Privacy Compliance via Model Transformations

Thibaud Antignac
French Atomic Energy Commission
Paris, France
Email: thibaud.antignac@cea.fr

Riccardo Scandariato
University of Gothenburg
Gothenburg, Sweden
Email: riccardo.scandariato@cse.gu.se

Gerardo Schneider
University of Gothenburg
Gothenburg, Sweden
Email: gerardo@cse.gu.se

***Abstract*—Due to the upcoming, more restrictive regulations (like the European GDPR), designing privacy preserving architectures for information systems is becoming a pressing concern for practitioners. In particular, verifying that a design is compliant with the regulations might be a challenging task for engineers. This work presents an approach based on model transformations, which guarantee that an architectural design encompasses regulation-oriented principles such as purpose limitation, or accountability of the data controller. Our work improves the state of the art along two main dimensions. The approach we propose (i) embeds privacy principles coming from regulations, thus helping to bridge the gap between the technical and the legal worlds, (ii) systematize the embedding of the privacy principles coming from regulations, thus enabling a constructive approach to privacy by design.**

***Keywords*-privacy-by-design; GDPR; MDE;**

I. Introduction

The upcoming European privacy regulation, the General Data Protection Regulation (GDPR), illustrates well how companies will be more and more subject to stringent obligations when it comes to user privacy in their digital products and services. Online systems will need to be designed with privacy in mind from the ground up (the so-called privacy by design approach), privacy risk assessment will be mandatory in most cases, and sanctions for data breaches will get tougher.

On one hand, development teams are required to think about privacy from the very beginning of each software development project. On the other hand, the obligations stated in the regulations cannot be easily translated into technical solutions, it is hard for engineers to assess whether a technical design is compliant with the law, and embedding a lawyer in each software project is downright impractical. Therefore, we identify the need to bridge the gap between the two worlds of the technical design and the privacy regulations, so that engineers would be more easily able to tackle the concerns related to privacy and the compliance to the law.

To this aim, this paper starts from the premise that Data Flow Diagrams (DFDs), boxes-and-arrows models, are often used during the conceptualization of digital systems, i.e., during the early phases of the software design. Further, DFDs are commonly used to analyze security and privacy issues of software systems [1], [2]. At the syntactical level, we extend the DFD notation to include a few key privacy concepts like the fact that some data is related to the users (and hence is privacy relevant) or that data is processed for a certain purpose, like marketing rather than service provisioning.

This minimal level of annotation on top of a business-oriented data flow diagram is used to identify the parts (i.e., hotspots) of the design model that are impacted by the principles stated in the regulations (and as an exemplification we refer to the GDPR). Such hotspots in the design are systematically transformed and, as a result, a more elaborated, privacy-aware data flow diagram (PA-DFD) is created automatically. In essence, we propose an approach to embed privacy principles in a software design by construction and by means of model-to-model transformations, which assure compliance with the recommendations of, e.g., the GDPR regulation. In future work, the semantic of the PA-DFD elements will be formally defined and we will provide the necessary formal proofs to show that the transformations preserve the business functionality and, at the same time, provide support for the intended privacy properties. However, this formal layer is hidden to the software designer. In summary, this approach represents an attempt to make it approachable to incorporate privacy principles into a software design, even for ‘regular’ designers without neither a formal nor privacy background.

This work follows the European GDPR as a guideline, as it is more restrictive and, hence, more challenging for practitioners than other regulations. Clearly, a regulation like the GDPR is quite extensive. Therefore, this work necessarily focuses on a subset of the privacy principles therein outlined. In particular, we address the principles of *purpose limitation* of the personal data processing, *accountability* of the data controller, user *right to erasure*, and *time-limited retention* of personal data.

The rest of this paper is organized as follows. In Section II we describe the proposed approach, in Section III introduce the notation, and in Section IV describe the model transformations. In Section V we discuss the related work. Finally, in Section VI we present the conclusive remarks.

