Land of the Lost: Privacy Patterns' Forgotten Properties

Enhancing Selection-Support for Privacy Patterns

Ala'a Al-Momani
Institute of Distributed Systems, Ulm University
Ulm, Germany
alaa.al-momani@uni-ulm.de

Kim Wuyts
imec-DistriNet, KU Leuven
Leuven, Belgium
kim.wuyts@cs.kuleuven.be

Laurens Sion
imec-DistriNet, KU Leuven
Leuven, Belgium
laurens.sion@cs.kuleuven.be

Frank Kargl
Institute of Distributed Systems, Ulm University
Ulm, Germany
frank.kargl@uni-ulm.de

Wouter Joosen
imec-DistriNet, KU Leuven
Leuven, Belgium
wouter.joosen@cs.kuleuven.be

Benjamin Erb
Institute of Distributed Systems, Ulm University
Ulm, Germany
benjamin.erb@uni-ulm.de

Christoph Bösch
Institute of Distributed Systems, Ulm University
Ulm, Germany
christoph.boesch@uni-ulm.de

ABSTRACT

Privacy patterns describe core aspects of privacy-enhancing solutions to recurring problems and can, therefore, be instrumental to the privacy-by-design paradigm. However, the privacy patterns domain is still evolving. While the main focus is currently put on compiling and structuring high-quality privacy patterns in catalogs, *the support for developers to select suitable privacy patterns is still limited.* Privacy patterns selection-support means, in essence, the quick and easy scoping of a collection of patterns to the most applicable ones based on a set of predefined criteria. To evaluate patterns against these criteria, a thorough understanding of the privacy patterns landscape is required. In this paper, (i) we show that there is currently a lack of extensive support for privacy patterns selection due to the insufficient understanding of pattern properties, (ii) we propose additional properties that need to be analyzed and can serve as a first step towards a robust selection criteria, (iii) we analyze and present the properties for 70 privacy patterns, and (iv) we discuss a potential approach of how such a selection-support method can be realized.

CCS CONCEPTS

• **Security and privacy** → Software security engineering; • **Software and its engineering** → *Software design engineering*; *Design patterns*;

KEYWORDS

Privacy Patterns, Privacy Engineering, Software Design.

ACM Reference Format:
Ala'a Al-Momani, Kim Wuyts, Laurens Sion, Frank Kargl, Wouter Joosen, Benjamin Erb, and Christoph Bösch. 2021. Land of the Lost: Privacy Patterns' Forgotten Properties: Enhancing Selection-Support for Privacy Patterns. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3412841.3441996

1 INTRODUCTION

The privacy-by-design (PbD) paradigm encourages the integration of privacy-enabling principles and privacy-enhancing technologies throughout the entire software development lifecycle [4, 17]. In the design phase, this PbD approach can be facilitated by applying privacy patterns, which describe the core of privacy-enhancing solutions to reoccurring problems. The use of patterns, however, is not a new concept in software engineering. Gamma et al. [16] described *software design patterns* which are commonly consulted and applied when designing and implementing software systems. When the potential benefits of design patterns became apparent, the concept was also applied to other aspects of software engineering. Security patterns were proposed and studied extensively in the literature [15, 34, 35] to provide well-documented solutions to recurring security problems. Analogously, privacy patterns have emerged, which target privacy-specific aspects and are still under continuous development [1, 2]. Several privacy patterns have been introduced in the literature such as the patterns proposed by Romanosky [33], Hafiz [18], and Thomborson [38]. Furthermore, dedicated online repositories [1–3] compile collections of such privacy patterns.

Despite the tremendous effort throughout the past years to populate and structure the collection of privacy patterns [8–10, 21], *there is currently a lack of concrete guidance to select the most suitable privacy pattern for a specific scenario.* Such selection-support depends on two main factors: 1.) the *context* of the system in which patterns is deployed, and 2.) the *properties* of the patterns themselves to help scoping the results with respect to a set of selection criteria. Introducing a selection support based only on the *contextual* information is unsatisfactory unless the *properties* of the patterns are synthesized in the overall process. Therefore, it is important to have a thorough understanding of the relevant properties of the patterns. However, an extensive analysis of the privacy patterns landscape and their properties is yet missing, and this is what we address in this paper as a first step toward introducing a holistic selection support method.

Currently, developers and software engineers need to browse and investigate a long list of privacy patterns along with their description to filter applicable ones and then select the best-fitting privacy pattern rather than being able to make a quick, yet well-informed, decision. Another drawback of lacking a guidance method for selecting privacy patterns is the inability to systematically justify the selection of specific privacy patterns while discarding others. This is particularly relevant when software engineers document their work for demonstrating compliance with certain regulations such as the GDPR.

In this paper, we (i) highlight the lack of extensive selection support for privacy patterns due to the insufficient understanding of their properties in Section 3, (ii) suggest a set of privacy pattern properties as basis for such selection support in Section 4, (iii) analyze 70 privacy patterns that are all the patterns found in a well-known privacy patterns catalog [2] with respect to these properties in Section 5, and (iv) discuss a potential approach of how such a selection-support method can be realized in Section 6.

2 BACKGROUND & RELATED WORK

Security Patterns. Before investigating literature on privacy patterns, we first look at related works on security patterns which have been studied extensively [15, 20, 34, 36, 43]. Here, several selection-support approaches have already been introduced [6, 29, 34, 41]. A key enabler to support the selection of appropriate security patterns is understanding their properties and then establish suitable categorizations [28, 34]. A simple approach to classify security patterns can be based on the fundamental security objectives; confidentiality, integrity, and availability. Hafiz et al. [20] demonstrated that such an approach places most of the patterns into multiple categories at the same time, rendering the selection process ineffective. Scandariato et al. [34] suggested enhancing this by including additional objectives, such as authorization and authentication, and additional classification aspects based on the development phase. Particularly, security patterns could be classified into application patterns (which are further classified into architecture and design patterns) or system patterns. A noteworthy observation from this study is pointing out the impact each security pattern has on supporting security qualities, such as privacy or integrity, and impeding non-security qualities, such as usability or performance. Additionally, Fernandez et al. [14] also pointed out that security patterns can be defined at several levels of abstraction, and have different properties in terms of, e.g., the architectural layers where the patterns belong, and the security concerns considered by the patterns. From a goal-oriented requirements perspective, Weiss et al. [41] proposed a selection approach for security patterns that fulfill security requirements. Furthermore, Li et al. [25] proposed a method for integrating security patterns with and select them for security requirements. In particular, they combined properties of the security patterns and other contextual information, and provided contextual goal models of security patterns which can then be matched with corresponding patterns. The tackled analysis of security patterns provided an adequate understanding of their properties and enabled introducing guidance for selecting appropriate patterns. We still lack a full understanding of privacy patterns properties, resulting in a very limited selection-support for them.

