Engineering Privacy by Design Reloaded

Gürses, Seda *
Princeton University
fgurses@princeton.edu

Troncoso, Carmela †
Gradiant
ctroncoso@gradiant.org

Diaz, Claudia ‡
COSIC/iMinds, Dept. of Electrical Engineering, KU Leuven
Claudia.Diaz@esat.kuleuven.be

1 Introduction

The concept of "privacy by design" has gained traction in policy circles in the last decade. However, The actual design, implementation and integration of privacy protection in the engineering of products and services remains however an open question. Different parties have proposed privacy-by-design methodologies that promise to be a holy grail for organizations collecting and processing personal data. These efforts aim at addressing the engineering aspects of privacy by design by pointing to design strategies, but fall short of relating how these strategies can be applied when building privacy preserving information systems [1, 9, 11].

In response to this status quo, we wrote a paper in 2011 on how data minimization can be applied to address privacy concerns in information systems [8]. In this paper, we used two case studies, a system for anonymous e-petitions [5], and a privacy- preserving Electronic Toll Pricing system (PrETP) [2], to illustrate in a concrete manner how a design process guided by the principle of data minimization would lead to a reduction of privacy risks, avoid function-creep, and provide users with maximum control over sensitive information.

The publication of our paper spurred a discussion with other experts in the field in which it became apparent that the "data minimization" metaphor may be misleading. In a system with a privacy-preserving design, the flow of sensitive data to a centralized entity (the service provider) is indeed "minimal", yet all the privacy-sensitive user data is captured and still stored on devices

*Some of this work was completed during the author's time at New York University Information Law Institute and at COSIC, KU Leuven

†Some of this work was completed during the author's time at COSIC, KU Leuven. This work is supported in part by the EU PRIPARE (FP7 GA No. 610613) and WITDOM (H2020 GA No. 64437) projects.

‡This work was partially funded by the projects FWO G.0360.11N, FWO G.0686.11N, and KU Leuven BOF ZKC6370 OT/13/070, and EU H2020 Panoramix project.

within the boundaries of the system. The difference to a "straightforward" implementation of a privacy preserving system is that the sensitive data only resides in components of the system under the control of the user. By that we mean that sensitive data may be kept on a user device; in encrypted form where the user holds the key; or data may be distributed across entities where the user is the only one who can re-compile the data. No matter which design is the enabler of user control, sensitive data resides somewhere in the system. As engineers refine their design, many things are being minimized, but certainly not data. So we asked ourselves, if the number of engineering activities that we engage in is not magically "vaporizing" data in the system, what is it doing?

After further examination of existing privacy preserving system designs, it became evident that a whole family of design principles are lumped under the term "data minimization". The term conceals a number of design strategies that experts apply intuitively when developing privacy preserving systems. A number of these are constraints on information flows like minimizing collection, disclosure, linkability, replication, retention and centrality. Systems engineered by applying these constraints intend to "minimize risk" by avoiding a single point of failure, and minimize the need to trust data collectors and processors by putting data under the user's control.

These findings are in line with the main contribution of another paper in which a host of privacy design principles – called privacy design strategies– are described [9]. In this paper, "privacy design strategies" refer to distinct approaches that can be used to achieve privacy protection. Privacy enhancing technologies (PETs) on the other hand are used to implement "privacy design patterns" – a commonly recurring structure of communicating components that solves a general design problem within a particular context. Privacy design strategies make explicit the different approaches that are available to protect privacy when designing systems. For each privacy design strategy the appropriate privacy design patterns and related PETs can be leveraged to enable the implementation of a privacy preserving system. For example, "hiding" information is defined as a strategy, and refers to "hiding any personal information that is processed from plain view". Hiding personal information can be achieved using the common privacy design pattern "mix networks" for anonymous communication. A privacy enhancing technology that enables hiding of traffic information using mix networks is Tor[1] [6].

This elegant distinction between privacy design strategies and how they relate to PETs is illuminating. However, in and of itself, having access to a catalog of design strategies and patterns is not sufficient to provide insight into the process through which these can be applied. It leaves open the question how these strategies can be put to use in practice. Experts know how to do this: they perform a number of activities during which they apply these strategies. However, this practice is not self-evident to non-experts that may want to integrate PETs into systems.

Our quest is to respond to this gap in knowledge by spelling out how ex-

[1] `https://www.torproject.org/`

perts apply privacy design strategies. We hope that a deeper understanding of their practice can inform future methods for engineering privacy by design. Specifically, providing greater insights into the definition and formalization of these strategies; how they can be related to the privacy design patterns; and, the way they guide the design process is in this context valuable and desirable. Moreover, our initial work shows that these three parts are interdependent. The definitions of the privacy design strategies and the process through which they can be applied needs some fitting, another aspect we believe would benefit from further elaboration.

In this paper we make the modest contribution of summarizing our initial conceptualization of how experts apply data minimization strategies. Specifically, based on a study of existing privacy preserving systems, we first elaborate the design strategies hidden behind the term data minimization. We then provide a preliminary description of the activities that a privacy engineer performs to apply the right data minimization strategies. Based on this process description, we then discuss where the definitions are useful, and where they need further tweaking. Through this exercise, we intend to make explicit some of the reasoning that PETs experts apply and that are difficult to grasp for outsiders. We intend this paper to start another round of discussion with experts but also method engineers on privacy engineering processes.

Why focus on PETs, security engineers and data minimization?

