

From GDPR to Privacy Design Patterns: The MATERIALIST Framework

Vita Barletta¹, Giuseppe Desolda¹, Domenico Gigante¹, Rosa Lanzilotti¹
and Marco Saltarella^{1,2}

¹Computer Science Department, University of Bari Aldo Moro, Via Edoardo Orabona, 4, 70125 Bari BA, Italy

²FINCONS SpA, Via Orfeo Mazzitelli, 258/E, 70124 Bari (BA), Italy

Keywords: Privacy Design Patterns, GDPR, ISO 9241-210, Code Vulnerabilities.

Abstract: Privacy is becoming an increasingly important factor in software production. Indeed, besides increasing software quality, privacy is a mandatory aspect of national and supranational regulations like GDPR. However, several aspects like lack of knowledge on privacy and data protection regulations ambiguities limit the adoption of proper privacy implementation mechanisms during the software lifecycle. To fill this gap, this paper presents a framework, MATERIALIST, which aims to guide developers in choosing privacy design patterns to be used during software development. In particular, this paper focuses on the selection of privacy design patterns starting from the GDPR requirements. In this way, what is currently prescribed by GDPR in a non-technical way becomes a practical solution that software developers can adopt during their work.

1 INTRODUCTION


Despite privacy is increasingly becoming a desired aspect of software systems, often required by data protection regulations like the General Data Protection Regulation (GDPR), different aspects are still limiting and affecting the right implementation of privacy features.


First, designers and developers often have *different views toward privacy*, which reflects in different approaches and solutions to implement it (Martín and Kung 2018). Second, *data protection regulations only provide legal indications* on how systems should be designed but no concrete or technical indications for developers are reported. Third, data protection regulations claim to involve the users (Sobolewski, Mazur, and Paliński 2017), but their *implementations are not user-centric*. Fourth, there is often a *lack of privacy and security knowledge* among developers and engineers (Hjerppe, Ruohonen, and Leppänen 2019). All these aspects contribute to improper integration of security and


privacy features during software lifecycles, determining vulnerable systems or insecure data protection solutions.


For these reasons, more adequate methodologies are required to ease the process of integrating privacy aspects in the software lifecycle without ambiguities, considering the end-users point of view and limiting the possible scarce knowledge of the developers and designers on privacy and security.


This paper presents the results of an ongoing wider research project whose goal is to support developers, designers and engineers integrate privacy aspects during software development. The main result of this project is the definition of a framework, *MATERIALIST* (Mapping dATa rEGulation softwaRe lIfecycle And vuLnerabilitieS paTterns), which guides the selection of privacy design patterns (PDP) starting from three different entry points, i.e., 1) GDPR articles, 2) phases of the ISO 9241-210 software development lifecycle and 3) vulnerabilities discovered during static code analysis. The first two entry points guarantee the possibility to include PDPs

^a <https://orcid.org/0000-0002-0163-6786>

^b <https://orcid.org/0000-0001-9894-2116>

^c <https://orcid.org/0000-0003-3589-6970>

^d <https://orcid.org/0000-0002-2039-8162>

^e <https://orcid.org/0000-0002-0021-9972>

when the development process starts from scratch (forward engineering), while the third entry point supports the re-engineering of a software system (backward engineering). To make more concrete the adoption of the selected PDPs, we are also extending such PDSs by defining both architectural patterns and User Interface patterns that better drive designers and developers.

The contribution of this paper to the MATERIALIST framework is the definition of the mapping between 14 GDPR articles and 72 PDPs. This allows developers to select the right PDPs when they have to consider specific GDPR articles. We also present the preliminary results of the ongoing mapping between PDPs and ISO 9241-210 lifecycle, and between PDPs and code vulnerabilities. Lack of space prevents us from presenting other details we are working on, for example, the architectural patterns and user interface patterns linked with the 72 PDPs.

The paper is structured as follows. Section 2 presents some existing frameworks for privacy and security (Section 2.1) and then discusses some background concepts on which this research was founded (Section 2.2 and Section 2.3). Then Section 3 presents the MATERIALIST framework and then illustrates the mapping phase between the privacy design patterns and the GDPR articles. Section 4 provides an overview of the ongoing works on the mappings with ISO 9241-210 phases and code vulnerabilities. Section 5 concludes the paper and reports some future works.

2 BACKGROUND AND RELATED WORK

2.1 Integrating Privacy and Security in Software Development

A growing number of frameworks and approaches addressing privacy or security aspects in software development have been proposed. However, very few solutions face both problems at the same time.