Privacy Patterns. Over the past few years, more and more privacy patterns have been suggested in the literature [18, 19, 22, 33, 37, 38], and several online resources were established to collect privacy patterns [1, 2]. Currently, privacy strategies are the most common approach to classify privacy patterns. Privacy strategies represent privacy-protection quality attributes and should be considered at an early stage of the system development. According to Hoepman [21], these strategies are *hide, minimize, separate, abstract, control, inform, demonstrate,* and *enforce*. This classification is, thus, based on the patterns' fundamental approaches to protect privacy. Privacy tactics [10] form another level of abstraction — more specific than strategies, but contributing to the their overarching goal. The nature of privacy patterns is even more detailed and forms a level of abstraction below tactics but still above privacy-enhancing technologies (PETs) which represent concrete implementations of (parts of) the patterns. For instance, the strategy *hide* can be achieved using four tactics; namely, *restrict, mix, obfuscate,* and *dissociate* [10]. Patterns exist to achieve each tactic goal and therefore, the overarching strategy goal. For example, the patterns *onion routing* and *anonymity set* are placed under the tactic *mix* [2]. Figure 1 shows the corresponding hierarchy of strategies, tactics, and patterns for the *hide* strategy. Additionally, Colesky et al. [8, 9] described relationships for privacy patterns of both the *control* and the *inform* strategies: *uses, leads to, refines, similar to, alternative to*, and *complements*. Apart from that, Pape and Rannenberg [30] demonstrated the applicability of privacy patterns — stated in the common catalog we consider in our work [2] — to the IoT domain and, particularly, the smart vehicles scenario. However, the applicability of privacy patterns, in general, is far from trivial [7]. Caiza et al. [7] addressed this issue and provided a roadmap to enhance privacy patterns' applicability. In particular, they distinguished between the process of introducing a privacy pattern, and the process of applying a privacy pattern. They pointed out that one of the crucial aspects when applying privacy patterns is pattern-selection which has, among

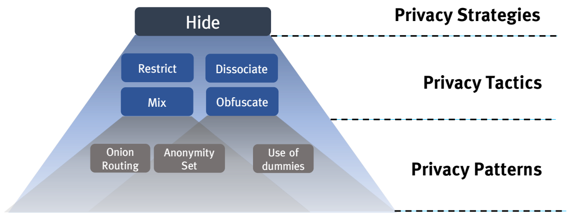

Figure 1: A snippet of the hierarchy of strategies, tactics, and patterns.

many other aspects, limited guidelines and resources. Apart from that, a different approach to introduce privacy patterns is taken by Meis and Heisel [27]. They proposed a pattern-based representation of PETs with the goal of having a common structure for the description of PETs, and therefore support the selection of PETs-based patterns.

(Privacy) Requirements Engineering. It is also worth looking at the broader privacy engineering spectrum. Kalloniatis et al. [23, 24] introduced the PriS method, a privacy requirements engineering method. PriS takes both privacy and business goals into account, and investigates their effect on the organizational processes by suggesting new processes, modifying existing ones, and removing others. PriS identifies a number of implementation techniques to realize such privacy-related processes but lets the developers select which of these techniques are best suited without extensive support for this selection. In terms of PriS, our proposal of a selection-support method for privacy patterns assists in selecting the patterns that describe the suitable processes to achieve a certain (combined) privacy- and enterprise-goal. Other related studies from the requirement engineering literature include Liu et al. work [26], and Lamsweerde's work [39]. Such work mainly focuses on eliciting system requirements, but with much less effort taken to *guide the selection of appropriate measures* to realize such requirements.

Selection Support for Privacy Patterns. Regarding the selection of privacy patterns, only limited resources are available. Drozd [13] proposed a catalogue that classifies a selection of privacy patterns according to the ISO/IEC 29100 standard's principles. The tool suggests all privacy patterns that meet or capable of achieving a certain principle. However, the selection is not supported by extensive patterns' properties nor various angles of the contexts of the patterns. Another work which specifically addresses the issue of selecting privacy patterns is done by Pearson and Shen [31]. Their work relies primarily on the contextual information of the system as a selection base for privacy patterns. However, applying their framework on the current large number of privacy patterns available nowadays (e.g., [1, 2]) seems to be a non-trivial task — especially, when the properties of all these patterns are not yet fully understood. Despite that the contextual information is crucial to select appropriate patterns, the properties of the patterns themselves are also of equal importance and should, therefore, be well-understood and combined with the contextual information to form a holistic selection method. Compared to the selection of security patterns, supporting the selection of privacy patterns can thus be considered to be still in its infancy and does not cover a wide range of properties yet. The observation of lacking extensive resources that support the holistic selection of privacy patterns has also been pointed out lately by Caiza et al. [7].

3 A CRITICAL DISCUSSION OF PRIVACY PATTERNS

With the increasing number of privacy patterns, selecting a suitable privacy pattern for a certain privacy requirement in a specific context becomes very challenging, especially, because of the limited support and guidance for selecting such patterns. We consider the lack of adequate knowledge of privacy patterns properties a major reason for the failure of appropriate selection procedures. A privacy pattern selection-support method entails offering a criteria to easily and quickly scope and align the set of applicable patterns. In order to point out suitable patterns, classification of privacy patterns based on their properties is required. Such properties should address, e.g., what privacy patterns are capable of fulfilling and what abstraction layer they apply to, which — together with the contextual information — are then used to guide the selection toward a specific pattern. While previous work on privacy patterns [10, 21] provides a significant advancement in the understanding of privacy patterns, we argue that this is still insufficient and additional research is required to define and foster the understanding of the *properties* of privacy patterns in order to introduce a simplified, easy-to-use, and validated privacy pattern-selection method.