Throughout our study, we assume we can learn from the existing practices of security engineers that develop PETs about the art of engineering privacy preserving systems. This may not seem sensible to someone who believes privacy engineering activities should be derived from data protection or privacy laws. Our justification for starting with these technical experts is twofold. First, we believe that engineering knowledge and experience, and not only those of security engineers, should be part and parcel of the conception of activities that can be summarized as privacy engineering. Second, while other fields of computer science and engineering contribute to privacy engineering practice, privacy enhancing technologies have predominantly been conceived and developed by security engineers. In engineering privacy preserving systems, the knowledge base of this community is unique and hence deserves our attention.

Furthermore, privacy engineering may benefit from the systematization of knowledge around designing systems using PETs. Security engineers engaged in PETs learn their trade by participating in a community of researchers and by implementing their ideas in concrete technical systems. Using the language in [9], these are experts at the cutting edge of defining novel privacy design patterns and enabling their implementation through (a combination of) concrete privacy enhancing technologies (PETs). It is then also unsurprising that they are the main figures who know how to put these privacy design strategies to work – valuable knowledge we hope to capture and make explicit to the best of our linguistic abilities.

Privacy engineering activities can be fruitful in tackling all aspects of data protection during system design. However, in this paper, our focus remains on "data minimization strategies". In other words, we are interested in those engineering activities that intend to minimize the risk of privacy breaches by minimizing trust in data collectors and processors handling sensitive data properly. Further privacy design strategies can be applied to increase the integrity and transparency of systems once sensitive data flows to data collectors and processors. For example, technical mechanisms may be introduced to guarantee that these entities respect their privacy policies with regard to data processing, to validate the integrity of algorithms, or demonstrate compliant handling of data. Such approaches are complementary to what we are doing and not in the scope of this paper.

As we turn our focus to privacy engineering, we make a number of assumptions about the world. Certain security issues, e.g., the security of users' devices, of the privacy enabling cryptographic mechanisms, as well as the secure execution of collection and processing activities are instrumental but orthogonal to the efforts we explain here. Furthermore, we assume we can trust the engineers with their designs, i.e., we trust that the systems they build will do approximately what they promise. This is a trust assumption which deserves many papers on its own.

The paper intends to contribute to the maturing field of privacy engineering by developing a clear vocabulary to express pertinent elements of its practice. By gaining a better understanding of the privacy engineering practice, we hope that policy makers will also be in a better position to articulate laws or other regulation by design frameworks. Finally, while it is unreasonable to expect the general public to understand the state of the art in privacy engineering, by making the reasoning explicit, we hope to contribute to making privacy engineering more accountable as a practice.

2 Unpacking Data minimization: a realm of strategies

Most intuitively, data minimization refers to not collecting certain data inputs, i.e., if not necessary for achieving the desired functionality of the system, data should not be collected in the first place. By ensuring that no, or no unnecessary, data is collected, the possible privacy impact of a system is limited [9]. In our previous paper [8], we argued that there is a less intuitive way to minimize data using state of the art in mathematical and computational capabilities. However, once we laid out and described how these capabilities are used, it became evident that many things were happening but the data in the system was not being minimized, reduced or removed using these capabilities. Rather, we found that with data minimization experts refer to a number of other design strategies that make it possible to constrain the flow of data from the user controlled domain to the domains controlled by other parties.

Through a systematic study of privacy preserving systems, we identified a set of data minimization strategies that we use to jump-start this paper. These strategies were inferred from the case studies presented in our previous paper, i.e., PrETP and privacy preserving e-petition, as well as other prominent PETs like Tor [6] or OTR [3]. To infer them, we did the cyclical exercise of identifying different data minimization strategies in a case study, and then testing the new set against another case study, until we were not able to identify additional strategies. Yet, as we discuss in Sect. 4 it is not clear whether these are the only strategies, nor whether our definitions are complete and coherent. We will therefore revisit and elaborate on these definitions once we have a better grasp of the process through which experts apply these strategies.

Before moving to the definitions of the data minimization strategies, it is important to clarify what we mean by a system. First, we assume the experts are about to develop a system that is going to be introduced into an environment. By system, we refer to all the entities that capture, process or further disseminate data, the technical parts of which the engineer is responsible for designing. For example, Fig. 1 describes an electricity smart metering system. This system includes all the users, the smart meters, as well as the servers of the utility provider. If the engineer were designing an app, then all entities running that app would also be seen as part of the system, including the software and hardware. In some cases, entities are hardware or software taken off the shelf, e.g., the phone of a user, app libraries, and the engineer has to decide whether this entity provides the necessary infrastructure for the privacy engineering task.

2.1 Data Minimization Strategies

We identified minimization of risk and the need for trust in other entities to be the primary privacy design strategies:

Risk whenever possible limit the likelihood and impact of a privacy breach.

Need for trust whenever possible limit the need to rely on other entities to behave as expected with respect to sensitive data.

A short clarification may be useful here. Minimizing need for trust is not about an emotional distrust towards any entity other than the user. Rather, it is about relying on entities to fulfill the functionality of the system, without this reliance being conditioned upon them collecting and handling large amounts of sensitive data that may later lead to privacy breaches. In most cases, minimizing the need for trust is seen as being equivalent to minimizing risk of privacy breaches materializing. However, there may be cases where the two are not aligned, e.g., cases where in order to avoid privacy breaches, sensitive data may be better handled by other parties.

The following are the strategies that can be used to minimize risk and the need for trust:

Minimize Collection: whenever possible limit the capture and storage of data in the system.

Minimize Disclosure: whenever possible constrain the flow of information to parties other than the entity to whom the data relates.

