Adaptive Blurring of Sensor Data to balance Privacy and Utility for Ubiquitous Services [*]

Assaad Moawad
University of Luxembourg
assaad.moawad@uni.lu

Thomas Hartmann
University of Luxembourg
thomas.hartmann@uni.lu

Francois Fouquet
University of Luxembourg
francois.fouquet@uni.lu

Jacques Klein
University of Luxembourg
jacques.klein@uni.lu

Yves Le Traon
University of Luxembourg
Yves.letraon@uni.lu

ABSTRACT

Given the trend towards mobile computing, the next generation of ubiquitous "smart" services will have to continuously analyze surrounding sensor data. More than ever, such services will rely on data potentially related to personal activities to perform their tasks, *e.g.* to predict urban traffic or local weather conditions. However, revealing personal data inevitably entails privacy risks, especially when data is shared with high precision and frequency. For example, by analyzing the precise electric consumption data, it can be inferred if a person is currently at home, however this can empower new services such as a smart heating system. Access control (forbid or grant access) or anonymization techniques are not able to deal with such trade-off because whether they completely prohibit access to data or lose source traceability. Blurring techniques, by tuning data quality, offer a wide range of trade-offs between privacy and utility for services. However, the amount of ubiquitous services and their data quality requirements lead to an explosion of possible configurations of blurring algorithms. To manage this complexity, in this paper we propose a platform that automatically adapts (at runtime) blurring components between data owners and data consumers (services). The platform searches the optimal trade-off between service utility and privacy risks using multi-objective evolutionary algorithms to adapt the underlying communication platform. We evaluate our approach on a sensor network gateway and show its suitability in terms of i) effectiveness to find an appropriate solution, ii) efficiency and scalability.

Keywords

Privacy, Blurring, Component-based architecture, Software-platform, Optimization, Sensors, Trade-off.

[*]The present research is supported by the National Research Fund, Luxembourg, CoPAInS project (code: CO11/IS/1239572).

.

1. INTRODUCTION

Given the trend towards mobile computing, the next generation of intelligent ubiquitous services must be more and more able to process sensor data of surrounding environment to enable new services for end-users. More than ever, such services will rely on data potentially related to personal activities to perform their tasks, *e.g.* to predict local weather using in-house weather station sensors, or urban traffic conditions as nowadays trending applications like Waze [1]. From the perspective of service providers high quality data are supposed to be: *1)* accurate, *2)* fresh enough for the service, and *3)* diverse to reveal potential correlations. However, such data are sensitive and often related to personal activities and therefore can lead to privacy risks, *e.g. profiling* [15] and *inventorying* [9], [11]. For example, sharing the precise electric consumption of a house enables an analysis of the energetic performance, for example in order to enhance the heating system. However, recent results [15], [11] demonstrate that sharing precise consumption data also leads to privacy issues. For instance, the currently watched TV program or inventory of a house can be inferred using physical signatures (electrical consumption, frequency, ...). Similarly an high precision position or in-house activity data gave a lot of information of user habits. This privacy problem can be generalized and applied to the forthcoming IoT domain, where sensors will be accessible to various stakeholders.

Despite a user wants to preserve his privacy as much as possible, he also needs to share enough data to allow services to work properly. The challenge is how to provide added-value services and preserve user privacy at the same time. Moreover, a user might install new services at any time and thus user's privacy configuration might change over time.

Classical access control systems, which only forbid or grant access (`'all or nothing'` access), can protect the access to personal data. Similarly, algorithms like K-anonymity can ensure the protection of sensor platform privacy but require that there is no relationship between sensor data and its origin. Such string assumption can forbid numerous smart services which will need at least a geographical zone origin to perform their computation. Usually this leads to situations where third-party services either can access all data or cannot access the data at all. In contrary, blurring techniques, by gradually decreasing data quality, offer a wide range of trade-offs between ultimate privacy (sharing nothing) and ultimate functionality for service providers (sharing every-