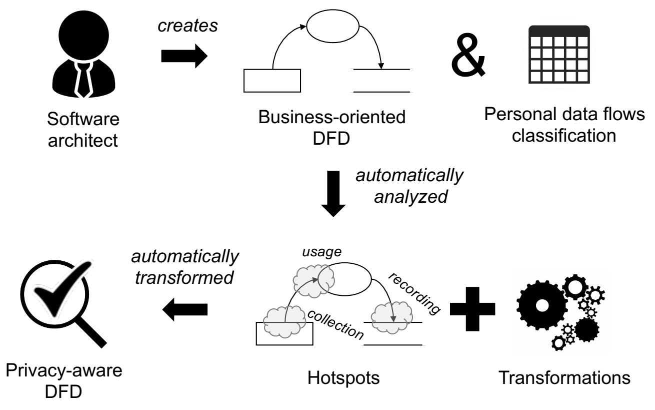

Figure 1. The privacy-by-construction approach proposed in this paper.

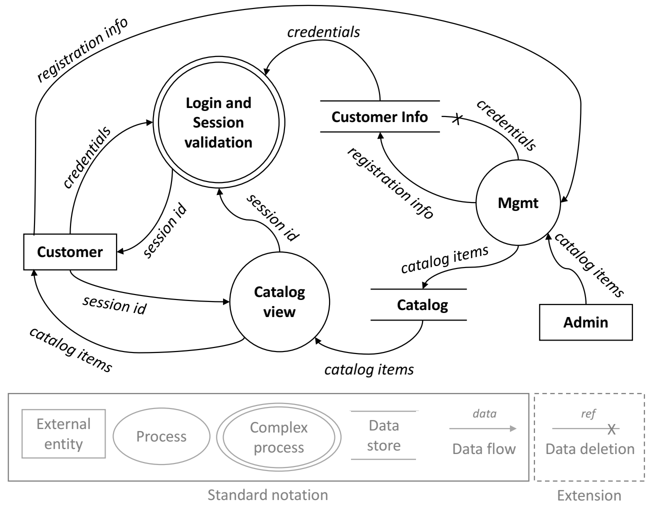

Figure 2. Example of a business-oriented DFD. The ‘data deletion’ element is an extension to the standard DFD notation.

II. An Overview of the Proposed Approach

Figure 1 illustrates the approach proposed in this paper. The designer (e.g., software architect) creates a model of the system using the DFD notation described in Section III and provides the classification of the data flows as personal and non-personal. The model is analyzed and all the privacy-sensitive operations are automatically identified as ‘hotspots’ in the model. There are 6 types of hotspots, which are described in Section IV-A. In these locations, the appropriate model transformations are injected. As described in Section IV-B, there is a specific transformation for each type of hotspot. The transformations guarantee that the resulting augmented model upholds important privacy properties by construction. The properties are explicitly mentioned in the GDPR and require, for instance, that (i) personal information is processed according to the purpose specified in the user consent, (ii) the system actions are accountable via event logging, and (iii) personal information is disposed after the retention time has expired or upon user request, and so on. Clearly, we cannot address all the provisions mentioned in the legislation. Rather, we focus on a selection of fundamental properties that can be enforced at a technical level. Further, in this paper we do not address the automation aspects related to identifying the hotspots and applying the transformations, which is subject to future work.

III. Business-Oriented DFD

A *data flow diagram* (DFD) is used to represent a digital system as a composition of functional parts that collaborate in order to deliver the business service to the intended user. As shown at the bottom of Figure 2, DFDs are composed of two sorts of elements, which are *activators* and *flows*. The activators can be *external entities* lying outside of the digital system (like end users and 3rd party systems), *processes* denoting the high-level computation applied to the data in the system (e.g., in a sub-system or in a software component), and *data stores* representing the locations where the data is stored. Processes can be *complex*, meaning that they represent a complex functionality that is detailed in an additional DFD. This mechanism allows for model refinement. Finally, data is exchanged among the above-mentioned elements by means of *data flows*.

We extend the standard notation by adding a second type of flow, called *data deletion*. It is used as an incoming flow for data stores and representing the deletion of previously stored data, whose reference is mentioned in the flow.

