Towards Organizing the Growing Knowledge on Privacy Engineering

Jose M. del Alamo, Yod-Samuel Martín, Julio C. Caiza

Towards Organizing the Growing Knowledge on Privacy Engineering

Jose M. del Alamo[1[0000-0002-6513-0303]], Yod-Samuel Martín[1[0000-0002-0065-5117]] and Julio C. Caiza[1,2[0000-0001-9910-582X]]

[1] Universidad Politécnica de Madrid, Madrid 28040, Spain
[2] Escuela Politécnica Nacional, Quito 170517, Ecuador
social@dit.upm.es

Abstract. Regulation asks engineers to stick to privacy and data protection principles and apply them throughout the development process of their projects. However, in spite of the availability of technological solutions to identify and address different privacy threats these have not seen widespread adoption in the engineering practice, and developers still find difficulties in introducing privacy considerations in their new products and services. In this context, privacy engineering has emerged as an inter-disciplinary field that aims to bridge legal, computer science and engineering worlds, as well as concepts from other disciplines. The goal is to provide engineers with methods and tools that are closer to their mindset, and allow them to systematically address privacy concerns and introduce solutions within the workflow and environment they are accustomed to. This paper provides an introduction to Privacy Engineering, describing a conceptual metamodel useful to organize the increasing knowledge in this emergent field and make it more accessible to engineers. We exemplify some of this knowledge focusing on privacy design patterns, a set of privacy engineering elements that distill best-practices available.

Keywords: Privacy, Data Protection, Software Engineering.

1 Introduction

The European Union (EU) General Data Protection Regulation (GDPR) [1], in force since 2016 and mandatory in May 2018, sets an array of binding data protection principles, individuals' rights, and legal obligations to ensure the protection of personal data of EU citizens. But the legal approach is not enough if it does not come along with technical measures to protect privacy and personal data in practice. As it is often said, "*[software] code is law*": the technological support regulates what we do by favoring, imposing or precluding specific actions, as much as the legal framework does so by allowing, enforcing or banning them.

Indeed, the notion that privacy and data protection must be proactively considered since the onset of a project and during the design and development of information systems is captured by the principles of Privacy by Design (PbD) [2]. This approach was

openly embraced by Data Protection Authorities in Europe [3] and worldwide [4], and afterwards it has explicitly become legally required by GDPR (rec. 78 and art. 25).

While there seems to be consensus on the benefits of the privacy and data protection by design approach its realization in engineering processes remains limited due to the divergent approaches taken so far from different disciplines. First, privacy is a multi-dimensional [5], plural [6] and essentially contested concept [7], which can thus be subject to multiple reference frameworks e.g. [8][9]. If engineers find difficulties to deal with abstracts principles coming from regulations, having several different privacy conceptualizations just worsens the problem. From the purely technological arena, solutions have long been researched and elaborated to create Privacy-Enhancing Technologies (PETs) that foster data protection and respond to privacy concerns [10]. However, PETs remain unknown for most engineers, due to the uncoupling between these technologies and the practice of systematic engineering and development, which makes engineers unaware or unknowledgeable of the proper applicability of such solutions. On top of that, software and systems engineering practices are also varied regarding e.g. the type of information system targeted and the development process followed, and thus preclude having a one-size-fits-all engineering approach for privacy.

In this context, Privacy Engineering is the nascent field of research and practice that aims to address these challenges by reconciling the different approaches and deliver methods to systematically identify and address privacy and data protection concerns throughout the software development process.

This paper describes our proposal for the organization and progress of this emerging field. In particular, we introduce a methodological framework to describe the concepts and elements that underlie the various contributions subsumed under the Privacy Engineering field, following a common model and an agreed vocabulary. We further detail one of such elements that exemplify the knowledge available in practice i.e. privacy design patterns.

2 From Privacy by Design to Privacy Engineering

For PbD to be viable, engineers must be effectively involved in the loop, as they are ultimately responsible for conceiving, elaborating, constructing and maintaining the systems, services, and software products. Indeed, Data Protection Authorities around the world have also recognized developers and engineers overall as key stakeholders to achieve effective data protection [11].

