy-friendly and agile development of large-scale service infrastructures can be exemplified.

DevOps emphasizes cross-functional collaboration to operate systems and accelerate delivery of any occurring changes [17]. For this purpose, it is practiced as a software development culture that integrates the following eight phases conducted in an endless cycle [71]. We will explain them briefly in our own words (for long-reads we recommend [5,63]). Additionally, we will hint at tangible activities that complement the phase with cloud native privacy engineering tactics. Therefore, we can now introduce a DevPrivOps lifecycle, that consists of the

already established DevOps loop (depicted in blue) and an enveloping “ring” that illustrates the possibility to add privacy-related activities in every phase (cf. Fig. 2).

First, the lifecycle is initialized by a **planning** phase. Working with agile project management tools, this could be a scrum planning phase, in which the tasks for the next sprint are to be defined. Moreover, this phase serves as a checkpoint to plan either a new functionality or fixes and enhancements to an existing one. Changes can, e.g., be prioritized based on the developer’s skill or the strategic business reason why a change is requested. In traditional software engineering, the plan phase is comparable to the requirements engineering phase, in which all functional and non-functional items are to be collected. Within the planning phase, it is convenient for the team to discuss which privacy pattern or design strategy (see Sect. 2.3) to employ. This phase may also entail the threat modelling or risk analysis to decide which technologies fit best.

Second, writing of source **code** begins. This activity is not meant to be limited to programming in the general purpose languages at hand, but can also be used to write configuration files, infrastructure as code definitions [3], test cases, API specifications or database queries (non-exhaustive list). Coding is assisted by integrated development environments (IDEs), a collection of tools to assist writing code, debugging, reading documentation and so on. With regard to privacy engineering, this phase is used to employ libraries or plugging in components that feature a design goal. For the security dimension one would, e.g., choose the encryption cipher suite and library. When focusing on transparency, all personal data indicators [28] would be documented (which also streamlines auditability) or (manual or automatic) instrumentation for logging, tracing, and monitoring tools would be added. Basically, this phase is crucial for every processing activity. Some IDEs automatically hint at uncaught exceptions, possible SQL injections, non-documented function parameters, missing type checking and many possible other security flaws in the source code [42]. In addition, version control systems are used to organize multiple developers working on the same files in different development branches. These can further be used to review code changes by another team member. This enables shared responsibilities and better code quality.

Third, the application is built using **build** automation tools. These tools help to check if all external and internal dependencies can be resolved or supervise the compilation process of respective programming languages. With regard to security, outdated versions of external libraries could be identified. Taking the data minimization and purpose limitation dimensions as examples, the tools can assist in building different versions for disparate target groups. For instance, if the business model contains a paid version without targeted advertising, the build automation could exclude third party tracking functionalities. Besides, in trustless setups, for instance, zero-knowledge proofs are generated in order to keep sensitive information private [18].

After the build phase, automatic **tests** are executed. Software testing can be an exhaustive task that includes thousands of test cases. Using the testing phase