Concerning privacy, one of the most recent solutions is Security Threat Oriented Requirements Engineering (*P-STORE*) a methodology for software development that proposes 10 mandatory steps, which starts by defining the software's data privacy goals and ends with the privacy elicited requirements. Another privacy-friendly approach is Structured Analysis Framework for Privacy (*STRAP*), which proposes 4 steps: analysis of the system, refinement of the found vulnerabilities, evaluation of the

resulting vulnerabilities, and iteration of the entire process until a reasonable level of risk is reached (Jensen et al. 2005).

Another interesting framework recently proposed by Brodin focuses on the GDPR compliance of small and medium enterprises (Brodin 2019). However, the proposed framework does not focus on the technical implementation of the regulation in software. A more technical work is *BPR4GDPR*, an H2020 project that proposes a framework composed of 8 different phases for supporting the re-engineering approach of business processes to provide compliance-by-design (Lioudakis et al. 2020). It includes a compliance toolkit that integrates different functionalities spanning from cryptography to notification mechanisms to support the implementation of privacy-aware systems.

Concerning the security aspect, we identify two important approaches, i.e., *SQUARE* and Microsoft SDL (which incorporated *SQUARE*). Security Quality Requirements Engineering (*SQUARE*) is a 10-steps sequential process for considering the security interests of all the stakeholders of the software system (Mead and Stehney 2005). It also involves threat agent identification and risk analysis for effective requirements engineering. In Secure Development Lifecycle (*SDL*) the core idea is that, to build a secure software system, an important aspect is to consider how an attacker might compromise the system by exploiting design flaws and, as a consequence, to build the necessary defense mechanisms in the system (Adam Shostack 2014). So, threat modeling plays a key role and *SDL* has integrated a systematic approach for security threat modeling using *STRIDE* (Microsoft).

An example of an approach facing both privacy and security at the same time is Restricted Misuse Case Modeling (*RMCM*) (Mai et al. 2018). It is a 4-step method whose output is a misuse case diagram, use case specifications, security use case specifications, misuse case specifications, and mitigation schemes. This method enables engineers to identify security threats and countermeasures but considers out-of-scope risk analysis, i.e., ranking and prioritizing of the resulting security threats.

It is worth mentioning that none of the presented works proposes an automatic tool to go through the phases of each approach or focuses on usability aspects. Our paper contributes to this research area by proposing a framework that helps developers, designers and engineers include privacy features in their software systems through the adoption of PDPs. The selection of the PDPs can start from the GDPR articles, the ISO 9241-210 phases, or security and

privacy vulnerabilities. In the following, the most important concepts behind the ongoing research are detailed. In particular, the most important aspects of source code vulnerabilities, privacy by design principles, privacy strategies, and privacy design patterns, are discussed with reference to the literature.

2.2 Code Vulnerabilities, Privacy by Design Principles and Privacy by Design Strategies

Cybersecurity vulnerabilities can be detected by adopting techniques like static analysis, automatic penetration testing and manual penetration testing. The current implementation of our framework considers vulnerabilities derived from static code analysis. They can be divided into *security vulnerabilities* and *privacy vulnerabilities*.

Security vulnerabilities are categorized according to *OWASP Top 10 2021* standard: this is based on data and information provided by firms specialized in application security or collected by using industry surveys. Its goal is to provide knowledge and information on the most common and important application security weaknesses. The vulnerabilities are divided into ten categories: A1 Broken Access Control, A2 Cryptographic Failures, A3 Injection, A4 Insecure Design, A5 Security Misconfiguration, 6 Vulnerable and Outdated Components, A7 Identification and Authentication Failures, A8 Software and Data Integrity Failures, A9 Security Logging and Monitoring Failures, A10 Server-Side Request Forgery (SSRF) (OWASP).

Privacy vulnerabilities are grouped by tools for static code analysis like Fortify SCA into four macro-categories, each one mapped with a single GDPR article. In particular, these categories are Access Violation, Indirect Access to Sensitive Data, Insufficient Data Protection, and Privacy Violation¹.

As proposed in (Baldassarre et al. 2019) these vulnerabilities can be mapped with Privacy by Design principles. Privacy by Design (PbD) is a methodology proposed in 1995 by Ann Cavoukian. This methodology consists of seven principles, each of which specifies actions and responsibilities for assessing “Privacy by Design Compliance” (Cavoukian 2012). An example of a PbD principle is *Proactive not Reactive*, which says that privacy threats must be anticipated and prevented while developing and/or running a system, rather than just reacting to privacy breaches once they have occurred.