However, despite the interest sparked by PbD in the regulatory arena, it has not yet gained widespread, active adoption in the engineering practice [12]. This responds to a mismatch between the legal and the technological mindsets [13]. Indeed, regulations tend to provide abstract guidance and provisions which are independent of specific technological contexts and can remain applicable as these evolve. However, technical requirements need be more concrete and anticipate the specific scenarios that may unfold. Unfortunately, this mismatch has caused privacy and data protection to be neglected or simply overlooked by most relevant works on data engineering. As a conse-

quence, from the engineers' mindset [12], privacy and data protection are usually considered just from the perspective of data security, if any; and they tend to rely for compliance on privacy policies rather than on the technical designs and architecture, which they chose instead depending on requirements and constraints other than privacy and data protection.

Furthermore, academic research has consistently shown [14][15] that developers and engineers (who usually are not privacy-savvy at all), find privacy and data protection alien to their work and, most importantly, seldom use privacy management tools, as they find these are more oriented to the legal arena rather than to the engineering activities. Same research has encountered that they will be more akin to take decisions that protect privacy and data protection when the process is embedded within their usual development workflow and tools.

Nonetheless, privacy and data protection regulatory innovations do have an impact on the engineering process. As a matter of example, the right to be forgotten or the right to data portability, besides entitling individuals to request data controllers to honor those rights, entail that the products need to implement any functionalities needed to support the user requests. This has a real impact throughout the development cycle of the product, as it implies, introducing the operational requirements to enforce those rights, modelling the categories of personal data affected, determining the functions and behavior of the system upon the users' requests, and implementing and validating those behaviors. Other regulatory innovations affect directly the process, such as accountability or data protection impact assessment.

We have identified the greatest impact in the following software and systems engineering disciplines:

- *Risk management*, which supports the execution of privacy and data protection impact assessments from the engineering perspective to identify, assess, evaluate and mitigate risks for the data subjects that may arise from processing activities dealing with their personal data.
- *Requirements engineering*, which supports the operationalization of high-level privacy and data protection goals (e.g. privacy principles, data subjects' rights, obligations of controllers and processors) into design requirements (privacy controls), and their overall systematic specification, management, analysis, traceability, validation and verification.
- *Modelling*, which supports engineers to analyse the systems under development from the perspective of privacy and data protection, and the appropriate choice of solutions (e.g. architecture, privacy patterns, PETs).
- *Assurance*, which supports the demonstration of compliance with the regulation and the observance of the principle of accountability through systematic capture of evidences, their association to requirements and artefacts, traceability to the regulation, and argumentation of compliance derived from those evidences.

The privacy engineering community have proposed dozens of novel contributions fitting in some of these engineering disciplines [16][17]. Even engineering methodologies have been developed [18][19][20][21] which define activities that deal with privacy aspects at different stages of the development process. Yet each proposal targets

specific aspects of the privacy problem, using different techniques, and following diverse methodologies to suit its own situation. This makes difficult to grasp the adequacy and assess the benefits of any such solution.

To overcome this problem, there is a need to deliver a comprehensive privacy engineering metamodel able to encompass and organize all the components available in the privacy engineering realm, including the different privacy conceptualizations as well as engineering methodologies and their elements. It can be thought of as a labelled rack, to each of whose compartments the contributions on privacy engineering can be anchored. This metamodel will further facilitate:

- The description of the different types of concepts and elements involved in existing privacy engineering methods, following a common model and agreed, shared vocabulary.
- The comparison, assessment, interoperability and integration of the distinct elements, both within and outside of the context of the method where they were originally defined, thanks to enrich descriptions of method elements that include well-defined connection hooks.
- The communication among privacy engineers, and with the other roles involved in the development process.

Method Engineering is the discipline dealing with these issues as it focuses on "*the design, construction and evaluation of methods, techniques and support tools*" [22]. Different conceptual frameworks have been historically developed for method engineering. All share a set of concepts that allow defining methodologies in compatible terms: processes, activities and tasks that can be executed, people carrying them out, products resulting from their application, and guidelines or constraints that tell how all those should be related in practice. The Software Engineering Metamodel for Development Methodologies (SEMDM), standardized as ISO/IEC 24744 [23], has perhaps been able to best capture all these concepts in its entirety, covering processes, producers (including people) and products, as well as given resources that are applied in a methodology as is (rather than instantiated or enacted). Thus, we build on SEMDM to elaborate our privacy engineering metamodel.