[1]http://waze.com

thing with best quality). Blurring mechanisms allow to tune the quality of shared data for dedicated purposes, (*e.g.* by reducing the sampling rate of electric consumption to avoid foot-printing of plugged devices). Instead of having an `'all or nothing'` access to data, we suggest to use *blurring controlled data flows* between data producers and consumers to allow proportional data access control. Composed as a *chain*, blurring algorithms take raw data generated from sensors as input and produce a blurred stream as output. However, the amount of sensors and services can quickly lead to an explosion of possible configurations of blurring algorithms, which are hardly manageable for users. In order to tackle this complexity, we propose a generic software platfom that automatically reconfigures and adapts (at runtime) the blurring chain between data owners and data consumers (services). Thus, this platform searches the optimal tradeoff between service utility and privacy risks using multiobjective evolutionary algorithms in order to drive an underlying communication platform. This enhances user privacy by adapting blurring components according to a users privacy configuration and service requirements.

We aim to enable an ubiquitous sensor mediation platform in order to control data flows. In order to realize this, we combine the idea of blurring with concepts from dynamic, component based platforms, the Models@Run.time paradigm, and techniques from search-based software engineering. More specifically:

- To control the "quality" of data to match a tolerated risk level, we use a set of dynamically deployable *blurring components*. By chaining these components between a data source (sensor) and a target (service) we can ensure the required privacy properties.
- We apply Models@Run.time techniques to use a model of the current running system as a reflection layer and to drive an underlying communication platform. We use the Kevoree [6] framework, which is an implementation of the Models@Run.time paradigm, in order to dynamically adapt blurring components at runtime.
- We leverage search-based software engineering methods (MOEAs) to search the best trade-off between service utility and privacy risks, in term of blurring components configuration, according to the tolerated risk level per service and context.

The rest of this paper is organized as follows. Section 2 discusses the background. In section 3 we present our concept of proportional data access. Section 4 discusses our blurring based platform in detail. We evaluate our approach on a sensor network gateway in section 5. Finally, we end on a discussion of the related work in 6 and conclude in 7.

2. BACKGROUND

In our platform, we combine the idea of blurring with concepts of component based development, the Models@Run.time paradigm, and techniques from search based software engineering. These concepts are introduced in this chapter.

Composable software components are well-known methods to design and describe software architectures. Components are reusable architectural elements, which only communicate with each other through well-defined interfaces. Each component implements one or several well-defined functionalities. The global software architecture of a system can be built by composing individual software components. These properties make components suitable to describe our blurring solution as a chain of reusable elements.

The dynamic Models@Run.time platform follows the idea to keep at runtime (during the execution of a system) a model of the current running system as a reflection layer. This reflexion model allows not only to introspect (analyze) the current state, but also acts as an intercession layer (ability to modify a system through model modifications). In our privacy framework we use Kevoree [6], which is a concrete implementation of the Models@Run.time paradigm, in order to dynamically adapt the blurring components at runtime. It leverages Models@Run.time concepts to build, adapt, and synchronize distributed systems using an architecture view, composed of *Nodes* (execution environments like Java, Android, Phones, sensors, ...) and *Components* (dynamic software modules deployed on top of nodes). In addition to classical features, Kevoree components can use *Parameters* to adapt the behaviour of a component at runtime, like setting the blurring intensity.

Search based software engineering has been applied to various software engineering problems in order to support software developers to optimize complex tasks. Search based algorithms rely on a *fitness function* in order to compare different solutions for a specific domain criterion, like the evaluation of the privacy risk. Multi-objective evolutionary algorithms (MOEAs) are a special class of search algorithms, dedicated to deal with the fact that a solution could be evaluated not only on one but on several axises. In order to explore different solutions, MOEAs apply different heuristics, among others the NSGAII [4] algorithms. We apply MOEAs to find a trade-off between service utility and privacy risks according to the tolerated risk level per service and context, which are two orthogonal optimization axises.