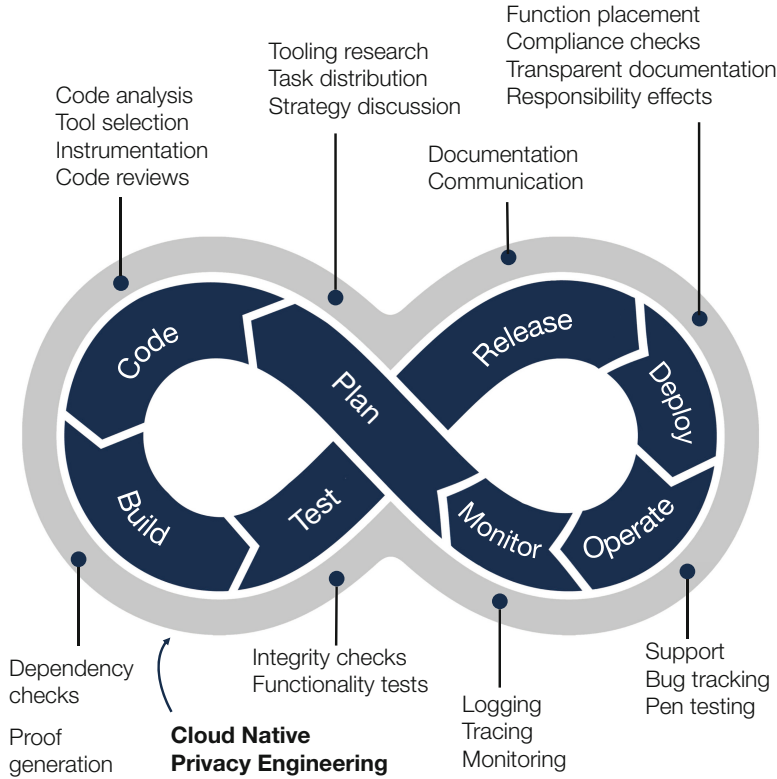


Fig. 2. Continuous DevPrivOps software development lifecycle.

for privacy-related tasks, the test suite can check several functions with different inputs if the expected accuracy or integrity are ensured. Test data sets can be used to check different behaviours or the correct calculation of obfuscation mechanisms. For instance, parts of a threat analysis and management can be automated in a CI pipeline [58]. Generally, integration tests can also be used to check the platform or infrastructure in various dimensions (esp. with security in mind) [36].

Next, the changes are **released**. Therefore, DevOps engineers automate an integration pipeline that is executed automatically. Such a pipeline may again carry out tests on different target platforms and then create a package for later delivery. Privacy engineering can play a role here again, e.g., by executing integrity checks or adding transparency-related information that can be generated out of automated analysis tasks. With this phase we leave the rather development-focused phases and enter the operation part of the loop.

Afterwards, the software is **deployed** to compute, storage, and network resources, provided by the cloud infrastructure. In automated scenarios, this may include the decision which (virtual) machine is used or at which edge device a

container is placed (in a fog computing scenario). In highly distributed scenarios these decisions can have a huge impact on the regulatory obligations that apply. Thinking about services that are deployed to a data center in a third country; this has direct implications on transparency, accountability or security dimensions [12]. However, also organizational responsibilities naturally change when software is deployed to infrastructure at different locations. Advanced deployment strategies (e.g., canary releases or A/B tests) are natural candidates for reducing the risks of potentially harmful processing of personal data.

Subsequently, **operating** the software is a key task for the responsible team. This does not only include to keep running the software technically, but may also include process-related activities like internal support or bug tracking. From a privacy perspective, security-related tasks such as pen testing or research for vulnerabilities are important. Moreover, during operation potentially lots of personal data is accessed, changed, added, or deleted. These activities need to be observed and cross-checked with the prior-made assumptions about, e.g., to which degree k/k_s -anonymity [49, 64], ℓ -diversity [43] or ε -differential [40] privacy can be guaranteed in real-world scenarios. In another dimension, data accuracy could be validated after each change.

Thereafter, the **monitoring** phase is entered. During this phase, cloud native architectures are watched using observability techniques. The most common tools perform logging, distributed tracing and collecting metrics [53]. These tools can also be used to achieve a higher level of privacy. First, logging helps to build an accountable system, since the controller can historically demonstrate that the system worked as intended by keeping the records of processing activity [22]. Secondly, distributed tracing can be used to observe service compositions. Thus, a data controller has full transparency over all personal data processing and can provide a summary to the data subject or supervisory authority in real-time. Moreover, all joint controllerships or de-facto processors are automatically detected independently from what was manually documented before. Additionally, purpose limitation can be guaranteed when there is a “watchdog” that detects unwanted or unlawful behaviour. Third, by the help of collected metrics we can detect adverse intrusions and therefore threats for personal data leakage. At the same time, key performance indicators allow to prove that data access or portability tasks were timely executed. As shown, this phase is exceptionally well suitable for all different kinds of cloud native privacy engineering.

In this section we showed how privacy engineering can be integrated into all phases of DevOps engineering. The term *DevPrivOps* was mentioned first in a recent position paper [57]. With this paper we want to further coin the term with regard to cloud computing environments and the chances of cloud-native tools to implement all privacy principles in practice. So far, the term DevOps was only augmented as *DevSecOps*. However, this perspective does not reflect the complexity of the privacy engineering discipline as a whole.