3 A Privacy Engineering Metamodel

Research has shown that, oftentimes, systems development activities concentrate on delivering the required functionalities at the expense of dismissing other, non-functional requirements (NFRs) [24] —such as those dealing with privacy properties. This phenomenon has been observed even in the presence of sizeable academic corpuses that deal with those requirements. Moreover, when NFRs are only considered as an afterthought, if any, to remediate blatant infringements, the correct application of Privacy by Design is eventually hindered. Nonetheless, method engineering has been proposed by same research as an approach to make existing knowledge attractive for practitioners, whose systematic application is cost-effective, whose benefits can be appraised, whose application can be customized, and which can leverage the help of computer-

aided software and systems engineering tools. Thus, we propose the systematic application of method engineering to privacy engineering methods so as to facilitate their adoption by engineers. Method engineering allows arranging the different concepts that usually underlie privacy engineering methodologies into a controlled vocabulary of methodological elements and a normalized set of connection points and relationships to organize those.

All in all, even though different privacy engineering methodologies exist, all of them can be modelled in terms of the above mentioned SEMDM metamodel. For instance, a given privacy engineering methodology may define:

- *Tasks*, which specify what must be done when enacting the methodology, e.g. mapping the types of personal data processed by the system, designing the appropriate architecture, etc. Each methodology will define its own set of tasks.
- *Techniques*, which describe procedures that tell how the tasks are to be completed; e.g. analyze a database model, apply a formal architecture analysis method, etc. Depending on the methodology, such techniques can be mandatory, recommended, optional or discouraged.
- *Processes* that group related tasks into larger units of work, within a common area of expertise, e.g. privacy impact assessment, application of PETs, etc.
- *Phases* of the software and systems application lifecycle where the tasks are applied e.g. inception, analysis, maintenance, operation, etc.
- *Work Products* that are consumed as inputs by the tasks and/or produced as their result. Different types of work products include *Models* (e.g. a dataflow diagram, a misuse case, etc.), *Documents* (a requirements specification, a risk assessment document), *Software* products, or even *Hardware* products.
- *Roles* that perform or take part in some of those tasks, e.g. a Data Protection Officer, a systems analyst, a software architect, an external auditor, etc.

It shall be noted that, in any case, a privacy engineering methodology need not be all-encompassing (specifying all the tasks, techniques, etc.). They may also require inputs (e.g. a system's architecture) that depend on the results of external tasks, refer engineers to external resources, leave unspecified techniques for some tasks, or even focus only on specific processes or phases. A detailed example of how a particular privacy engineering methodology (LINDDUN) can be described in terms of the SEMDM metamodel may be found at [25].

Even though the SEMDM metamodel may cater for a large variety of methodologies, non-functional requirements may entail the addition of new elements into the metamodel. In our case, and in order to deal with privacy NFRs, we have extended SEMDM with a set of Resources that are usually encountered in privacy engineering methods. In SEMDM, a Resource is a methodology element to be used 'as is' at the project level, without requiring any instantiation. In particular, and in order to take privacy into account when designing systems, engineers can be provided with four kinds of Resources, each of them dealing with privacy from different, complementary perspectives (Table 1).

Table 1. Types of resources provided by privacy engineering methodologies.

Resource	**Privacy Conceptual Model (PCM)**	**Privacy Normative Framework (PNF)**	**Privacy Engineering Code (PEC)**	**Privacy Knowledge Base (PKB)**
Perspective	Ontological	Deontological	Situational	Epistemological
Source	Theory of privacy	Binding regulations, non-binding best practices	Codes of conduct, codes of practice	Community of practice, repositories
Purpose	Describes	Prescribes	Refines, clarifies, documents	Compiles, arranges, endorses
Contents	Essential concepts	Method constraints and product requirements	Guidelines	Applicable knowledge models

A *Privacy Conceptual Model (PCM)* deals with privacy from an ontological perspective. Any privacy engineering method is framed by and grounded on a particular, underlying theory of privacy (even if different, competing theories currently exist). That theory describes the essential concepts of privacy in terms of principles, harms, goals, etcetera; as well as it defines the subject and the object of privacy itself. Often (but not always) the concept of privacy is partitioned into a list of unitary concepts. For instance, ISO29100 privacy framework [9] provides a list of fundamental, privacy principles, besides defining personal information, actors involved and their interactions, etc. Or, LINDDUN [21] methodology defines nine privacy properties in opposition to seven threat categories.