3. PROPORTIONAL DATA ACCESS

Our approach aims at allowing users to control and limit privacy risks when sharing data collected from sensors. In order to make this possible, we use blurring components to dynamically adapt the quality of the shared data between real sensors and dedicated service *proxies* exposed to data consumers. In the scope of this paper, we do not consider the security of the communication between the proxies and data consumers, nor the problem of authentication. We suppose that this communication channel is secured (*e.g.* via SSL) and an authentication mechanism to identify data consumers is implemented on the platform (*e.g.* HTTPS).

Instead, in this paper we focus on the communication flow and the privacy risks, which occur when sharing high quality data. Thus, our research question is how to find an optimal chain of blurring components (and settings) between sensors and connected data consumers (*e.g.* identified as HTTP session) in order to balance user privacy and utility for ubiquitous services. Figure 1 shows an example of a blurring chain between a sensor x and a service proxy y.

Let's denote by X the total number of sensors available on a platform and by Y the total number of services (data consumers) currently connected to this platform. Each service wants to get data from a specific subset of sensors available on the platform. The data flow from a sensor $x \in X$ to a service $y \in Y$ introduces a privacy risk R_{xy}. To identify such

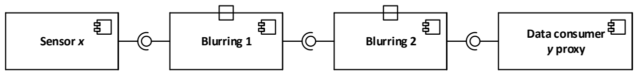


Figure 1: Chain of blurring components between a sensor and a data consumer.

privacy risks we assume the availably of a risk knowledge base, filled and maintained by security and privacy experts. Moreover, a sensor platform user (or owner) might not trust each data consumer to the same degree. Therefore, our approach foresees that a user can specify different privacy risk tolerances for different data consumer groups (*i.e.* family, friends, or strangers). This is inspired from the social network model of managing privacy [14], which define groups of different trust levels. The amount to which a data flow between a sensor and a service should be blurred depends on these risk and trust levels.

In order to counter the privacy risks we provide in our platform different blurring algorithms. All counter strategies are indexed in a knowledge base, which we call *countermeasure database*. Every counter-measure (blurring strategy) is implemented in a dynamically deployable software component. A third criteria that we include in the optimization is the performance of the dynamic communication platform. This means, besides the privacy and utility requirements we also take the resulting performance of the blurring components into account.

Given the fact that we map blurring algorithms to components, we reduce the problem to an architectural configuration. In order to select and configure such components we use an evolutionary algorithm to explore potential solutions.

Our framework is built around a reasoning engine, which reads the current state of the platform using the reflexion layer of Models@Run.time and optimizes it. In order to find alternatives our reasoning engine is fed with data requirements of currently connected services, privacy risks, and counter-measures from our knowledge base (outside the scope of this paper, for instance see the work of Neustaedter et al. [17]). Figure 2 shows an overview of the proposed architecture. Applying the Models@Run.time paradigm, this alternative solution can be deployed without interrupting the running system by updating only impacted elements.

To summarize, the proposed framework is build around a continuous adaptation loop, which monitors and drives an underlying component platform through architecture models. Our framework takes as input *(i)* privacy requirements from users (data owners), *(ii)* utility requirements (quality of data) from services (data consumers), *(iii)* risks and counter-measures provided by security experts.

Figure 3 shows an example of the state of the dynamic communication platform after a first deployment. Each service is in a different privacy group and had asked for different data sources, resulting in independent blurring chains. Our contribution is scoped between sensors and services in order to dynamically balance user privacy and utility requirements of connected ubiquitous services.

4. ADAPTIVE BLURRING PLATFORM

In this section, we describe the major elements of our proposed adaptive blurring framework.