5 Discussion and Conclusion

Future privacy engineering tools need to be in line with the actual givens of practical information systems engineering. Without doubt, conceptual models provide us with guidelines for a better architecture of real-world systems. At the same time, a reality check is necessary to align with the environment of industry-grade cloud native computing. So far, we observed that the goal of privacy by design can then be reached when the software development lifecycle is focused on all dimensions as laid out in Sect. 3.

In this paper, malicious practices from within the corporation were not considered. It is evident that appropriate measures must also be taken in this regard. These are, however, primarily of an organizational nature. Consequently, clear processes and, above all, automated tests can also be effective protective safeguards. Manual manipulation away from the log is made significantly more difficult by DevPrivOps practices, e.g., when CI/CD pipelines alert or stop non-compliant code from running in production.

To the best of our knowledge, there have been no studies on the acceptance of privacy engineering methods that encompass all the dimensions identified above. Rather, there is the impression that priority is given to security- and data minimization related measures, while most other principles are neglected. To counteract this, further incentive models and easy-to-integrate technical options need to be developed. In this paper, we have provided first suggestions; in turn, a more comprehensive set of options for implementing privacy by design needs to be extracted from existing DevOps implementations.

In the same vein, we need evaluation methods for each of the elements in the cloud native privacy engineering matrix. For the development of these, we can first borrow ideas from both legal and technical methodology. Then, we need to carry out a cross-disciplinary discourse on the exact design of said approaches. This process is considered future work needed to be coming up next. Having these evaluation methods set up, we then can also better evaluate the “level of coverage” within each matrix element (possibly indicated by a filling level instead of binary hatching).

Finally, technological possibilities continue to grow at a rapid pace. With increasing connectivity through powerful mobile networks, the number of internet-enabled devices that will process personal data is exploding. For computing approaches such as fog computing and (I)IoT (Industrial Internet of Things), appropriate strategies need to be developed and tested to effectively implement the above privacy dimensions. However, they can also be beneficial and helpful for said principles [50].

So far, this work has presented two key models that bring privacy engineering by design and by default closer to the realities of information systems engineering. At the same time, it is intended to improve these designs in future iterations – just as suggested by the infinite loop presented above.

Acknowledgements. The work behind this paper was partially conducted within the project DaSKITA (Data sovereignty through AI-based transparency and access), supported under grant no. 28V2307A19 by funds of the Federal Ministry of Justice and Consumer Protection (BMJV) based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program.

References

1. Agarwal, A., Agarwal, A.: The security risks associated with cloud computing. *Int. J. Comput. Appl. Eng. Sci.* **1**, 257–259 (2011)
2. Al-Slais, Y.: Privacy engineering methodologies: a survey. In: 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), pp. 1–6 (2020). <https://doi.org/10.1109/3ICT51146.2020.9311949>
3. Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., Tamburri, D.A.: DevOps: introducing infrastructure-as-code. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pp. 497–498. IEEE (2017)
4. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Microservices architecture enables DevOps: migration to a cloud-native architecture. *IEEE Softw.* **33**(3), 42–52 (2016)
5. Bass, L., Weber, I., Zhu, L.: DevOps: A Software Architect’s Perspective. Addison-Wesley, Boston (2015)
6. Bednar, K., Spiekermann, S., Langheinrich, M.: Engineering privacy by design: are engineers ready to live up to the challenge? *Inf. Soc.* **35**(3), 122–142 (2019). <https://doi.org/10.1080/01972243.2019.1583296>
7. Cavoukian, A.: Understanding how to implement privacy by design, one step at a time. *IEEE Consum. Electron. Mag.* **9**(2), 78–82 (2020). <https://doi.org/10.1109/MCE.2019.2953739>
8. Cavoukian, A., et al.: Privacy by design: the 7 foundational principles. *Inf. Priv. Comm. Ontario, Canada* **5**, 12 (2009)
9. Clifford, D., Ausloos, J.: Data protection and the role of fairness. *Yearbook Eur. Law* **37**, 130–187 (2018). <https://doi.org/10.1093/yel/yey004>
10. Cloud Native Computing Foundation (CNCF): Cloud Native Definition v1.0 (2018). <https://github.com/cncf/toc/blob/main/DEFINITION.md>
11. California Civil Code: California consumer privacy act (CCPA) (2018)
12. Crabtree, A.: Building accountability into the Internet of Things: the IoT databox model. *J. Reliable Intell. Environ.* **4**(1), 39–55 (2018)
13. Cranor, L.F.: *Web Privacy with P3P*. O’Reilly Media Inc., Sebastopol (2002)
14. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Eng.* **16**(1), 3–32 (2011)
15. Diker Vanberg, A., Ünver, M.B.: The right to data portability in the GDPR and EU competition law: odd couple or dynamic duo? *Eur. J. Law Technol.* **8**(1) (2017)
16. Dragoni, N., et al.: Microservices: yesterday, today, and tomorrow. In: *Present and Ulterior Software Engineering*, pp. 195–216. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67425-4_12
17. Dyck, A., Penners, R., Lichter, H.: Towards definitions for release engineering and DevOps. In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering, p. 3 (2015). <https://doi.org/10.1109/RELENG.2015.10>