Composition of these elements must obey to well-formedness rules that preserve the consistency of the diagram [3]. For instance, there cannot be any isolated element and each flow must start from or end at a process, i.e., there cannot be a direct data flow between two external entities or between two data stores. Also, both processes and data stores must have at least a flow in and a flow out.

The example in Figure 2 shows an online shop that allows customers (external entity) to browse a catalog of product items, which is populated by an administrator (also an external entity). In order to access the system functionality, customers must first register via a management subsystem and provide some personal information, including their chosen credentials. Then, customers must login and obtain a session ID, which is used as a means of identification for any subsequent interaction.

Next to the DFD, the software architect must identify which data flow is carrying personal data. In Figure 1, this is referred to as the ‘personal data flows classification’. For instance, in the example of Figure 2, the ‘registration info’ flows are certainly to be considered as personal (as well as ‘session id’ and ‘credentials’). For each personal data flow, the architect must provide the following information: (i) the data subject (i.e., external entity) to whom the personal data belongs to, (ii) the purpose for the flow (to be checked against the user consents), and (iii) the retention time for the

Table I
PRIVACY PROPERTIES ENFORCED AT EACH HOTSPOT (PART 1).

	Privacy properties
Collection	**Purpose limitation.** Personal data can be collected only if the current consent given by the data subject (external entity) covers the purpose of this collection. **Accountability.** Personal data can be collected only if this collection is logged. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose of collection of their personal data.
Disclosure	**Purpose limitation.** Personal data can be disclosed only if the current consent given by the data subject covers the purpose of this disclosure. **Accountability.** Personal data can be disclosed only if this disclosure is logged. **Policy propagation.** Personal data can be disclosed only if the purpose mentioned in the current consent (and retention time if applicable) for collection, disclosure, usage, recording, retrieval, and erasure is propagated. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose of disclosure of their personal data.
Usage	**Purpose limitation.** Personal data can be used only if the current consent given by the data subject covers the purpose of this usage. **Accountability.** Personal data can be used only if this usage is logged. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose of usage of their personal data.

personal data. As explained later, this information is used to guide the identification of the hotspot and the transformation of the model.

IV. MODEL TRANSFORMATIONS

Effective model transformations require to identify where they should be performed. As a consequence, we first identify privacy hotspots before applying the transformations.

A. *Privacy Hotspots*

A conceptual model of privacy-sensitive operations with regard to personal data processing has been defined by Antignac et al. [4]. As shown on the left-hand side of Figure 3, there are six operations in total, which correspond to a step of the personal data life-cycle as described in the regulations: data collection, disclosure, usage, recording, retrieval, and erasure. These are considered hotspots for potential privacy violations. As such, the model needs to be modified at each hotspot so that certain privacy properties are entailed by construction. Hence, the privacy hotspots are the target of pre-defined model transformations. The privacy properties of interest for each hotspot are listed and described in Tables I and II. These properties are derived from the GDPR, which is the reference regulation in this work.

Finally, note that though the identification of the hotspots in a DFD to be automated is a relatively easy task (it reduces to matching the patterns sketched in Figure 3—left-hand side), the definition of what would be a good target model is