3.1 Privacy Strategies as Main Selection Criterion

The online repositories for privacy patterns [1, 2] are currently all categorized according to privacy strategies [21]. This is however not perfectly adequate for selection. The main reason behind this inadequacy is the coarse-grained and high-level nature of privacy strategies. We now elaborate on this further. For selection, ideally, every pattern would fall under a specific tactic and a specific strategy, but in practice, some privacy patterns naturally fall under multiple privacy strategies at the same time. While this is generally a positive remark that a pattern achieves the goal of multiple strategies simultaneously, it creates a two-fold issue for the selection which we explain here. Classifying privacy patterns into their strategies may take two approaches: 1.) to place a pattern into its primary overarching strategy *only*, or 2.) to place a pattern into *multiple* orthogonal strategies. The *first approach* leads to not including some relevant patterns under a certain strategy because their overarching goals fit better into other strategies. The *second approach* leads to the case where each strategy would consist of a long list of privacy patterns that need to be individually examined for the selection, yet, without guidance. An example of the first approach is the following: Assigning the pattern *encryption with user-managed keys* (also see Section 4) only to the *control* strategy, but not to the *hide* strategy [2] — despite that in many cases, this pattern achieves the actual goal of the *hide* strategy. An example of the second approach is: placing this pattern under both the *control* and the *hide* [2] strategies, which together with other patterns, result in a long list of, e.g., the *control* strategy's patterns.

At this point, one may think that privacy tactics, which has a lower level of abstraction than the strategies, would solve the previous issues. Privacy tactics, in fact, solve the drawback of the coarse-grained strategies in general but are still nonoptimal for a selection support to be based on them. The reason is that the first selection is still made based on the privacy strategy, which can be both challenging and limiting. You already need to have sufficient privacy expertise to determine which strategy is best suited to address a certain issue, and then which tactic. Furthermore, privacy tactics still fall under similar categorial aspect like strategies and do not take additional (orthogonal) aspects and properties into account. Several patterns might not apply to a specific problem due to certain properties; for instance, a pattern related to a communication channel may not be applicable for a database privacy issue. These patterns should, therefore, be discarded early on in the selection process and neither strategies nor tactics address this. There is, thus, a need for additional properties to be included in the selection criteria.

Apart from that, additional assessment in selecting privacy patterns is provided through *tags* [2]. Such labels provide additional references, e.g., to the context of the system (such as *cloud* scenarios), or the goal of the pattern (such as *anonymous communication*, or *routing*). At the time of writing, 36 tags have been used in the pattern repository [2]. As tags usually represent unstructured and evolving taxonomies, it is unclear on what basis such tags are instantiated as a categorization aspect. It is, therefore, crucial to properly and systematically understand the properties and features of privacy patterns, and base such a tagging system on them.

3.2 The Need for Additional Selection Criteria

When selecting privacy patterns, software developers often look for patterns with certain properties and features beyond only knowing their strategies or tactics. For instance, some developers might particularly look for architectural solutions to certain privacy risks while others might look for a lower-level design solution that does not require architectural changes because, e.g., the architecture of the application has been already set. Additionally, developers may also look for patterns that apply to data regardless where its presence in the system is. Such data-focus patterns can, for example, be applied where data enters the systems and do not require system-wide changes. Other developers may look for patterns that target processing of the data in the system. Furthermore, developers may also look for patterns that apply at certain parts of the system, e.g., the (user)-(application) interaction, but not at other parts, e.g., (service provider)-(third party) interaction. As patterns, in general, bring different properties and features, privacy patterns should do so as well. In other words, privacy patterns should have, e.g., different levels of abstraction, applicability-scope, impact on the system's attributes, and entail different fundamental approaches to be instantiated and deployed.

Concerning privacy patterns, a thorough discussion of such properties cannot be found in the literature yet. As this understanding is mandatory to improve the selection process of privacy patterns, we next propose a minimal set of properties that form the basis for an adequate selection-support method.

4 PRIVACY PATTERNS PROPERTIES AS BASIS FOR A SELECTION CRITERIA

Section 3 highlighted that one of the main reasons for lacking selection support of privacy patterns is the limited understanding of their properties and characteristics to guide the selection. Additional analysis and investigation are thus required to understand these properties. In this section, we describe the minimal set of properties that we believe is required to provide additional input and support for a suitable selection criteria.

Previous analyses of security patterns [14, 34] showed that such patterns have different properties in terms of, e.g., their abstraction level according to the development phases of a system lifecycle, which supported the selection process of fitting security patterns. We consider this while pointing out and discussing further additional classification aspects that can be applied as useful selection criteria. Those classification aspects form *dimensions* in a multidimensional graph representation of the privacy patterns. Each of these dimensions contains a set of values that represent *properties* privacy patterns may have. Therefore, the *property* of a privacy pattern is the relation between a privacy pattern and a certain dimension. We note that each pattern may take one or more values of a certain dimension. The concept of dimension graph to describe patterns in multidimensional space has been introduced by Washizaki et al. [40] and used for security patterns by Fernandez et al. [14]. Concerning privacy patterns, we propose the following initial set of dimensions and properties to be used as a potential selection criteria. These dimensions and properties are proposed based on 1.) the literature review tackled for security patterns' properties as discussed in Section 2, and 2.) our analysis of privacy patterns upon selecting them to solve certain privacy risks. We note that such a list is not exhaustive. Additional research is required to determine how useful each dimension and property is for the sake of supporting the selection.

Applicability Scope. Privacy patterns can be instantiated at and be targeting the application's *overall architecture* (as a higher abstraction level) or the application's *implementation-specific* aspects (as a lower abstraction level). In the field of security patterns, a similar distinction has been made between application architecture and application design [34] in a dimension of development phase. However, such terms used in this study [34], i.e., application architecture and application design, may have an overlapping intuition from a software engineering perspective, as developers actually design an architecture for the application. Therefore, we decide to use a different terminology. Hence, we form the applicability scope dimension to take the following values as properties: {*Overall Architecture, Implementation-specific*}. An example of a privacy pattern that applies to the overall architecture is the *aggregation gateway* pattern [2] which frequently requires deploying extra third party components to, e.g, compute aggregated values over a group of users. An example of a privacy pattern that targets implementation-specific aspects is *privacy-aware wording* [2] which mainly deals with how privacy policies should be presented to users, not requiring any architectural modifications.