18. Eberhardt, J., Tai, S.: ZoKrates - Scalable privacy-preserving off-chain computations. In: IEEE International Conference on Blockchain, pp. 1084–1091. IEEE (2018)
19. Erich, F., Amrit, C., Daneva, M.: A qualitative study of DevOps usage in practice. *J. Softw.: Evol. Process* **29**(6), e1885 (2017)
20. European Parliament and Council of the European Union: Regulation (EU) 2016/679 of 27 April 2016. General Data Protection Regulation (2018)
21. Federal Trade Commission: Privacy online: Fair information practices in the electronic marketplace (2000). <https://www.ftc.gov/reports/privacy-online-fair-information-practices-electronic-marketplace-federal-trade-commission>
22. Felici, M., Koulouris, T., Pearson, S.: Accountability for data governance in cloud ecosystems. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, vol. 2, pp. 327–332. IEEE (2013)
23. Finck, M., Biega, A.J.: Reviving purpose limitation and data minimisation in data-driven systems. *Technol. Regul.* **2021**, 44–61 (2021). <https://techreg.org/index.php/techreg/article/view/63>
24. Gannon, D., Barga, R., Sundaresan, N.: Cloud-native applications. *IEEE Cloud Comput.* **4**(5), 16–21 (2017)
25. Gill, S.S., et al.: Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: evolution, vision, trends and open challenges. *Internet of Things* **8**, 100118 (2019)
26. Goldberg, I., Wagner, D., Brewer, E.: Privacy-enhancing technologies for the internet. In: Proceedings IEEE COMPCON 97. Digest of Papers, pp. 103–109. IEEE (1997)
27. Grünewald, E., Pallas, F.: TILT: a GDPR-aligned transparency information language and toolkit for practical privacy engineering. In: Proceedings of the 2021 Conference on Fairness, Accountability, and Transparency. ACM, New York (2021). <https://doi.org/10.1145/3442188.3445925>
28. Grünewald, E., Wille, P., Pallas, F., Borges, M.C., Ulbricht, M.R.: TIRA: an OpenAPI extension and toolbox for GDPR transparency in RESTful architectures. In: 2021 International Workshop on Privacy Engineering (IWPE). IEEE Computer Society (2021)
29. Gürses, S., Del Alamo, J.M.: Privacy engineering: shaping an emerging field of research and practice. *IEEE Secur. Priv.* **14**(2), 40–46 (2016)
30. Gürses, S., Troncoso, C., Diaz, C.: Engineering privacy by design. *Comput. Priv. Data Protect.* **14**(3), 25 (2011)
31. Gürses, S., van Hoboken, J.: Privacy after the Agile Turn. In: Cambridge Law Handbooks, pp. 579–601. Cambridge University Press (2018). <https://doi.org/10.1017/9781316831960.032>
32. Hansen, M.: Data protection by design and by default à la European general data protection regulation. In: Lehmann, A., Whitehouse, D., Fischer-Hübner, S., Fritsch, L., Raab, C. (eds.) *Privacy and Identity 2016*. IAICT, vol. 498, pp. 27–38. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-55783-0_3
33. Hansen, M., Berlich, P., Camenisch, J., Clauß, S., Pfitzmann, A., Waidner, M.: Privacy-enhancing identity management. *Inf. Secur. Tech. Rep.* **9**(1), 35–44 (2004)
34. Heurix, J., Zimmermann, P., Neubauer, T., Fenz, S.: A taxonomy for privacy enhancing technologies. *Comput. Secur.* **53**, 1–17 (2015). <https://doi.org/10.1016/j.cose.2015.05.002>
35. Hoepman, J.-H.: Privacy design strategies. In: Cuppens-Boulahia, N., Cuppens, F., Jajodia, S., Abou El Kalam, A., Sans, T. (eds.) *SEC 2014*. IAICT, vol. 428, pp.