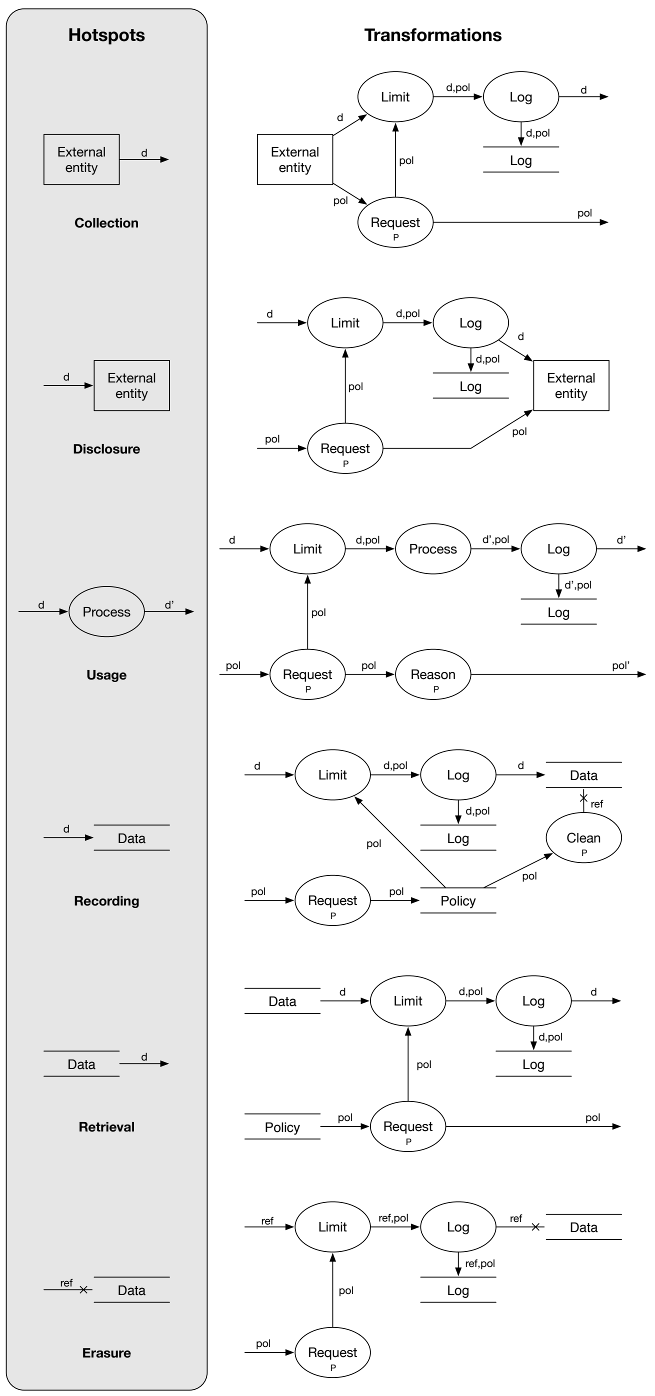

Figure 3. Privacy-sensitive parts (i.e., hotspots) of a DFD and corresponding privacy-aware transformations.

not. Some of the sources for this not being trivial are: i) The Process element needs to be split into different "subtypes" to better capture the flow in a private way; ii) A "priority" between flows needs to be added, again to enforce that certain privacy principles are respected; iii) Besides "preserving" the functionality of the original DFD, the generated

Table II
PRIVACY PROPERTIES ENFORCED AT EACH HOTSPOT (PART 2).

	Privacy properties
Recording	**Purpose limitation.** Personal data can be recorded only if the current consent given by the data subject covers the purpose of this recording. **Time retention.** Personal data can be retained as recorded only if the current retention time given by the data subject has not expired. **Accountability.** Personal data can be recorded only if this recording is logged. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose *and* retention of recording of their personal data. **Right to erasure.** At any moment, a data subject can request to erase their personal data.
Retrieval	**Purpose limitation.** Personal data can be retrieved only if the current consent given by the data subject covers the purpose of this retrieval. **Accountability.** Personal data can be retrieved only if this retrieval is logged. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose of retrieval of their personal data.
Erasure	**Purpose limitation.** Personal data can be erased only if the current consent given by the data subject covers the purpose of this erasure. **Accountability.** Personal data can be erased only if this erasure is logged. **Right to change.** At any moment, a data subject can request to change their current consent for what concerns the purpose of erasure of their personal data.

privacy-aware DFDs need to be enhanced with additional management and private structures, including logs, policy flows, and additional checks. Briefly, the transformation is the consequence of a careful thought on how to add privacy issues and accountability while preserving the underlying semantics of the original DFD. In the figure, ‘d’ is a personal data in the data flows classification and ‘ref’ is a reference to a piece of data.