Privacy Objective. A sensible requirement for a privacy pattern — in order to be recognized as a *privacy* pattern — is contributing to, at least, one privacy objective. Therefore, it is crucial to analyze privacy patterns in terms of distinct privacy protection goals; i.e., the properties of this dimension, which are: {*Anonymity, Unlinkability, Confidentiality, Plausible Deniability, Undetectability, Manageability, Intervenability, Transparency*} [5, 12, 32]. We note here that many classifications of privacy objectives such as, e.g., [12], consider each of compliance, transparency, and intervenability as standalone privacy objectives despite the fact that transparency and intervenability are actually aspects of compliance. Furthermore, the compliance objective differs based on the regulation or law the system is required to comply with. In our work, we consider the privacy objectives, e.g., manageability, intervenability, and transparency, that represent most factors based on which compliance requirement is usually formed. The manageability objective deals with managing risks and obligations with the authorities and stakeholders. The rest of the objectives are defined similarly to previous work in this regard [12, 32]. Apart from that, while the privacy objectives are partially addressed by the *tags* of the online repositories of privacy patterns [1, 2] as we mentioned earlier, additional structure of these tags is required.

Qualities. An additional and critical aspect to be taken into account while analyzing privacy patterns is the impact of a pattern on the functional and non-functional properties of a software system. This dimension includes {*Performance Impact, Complexity of Interaction, Utility, Security*} as properties privacy patterns may impact the system with. That is, a pattern may affect the performance (e.g., latency) of the application, increase the complexity of interaction required from end users, decrease the utility of the application, or impact other security goals envisioned in the application. Such properties play a vital role in selecting proper privacy patterns. For instance, if we consider low-latency as a critical quality attribute for an application, implementing certain patterns such as *onion routing* [2] to achieve anonymity is infeasible due to the large communication overhead this pattern incurs. We note that the impact of privacy patterns on other qualities of the system can also be positive, such as the impact of the pattern *informed secure passwords* [2] on *security*.

Data Focus. A common distinction for privacy protection is data focus and process-oriented approaches [10]. That is, some approaches target the data elements used in an application, while other approaches target how such data elements are processed in an application. Accordingly, this dimension classifies patterns into the following set of properties: {*Data-focused, Process-oriented*}. For privacy patterns particularly, such a distinction has not been analyzed despite its validity for other privacy protection tools and approaches. For instance, the pattern *added-noise measurement obfuscation* [2] targets data elements themselves in an application and suggest modifying them. In other words, it does not target the architecture of the system nor the process itself albeit it may have some effect on these depending on the actual deployment of the pattern. Other patterns may target both the data element and the processing of it, such as the pattern *user data confinement pattern* [2] which states that the processing of personal data should be shifted to user-trusted environment.

Hotspot. Software systems, in general, are collections of several components along with their interactions. Within a software system, data is either *in-motion*, i.e., communicated or processed, or *at-rest*, i.e., stored [21]. To reflect this in terms of properties, we adopt the idea of a *hotspot* utilized in the recently proposed LINDDUN Go [42]. LINDDUN Go defines a hotspot as an area of interest in the system where a specific threat can originate. In our work, we consider a hotspot to be an area of interest in the system where a pattern can be applied. Thus, this dimension takes the following hotspot values: {*inbound, outbound, inbound from user, outbound to user, store, retrieve, process*}. *Inbound* and *outbound* refer to the flow into and from the system with an external entitiy, consequently. If the user is mentioned in the properties, then the external entity (with respect to the system) in this case is the user. *Store* and *retrieve* refer to the actions to data storage. *Process* refers to the processing operations in the system. An example of a pattern that may apply to inbound & outbound flows from & to the user is *onion routing* [2], while other patterns, such as *location granularity* [2], apply not only to user-inbound flows, but also to storing (in order to store less granular location after offering a service) and outbound flows to, e.g., third parties.

5 REPRESENTATION OF PRIVACY PATTERNS IN THE MULTIDIMENSIONAL SPACE

Section 4 discussed the minimal set of properties to support the selection process. This section illustrates the use of these properties by discussing two pattern examples along with the proposed properties (in Section 5.1) in detail, and by presenting the analysis results of these properties for a large set of privacy patterns (in Section 5.2).

5.1 Patterns Examples in the Multidimensional Space

In order to further illustrate the concept of the multidimensional space and the properties of privacy patterns, we now discuss the privacy pattern *encryption with user-managed keys* [2] (shown below) with respect to the multidimensional space we present in this paper. Additionally as a second example, we briefly discuss the privacy pattern *layered policy design* [2] along with its properties.

5.1.1 Encryption with User-managed Keys. This privacy pattern is described in the privacy patterns portal [2] and summarized below.

Pattern Title: Encryption with user-managed keys
Summary: Use encryption in such a way that the service provider cannot decrypt the user's information because the user manages the keys.
Enable encryption, with user-managed encryption keys, to protect the confidentiality of personal information that may be transferred or stored by an untrusted 3rd party.
Context: User wants to store or transfer their personal data through an online service and they want to protect their privacy, and specifically the confidentiality of their personal information.
Problem: How can a user store or transfer their personal information through an online service while ensuring their privacy and specifically preventing unauthorized access to their personal information?
Solution: Encryption of the personal information of the user prior to storing it with, or transferring it through an online service. In this solution the user shall generate a strong encryption key and manage it themselves, specifically keeping it private and unknown to the untrusted online service or 3rd parties.
Examples:

- Spider Oak, Least Authority, LastPass

- -

Our Multidimensional Representation

- -

- ***Applicability Scope.*** Implementation-specific.
- ***Privacy Objective.*** Confidentiality, Intervenability.