A *Privacy Normative Framework (PNF)* deals with privacy from a deontological perspective. Many privacy engineering methods claim to abide by some binding regulations (established by e.g. laws, quasi- and co-regulations, binding policies, etc.) or non-binding recommended best practices. These prescribe both constraints that refer to the application of the method itself (e.g. impose the existence of specific method elements, or that they be applied according to a precise temporal order), and requirements that refer to the products created when the method is enacted. For example, the EU GDPR requires (in certain cases, and among others) that:

- a Data Protection Officer (DPO) is nominated who performs specific tasks (e.g. monitoring compliance, training, etc.);
- an impact assessment process is carried out before processing any personal information;
- technical measures are implemented so as to ensure confidentiality, integrity, availability and resilience of processing systems and services.

Note that GDPR defines as well a set of principles for personal data processing which are not part of the PNF, but rather of a PCM, even if defined within the same document.

A *Privacy Engineering Code (PEC)* deals with privacy from a situational perspective. There exist many codes of conduct and codes of practice which provide different

sets of guidelines that document how normative requirements can be better applied on specific contexts or situations, thus refining or clarifying the application of the corresponding PNF. The compliance with such codes can be usually subject to audits.

A *Privacy Knowledge Base (PKB)* deals with privacy from an epistemological perspective. The community of practice and research of privacy engineering has developed an amount of generally recognized knowledge, whose value and usefulness are collectively endorsed for their application by privacy engineering practitioners. This knowledge is sometimes compiled into repositories, modelled according to a homogeneous template, and arranged into a structure that facilitates their systematic application. Such PKBs have been defined which gather e.g. privacy design strategies [26], privacy threats [21], and privacy design patterns [27].

Indeed, the PKBs of privacy design patterns particularly illustrate the usefulness of having systematic, reusable knowledge at hand, as advocated by method engineering proponents. A privacy design pattern (privacy pattern for short) provides a commonly applied, well-proven design solution to common privacy problems in particular contexts. Further, privacy pattern repositories gather patterns endorsed by the community and provide navigation mechanisms that allow engineers to easily choose the most appropriate design solution(s) to apply whenever they need to cope with privacy issues in a specific context. Next section elaborates in detail into privacy patterns and their repositories.

4 Privacy Design Patterns

Patterns researchers have defined a path to improve their applicability in system's development: patterns collections can evolve from being mere patterns catalogs, through patterns systems, until achieving a pattern language level [28]. Each provides more operationalization benefits than the previous one. A pattern catalog maintains together and classifies a set of design patterns. A pattern system goes further and presents a set of patterns with a uniform structure, some relationships between them, and as sufficient base to build the foundations of an information system. Finally, a pattern language should eventually support the complete construction of an information system, but in a very specific domain.

The state of the art already includes different contributions on privacy design patterns. Some authors have identified single patterns [29][30]; other have proposed catalogs of privacy patterns classified by different approaches [27]; and there has been even a pattern language proposal revolving around anonymity [31].

The existing privacy pattern catalogs have remained isolated and approached from different perspectives for classification and implementation. For instance, Colesky et al. classify patterns according to strategies and tactics [32], while Drozd uses ISO 29100 privacy principles [33]. In an attempt to generate a uniform knowledge base for privacy engineering practice, some authors have joined efforts to set up a common repository of privacy patterns, which could be used as a toolbox to help system designers.

The efforts of this community have concentrated in gathering the privacy patterns together, describing them according to an agreed template, and using a common categorization schema for their classification [27].

As part of this community, we have evolved a part of this catalogue into a system of patterns. To this end we have (1) proposed a taxonomy of types of relationships to describe the patterns connections [35], (2) dug into the available patterns to identify these connections, and (3) built a patterns system out of the individual patterns [34]. Table 2 enumerates and describes a sample of patterns focused on the selective disclosure of personal data. Fig. 2 further shows the relationships identified.

Table 2. Patterns supporting the selective disclosure of personal information.