B. Model Transformations and Privacy-Aware DFD

The model-to-model transformations on the right-hand side of Figure 3 transform a business-oriented DFD (source model) into a privacy-aware DFD (target model). In summary, there are two main differences in the privacy-aware DFD w.r.t. the description of DFDs given earlier in Section III. First, we define five subtypes for the process element: *Limit*, *Reason*, *Request*, *Log* and *Clean*. Second, processes can be decorated with a label (‘p’) indicating that the process has to be executed before non-priority processes in order to preserve the privacy properties. In this figure, ‘pol’ is a policy related to data ‘d’.

The transformations have many common elements. First, a *Limit* process is always the first step through which the personal data *d* should flow. This process limits data processing to the purposes that the data subject of *d* has given their consent. This, in turn, requires a policy *pol* to have been given beforehand thanks to a *Request* process in order to be able to perform this limitation. Another common part of the transformations is the *Log* process. It is used to log in the *Log* store a trace of the data processing on *d* in the context of its *pol*. The personal data is then let flow towards the rest of the data flow.

While relying on the common elements described above, each transformation also has its specificities. A **collection** receives the personal data *d* and its corresponding policy *pol* (e.g., a consent) from an *External entity* and forwards them to the next processing after application of the *Request*, *Limit*, and *Log* processes as described previously. A **disclosure** can be seen as the dual of a collection since it takes both the personal data *d* and its corresponding policy *pol* and forwards them to an *External entity*. A **usage** takes the personal data *d* and its corresponding policy *pol* from the system to apply to them the *Process* process to get a computed data *d’* and the *Reason* process to get an updated policy *pol’* corresponding to *d’*. These operations are also mixed with the common pattern before *d’* and its *pol’* are forwarded to the next operation. A **recording** takes the personal data *d* and its policy *pol* and stores them in a *Data* store and in a *Policy* store, respectively. Additionally, a *Clean* process ensures that the personal data *d* (with reference *ref*) is erased from the *Data* store, when required by *pol*. This happens notably when the consent given by the data subject of *d* changes and is no longer valid or when the current retention time of *d* has expired. A **retrieval** takes the personal data *d* and *pol* from a *Data* store and a *Policy* store and forwards them to the rest of the system after application of the common pattern of checking and logging. Finally, an **erasure** takes a reference *ref* and the policy *pol* corresponding to the referenced data and removes the data from the *Data* store.

It can be noticed that apart from the *Log* store, the first three transformations (i.e., collection, disclosure, and usage) are stateless while the last three (i.e., recording, retrieval and erasure) are stateful. In particular, there is no need to store *pol* as long as its corresponding *d* is not stored in a data store. A consequence is that processes mainly dedicated to handle policies (*Request*, *Reason*, and *Clean*) have priority over the processes dedicated at data handling (*Limit*, *Process* and *Log*). This ensures that the *Data* stores are always up-to-date with the current status of the policies.

In summary, the transformations are intended to provide support for four regulation-oriented privacy principles: (i) purpose limitation, (ii) accountability of the data controller, (iii) retention time for personal data, and (iv) right to erasure.

Purpose limitation dictates that the data controller should only process data in conformance to the consent given by the data subject. To implement this, we attach a purpose to personal data flows and check whether the purpose of the data processing is compatible with the consent. To this aim, the transformations introduce a ‘Limit’ process.

Accountability of the data controller can be improved by means of events logging. To enable this feature, the transformations add a ‘Log’ process dedicated to the logging

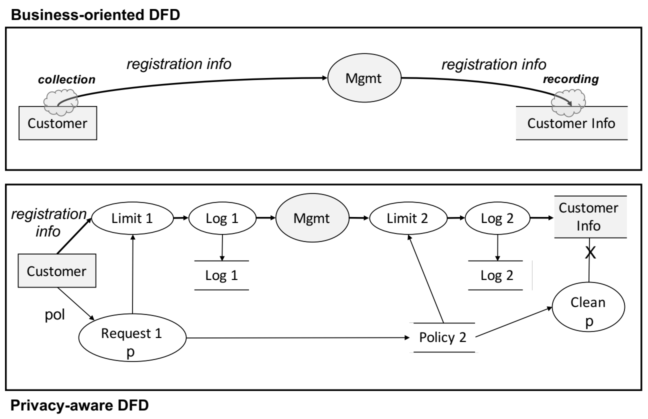

Figure 4. Example of privacy-aware DFD.

of events in the system.