- ***Qualities.*** Performance Impact, Complexity of Interaction, Security.
- ***Data Focus.*** Data-focused.
- ***Hotspot.*** Inbound from user, Store, Outbound.

From the pattern's description, one can see that this pattern does not target the architecture of an application, e.g., does not necessarily require adding an architectural component such as a third party, but rather targets the application *implementation-specific* aspects to include new features of encrypting user's data, e.g., at the user side. A noteworthy remark here is that this pattern can also have architectural impact and changes depending on the developers and how they want to implement it. However, implementing it does not necessarily require architectural level changes in the application. Out of the privacy objectives, this pattern addresses *confidentiality* since it suggests encrypting user's data, as well as *intervenability* as it gives the user the control over the encryption keys. However, this pattern has some impact on the *performance* and the *complexity of interaction* as users need additional steps to encrypt their data before the service provider (or a 3P) obtains the data, which also supports *security.* This pattern is a *data-focused* pattern since that it targets and modifies the data itself by encrypting it. From the examples and known uses of this pattern, one can clearly conclude that this pattern applies to the hotspots *inbound from user, store* and *outbound* (to external entities) interactions in the system.

5.1.2 Layered Policy Design. This pattern aims to make privacy policy easier for users to understand, by layering the details of privacy policies in a concise fashion. We refer the reader to the privacy patterns portal [2] for a detailed description of this pattern. Unlike the previous pattern, this pattern addresses the *transparency* of privacy policy of collecting and processing user's data, but does not address the confidentiality of, e.g., the collected data. This pattern is *process-oriented* and *data-focused* as it targets both what data is collected and what processing is done on user's data, and present them to the user. This pattern does not have an impact on any of the functional qualities and does not affect the security of the application. It can be thoroughly addressed in the lower-level and implementation-specific aspects without impacting the high-level architecture of the application. In an application's data flow diagram, this pattern applies to the *outbound to the user* flow. Therefore, in our multidimensional space, this pattern can be represented as the following:

- ***Applicability Scope.*** Implementation-specific.
- ***Privacy objective.*** Transparency.
- ***Qualities.*** ϕ.
- ***Data Focus.*** Data-focused, Process-oriented.
- ***Hotspot.*** Outbound to the user.

5.2 Multidimensional Space Representation of Current Privacy Patterns

In this paper, we analyze the full set of 70 privacy patterns that are described in the privacy patterns catalog [2] with respect to our multidimensional space of patterns properties. In future work, this list will be extended with additional privacy patterns from other repositories (e.g., [1]). The analysis results are visualized in Table 1, which provides an overview of the multidimensional properties for each patterns.

This work lays the foundation to provide better support for privacy patterns selection. Let's consider the following example to illustrate this further: assume a service that uses identifiable credentials to authenticate its users so that they are able to use the service. Such a system, thus, suffers from an identifiability threat that applies to the inbound data flow to the service provider, and to the process of authentication. To remedy this threat, system engineers form the requirement that the service should be offered anonymously and look for privacy patterns that satisfy this requirement. At this point, system engineers would navigate the patterns portal website and filter out related patterns based on the unstructured list of tags. Related tags in this case include *anonymity, anonymous communication*, and *authentication.* Those tags suggest the following patterns; for anonymity, *protection against tracking, pseudonymous identity, attribute based credentials, and anonymity set*, for anonymous communication, *onion routing*, and for authentication, *attribute based credentials, and unusual activities.* There is no additional guidance on selecting: (i) appropriate tags, and thus, system engineers need to check the 36 available tags to ensure completeness of the outcome, and (ii) a pattern out of these suggested patterns. Therefore, system engineers are required to go through each one of those patterns and investigate its implementations to decide on which one to select. At this point, our multidimensional analysis and representation of privacy patterns helps system engineers to eliminate irrelevant patterns without the need to investigate them further. Assuming that latency of communication is crucial for our example here, then the *onion routing* pattern can be eliminated according to Table 1. Furthermore, for the selected pattern, system engineers and developers form a wider understanding of additional properties of that particular pattern other than the main property it has been selected upon. Assuming that, in our example, the pattern *pseudonymous identity* is selected, then system engineers perceive that this pattern can be instantiated at either the architecture level or at the lower implementation-specific level. System engineers will also perceive that this pattern supports plausible deniability but has an impact on security, while it is a data-focused pattern that targets the data (credentials) used for authentication. It is, therefore, worthy to remark that the multidimensional representation and the properties we present here provide a more structural way for the trade-off analysis developers often face when selecting privacy patterns.

However, with this example one can see that, despite the benefits privacy patterns properties bring, these properties *alone* are still *insufficient* to *fully* guide the selection process. One needs to complement those analysis with the other main factor the selection-process should be based on which is the *context* in which a privacy pattern will be deployed in, and the *context* a privacy pattern supports. Those two main factors; i.e., the properties of the patterns and the context of the system, have to be aligned in order to introduce a holistic selection method. We elaborate on this further in Section 6.

Observations, Lessons Learned, & Recommendations. Our analysis of the current landscape of privacy patterns provided us with some valuable insights. First, we see that some patterns do not actually contribute to any of privacy goals, e.g., *pay back.* This pattern states