Minimize Replication: whenever possible limit the amount of entities where data is stored or processed.

Minimize Centralization: whenever possible avoid single point of failure in the system.

Minimize Linkability: whenever possible limit the inferences that can be made by linking data

Putting temporal limitations is orthogonal to the five strategies above and can be applied to all data and information flows in the system:

Minimize Retention: whenever possible minimize the retention of data in the system.

Privacy-preserving systems typically aim to protect privacy by combining these principles. For example, in PrETP [2], collection is not minimized, i.e, data collected on the On Board Units (OBUs), devices in the vehicle doing local computations assumed to be under control of the user, does not get removed and hence remain in the system. However, disclosure, replication and centralization are all minimized. The location data remains on the OBU, while only the information necessary to fulfill the functionality of the system, the final fee, plus some data needed for service integrity flows to the service provider. This avoids the replication and the centralization of location data. Users are registered with the service provider, hence fees can be linked to the user, but not the location data. Spot checks are used for fraud detection, but these are designed such that only the location information that is being probed is released to the service provider, minimizing disclosure. As a result of this design, users do not need to trust the service provider with the protection of their location data. Since the service provider does not have a large database of location data, the risk of privacy breaches are also minimized.

It was a considerable task to unpack the different data minimization strategies that were followed in PrETP, but how did the experts get to this design? How did they decide where which data will be collected, to whom the information will flow, and which privacy design patterns would best help them get there? In the next section, we scratch the surface of how this process unfolds for the experts.

3 Engineering Privacy by Design with Data minimization strategies

In the previous sections we have identified a set of strategies that steer the design of ICT systems towards privacy-preserving implementations. In this

section we continue our reflection about how and when experts apply the data minimization strategies. The idea is to provide designers and engineers with insights that shall help them to make choices that increase the level of users' privacy in the system.

Before diving into details, it is important to note that the thoughts reflected in this paper only deal with choices taken when designing systems from scratch, and it is not clear that they are of use when re-designing or modifying systems. Furthermore, we note that the paper only tackles the design step and not previous steps (e.g., requirements elicitation, the threat analysis), nor posterior steps (e.g., concrete implementation).

3.1 Starting assumptions

At the beginning we assume that the engineer has an idea of a "straightforward" design of the desired system. For example, if they are going to develop a road tolling system, they have an imagination of the basic elements of the design of such a system with a database and some sort of tracking mechanism. Typically, similarly to most deployed ICT systems, this idea would be engineered in such a way that the entity providing a service must have access to all of the data produced in the system in order to fulfill the required functionality. We call this straightforward design the *reference system*, and we consider its privacy protection level the baseline against which privacy-preserving systems can be compared to. More concretely, we assume that at the beginning:

1. There exists an initial reference system that allows to fulfill the desired functionality, whether it is based on an existing system or concocted by the designer. We assume that for this reference system:
   (a) There exists a system model: an abstract architecture of the reference system that could fulfill the functional requirements. Stakeholders are identified, and situated in the architecture (i.e., their interactions with the different system components).
   (b) There exists an information model: a model reflecting the data that will be collected and/or processed by the reference system.

As it will become apparent later in this section, in order to enable the designer to take privacy-preserving choices we must further assume that:

2. The functionality of the desired system is well defined. This means that the goal of the system is concrete and specific.
3. The privacy concerns of the system's stakeholders, and the service integrity requirements of the system are identified.[2] By service integrity requirements we mean those that guarantee that interactions in the system are

[2]We are on purpose overlooking other fundamental security requirements (e.g., availability, data integrity, etc.) since, as already mentioned in the introduction, the techniques to achieve such properties are well known and orthogonal to the purpose of this paper.

complete, coherent and accountable. In layman terms, requirements that allow parties to check that others acted responsibly within the system.

Example: Electricity smart metering system

1. The **reference system** consists on the following:
   (a) *System model*: the stakeholders are identified (Users, Utility, Regulatory authorities such as governmental agencies or industry self-regulation bodies) and there is a reference architecture of the system, where stakeholders roles and their interactions are identified (see Fig. 1).
   (b) *Information model*: the data flowing in the system is identified:
   - Personal data of users subscribed to the system: the data needed by the Utility to identify customers (e.g., name, address, etc.)
   - Billing data of users subscribed to the system: the data needed by the Utility to bill users (bank account and amount to be billed)
   - Consumption data of users, i.e., their consumption records
   - Transaction data, i.e., log of transactions required by regulation authorities (e.g., proofs of billing, proofs of payment, etc.)
2. The **well-defined goal** of the system is "to bill users depending on how much electricity they consume at each billing rate"[3]as opposed to a less specific description such as: "to bill users depending on their energy consumption habits"
3. The **privacy and security requirements** are the following:
   - *Privacy Requirements*:
     - Users: to hide their fine-grained consumption from all actors in the system, to hide their billing information and other personal data from all actors but the Utility
     - Utility: -
     - Regulatory authorities: -
   - *Service Integrity requirements*
     - Users: must be billed accurately for their consumption (i.e., the utility cannot charge them for more than what they actually consumed)
     - Utility: requires service integrity, i.e., the bill must include the full consumption record (i.e., the users cannot pay for less than what they actually consumed)

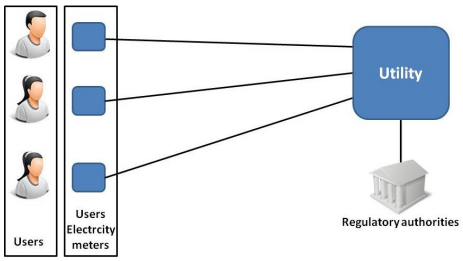

Figure 1: Electricity smart metering system – Reference abstract architecture.