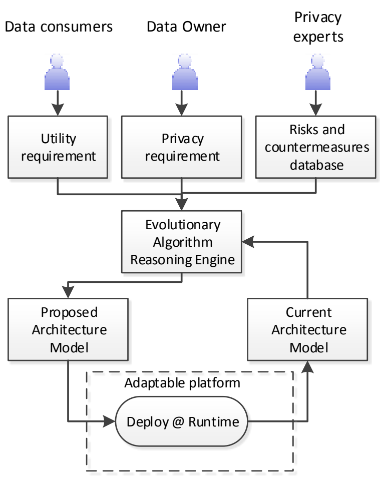

Data consumers
Data Owner
Privacy experts
Utility requirement
Privacy requirement
Risks and countermeasures database
Evolutionary Algorithm Reasoning Engine
Proposed Architecture Model
Current Architecture Model
Adaptable platform
Deploy @ Runtime

Figure 2: Adaptive blurring framework architecture.

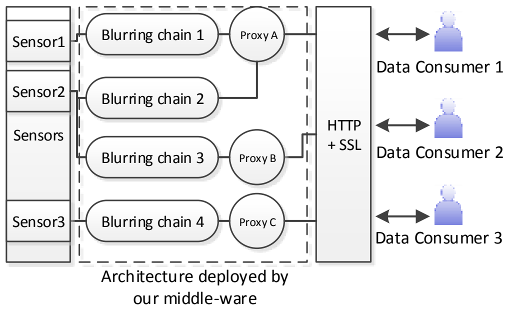


Figure 3: Example of a deployed architecture

4.1 Blurring Components

We propose to use *blurring components* in order to control the privacy risks that may occur when sharing high quality sensor data with data consumers. Under the term “blurring component” we understand a software component, which takes raw data as input and produces a blurred stream as output. The idea behind this approach is that, by gradually decreasing the data quality, a blurring component is able to hide some of the personal data delivered by sensors. However, at the same time, a blurring component still reveals enough information to allow services to work properly. For example, a service could automatically provide a user with the current weather of his whereabout. This service would need only the approximate location of the user to check the current weather in this area (typically a city). However, it is not necessary to share the exact location of the user with this service. A blurring component could take the raw data from a sensor (exact location of the user), blur these data (approximate location), and provide the weather service only with the blurred data. This would allow the weather service to fulfill its task but, on the same time, increases the protection of personal data. The *blur intensity* can be parametrized to provide different blurring levels, ranging from directly sharing sensors’ raw data without blurring to not sharing data at all. In order to blur data, different concrete blurring strategies can be applied on (raw) data. We provide two main categories of blurring components. The first category directly operates on *data values*, whereas the second category operates on the *time dimension* of data.

Time blurring components: We define three concrete blurring components in this category. The first one is *averaging.* This component takes the input data and calculates an average value (over a configurable time window). The blurring effect achieved by this component smooths and streamlines the output value. An example is the smart meter domain, where it is recommended to average the measurements over a window of 15-minutes [10]. The second blurring component we provide in this category is *frequency reducing.* By limiting the number of output values (not all input values are used as output) per time window. This decreases the data quality by reducing the frequency. An use case example is the location privacy preservation domain: if the frequency of the data is high enough, an attacker can link different locations together and trace a full route of a single user [8]. *Access control* is the last blurring component we provide in this category. The idea behind is to restrict the access to the data during specific time periods. An example is the public video surveillance domain, where the video stream can be shared or not in certain periods of time [3].

Value blurring components: For this category we define three subcategories, *noise*, *pass filters*, and *generalize. Noise blurring* effects add specific properties and variances to the original data to make them less precise. These components are usually used for real-time applications, due to their fast performance. *Gaussian blurring* is a specific type of noise blurring. It uses a Gaussian distribution model to generate the noise [5]. *Pass filters*, be it *above threshold* or *below threshold* filters, output the input data only if they pass above or under a certain threshold level or just use the boolean state when a value is above/under a defined threshold. These filters are useful, for instance, in the health care domain to share the data flow of sensors when data measurements go above or under certain medical critical limits. *Generalizing blurring* techniques reduce data accuracy, *e.g. trimming* rounds floating point numbers to a specific precision. Other generalizing techniques can include sending an interval or a range of answers instead of one value.

Blurring components can be domain independent (like a Gaussian noise generator) or domain dependent (such as location cloaking algorithms). We categorize these different blurring components using a type hierarchy meta-model. Figure 4 presents the different blurring components integrated in our platform.