Each of these principles can be further mapped with one or more Privacy Strategies (Baldassarre et al. 2019). A privacy design strategy (PDS) is defined as an approach to achieve some level of privacy protection. In this study, the Hoepmann’s PDSs are considered (Hoepman 2014). These strategies are divided into two categories:

- 1) Data-oriented, and the strategies are: *minimize*, to reduce to the minimum possible the amount of data collected and processed; *separate*, to (physically and/or logically) separate data processing and storage; *abstract*, to limit the level of detail of processed data; *hide*, to protect personal data from unauthorized third parties;
- 2) Process-oriented, and the strategies are: *inform*, to notify users in an exhaustive yet simple way about the whole data processing lifecycle; *control*, to provide users full control over their data; *enforce*, to implement privacy-friendly data processing; *demonstrate*, to prove the enforcement of compliant data processing.

2.3 Privacy Design Patterns

Generally speaking, a design pattern provides knowledge collected by experts in a specific field; this knowledge is provided in a structured, documented, and reusable manner (Colesky, Hoepman, and Hillen 2016) and helps practitioners build information system. In a cybersecurity context, PDPs address and provide a common solution to privacy problems. They can be seen as a way to translate “privacy-by-design” into practical solutions for software engineering: they help improve the (re-)engineering process by describing classes, collaborations between objects, and their purposes, but also can help designers identify and address privacy concerns during the initial phases of the Software Development Life Cycle (SDLC).

In our study, we considered the PDPs resulting from a joint research work between the U.S. Department of Homeland Security and the National Institute of Standards and Technology. The entire set of patterns is published at <https://privacypatterns.org/>. It consists of 72 privacy patterns grouped by 7 PDSs (control, abstract, separate, hide, minimize, inform, enforce) described in terms of the following dimensions: context, problem, solution, consequences, and examples.

¹ <https://www.microfocus.com/it-it/products/static-code-analysis-sast/overview>

3 A FRAMEWORK TO SELECT PRIVACY DESIGN PATTERNS STARTING FROM GDPR

As reported in the Introduction, this study is part of a broader research that aims to help developers, designers and engineers (stakeholders in the following) include privacy features in their software systems. The main contribution to this research is the MATERIALIST (Mapping dATa rEgulation softwaRe lfcycle And vuLnerabilitieS paTterns) framework summarized in Figure 1. The next subsections present the overall MATERIALIST framework (Section 3.1) and how it guides stakeholders in choosing PDSs starting from GDPR articles (Section 3.2).

3.1 The MATERIALIST Framework

The core of this framework is represented by the set of 72 privacy design patterns reported at <https://privacypatterns.org/>. In addition, to make the adoption of these patterns more concrete, we are also working on their extension through architectural and user interface patterns that guide more practically and less ambiguously developers in the creation of code (architectural patterns) and designers in the design of user interfaces (interface patterns). For sake of simplicity, in the following, PDPs refer to the extended set of the original 72 patterns.

The underlying idea is to guide stakeholders in selecting such PDPs starting from three different entry points: i) GDPR articles, ii) ISO 9241-210 phases and iii) privacy vulnerabilities discovered during static code analysis. The use of these PDPs aims to solve two of the four issues mentioned in the Introduction, specifically *different views about privacy* and *lack of privacy and security knowledge*. Indeed, PDPs are intrinsically a standardized language for privacy aspects, thus they contribute to providing the same view on privacy. Lack of knowledge is addressed since patterns have been created by privacy experts and provide robust and practical solutions.

In the MATERIALIST framework, we considered the GDPR instead of other regulations because it is the main privacy regulation in the European Union, where the authors of the paper live. The ISO 9241-210 has been considered since, differently from other software development processes, it is human-centered and thus contributes to developing user-centric software systems. This choice aims to mitigate the issue *implementations are not user-*

centric. The vulnerabilities are the ones provided by OWASP Top 10 2021, which is one of the most important standard de-facto in this field.

A peculiarity of this framework is that ‘traversing’ is also allowed between any entry points passing through the privacy patterns. For example, a stakeholder may require the list of the GDPR articles violated by a specific vulnerability and vice versa; or, a stakeholder may need to know which are the ISO 9241-210 phases that can trigger some vulnerabilities and vice versa. This flexible traversing guarantees the inclusion of PDPs both when the development process starts from scratch (forward engineering) and in case of re-engineering of a software system (backward engineering).