- 446–459. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55415-5_38
36. Hsu, T.H.C.: Hands-On Security in DevOps: Ensure Continuous Security, Deployment, and Delivery with DevSecOps. Packt Publishing Ltd. (2018)
 37. Kostova, B., Gürses, S., Troncoso, C.: Privacy engineering meets software engineering. on the challenges of engineering privacy by design. arXiv preprint [arXiv:2007.08613](https://arxiv.org/abs/2007.08613) (2020)
 38. Kratzke, N., Quint, P.C.: Understanding cloud-native applications after 10 years of cloud computing—a systematic mapping study. *J. Syst. Softw.* **126**, 1–16 (2017)
 39. Kuner, C., Bygrave, L.A., Docksey, C.: Background and evolution of the EU general data protection regulation (GDPR). In: *The EU General Data Protection Regulation (GDPR)*. Oxford University Press (2020)
 40. Lee, J., Clifton, C.: How much is enough? Choosing ϵ for differential privacy. In: Lai, X., Zhou, J., Li, H. (eds.) *ISC 2011*. LNCS, vol. 7001, pp. 325–340. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24861-0_22
 41. Lenk, A., Klems, M., Nimis, J., Tai, S., Sandholm, T.: What’s inside the cloud? An architectural map of the cloud landscape. In: *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp. 23–31. IEEE (2009)
 42. Li, J., Beba, S., Karlsen, M.M.: Evaluation of open-source IDE plugins for detecting security vulnerabilities. In: *Proceedings of the Evaluation and Assessment on Software Engineering*, pp. 200–209 (2019)
 43. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkatasubramanian, M.: l-diversity: privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data (TKDD)* **1**(1), 3 (2007)
 44. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing—the business perspective. *Decis. Support Syst.* **51**(1), 176–189 (2011)
 45. Mell, P., Grance, T., et al.: *The NIST definition of cloud computing* (2011)
 46. Mulligan, D.K., Koopman, C., Doty, N.: Privacy is an essentially contested concept: a multi-dimensional analytic for mapping privacy. *Phil. Trans. R. Soc. A.* **374**(2083) (2016). <https://doi.org/10.1098/rsta.2016.0118>
 47. Nieuwenhuis, L.J., Ehrenhard, M.L., Prause, L.: The shift to cloud computing: the impact of disruptive technology on the enterprise software business ecosystem. *Technol. Forecast. Soc. Chang.* **129**, 308–313 (2018)
 48. OECD: *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* (1980)
 49. Pallas, F., Legler, J., Amslgruber, N., Grünewald, E.: RedCASTLE: practically applicable k_s -anonymity for IoT streaming data at the edge in Node-RED. In: *Proceedings of the 8th International Workshop on Middleware and Applications for the Internet of Things*. Association for Computing Machinery, New York (2021). <https://doi.org/10.1145/3493369.3493601>
 50. Pallas, F., Raschke, P., Bermbach, D.: Fog computing as privacy enabler. *IEEE Internet Comput.* **24**(4), 15–21 (2020). <https://doi.org/10.1109/MIC.2020.2979161>
 51. Pallas, F., et al.: Towards application-layer purpose-based access control. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1288–1296 (2020)
 52. Pfitzmann, A., Borcea-Pfitzmann, K., Camenisch, J.: Primelife. In: Camenisch, J., Fischer-Hübner, S., Rannenberg, K. (eds.) *Privacy and Identity Management for Life*, pp. 5–26. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20317-6_1