	Scope		Privacy Objectives								Qualities				Data Focus		Hotspot
Pattern Name	Implementation specific	Overall Architecture	Anonymity	Unlinkability	Confidentiality	Plausible Deniability	Undetectability	Manageability	Intervenability	Transparency	Performance Impact	Complexity	Utility	Security	Data-focused	Process-oriented	Hotspot Properties
Protection against Tracking	●		●	●									–		●		Ib-U / R
Location Granularity	●	●	●	●		●	●						–		●		Ib-U / Ob / R
Minimal Information Asymmetry	●								●	●					●		Ib-U
Informed Secure Passwords	●				●				●	●		+		+	●		Ib-U / Ob-U
Awareness Feed	●								●	●					●		Ob-U
Encryption with User-Managed Keys	●				●				●		●	+		+	●		Ib-U / S / Ob
Federated Privacy Impact Assessment	●	●						●							●	●	Ib / Ob / Ib-U / Ob-U / P / R / S
Use of Dummies		●				●	●				●		–	–	●		Ib-U / S
Who's Listening	●								●	●		+			●		Ob-U
Privacy Policy Display	●									●					●	●	Ob-U
Layered Policy Design	●									●					●	●	Ob-U
Discouraging Blanket Strategies	●	●							●			+			●	●	Ib-U / Ob-U
Reciprocity	●	●											+			●	Ib-U / Ob-U / P
Asynchronous Notice	●									●					●	●	Ob-U
Abridged Terms and Conditions	●									●					●	●	Ob-U
Policy Matching Display	●									●					●	●	Ob-U
Incentivized Participation	●	●											+		●		Ib-U / Ob-U
Outsourcing [With Consent]	●								●						●	●	Ib-U / Ob
Ambient Notice	●									●					●	●	Ob-U
Dynamic Privacy Policy Display	●									●					●	●	Ob-U
Privacy Labels	●									●					●	●	Ob-U
Data Breach Notification Pattern	●	●						●		●					●		Ob-U
Pseudonymous Messaging		●	●	●							●				●		Ib-U
Onion Routing		●	●	●		●					●				●	●	Ib-U / Ob-U
Strip Invisible Metadata	●	●									●		–		●		Ib-U
Pseudonymous Identity	●	●	●	●		●								–	●		Ib-U / Ob-U
Personal Data Store	●	●			●				●			+			●		Ib-U / S / R
Trust Evaluation of Services Sides		●						●		●					●	●	Ob
Aggregation Gateway		●			●		●				●				●		Ib-U
Privacy Icons	●									●					●	●	Ob-U
Privacy-Aware Network Client		●								●					●	●	Ob-U
Sign an Agreement to Solve Lack of Trust on the Use of Private Data Context	●							●							●	●	Ib-U/Ob-U
Single Point of Contact		●			●				●		●				●	●	Ib-U
Informed Implicit Consent	●									●					●	●	Ob-U
Enable/Disable Functions		●							●				–			●	Ib-U / P
Privacy Color Coding	●									●					●	●	Ob-U
Appropriate Privacy Icons	●									●					●	●	Ob-U
User Data Confinement Pattern		●			●						●				●	●	Ib-U / P
Icons for Privacy Policies	●									●					●	●	Ob-U
Obtaining Explicit Consent	●								●	●					●	●	Ib-U
Privacy Mirrors	●									●					●		Ib-U/Ob-U
Appropriate Privacy Feedback	●									●					●		R / Ob-U
Impactful Information and Feedback	●									●					●		Ib-U / Ob-U
Decoupling [Content] and Location Information Visibility	●			●					●			+	–		●		Ib-U / S / R / Ob
Platform for Privacy Preferences	●									●					●	●	Ob-U
Selective Access Control	●	●			●				●					+	●		R
Pay Back	●												+		●		Ib-U / Ob-U / P
Privacy Dashboard	●									●					●		Ob-U
Preventing Mistakes or Reducing Their Impact	●									●					●		Ib-U / Ob-U
Obligation Management		●						●							●	●	Ob
Informed Credential Selection	●								●	●					●		Ib-U / Ob-U
Anonymous Reputation-Based Blacklisting		●	●	●									+	+			Ob-U
Reasonable Level of Control	●	●							●				–		●	●	Ib-U / P / S
Masquerade	●								●						●		Ib-U / Ob-U
Buddy List	●	●			●				●						●		Ib-U /,Ob-U / Ob
Privacy Awareness Panel	●									●					●		Ib-U /,Ob-U
Lawful Consent	●								●	●					●	●	Ib-U / P / S Ob
Privacy Aware Wording	●									●					●	●	Ob-U
Sticky Policies	●	●						●	●	●			+		●	●	Ob / S /P
Personal Data Table	●								●	●		–			●	●	Ib-U / Ob
Informed consent for Web-Based Transactions	●								●	●					●	●	Ib-U / S / P / Ob
Added-Noise Measurement Obfuscation	●	●			●	●							–	–	●		Ib-U / Ob
Increasing Awareness of Information Aggregation	●									●					●		Ib-U / Ob-U
Attribute Based Credentials		●	●	●		●					●	+		–	●		Ib-U / S / R / Ob / P
Trustworthy Privacy Plug-In		●		●	●		●				●				●		Ib-U
(Support) Selective Disclosure		●			●				●		●	+	–		●	●	Ib-U / P / Ob
Private Link	●				●				●						●		Ib-U / Ob
Anonymity Set	●	●	●			●								–	●		Ib-U
Active Broadcast of Presence	●			●	●		●		●		●	+	–		●		Ib-U
Unusual Activities	●				●					●		+		+	●	●	Ob-U

Table 1: Properties of privacy patterns in our multidimensional space representation. (●) means that the pattern has that particular property or the patter affects the quality attribute, (+) means the pattern increases the -quality- property, (–) means the pattern decreases the -quality- property, the hotspot properties are: (Ib-U) inbound from user, (Ob-U) outbound to user, (Ib) inbound to the system, (Ob) outbound from the system, (P) process, (R) retrieve, and (S) store.

that users need motivation to provide and maintain content so that the service will function properly, and suggests to reward users for providing their information, e.g, financially. This pattern does not align with the fundamentals of privacy protection and does not contribute to any of the privacy goals, which makes the validity of such a privacy pattern as a good privacy practice questionable. Additional patterns that fall under this observation include reciprocity and incentivized participation. This observation raises several concerns out of which is that, not all privacy patterns — as they are presented nowadays — support privacy, and therefore we should *refine* those privacy patterns and *carefully* introduce them in a way that truly supports privacy protection. Introducing them carefully also means avoiding strong overlapping between the ideas and intuitions behind privacy patterns.

Furthermore, one can clearly see in Table 1 that most privacy patterns support *soft privacy* goals [11, 12], and mainly focuses on transparency and intervenability. There are far fewer privacy patterns focusing on *hard privacy* [11, 12] like anonymity and unlinkability. The current landscape of privacy patterns seems to be considerably influenced by certain GDPR principles and articles that focus on user awareness and user control. This is reflected in the privacy patterns landscape by having the biggest subset of the landscape dealing with these certain principles and articles. A complete landscape of privacy patterns should equally consider both hard and soft privacy aspects. In other words, in order to see more trustworthy systems and applications supporting user's privacy in practice, it is crucial to introduce privacy patterns that not only support soft privacy goals, but also hard privacy goals.