– Regulatory authorities: require to be able to check that all transactions have been done correctly

3.2 Guidelines to apply the strategies

Departing from the assumptions in the previous section, we now propose four activities that are intended to help the designer decide when and how to apply the strategies in order to safeguard the privacy of users in the designed system. We have separated and ordered activities to improve readability, but we must stress that while articulating the activities we recognized that they are not always disjoint and that the order does not necessarily need to be as stated in this paper. From here on we also refer to our observations of how experts perform these activities, the generalization of these practices into a useful and practical methodology is a topic of future research.

3.2.1 Activity 1: Classification of system entities in domains

A first hidden assumption taken by experts is the implicit classification of entities in the system in two domains:

- **User domain**: these are entities which are assumed to be under the control of the user. Hence, experts consider it to be ok to collect or process the user's sensitive data in these entities.
- **Service domain**: these are entities which are not under the control of the user. They include data processors and data controllers, but can include other entities involved in the system. Since they are not under the control of the user, experts consider that sensitive data should not be accessible to these entities.

Figure 2 shows a possible definition of these domains in the Smart Energy example. The users, as well as the smart meters, are considered under the

[1]We are aware that Electricity smart metering system could be based in more complex policies, we chose a simple one to ease the explanation. We note that these policies could also be well defined and limited.

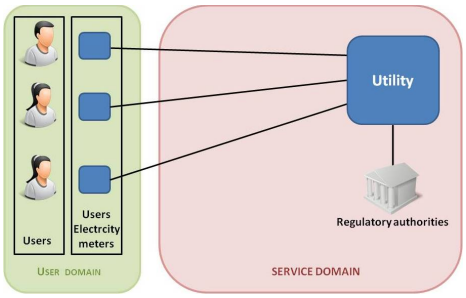

Figure 2: Electricity smart metering system – User and Service Domains.

control of the user[4], and hence they compose the *User domain*. The Utility provider and Regulatory bodies are not considered to be under control of the user and hence they form the *Service domain*.

3.2.2 Activity 2: Identification of necessary data at the service domain

A second activity taken by experts that is difficult to grasp without years of training is the identification of the set of data necessary at the service domain for achieving the purpose of the system. It is important to note that in general there is no established minimal set of data since it strongly depends on the system purpose, its context, etc.

Typically, designers aim at collecting as much data as possible in the service domain driven by: i) the aforementioned feeling that all data should be accessible by the entity providing the service, and ii) the pressure from marketing and/or business units who also push for collecting as much data as possible. We call this the "collect-all-data" approach. In general, there are some limits on this collection imposed most often by regulations, and less often stemming from social pressure. This limitation results in the collection of a smaller set of data than initially intended, though a lot of personal and sensitive data can still be collected provided that there is consent from the user, regardless of whether it is necessary for the functionality or not. A slightly improved version of this approach with respect to privacy is "select-before-collect" [9]. This approach, inspired by Data Protection principles, encourages the designer to think about the need for every piece of data that could be collected in the system, so that pieces that are not necessary are removed from the set of collected data.

On the other hand experts, whose approach we call "only-collect-necessary-data", start by thinking about the minimum data necessary to fulfill a purpose. Such thinking is very much influenced by their knowledge of the possibilities offered by technology, and is one of the sources of the intertwine between the

[4]As mentioned in the introduction, this vision of "entities controlled by the user" hides an assumption often made by experts: that the code running in the user devices is trustworthy, i.e., that it will act as expected and will not operate in any way that may harm the privacy of the user explicitly or in a stealthy manner.

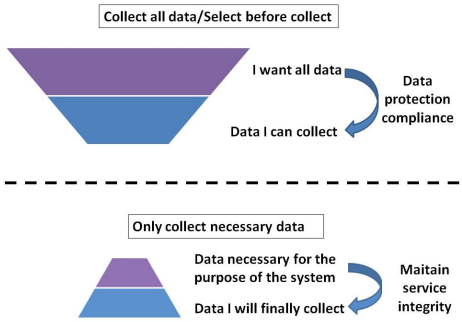

Figure 3: Identification of necessary data: Typical vs. Experts approaches.

Activities described in this paper. This set of data, while sufficient to fulfill the system's functionality, may not be enough to guarantee service integrity and hence experts are often forced to collect more data than in principle desired. This extra information is limited to data required to ensure correct functioning and most often than not, again thanks to the use of advanced technology, does not include sensitive information in "clear form". What we mean by information not being available in the clear shall become clearer when reading the description of Activity 4.

The way these two approaches work is illustrated in Fig. 3, where purple is a representation of the data that initially seemed to be necessary to collect, and blue represents the data that will finally be collected. The fundamental difference between the two approaches is apparent. While the "Collect all data/Select before collect" approach is based on starting by a tentative set of data and shrinking this set to find the final collected data; the experts "Only collect necessary data" approach consists on starting by the minimal data to fulfill the functionality and only increase this set when necessary for service integrity. Therefore, the experts' approach is in general bound to end up requiring less data, and in particular less sensitive data, than the typical approach.

3.2.3 Activity 3: Distribution of data in the architecture to achieve the functionality

This activity consists of mapping the data in the information model to the entities in the User and Service domains, guided by the identification of data performed in Activity 2. This again highlights that activities are not independent, and that they may need to be revisited during the design process. The mapping of data in domains responds to the following reasoning (The relation between the inputs and the outputs of domains is shown in Fig. 4.):

- **Data necessary at the Service Domain**: this is data that must flow to the Service domain in order for the entities in this domain to be able to carry out operations for achieving the functionality of the system.