53. Picoreti, R., do Carmo, A.P., de Queiroz, F.M., Garcia, A.S., Vassallo, R.F., Simeonidou, D.: Multilevel observability in cloud orchestration. In: 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pp. 776–784. IEEE (2018)
54. Rajkumar, M., Pole, A.K., Adige, V.S., Mahanta, P.: DevOps culture and its impact on cloud delivery and software development. In: 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), pp. 1–6. IEEE (2016)
55. Rauhofer, J.: “Privacy is dead, get over it!” information privacy and the dream of a risk-free society. *Inf. Commun. Technol. Law* **17**(3), 185–197 (2008). <https://doi.org/10.1080/13600830802472990>
56. Schwaber, K.: *Agile Project Management with Scrum*. Microsoft Press, Redmond (2004)
57. Sion, L., Landuyt, D.V., Joosen, W.: The never-ending story: on the need for continuous privacy impact assessment. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), pp. 314–317. IEEE (2020). <https://doi.org/10.1109/EuroSPW51379.2020.00049>
58. Sion, L., Van Landuyt, D., Yskout, K., Verreydt, S., Joosen, W.: Automated threat analysis and management in a continuous integration pipeline. In: 2021 IEEE Secure Development (SecDev) (2021)
59. Spiekermann, S.: The challenges of privacy by design. *Commun. ACM* **55**(7), 38–40 (2012)
60. Spiekermann, S., Cranor, L.F.: Engineering privacy. *IEEE Trans. Softw. Eng.* **35**(1), 67–82 (2009). <https://doi.org/10.1109/TSE.2008.88>
61. Spiekermann, S., Korunovska, J., Langheinrich, M.: Inside the organization: why privacy and security engineering is a challenge for engineers. *Proc. IEEE* **107**(3), 600–615 (2019). <https://doi.org/10.1109/JPROC.2018.2866769>
62. Srivastava, P., Khan, R.: A review paper on cloud computing. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **8**(6), 17–20 (2018)
63. Stahl, D., Martensson, T., Bosch, J.: Continuous practices and DevOps: beyond the buzz, what does it all mean? In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 440–448. IEEE (2017)
64. Sweeney, L.: k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **10**(05), 557–570 (2002)
65. Tai, S.: Continuous, trustless, and fair: changing priorities in services computing. In: Lazovik, A., Schulte, S. (eds.) *ESOC 2016*. CCIS, vol. 707, pp. 205–210. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72125-5_16
66. United Nations General Assembly: *Universal Declaration of Human Rights (UDHR)* (1948)
67. Voigt, P., von dem Bussche, A.: Enforcement and fines under the GDPR. In: *The EU General Data Protection Regulation (GDPR)*, pp. 201–217. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57959-7_7
68. Voigt, P., Von dem Bussche, A.: *The EU General Data Protection Regulation (GDPR). A Practical Guide*, 1st edn. Springer, Heidelberg (2017). 10, 3152676
69. Warren, S.D., Brandeis, L.D.: The right to privacy. *Harv. Law Rev.* **4**(5), 193–220 (1890). <https://doi.org/10.2307/1321160>
70. Whitman, J.Q.: The two western cultures of privacy: dignity versus liberty. *Yale LJ* **113**, 1151 (2003)

71. Yarlagadda, R.T.: DevOps and its practices. *Int. J. Creat. Res. Thoughts (IJCRT)*, ISSN, pp. 2320–2882 (2021)
72. Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A.: Security and privacy in cloud computing: a survey. In: 2010 Sixth International Conference on Semantics, Knowledge and Grids, pp. 105–112 (2010). <https://doi.org/10.1109/SKG.2010.19>
73. Zimmermann, C.: Automation potentials in privacy engineering. In: Roßnagel, H., Schunck, C.H., Mödersheim, S., Hühnlein, D. (eds.) *Open Identity Summit 2020*, pp. 121–132. Gesellschaft für Informatik e.V., Bonn (2020). https://doi.org/10.18420/ois2020_10
74. Zuboff, S.: *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*. Profile Books, London (2019)