In forward engineering, the entry points are the GDPR and the ISO 9241-210. In the former case, PDPs are suggested to comply with the GDPR articles. In the latter case, to support the stakeholders along with all the ISO 9241-210 phases, patterns are suggested for each phase.

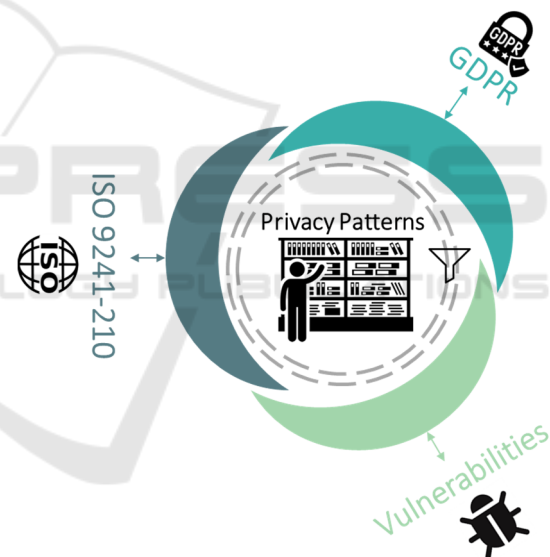


Figure 1: The MATERIALIST framework guides stakeholders in selecting the right privacy design patterns.

In backward engineering, the re-engineering software process may start from a scan performed by static code analysis tools such as SonarQube and Fortify SCA, which detect common security and privacy vulnerabilities (i.e., OWASP Top 10). Starting from a specific vulnerability, the framework suggests the PDPs that, when re-engineering the software, help mitigate such vulnerability. At the current stage, the MATERIALIST framework covers the mapping between the GDPR articles and the PDPs. In the next section, we detail the process we

followed to perform this mapping activity, which represents the main contribution of this paper. The other mappings between vulnerabilities with patterns and between ISO 9241-210 with patterns are still in progress; an overview of the results of these mappings is reported in Section 4.

3.2 Mappings between the GDPR and Vulnerabilities

In this paper, we focused on the forward engineering that starts from the GDPR entry point up to PDPs. This side of the framework addresses one of the current limitations reported in the Introduction, i.e., *data protection regulations only provide legal indications*. To this end, PDPs come in handy as a way to guide the design of systems by proposing a common and reusable solution to common privacy problems. However, PDPs alone do not provide any indication of how to comply with the GDPR, leaving developers with no clear references on the regulation. Moreover, the GDPR is composed of 99 articles, most of which are about purely legal and bureaucratic aspects.

To fill this gap and guide stakeholders in selecting the right privacy design patterns according to the GDPR articles they must be compliant with, we performed a systematic mapping between the 72 privacy patterns and the GDPR articles. This activity has been conducted by two researchers that are experts in both GDPR and PDPs. In addition, to increase robustness and reduce the possibility of biases, this mapping has been performed starting:

1. from each PDP towards the GDPR articles;
2. from each GDPR article towards the PDPs.

The two researchers started by performing independently the mapping phase. Each of them spent around 40 hours for this phase. Then, they compared their results and the reliability value was 67%; thus, they discussed the differences and reached a full agreement on the remaining mappings.

This process led to the identification of different relationships between each privacy pattern and 14 different articles of the GDPR, thus providing a clear indication of which PDPs can support compliance with what article. Three patterns, namely “Incentivized Participation”, “Pay Back” and “Reciprocity” are left without any assignation because the GDPR does not incentivize any remuneration in return for data shared by users. The result of the mapping represents the main contribution of this work and, due to the huge amount of content, the report of this mapping is reported as a Web page available at <http://90.147.170.155/mapping.html>.

Stakeholders can exploit this table, as part of the overall framework, to choose the right PDPs starting from the GDPR articles they must be compliant to.

4 TOWARD THE EXTENSION OF MATERIALIST

As reported before, as a work in progress, we are extending the MATERIALIST framework by associating PDPs with both vulnerabilities and ISO 9241-210 phases. We are also augmenting MATERIALIST by extending the 72 PDPs with more concrete architectural and UI patterns; however, a lack of space limits the presentation of these details. In the next sub-sections, we provide an overview of the mappings with ISO 9241-210 and vulnerabilities. We also present the adoption of user stories to furtherly filter PDPs selected starting from any entry point.