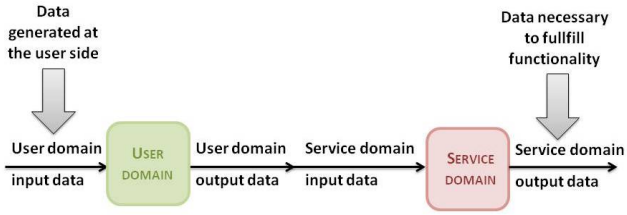

Figure 4: Data flow in User and Service domains.

- **Data necessary at the User Domain**: this is data that needs to exist in the User domain so that the entities in this domain can produce adequate inputs to the Service domain for the fulfillment of the system functionality.

The grey box below shows the data placement in our running example.

Example: Electricity smart metering system

User Domain	Service Domain
Personal data, Billing data Consumption data	Personal data, Billing data Consumption data, Transaction data

First of all we note that, regardless of whether the chosen design approach is privacy invasive or privacy preserving, there is some data that will always exist at the User domain since either it is generated there (e.g., the Consumption data), it is inherent to the user (e.g., her Personal data), or it is at some point forwarded to the user (e.g., the Billing data). This said, at first sight all data seems to be necessary at the Service domain to fulfill the goal of "billing users depending on how much electricity they consume at each billing rate": Personal data is necessary to identify the user; Consumption data is necessary to i) compute the bill and ii) run checks to guarantee service integrity (e.g., detect anomalies); Billing data is necessary to charge the user; and Transaction data is necessary to comply with regulation authorities. This distribution is represented in Fig. 5. According to Fig. 4, the output of the User Domain would consist of the Personal and Consumption data that serve as input for the operations carried out by entities in the Service domain.

However, the amount of data collected differs depending on which of the approaches to identify necessary data is followed in Activity 2. Let us consider two paradigmatic examples: personal data and consumption data.

Personal data: When following the "Collect all data" approach, typically the designer will try to collect all types of data: name, postal address, phone, email, gender, usage preferences, and any other data deemed relevant for marketing purposes. Then, usually regulations kick in and some data cannot be collected because it is considered excessive resulting, for instance, in usage preferences not being collected. The "Select before collect" approach may result in a reduced set of data, since while reflecting about the necessity is likely that

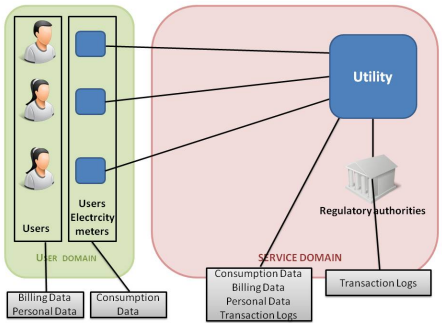

Figure 5: Electricity smart metering system – Data distribution in domains.

some data (e.g., gender) is deemed not necessary, although this will depend very much on the designer's discretion.

On the other hand, when following the "Only collect necessary data" approach, experts would solely require name (for billing purposes), address (to identify the users' meter), and depending on the means to communicate with the user, her phone number or email.

Consumption data: When following the "Collect all data" approach, typically the designer will try to collect as fine grained information as possible, regardless of the billing policies, e.g., a reading every minute. Then, usually regulations kick in and some data cannot be collected because it is considered excessive resulting. In the current case often regulation only allows to collect one reading every 15 minutes. The "Select before collect" approach may result in a reduced set of data, since it may become apparent that for certain billing policies (e.g., bill according to consumption per hour) less readings are necessary.

On the other hand, when following the "Only collect necessary data" approach, experts would first study the billing policy, and then focus on collecting the data necessary to compute the bill according to the policy. This results in much less sensitive data being collected, since collection will be coarse and limited in frequency.

3.2.4 Activity 4: Follow the strategies through the use of technology

It may be surprising that the above Activities have not dealt with the inclusion of Privacy Enhancing Technologies in the system design. Rather, we have introduced the reader to the preliminary exercises done by the experts to arrive to the point where these technologies can play their role to increase the privacy protection provided by the system.

Up to this point the previous Activities would have led the engineer to have in mind an architecture where data is placed, and whose entities are divided in two domains: the User domain, where experts consider that sensitive data can be stored and processed since the entities in this domain are under the

control of the user and hence cannot harm her privacy; and the Service domain, where sensitive data should not flow into since the entities in this domain are not under the user's control and hence could harm user privacy accidentally or intentionally.

Therefore, the goal of the expert shall be to remove as much data as possible from the Service domain, or in other words to keep as much data as possible in the User domain with respect to the initial data distribution established in Activity 3. In order to achieve this goal, experts pose the question: "Does this data really need to flow to the Service domain or is there a technology that would allow to keep this data in the User domain (i.e., under the control of the user)?". A non exhaustive list of approaches, in Hoepman's words "privacy design patterns" [9], in which technology helps keeping data under user control are:

1. **Not sending the data**: perform any computation on sensitive data on the entities in the User domain and only send to the Service domain the result of these operations so that the service provider can fulfill system functionality.