Retention time is a privacy principle aiming at preventing data from resting in data stores longer than allowed by the data subject. To add this to our model, we enrich the personal data flows with information concerning the retention time and the transformations augment the data stores with 'Clean' processes, which are dedicated to performing the clean-up as necessary.

Right to erasure refers to the possibility given to the data subject to have their personal data erased. This is managed via the withdrawal of the consent for the personal data to be erased, which, in turn, triggers an automatic clean-up of such data thanks to the priority given to the 'Request' and 'Clean' processes.

We remark that the GDPR is an extensive piece of regulation and we have concentrated on a subset of its provisions, namely on those aspects that more directly translate to technical solutions. However, we argue that we cover, to a sufficient extent, the most recurring issues with respect to privacy compliance.

C. Example

In Figure 4, two transformations (collection and recording) are applied to the business-oriented DFD of Figure 2, of which, only a relevant subset is shown. Note that the privacy-aware DFD can be generated automatically and is not necessarily meant to be presented to the architect. Rather, it could be used to drive the model-driven generation of a blueprint of the system implementation. Further, it could be presented to a privacy expert in charge of streamlining and refactoring the generated model.

Note that the transformed model (i.e., after all transformations are applied) could be refactored in order to centralize the logging. That is, the 'Log 1' and 'Log 2' data stores could be unified. Further, the 'Log 1' and 'Log 2' processes could be placeholders that delegate the actual event logging activity to a central logging process. A similar strategy could be applied to refactor the parts that enforce the purpose limitation imposed by the user consents (see the 'Limit 1' and 'Limit 2' processes).

Although this paper does not cover such refactoring issues, the transformations we propose suggest that a (service oriented) *reference architecture* (i.e., a macro architectural pattern) could be devised in order to have a unified solution that deals with some recurring activities, like, for instance, logging events, checking the consents, handling data subject requests, and so on. The definition of such reference architecture is subject to future work. However, we remark that such reference architecture would not diminish the value of the transformations presented here, as they provide a principled way to reason about where and how to 'weave' the reference architecture into the existing business-oriented DFD.

V. RELATED WORK

This work lies at the intersection of the research areas of privacy by design, privacy at architectural level, and privacy threat analysis.

Privacy by Design: The most well-known principles of privacy by design come from the proposal of Cavoukian [5]. They divide privacy into high-level non-functional properties a system should satisfy and do not give real insights to architectural ways to meet these. However, Cavoukian and her team proposed different (still high-level) generic architectures and solutions for specific domains such as geolocation services, smart grids, biometric solutions, and cloud-computing [6], [7], [8], [9].

Technical committees at standardization organizations proposed several frameworks or guidelines to help software designers to embed privacy in their system. The OASIS PbD-SE [10] brings more precise definitions and ways to make real the seven foundational principles of privacy by design as proposed by Cavoukian. The OASIS PMRM [11] proposes a methodology to embed privacy in principles. On the other hand, the ISO 29100 [12] keeps a very definitional purpose but is nonetheless useful for the definitions of privacy properties in principles as they are understood in the industry.

A top-down approach for privacy by design is described in [13] and relies on a layered approach from strategies to technologies (such as PETs) with patterns as the intermediate step. These patterns can for instance be built by relying on a catalog such as [14]. A top-down approach to derive PETs from higher-level privacy goals is described in [15].

Finally, [16] details the link that can be made between formal methods and privacy and how the former can help to define and implement the latter. More particularly, the connections between privacy and models, logics, policies, abstractions and refinements, and static analysis are detailed.