4.1 Mappings with Vulnerabilities and ISO 9241-210

The mapping between vulnerabilities and PDPs is inspired by the work of Baldassarre et al. (Baldassarre et al. 2019): we are performing the same mapping by considering the updated *OWASP Top 10 2021* instead of the *OWASP Top 10 2017* used in the existing mapping. It is worth noticing that there is not a direct mapping between vulnerabilities and patterns, indeed vulnerabilities are mapped first with the PbD principles, which are mapped with PDSs, and are in turn mapped with the PDPs. Table 1 reports an extract of the results of this mapping. In particular, we can see five OWASP Top 10 2021 vulnerabilities (A1, A2, A3, A7, A8) mapped with the same PDPs.

Table 1: Mapping between some OWAPS Top 10 2021 vulnerabilities and privacy design patterns.

Vulnerability	Privacy Design Pattern
A1 - Broken Access Control	Abridged Terms and Conditions
A2 - Cryptographic Failures	Ambient Notice
A3 - Injection	Appropriate Privacy Icons
A7 - Identification and Authentication Failures	Asynchronous notice
A8 - Software and Data Integrity Failures	

Regarding the ISO 9241-210, we are also carrying out a mapping between all the process phases and each PDP. The ISO 9241-210 proposes a complementary and iterative process, divided into an initial planning phase and 4 iterative phases (understand and specify the context of use; specify the user requirements; produce design solutions to meet user requirements; evaluate the designs against requirements) that are cycled until the solution satisfies all the user requirements. To this end, the mapping aims to identify, during the development process, when stakeholders should start considering the implementation of a PDP. This is necessary because security must be considered as a process throughout the whole SDLC. Although, one may think that patterns, being practical solutions, may be considered only during the coding phase, their implementation heavily depends on the context and on the requirements of the system. As an example, the *Data Breach Notification*² pattern should be considered from the requirements phase as it is mandatory, by the GDPR, that the authorities are notified of the incident within 72 hours. Hence, specific requirements must be set to duly implement this pattern and thus comply with the regulation. On the other hand, a pattern such as the *Aggregation gateway*³, which deals with more architectural and technical aspects instead, belongs to the design production phase.

In this way, having a clear view of when engineers should start considering a pattern, helps in easing the secure implementation of the system. Moreover, the adoption of a user-centered development process such as the ISO 9241-210 further supports the implementation of more usable approaches in security, as usability is frequently overlooked in security and privacy features (Jakobi et al. 2019) (Alpers et al. 2018), thus solving the aforementioned issue.

4.2 Mappings with User Stories

Traversing the MATERIALIST framework from any entry point to PDPs can undoubtedly simplify the work of the stakeholders, also avoiding choosing the wrong PDPs due to missing or different knowledge on privacy. However, sometimes this traversing leads to the selection of several patterns, which have to be further refined by the stakeholders if not properly guided. For example, starting from the Article 5 of the GDPR, 23 patterns can be used and the selection of the right one(s) is in charge of the stakeholders, which

has not furthered elements to guide the choice.

To assist the selection of relevant patterns, we defined an intermediate layer between the three entry points and the PDPs. This layer consists of a set of common use case scenarios (e.g., user registration on a website, user deletion) mapped with the PDPs. We started from a set of user stories proposed by the *Agency for Digital Italy* (AgID), which is the technical agency of the Presidency of the Council of Ministers (AgID). Such scenarios represent the majority of the situations that can cause privacy and security vulnerability.

To perform this map, an analysis of the privacy patterns was made to identify which PDPs are involved in each user story and can be implemented to provide a secure and GDPR-compliant procedure (e.g., secure credentials storage during registration). As an example, we can consider the *users' registration* scenario; this is an interactive activity where users are required to input their data, choose a secure password and possibly make some choices regarding how to share their data with the website or any other related third-party services they are registering with (e.g., share the email with a third-party service for marketing purposes). In this context, the designers not only have to design a privacy-aware component but also they have to work on the front-end to provide an intuitive and clear user interface that can guide users during the whole registration process and can help take appropriate decisions, exactly in line with what GDPR asks for.

The PDPs related to this scenario are proposed by the MATERIALIST framework, for example, the *Protection against Tracking* and *Strip Invisible Metadata*. In particular, these are the suggested PDPs:

PDP: Protection against Tracking

SUMMARY: Do not collect unnecessary cookies, especially if they are useful only in the future and not at the moment

USER STORY: Collecting cookies during the registration

NOTES: If it is necessary to collect cookies for system functionality, collect only those strictly necessary and not the optional ones (i.e. technical and profiling ones). For each cookie used, keep correspondence in the DB between the cookie and the user

PDP: Strip Invisible Metadata

SUMMARY: At least warn the user of what he is sharing and give him a chance to rectify it

² <https://privacypatterns.org/patterns/Data-breach-notification-pattern>

³ <https://privacypatterns.org/patterns/Aggregation-gateway>

USER STORY: Metadata collection during registration

NOTES: If it is necessary to collect metadata for system functionality, collect only what is strictly necessary and inform the user of all information that can be deduced from the collected metadata.