2. **Encrypt the data**: encrypt data locally and send the encrypted version to the Service domain while keeping the key in the User domain. Depending on the type of encryption used it may be impossible for the entities in the Service Domain to operate with the data (e.g., using traditional encryption schemes such as AES, RSA, etc), or some operations may be possible (e.g., using advanced cryptographic techniques such as homomorphic encryption, a form of encryption that allows computations to be carried out on ciphertext generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext)

3. **Use privacy-preserving cryptographic protocols**: process data locally to obtain inputs to a protocol in which, by interacting with the entities on the Service domain, the user can obtain or prove information while limiting the information leaked to the Service domain entities.[5] Examples of these protocols are Zero Knowledge proofs [12], that allows to prove the value of an attribute without revealing the value of the attribute; Attribute-Based Credentials [10], a particular use of Zero Knowledge proofs that allows users to authenticate themselves based on the value of an attribute (e.g., being older than 18years old); Private Information Retrieval [4], that allows to perform a search on a database without revealing the query to the database holder; or Cryptographic commitments [7] that allow to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later on. We must stress that this is a non exhaustive list and there exist other protocols that allow many other operations in a privacy-preserving manner.

[5] By "limiting the leaked information" we mean that the protocol itself guarantees that no more information than intended can be inferred by the Service domain entities

4a **Obfuscate the data**: process the data locally to obtain a modified version that is sent to the Service domain. This modified version, while enabling the entities in the Service domain to provide the service, hinder inferences about sensitive data from the user.

4b **Anonymize the data**: process the data locally to remove identifiable information that is sent to the Service Domain through an anonymous communication channel (e.g., Tor [6])

The above privacy design patterns, together with the "Only collect necessary data" approach described in Activity 2 reflect the data minimization strategies described in Section 2. As shown in Fig. 3, the experts approach to identifying necessary data helps minimize collection. If the data that has to be collected, then technological alternatives can be leveraged to limit the likelihood of disclosure of data to the Service Domain and, by avoiding the flows towards the Service Domain, also limit data replication. Since the goal is to keep as much data as possible in the User domain, the above privacy design patterns inherently limit centralization, since they avoid having an entity in the Service domain that could be a single point of failure for protecting privacy of all users in the system. All of these strategies combined implicitly reduce the amount of trust the user needs to put in entities in the Service domain to safeguard her privacy, and reduce the risk that a privacy breach happens. As mentioned in Sect. 2, independently from following the mentioned strategies, it is desirable that the designer minimize the retention of data and the flow of information. This can be done using techniques orthogonal to the privacy design patterns above, such as deleting data after a given period of time.

In our study of privacy experts' engineering activities we could not identify means to decide on the best technological option, nor the data minimization strategy that should be prioritized, over others for a given scenario. However, we have found that experts usually evaluate the approaches in the order expressed above. The rationale is that given that these experts' background is usually greatly influenced by security engineering, they consider first options that give the strongest privacy guarantees (we refer the reader to Sect. 4 for a discussion on the meaning of privacy guarantee). For instance, not sending the data, or encrypting it provides stronger protection with respect to privacy than obfuscation or anonymity. Note that we have considered anonymity and obfuscation to be in the same position of the list, since both privacy design patterns offer less than perfect protection. We refer the reader to [8] for a discussion on the difference between these two approaches: hide identity of data subject, or keep identity but hide data.

While alternatives that provide stronger guarantees are in general preferred, it is not always possible to select them. Factors such as the limited functionality permitted by encryption schemes, performance of privacy-preserving cryptography, or complexity of implementation and deployment of such technologies may force experts to opt for alternatives from the bottom of the list even though they are known to provide weaker privacy protection.

Going back to our running example, when studying the data and trying to answer the question of whether it is possible to construct the system without data flowing to the Service domain one arrives to a number of conclusions. First, Personal and Billing data must flow to the Service domain so that the user can be charged for her electricity usage (though the amount of Personal data that flows can be substantially reduced, see Activities 2 and 3). Second, Transaction data is generated at the Service domain, hence it cannot be removed. Finally, with respect to Consumption data one would intuitively realize that given the computation power of current embedded systems such as the Smart Meters in the User domain it should be possible to perform the billing operations on the consumption locally. This way Consumption data do not need to be sent to the Service domain, where the Utility only needs to receive the amount to be billed in order to charge the user.

While local computation of the electricity bill at the User domain seems to be a nice privacy-preserving alternative, it introduces problems with respect to the service integrity. Since the Utility does not receive consumption data, it cannot run checks to ensure that the user has included all the records and has not tampered with them. Hence, as announced in Activity 2, more data will need to flow to the Service domain to ensure that users cannot cheat the utility provider. An example of this extra data is to rely on cryptographic commitments [13]. The system could be built in such a way that, besides the bill, the output of the User domain (i.e., the input to the Service domain) includes cryptographic commitments to the user's consumption. These commitments, while hiding the consumption from the Utility, force the user to commit to a consumption record which the Utility can request to open (i.e., reveal its content) when there is a suspicion of fraud.

Since we decided that local computation is a suitable alternative enabled by the use of commitments, then it is also possible to minimize the retention of this data by deleting them once they are processed to obtain commitments and the billing information. Ideally, from a privacy point of view this would be the best option, but deleting all data would leave the users' with no arguments to refute the bill if they do not agree with the bill provided by the Utility provider. Therefore, it is desirable to retain the data in the user domain so that the user can contest the service provider actions if needed. This again shows that extra data may have to be collected to guarantee service integrity, this time for the user. The example also highlights that even though extra data may be needed to guarantee correct and fair functioning for all parties, it is not necessary that this extra data flows to the Service domain.