Following the previous observation, transparency-related patterns might require additional classification and sub-categorization because they mainly deal with fine-grain details of how to keep the user informed, and those details are often not captured in the properties we presented in this paper. Mostly all transparency-related patterns share similar properties of certain dimensions; they, for instance, all incur implementation-specific aspects and do not require any architectural changes in an application. This indicates that, for a selection-support, additional fine-grain properties concerning those patterns need to be introduced to capture the fine-grained details these patterns suggest. Same observation applies to patterns related to hard privacy goals. There might exist some additional properties that the subset of hard privacy patterns share in common differently from the soft patterns properties. Apart from that, we discovered that differentiating between the applicability scope properties, i.e., the overall-architecture and the implementation-specific properties, is critical as it depends heavily on the way developers implement it. Some patterns do not require any architectural changes, but can still be implemented in some way that requires adding, e.g., an architectural building block. At last, a noteworthy remark we noticed when working with the current landscape of the privacy patterns [1, 2] that the patterns' style is not a GoF-like style [16]. The current privacy patterns mostly describe some ideas used in websites and applications, like *private link* [2] or *unusual activity* [2], which are fundamentally different from GoF software patterns [16].

6 FUTURE WORK

Selection Support using Decision Trees. Defining properties for privacy patterns is evidently only the first step towards selection support. The next challenge is determining how these *properties* can be applied together with the *context* in the selection procedure. We see the use of decision trees as a potential and promising selection support method. Each decision tree may address a privacy objective that needs to be achieved or may address a certain applicability scope, e.g., overall-architecture or implementation-specific. Information on an application and the context under which it operates is fed into the tree as an input which ideally leads to a tree leaf that is equivalent to one or multiple suitable privacy patterns. There, the context rules for systems could be adapted from previous work in context-aware requirements engineering we discussed in Section 2, e.g., [25, 31]. One can see such method as two steps: first, filtering out inapplicable patterns due to their undesired properties, and second, using both the context information and the qualities impacted to select the most suitable pattern for that given scenario in a guided manner. The particular order of the properties and type of questions are still to be determined. Once a pattern is selected, the relationships to other patterns may be investigated to check whether additional pattern(s) should also be deployed together with the selected pattern to achieve, e.g., the overall objective, or whether another pattern should be completely avoided (or favored). The properties we proposed in this paper provide a further assistance in structuring such (unavoidable) trade-off analysis. The current relationships between privacy patterns address only intra-strategy relationships of patterns within one strategy [8, 9]. With the described and demonstrated arguments in this paper, one should realize that relationships not only (1) exist among patterns associated with different strategies, but also (2) may exist between patterns from different abstraction levels and different properties. Another noteworthy remark we want to make here is that, working on such a holistic selection-support for privacy patterns will provide feedback on the set of the dimensions and properties, and will potentially result in extending this set to include additional dimensions and properties. For instance, the dimension *qualities* can be extended to include detailed performance metrics such as, e.g, latency, load, throughput, scalability properties. In fact, this dimension is even more complex in practice. It may affect different phases of the lifecycle: design, implementation, and actual operation of the system, and can be seen from different perspectives: developer, end-user, and operator. Another aspect that this dimension should take into consideration is the evolvability of a software system and how certain patterns may restrict future development. Such additional features of the quality dimension overlap strongly with the contextual information that should be considered for selection-support, and align pretty well with our future work direction. Future work should, therefore, focus on evaluating the usefulness of the multi-dimensional representation we propose for privacy patterns, and investigate how to utilize this knowledge with the contextual information in order to introduce a simplified selection-support method for privacy patterns.

7 CONCLUSION

In this paper, we pointed out the lack of a selection support method for privacy patterns. The goal of such selection-support is to enable developers to confidently and more efficiently select fitting privacy patterns, and also enable them to justify their choices. Particularly, such a method should scope applicable and fitting patterns easily even for practitioners with less extensive privacy expertise. Scoping privacy patterns requires categorizing them according to some shared properties among them. Such properties may include their abstraction level concerning the development phase, their applicability scope in the system, and their impact on the functionality of the system. These properties are not well-understood for privacy patterns yet and additional analysis is needed to determine such properties. In this paper, we study and define such properties for privacy patterns. We analyze a total of 70 privacy patterns, and present them in a multidimensional space in which each dimension takes a set of values that determine the properties of the patterns. The next challenge is to combine the contextual information of systems along with the properties to offer a holistic selection-support method for privacy patterns. The best approach to utilize such information and represent the selection-support method is still to be determined.

REFERENCES

[1] [n. d.]. Privacy Patterns. https://privacypatterns.eu/ Last Checked: Sep. 2020.

[2] [n. d.]. Privacy Patterns. https://privacypatterns.org/patterns/ Last Checked: Sep. 2020.

[3] [n. d.]. Privacypatterns.org mirror. https://privacypatterns.cs.ru.nl/ Last Checked: July. 2020.

[4] Ala'a Al-Momani, Frank Kargl, Robert Schmidt, Antonio Kung, Christoph Bösch, et al. 2019. A Privacy-Aware V-Model for Software Development. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 100–104.

[5] Kaitlin R Boeckl and Naomi B Lefkovitz. 2020. NIST Privacy Framework: A Tool for Improving Privacy Through Enterprise Risk Management, Version 1.0. (2020).

[6] Rahma Bouaziz and Slim Kammoun. 2015. A Decision Support Map for Security Patterns Application. In *Computational Science and Its Applications – ICCSA 2015*. Cham, 750–759.

[7] Julio C Caiza, Jose M Del Alamo, and Danny S Guamán. 2020. A framework and roadmap for enhancing the application of privacy design patterns. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 1297–1304.

[8] Michael Colesky and Julio C. Caiza. 2018. A System of Privacy Patterns for Informing Users: Creating a Pattern System. In *European Conference on Pattern Languages of Programs (EuroPLoP '18)*. Article 16, 11 pages.