Privacy at Architectural Level: Several authors mention the importance of privacy engineering to build systems conforming to personal data regulations (and customers' expectations). The difference between privacy (provided) by policy and privacy (provided) by architecture is explained in [17] where the authors emphasize the importance of the architecture to mitigate the privacy risks. The data minimization principle is identified as one of the core principles by Cavoukian and applied to different case studies in [18]. The difficulty to get a satisfactory system make the authors to conclude that such methodologies are not very recommendable, and warn against the use of too simple ways such as check-lists.

The software engineering method LIND(D)UN [2] has been proposed as a systematic approach to model and help a designer to build a system by choosing appropriate PETs to mitigate the risks identified. Another method to build systems with privacy in mind is PriS [19]. This is a goal-oriented method to take into account privacy properties early in the development cycle. Although they claim to rely on a formal model, the semantics of their model only describes their process instead of the privacy properties to be verified. A comparison of PriS and LIND(D)UN, two methods having a strong impact at the architectural level, is proposed in [20].

Some proposals have been given to exemplify how an architecture and a system can be built in a specific domain such as geolocation services for [21]. This work proposes a conceptual architecture with a formalization of the k-anonymity property and shows how the proposed algorithm conforms to the policy. Another collection of work has been performed to propose specific architectures in different contexts, such as ubiquitous computing, vehicular ad-hoc networks, or online social networks for instance [22], [23], [24].

Privacy Threats: The main work in the definition of privacy threats is Solove's taxonomy [25]. The different way by which the privacy of individuals can be endangered is systematically analyzed for the collection, processing and dissemination phases of the life cycle of data in a system.

More recently, new privacy threats and issues arising with cloud-computing have been extensively studied [26], [27]. Both papers propose high-level solutions to mitigate the identified threats that may have a strong impact on the architecture of a system. The second work also deals with legal requirements.

Risk-Based Analysis: The most general work on risk-based analysis details how an iterative risk-based analysis should be integrated in the development cycle of a system [28]. Impacts at the architectural levels are mentioned and detailed, especially concerning n-tier architectures.

An example of an application of a risk-based analysis in the context of HIPAA conformant systems is provided in [29] by the Department of Health and Human Services of the US. This relies on the risk management guide (for general purpose) established by the NIST [30]. Recently, the NIST developed a risk management framework dedicated to privacy analysis [31].

VI. Conclusion

We have enhanced the broadly used DFD models with privacy-aware concepts with the objective of achieving privacy by design and privacy-by-construction in software systems. We called such extended models Privacy-Aware DFDs (or PA-DFDs). We started with the assumption that software architects are in general more inclined to focus on the business-oriented functionality of the system under construction and not on other properties, like privacy, which require a high level of expertise. Thus, our solution has focused on providing an automatic translation from quasi-standard DFDs into PA-DFDs, where the checking of a number of privacy aspects (e.g., purpose limitation and accountability of the data controller) are added by construction in a transparent way.

Future Work: In future work, we will formally prove that the transformations preserve the functionality of the original DFD as well as a number of privacy principles that are prominent in the upcoming European GDPR. Also, we plan on providing tool support for the presented approach. Additionally, we will investigate an extension of the transformations in order to encompass additional user rights mentioned in the GDPR, like the right of access, the right to correct inaccurate data, and the right to data portability. Finally, we are interested in defining suitable heuristics for the refactoring of the resulting transformed model, e.g., by means of a privacy-aware reference architecture or by applying other architectural patterns.

Acknowledgment

This research has been supported by the Swedish funding agency VINNOVA under the grant *HoliSec: Holistic Approach to Improve Data Security* and by the Swedish funding agency SSF under the grant *DataBIn: Data Driven Secure Business Intelligence*.

References

[1] A. Shostack, *Threat Modeling: Designing for Security*. Wiley, 2014.

[2] K. Wuyts, R. Scandariato, and W. Joosen, "Empirical evaluation of a privacy-focused threat modeling methodology," *Journal of Systems and Software*, vol. 96, pp. 122–138, 2014.