The leftmost image of Fig. 6 illustrates the change in terms of information flow reflecting that the output of the User domain is not anymore the consumption; while the rightmost image reflects the distribution of data in the privacy-preserving design. Note how the inclusion of a PET in the system enables the engineer to follow the strategies: less data is disclosed to the utility, less data is replicated (Consumption data only appears in the user domain), the system is not centralized anymore; and the need for trust entities to preserve user's privacy and the risk for privacy breaches have been reduced.

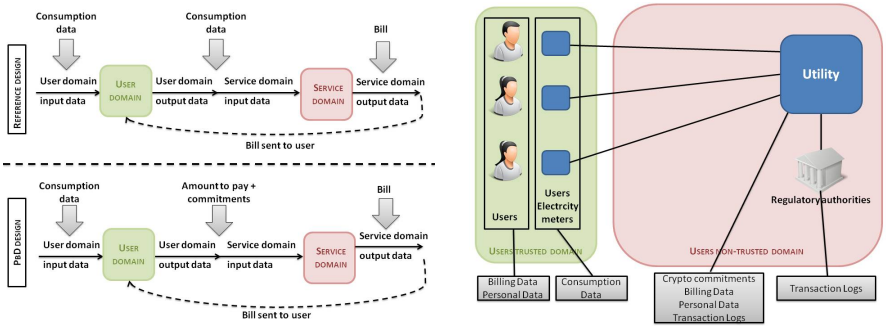

Figure 6: Electricity smart metering system: data flow modifications – reference design, above; privacy-preserving design, below (left) and privacy-preserving design (right).

4 Discussion

There are numerous points that came up as we wrote up the different activities. These discussions show that the definition of the privacy design strategies needs revision, something we leave for a future version of this paper. Another future project is to assess their differences and similarities to the privacy design strategies defined in [9]. For now, we highlight some of the topics of discussion.

The information model of any system-to-be is a matter of imagination and negotiation: The exercise of writing down how security engineers approach privacy engineering problems surfaces a conflict as to "where privacy requirements come from". When we first listed our assumptions in Section 3.1, we said we assume that privacy concerns come from the stakeholders and privacy requirements are well defined. This suggests that which information flows are permissible in the future system is defined a priori by the stakeholders. However, when we wrote up the process, it was very clear that the experts have a normative understanding of privacy, it is better if data resides in the user domain, which may not always coincide with what the stakeholders imagine when they express their requirements. The experts presuppose that privacy is best protected when sensitive user data remains under the control of the user. For many reasons, the position that these experts take here may be dismissed. Some may argue that privacy is not just about user control, others may argue that information flows should reflect social norms and not that of the engineers, yet others may raise the spectacle of the engineer whose reasoning lacks shades of grey etc. Our discussions, however, led us to think that something else is going on here.

Engineering a new system is inevitably about bringing about change into an environment: it is about imagining a possible future through the introduction of a new system. With this change in mind, a system is ideally shaped by the requirements of different stakeholders of the system, interests of prospective

users, but also through the design decisions of engineers. Hence, engineering a system is a moment of re-imagining and negotiating a new environment [6]

We see the articulation of privacy concerns and their translation into system requirements as part of this negotiation process. In this negotiation, what is technically feasible either constrains or provides possibilities to expand what can be done with state of the art technologies to protect privacy. As we develop new technologies, there will be novel ways to enable and constrain information flows. This means we can also have different ways of translating privacy concerns to system requirements, and hence that we can build different systems that protect privacy in unusual ways.

Thinking of privacy engineering as a one way road that starts with user and legal requirements, which are then turned into system specifications cuts this negotiation short and carves out the value of the engineering expertise from the process. It also doesn't make sense for non-experts to have to imagine how a system can be designed to do justice to protecting their privacy. Moreover, when imagining a new system, we tend to be limited by our mental models of what we in this paper call reference systems. This is where experts can come in and make a difference by showing other possible futures. This is also why we went through the somewhat painful process of describing what PETs experts do, so that we can come to understand other mental models and ways of thinking about privacy engineering.

The experts hence surface possible architectures and designs, the virtues of which should then be up for discussion and evaluation. The final word does not lie with the engineers, although, we heard, the final design does :)

Data in the system, information flow and inference quality: In speaking of putting constraints on information flow, we find we may benefit from making a distinction between the quantity of flow from the user to the service domain vs. the quality of that flow to enable inferences that can be considered a privacy breach. For example, let us say that a design is based on encryption, meaning that the user uploads a 200 GB encrypted file to a cloud service provider. In such a system a large amount of data, 200GB, flows from the User to the Service domain, but the quality for inference is null since encryption schemes do not leak information about the plaintext. Also, in this system the data is replicated, but the information cannot be used to make inferences about the user that may lead to a privacy breach.

In the case of privacy-preserving crypto protocols data remains mostly in the user domain minimizing the flow quantity, but still allowing some inferences to be performed in the Service domain. In the case of obfuscation, quantity of data flow may be even increased with respect to the reference information model (e.g., introducing noise) but, ideally, the inference quality is minimized. This minimization, as pointed out by Shokri et al. for the case of location privacy [14], can be multidimensional. In fact, it is the result of three vari-

[6] How the system actually performs when it is in use (or when it fails) in comparison to what we intend with the design is another issue that we leave for later.

ables: accuracy, certainty, and correctness and depending on the value of these variables, a user may have better or worse privacy protection.