The stakeholders that start from an entry point thus can further refine the selection of PDPs considering the scenarios they have to cover in their system. As for the ISO 9241-210 and for the vulnerability this is a work in progress.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we presented MATERIALIST, a framework, which supports secure and privacy-aware forward and backward software engineering processes with a user-centric approach. Specifically, we focused on developing the MATERIALIST component that guides the selection of PDPs starting from the GDPR articles. This framework is going to be extended to facilitate the selection of PDPs also starting from the ISO 9241-210 phases, as well as from privacy and security code vulnerabilities. Some results of these extensions are already reported in this paper.

To improve the selection of PDPs we are also working on the definition of intermediate layers that further guide the choice of PDPs selected starting from any entry point. Further approaches based on heuristics and metrics are also under development to suggest proper PDPs also depending on the context of use. We are also working on the extension of the 72 PDSs by defining, for each of them, architectural patterns and User Interface patterns, which better drive designers and developers during software development activities like coding and design.

REFERENCES

- AgID "Linee Guida per Lo Sviluppo Del Software Sicuro|Agenzia per l'Italia Digitale." (<https://www.agid.gov.it/it/sicurezza/cert-pa/linee-guida-sviluppo-del-software-sicuro>).
- Alpers, S., Oberweis A., Pieper M., Betz S., Fritsch A. 2018. "PRIVACY-AVARE: An Approach to Manage and Distribute Privacy Settings". In *IEEE International Conference on Computer and Communications (ICCC 2017)*, pp. 1460-1468.
- Baldassarre, M.T., Barletta V.S., Caivano D., and Scalera M. 2019. "Privacy Oriented Software Development". In *International Conference on the Quality of Information and Communications Technology*, pp. 18-32. Springer, Cham.
- Brodin, M., 2019. "A Framework for GDPR Compliance for Small- and Medium-Sized Enterprises." *European Journal for Security Research*
- Cavoukian, A., 2012. "Operationalizing Privacy by Design: A Guide to Implementing Strong Privacy Practices."
- Colesky, Michael, Jaap Henk Hoepman, and Christiaan Hillen. 2016. "A Critical Analysis of Privacy Design Strategies". In *IEEE Symposium on Security and Privacy Workshops (SPW '16)*, pp 33–40.
- Hjerpe, K., Ruohonen J., and Leppänen V. 2019. "The General Data Protection Regulation: Requirements, Architectures, and Constraints". In *IEEE International Conference on Requirements Engineering*. Vols. 2019-September.
- Hoepman, J.H. 2014. "Privacy Design Strategies." Pp. 446–59 in *IFIP Advances in Information and Communication Technology*. Vol. 428.
- Jakobi, T., Patil, S., Randall, D., Stevens, G., & Wulf, V. 2019. It is about what they could do with the data: A user perspective on privacy in smart metering. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(1), 1-44.
- Jensen, Carlos, Joe Tullio, Colin Potts, and Elizabeth D. Mynatt. 2005. "STRAP: A Structured Analysis Framework for Privacy." *Technology*.
- Lioudakis, Georgios V., et al. "Facilitating GDPR compliance: the H2020 BPR4GDPR approach." *ION Conference on e-Business, e-Services and e-Society*. Springer, Cham, 2019.
- Mai, P. X., Goknil, A., Shar, L. K., Pastore, F., Briand, L. C., & Shaame, S. (2018). Modeling security and privacy requirements: a use case-driven approach. *Information and Software Technology*, 100, 165-182.
- Martin, Y. S., & Kung, A. 2018. Methods and tools for GDPR compliance through privacy and data protection engineering. In *IEEE European symposium on security and privacy workshops (EuroS&PW)*, pp. 108-111.
- Mead, N. R., & Stehney, T. 2005. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4).
- Sobolewski, M., J. Mazur, and M. Paliński. 2017. "GDPR: A Step towards a User-Centric Internet?" *Intereconomics*. 52(4), 207-213.
- Shostack, A. 2014. *Threat Modeling: Designing for Security* (1st. ed.). John Wiley & Sons. <https://dl.acm.org/doi/10.5555/2829295>