[9] Michael Colesky, Julio C Caiza, José M Del Alamo, Jaap-Henk Hoepman, and Yod-Samuel Martín. 2018. A system of privacy patterns for user control. In *ACM SAC*. 1150–1156.

[10] Michael Colesky, Jaap-Henk Hoepman, and Christiaan Hillen. 2016. A critical analysis of privacy design strategies. In *Security and Privacy Workshops (SPW)*. IEEE, 33–40.

[11] George Danezis. 2008. Talk: an introduction to u-prove privacy protection technology, and its role in the identity metasystem–what future for privacy technology.

[12] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. 2011. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering* 16, 1 (2011), 3–32.

[13] Olha Drozd. 2015. Privacy pattern catalogue: A tool for integrating privacy principles of ISO/IEC 29100 into the software development process. In *IFIP International Summer School on Privacy and Identity Management*. Springer, 129–140.

[14] Eduardo B Fernandez, Nobukazu Yoshioka, Hironori Washizaki, Jan Jurjens, Michael VanHilst, and Guenther Pernu. 2011. Using security patterns to develop secure systems. In *Software Engineering for Secure Systems: Industrial and Research Perspectives*. IGI Global, 16–31.

[15] Eduardo Fernandez-Buglioni. 2013. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns* (1st ed.). Wiley Publishing.

[16] Erich Gamma. 1995. *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

[17] Seda Gürses, Carmela Troncoso, and Claudia Diaz. 2015. Engineering privacy by design reloaded. In *Amsterdam Privacy Conference*. 1–21.

[18] Munawar Hafiz. 2006. A Collection of Privacy Design Patterns. In *Proceedings of the 2006 Conference on Pattern Languages of Programs (PLoP '06)*. Article 7, 13 pages.

[19] Munawar Hafiz. 2013. A pattern language for developing privacy enhancing technologies. *Software: Practice and Experience* 43, 7 (2013), 769–787.

[20] Munawar Hafiz, Paul Adamczyk, and Ralph E Johnson. 2007. Organizing security patterns. *IEEE software* 24, 4 (2007), 52–60.

[21] Jaap-Henk Hoepman. 2014. Privacy Design Strategies. In *ICT Systems Security and Privacy Protection*. 446–459.

[22] Jörn Kahrmann and Ina Schiering. 2014. Patterns in privacy-a pattern-based approach for assessments. In *IFIP International Summer School on Privacy and Identity Management*. Springer, 153–166.

[23] Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. 2007. Using privacy process patterns for incorporating privacy requirements into the system design process. In *The Second International Conference on Availability, Reliability and Security (ARES'07)*. IEEE, 1009–1017.

[24] Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. 2008. Addressing privacy requirements in system design: the PriS method. *Requirements Engineering* 13, 3 (2008), 241–255.

[25] Tong Li, Jennifer Horkoff, and John Mylopoulos. 2014. Integrating security patterns with security requirements analysis using contextual goal models. In *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 208–223.

[26] Lin Liu, Eric Yu, and John Mylopoulos. 2003. Security and privacy requirements analysis within a social setting. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003*. IEEE, 151–161.

[27] Rene Meis and Maritta Heisel. 2017. Pattern-based representation of privacy enhancing technologies as early aspects. In *International Conference on Trust and Privacy in Digital Business*. Springer, 49–65.

[28] Anas Motii, Brahim Hamid, Agnes Lanusse, and Jean-Michel Bruel. 2015. Guiding the selection of security patterns based on security requirements and pattern classification. In *20th European Conference on Pattern Languages of Programs*. 1–17.

[29] Anas Motii, Brahim Hamid, Agnes Lanusse, and Jean-Michel Bruel. 2016. Guiding the selection of security patterns for real-time systems. In *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 155–164.

[30] Sebastian Pape and Kai Rannenberg. 2019. Applying privacy patterns to the internet of things'(iot) architecture. *Mobile Networks and Applications* 24, 3 (2019), 925–933.

[31] Siani Pearson and Yun Shen. 2010. Context-aware privacy design pattern selection. In *International Conference on Trust, Privacy and Security in Digital Business*. Springer, 69–80.

[32] Andreas Pfitzmann and Marit Hansen. 2010. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. (2010).

[33] Sasha Romanosky, Alessandro Acquisti, Jason Hong, Lorrie Faith Cranor, and Batya Friedman. 2006. Privacy Patterns for Online Interactions. In *Proceedings of the 2006 Conference on Pattern Languages of Programs (PLoP '06)*. Article 12, 9 pages.

[34] Riccardo Scandariato, Koen Yskout, Thomas Heyman, and Wouter Joosen. 2008. *Architecting software with security patterns*. Technical Report. Department of Computer Science, K.U.Leuven; Leuven, Belgium.

[35] Markus Schumacher. 2003. *Security engineering with patterns: origins, theoretical models, and new applications*. Vol. 2754. Springer Science & Business Media.

[36] Chritopher Steel and Ramesh Nagappan. 2006. *Core Security Patterns: Best Practices and Strategies for J2EE", Web Services, and Identity Management*. Pearson Education India.

[37] T. Suphakul and T. Senivongse. 2017. Development of privacy design patterns based on privacy principles and UML. In *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 369–375.

[38] Clark Thomborson. 2016. Privacy patterns. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 656–663.

[39] Axel Van Lamsweerde. 2001. Goal-oriented requirements engineering: A guided tour. In *Proceedings fifth ieee international symposium on requirements engineering*. IEEE, 249–262.

[40] Hironori Washizaki, Eduardo B Fernandez, Katsuhisa Maruyama, Atsuto Kubo, and Nobukazu Yoshioka. 2009. Improving the classification of security patterns. In *2009 20th International Workshop on Database and Expert Systems Application*. IEEE, 165–170.

[41] Michael Weiss and Haralambos Mouratidis. 2008. Selecting security patterns that fulfill security requirements. In *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 169–172.

[42] Kim Wuyts, Laurens Sion, and Wouter Joosen. 2020. LINDDUN GO: A Lightweight Approach to Privacy Threat Modeling. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE.

[43] Koen Yskout, Riccardo Scandariato, and Wouter Joosen. 2015. Do security patterns really help designers?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 292–302.