If the distinction between quantity and quality of information flow matters, then it might be reasonable to indicate this distinction in the definition of the data minimization strategies. For example, the definition of disclosure may be sharpened to indicate something about inference quality, whereas replication may be better suited to indicate how much data is flowing. Whether quantity and quality can actually be separated and how to best express this difference is a topic of future discussion.

Risk and trust: The risks and trust models associated with each privacy design pattern described in Activity 4 in the previous section may also differ. Where data resides, regardless of its implications for privacy, is of importance to the minimization of risks and to minimizing the need for trust in others. In other words, while sending encrypted data flows to the Service domain minimizes privacy risks, the risk of loosing that data due to breach of availability becomes a concern. Our intuition is that greater attention needs to be given to the changes in the risk and trust in general, vs gains made in protecting privacy.

Social computing and big data: The curious reader may ask whether the data minimization strategies outlined in this paper are at odds with the era of social computing and big data. It is valid to consider whether social applications, which require a lot of data exchange for social signaling, may trump the data minimization strategies we identify in this paper. Indeed, the mental model introduced through Sect. 3 implicitly assumes that users are living in their data islands, which may make it difficult to bend this model to social applications.

We note, however, that not all services are social and in fact most services work with the same model as the Smart Energy example in Sect. 3, i.e., treating users as individuals who hold stock of their data, e.g,, banks, e-commerce sites, government sites. In these domains, it is clear that there is much to improve using the data minimization strategies. Further, data minimization strategies can also provide support to embed better privacy in social applications by, for instance, putting other users either in the user or service domain, and constraining flows accordingly. However this is just conjecture, and whether this is as simple as it sounds or other models may be more useful is also a topic of future research.

Whether data minimization strategies are at odds with big data depends on what data is of concern. PETs experts are very concerned about the current status quo of "big *personal* data" with a keen interest in controlling and influencing populations in the interest of those who hold the data. There is also a lot of mythology about the need to have all the data to make on-demand changes to systems. For example, in energy consumption, it may not be legal to change the prices all the time. The consumers need to have some way of knowing approximately what they will pay at the end of the month. In a road toll system, congestions due to disruptions, e.g. a traffic accident, should not end up as costs

for the drivers. Hence, the need to have data at such fine granularity is neither reasonable nor is it always legal. If congestion services are offered, this can be done without having to track everyone all the time. Using sensors and smart environments to find ways to improve them using big data is interesting, but should not come at the cost of people's privacy. If anything, most PETs experts demonstrate that big data and privacy is possible. Further social, economic and political implications of the use of big data in these contexts is beyond the scope of these experts.

5 Conclusion

In this paper, we summarized discussions we have had over the last three years about data minimization and privacy engineering. We aligned some of this with the terminology provided in [9] with a focus on what we call data minimization strategies. We also made a first attempt to describe the way in which these data minimization strategies are deployed in engineering activities when building privacy preserving systems.

This is clearly work in progress. Neither the definitions of the data minimization strategies nor the description of the process are complete. However, we believe we have a good draft that will hopefully help kick off a lively discussion. We identified some of these topics in Section 4. In the process, we assumed that these would apply to all sorts of systems, e.g., an app, as well as when engineering large infrastructures, an assumption that will have to be more differentiated. We also assumed that these design strategies will be applied in order to build a system from scratch. Whether the same strategies can be used to evolve an existing system is an open research question. Especially how these strategies can be adapted to service architectures needs further sharpening. There are surely many other topics that we didn't address, including where these engineering practices fit within the greater scheme of privacy by design and privacy regulation. We see all of these as great topics of future research in the nascent field of privacy engineering.

References

[1] Thibaud Antignac and Daniel Le Metayer. Privacy by design: From technologies to architectures. In Bart Preneel and Demosthenes Ikonomou, editors, *Privacy Technologies and Policy*, volume 8450 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2014.

[2] Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. Pretp: Privacy-preserving electronic toll pricing. In *USENIX Security Symposium*, pages 63–78. USENIX Association, 2010.

[3] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. Off-the-record communication, or, why not to use PGP. In Vijay Atluri, Paul F. Syverson, and Sabrina De Capitani di Vimercati, editors, *Workshop on Privacy in the Electronic Society (WPES 2004)*, pages 77–84. ACM, 2004.

[4] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[5] Claudia Diaz, Eleni Kosta, Hannelore Dekeyser, Markulf Kohlweiss, and Girma Nigusse. Privacy preserving electronic petitions. *Identity in the Information Society*, 1(1):203–209, 2009.

[6] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *13th USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[7] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[8] Seda Gurses, Carmela Troncoso, and Claudia Diaz. Engineering Privacy by Design. In *Computers, Privacy & Data Protection*, page 25, Brussels,Belgium, 2011.

[9] Jaap-Henk Hoepman. Privacy design strategies. In Nora Cuppens-Boulahia, Frdric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT Systems Security and Privacy Protection*, volume 428 of *IFIP Advances in Information and Communication Technology*, pages 446–459. Springer, 2014.

[10] Merel Koning, Paulan Korenhof, and Jaap-Henk Hoepman. The ABC of ABCs – An analysis of attribute-based credentials in the light of data protection, privacy and identity.

[11] PRIPARE EU Project. D1.2 privacy and security-by-design methodology., 2014.

[12] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Thomas A. Berson. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631. Springer, 1990.

[13] Alfredo Rial and George Danezis. Privacy-preserving smart metering. In Yan Chen and Jaideep Vaidya, editors, *10th Workshop on Privacy in the electronic (WPES 2011)*, pages 49–60. ACM, 2011.

[14] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy (S&P 2011)*, pages 247–262. IEEE Computer Society, 2011.