Pattern name	**Pattern description**
Buddy List	Use a short list of close and trusted contacts for the user and allow the expansion of the list.
Enable/Disable Functions	Allow the users to define which functions (and provided data) they require inside an application.
Decoupling Content and Location	Allow the users to configure the privacy level of location to be disclosed associated to a content depending on the context.
Discouraging Blanket Strategies	Give the users a range of possibilities to select the privacy level associated to a content to be shared.
Negotiation of Privacy Policy	Allow the users opt-in and opt-out in the privacy configuration since the beginning of the service use.
Reasonable Level of Control	Give the users a selective control on the information they provide and to whom. Explore push and pull mechanisms for achieving this goal.
Selective Access Control	Allow the users to specify (granularly) the audience for the content during and after sharing.
Support Selective Disclosure	Instead of the massive collection of personal data, even before the use of a service, allow the users to configure the privacy level they feel comfortable with before, during and after sharing content.

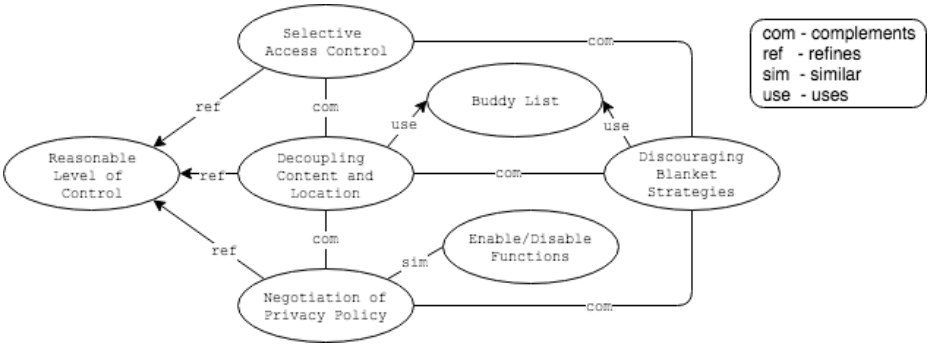

Fig. 1. Privacy patterns for the selective disclosure of personal information.

5 Conclusion

Regulation asks engineers to stick to Privacy by Design principles and apply data protection solutions throughout their projects. However, to accomplish that, engineers demand methodological elements that are closer to their mindset, and allow them to systematically introduce such solutions within the workflow and environment they are accustomed to. This paper has introduced a conceptual model to organize the growing number of methodological elements already available for privacy engineering, and has further elaborated on privacy pattern systems as means to gather systematic, reusable knowledge.

Our next steps point towards introducing some of these privacy engineering methodological elements into existent mainstream software engineering methods and tools, so as to ease their adoption by engineers even when they are not savvy in the privacy field. This is aligned with the recommendations issued by the EU Agency for Network and Information Security (ENISA) [36]: "Providers of software development tools and the research community need to offer tools that enable the intuitive implementation of privacy properties.".

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731711 and from the Spanish Government's Agencia Estatal de Investigación under grant agreement No EUIN2017-87180.

References

1. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).
2. Cavoukian, A.: Privacy by design: The 7 foundational principles. implementation and mapping of fair information practices. Information and Privacy Commissioner of Ontario, Canada (2009).
3. Article 29 Data Protection Working Party: The Future of Privacy: Joint contribution to the Consultation of the European Commission on the legal framework for the fundamental right to protection of personal data (WP168).
4. Data Protection and Privacy Commissioners: Resolution on Privacy by Design, 32nd International Conference of Data Protection and Privacy Commissioners (2010).
5. OHara, K.: The Seven Veils of Privacy. IEEE Internet Computing 20(2), 86-91 (2016).
6. Solove, D.J.: A taxonomy of privacy. U. Pa. L. Rev. 154, 477 (2005).
7. Mulligan, D.K., Koopman, C., Doty, N.: Privacy is an essentially contested concept: a multi-dimensional analytic for mapping privacy. Philos Trans R. Soc. A 374 (2016).
8. Nissenbaum, H.: Privacy in context: Technology, policy, and the integrity of social life. Stanford University Press (2009).