[3] E. Falkenberg, R. V. D. Pols, and T. V. D. Weide, "Understanding process structure diagrams," *Information Systems*, vol. 16, no. 4, pp. 417 – 428, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0306437991900325

[4] T. Antignac, R. Scandariato, and G. Schneider, "A privacy-aware conceptual model for handling personal data," in *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, ser. LNCS, vol. 9952. Springer, 2016, pp. 942–957.

[5] A. Cavoukian, "Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada*, 2009.

[6] ——, "Redesigning ip geolocation: Privacy by design and online targeted advertising," *Information and Privacy Commissioner, Ontario, Canada*, 2010.

[7] ——, "Operationalizing privacy by design: the ontario smart grid case study," *Information and Privacy Commissioner, Ontario, Canada*, 2011.

[8] A. Cavoukian and A. Stoianov, *Privacy by Design Solutions for Biometric One-to-Many Identification Systems*, 2014.

[9] K. Zeng, A. Cavoukian, and N. D. K. Kaisha, *Modelling cloud computing architecture without compromising privacy: A privacy by design approach*. Information and Privacy Commissioner of Ontario, 2010.

[10] O. Committee, "Privacy by design documentation for software engineers (pbd-se)," Tech. Rep., 2014.

[11] ——, "Privacy management reference model and methodology (pmrm)," Tech. Rep., 2013.

[12] I. Committee, "Privacy framework (iso 29100)," Tech. Rep., 2011.

[13] J.-H. Hoepman, "Privacy design strategies," in *ICT Systems Security and Privacy Protection*. Springer, 2014, pp. 446–459.

[14] M. Hafiz, "A collection of privacy design patterns," in *Proceedings of the 2006 conference on Pattern languages of programs*. ACM, 2006, p. 7.

[15] K. Wuyts, R. Scandariato, B. D. Decker, and W. Joosen, "Linking privacy solutions to developer goals," in *International Conference on Availability, Reliability and Security (ARES)*, 2009.

[16] M. Tschantz and J. Wing, "Formal methods for privacy," in *FM 2009: Formal Methods*, ser. Lecture Notes in Computer Science, A. Cavalcanti and D. R. Dams, Eds. Springer Berlin Heidelberg, 2009, vol. 5850, pp. 1–15. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05089-3_1

[17] S. Spiekermann and L. F. Cranor, "Engineering privacy," *Software Engineering, IEEE Transactions on*, vol. 35, no. 1, pp. 67–82, 2009.

[18] S. Gürses, C. Troncoso, and C. Diaz, "Engineering privacy by design," *Computers, Privacy & Data Protection*, vol. 14, 2011.

[19] C. Kalloniatis, E. Kavakli, and S. Gritzalis, "Addressing privacy requirements in system design: the pris method," *Requirements Engineering*, vol. 13, no. 3, pp. 241–255, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00766-008-0067-3

[20] K. Beckers, "Comparing privacy requirements engineering approaches," in *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, Aug 2012, pp. 574–581.

[21] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 1–18, Jan 2008.

[22] J. I. Hong and J. A. Landay, "An architecture for privacy-sensitive ubiquitous computing," in *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 177–189. [Online]. Available: http://doi.acm.org/10.1145/990064.990087

[23] K. Plossl, T. Nowey, and C. Mletzko, "Towards a security architecture for vehicular ad hoc networks," in *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*. IEEE, 2006, pp. 8–pp.

[24] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "Decent: A decentralized architecture for enforcing privacy in online social networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 326–332.

[25] D. J. Solove, "A taxonomy of privacy," *University of Pennsylvania law review*, pp. 477–564, 2006.

[26] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, no. 6, pp. 24–31, 2010.

[27] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 693–702.

[28] D. Verdon and G. McGraw, "Risk analysis in software design," *Security & Privacy, IEEE*, vol. 2, no. 4, pp. 79–84, 2004.

[29] D. Committee, "Basics of risk analysis and risk management for hipaa," Tech. Rep., 2005.

[30] N. Committee, "Guide for conducting risk assessments (nist 800-30)," Tech. Rep., 2012.

[31] ——, "Privacy risk management for federal information systems (nistir 8062)," Tech. Rep., 2015.