9. ISO/IEC JTC 1/SC 27: ISO/IEC 29100:2011. Information technologies – Security techniques – Privacy framework. Geneva (CH), 2011.
10. Heurix, J. et al: A taxonomy for privacy enhancing technologies. Computers & Security 53, 1-17 (2015).
11. Data Protection and Privacy Commissioners: Warsaw declaration on the "appification" of society, 35th International Conference of Data Protection and Privacy Commissioners (2013).
12. Hadar, I. et al.: Privacy by designers: software developers' privacy mindset. Empirical Software Engineering, 1-31 (2017).
13. Birnhack, M., Toch, E., Hadar, I.: Privacy mindset, technological mindset. Jurimetrics 55, 55-114 (2014).
14. Balebako, R., Cranor, L: Improving app privacy: Nudging app developers to protect user privacy. IEEE Security & Privacy 12(4), 55-58 (2014).
15. Van Der Sype, Y.S., Maalej, W.: On lawful disclosure of personal user data: What should app developers do? 7th International Workshop on Requirements Engineering and Law (2014).
16. IWPE homepage, http://iwpe.info, last accessed 2017/11/25.
17. ESPRE homepage, http://espre2017.org/, last accessed 2017/11/25.
18. Notario, N. et al.: PRIPARE: integrating privacy best practices into a privacy engineering methodology. IEEE Security and Privacy Workshops (2015).
19. Shapiro, S. et al.: Privacy Engineering Framework. The MITRE Corporation (2014).
20. Brooks, S. et al.: An Introduction to Privacy Engineering and Risk Management in Federal Systems. National Institute of Standards and Technology Internal Report 8062 (2017).
21. LINDDUN homepage, https://linddun.org/, last accessed 27/11/2017.
22. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and software technology 38(4),275-280 (1996).
23. ISO/IEC JTC 1/SC 27: ISO/IEC 24744:2014. Software engineering -- Metamodel for development methodologies. Geneva (CH), 2014.
24. Franch, X. et al.: Bridging the gap among academics and practitioners in Non-Functional Requirements management: Some reflections and proposals for the future. In: Essays Dedicated to Martin Glinz on the Occasion of His 60th Birthday, pp. 267-274. Monsenstein und Vannerdat (2012).
25. Martin, Y.S., Del Alamo, J.M.: A Metamodel for Privacy Engineering Methods. In: 3rd International Workshop on Privacy Engineering, pp. 41-48. CEUR workshop proceedings, San Jose (2017).
26. Hoepman, J.-H.: Privacy Design Strategies. In: ICT Systems Security and Privacy Protection SE - 38, vol. 428, pp. 446–459. Springer Berlin Heidelberg (2014).
27. PrivacyPatterns homepage, https://privacypatterns.org/, last accessed 27/11/2017.
28. Buschmann, F. et al.: Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, (1996).
29. Romanosky, S. et al.: Privacy Patterns for Online Interactions. In: Proceedings of the 2006 conference on Pattern languages of programs - PLoP '06. ACM, (2006).
30. Fernandez, E., Mujica, S.: Two Patterns for HIPAA Regulations. In: Proceedings. of AsianPLoP (Pattern Languages of Programs) 2014. (2014).
31. Hafiz, M.: A Pattern Language for Developing Privacy Enhancing Technologies. Software - Practice and Experience 43(7), 769 -787 (2013).
32. Colesky, M., Hoepman, J.-H., Hillen, C.: A Critical Analysis of Privacy Design Strategies. In 2016 IEEE Security and Privacy Workshops (SPW), pp. 33-40. IEEE, (2016).

33. Drozd, O.: Privacy pattern catalogue: A tool for integrating privacy principles of ISO/IEC 29100 into the software development process. In: IFIP Advances in Information and Communication Technology, pp. 129-140. Springer, (2015).
34. Colesky, M. et al.: A System of Privacy Patterns for User Control. In: Proceedings of the 33rd ACM Symposium on Applied Computing. ACM, Pau (2018).
35. Caiza, J. et al.: Organizing Design Patterns for Privacy: A Taxonomy of Types of Relationships. In: Proceedings of the 22Nd European Conference on Pattern Languages of Programs EuroPLOP ’17. ACM, Germany (2017).
36. Danezis, G. et al.: Privacy and Data Protection by Design-from policy to engineering. European Union Agency for Network and Information Security (2014).