4.2 Risk and Counter-Measure Model

In order to quantify privacy risks and counter-measures we use a knowledge base that has to be filled, maintained, and updated by domain privacy experts. For each sensor in the platform, these experts have to define a list of potential privacy risks that are related to that particular sensor and add a criticality weight for each of the risks. Then, different counter-measures for each defined risk have to be added.

A counter-measure for a risk consists of defining the blurring component and its risk reduction profile. For the sake of convenience and simplicity, we define the risk reduction profile by two mapping points and a mathematical profile. A mapping point links the blurring intensity to a risk reduction factor. The mathematical profile dictates the shape of the risk curve in between these two mapping points. In our implementation it can be linear, exponential, or logarithmic. In future work we plan to integrate more complex mathematical profiles to the platform.

For example, for the Gaussian noise blurring an expert can state that an intensity of 0 (variance of the noise=0) will have no impact on the risk in question (risk reduction factor=0) and an intensity of 3 (variance=3) is enough to remove 95% of the risk, thus a risk reduction factor of 0.95. Then, for example he can state that the risk reduction is linear between these two mapping points.

Figure 5 shows an simplified excerpt of the risk and counter-measure meta-model as implemented in our platform. The

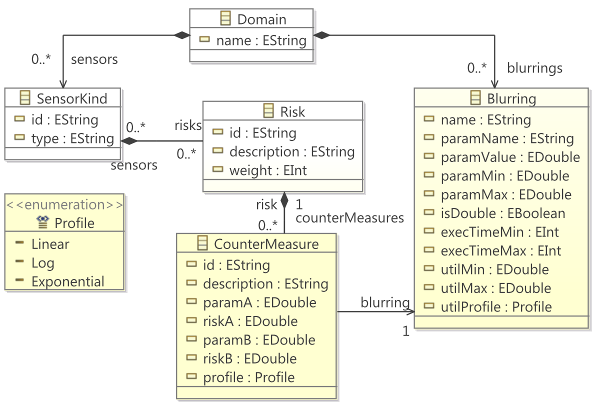


Figure 5: Risks and Counter-measures meta-model

meta-model contains five main concepts: domain, sensor, risk, counter-measure, and blurring.

Now, having this knowledge base the list of blurring components, and their intensity parameters we can derive a global privacy risk estimation. In order to do so, we calculate the weighted average of the different existing privacy risks multiplied by the maximum risk reduction factors induced from the different blurring components and their settings.

More formally: let Y be the set of all the services y connected to the platform. X_y denotes the set of the sensors connected to a service y, let R_x denotes the set of privacy risks present in the knowledge base related to a sensor x in X_y, B_{xy} the set of blurring components available on the chain between a sensor x and a service y. Let f_{ij} be the risk reduction factor of the blurring component j on a particular privacy risk i. With $f_{ij} \in [0, 1]$; 1 means that the blurring component j in its current settings does not have any effects on the risk i and 0 meaning it counter-effects the risk totally. f_{ij} is calculated using the privacy counter measure knowledge base and it depends on the current blurring configuration. The risk reduction factor f_i on the risk i is defined as being the minimum risk reduction factor on this particular risk i over all the blurring components in the chain. In other words: $f_i = Min(f_{ij}), j \in B_{xy}$. Let w_i be the weight associated to the risk i in the knowledge base. The privacy risk P_{xy} between the sensor x and the data consumer y is calculated as following: $P_{xy} = (\sum_{i \in R_x} w_i * f_i)/|R_x|$. The privacy risk induced by sharing data to service y is calculated by: $P_y = (\sum_{x \in X_y} P_{xy})/|X_y|$. Finally, the global privacy risk of the platform is defined by: $P = \sqrt{(\sum_{y \in Y} (P_y - T_y)^2)}$, where T_y is the tolerated privacy risk for the service y defined by the data owner. P will serve as the fitness function representing the privacy risk to minimize by the evolutionary algorithm. This is presented in the next section.

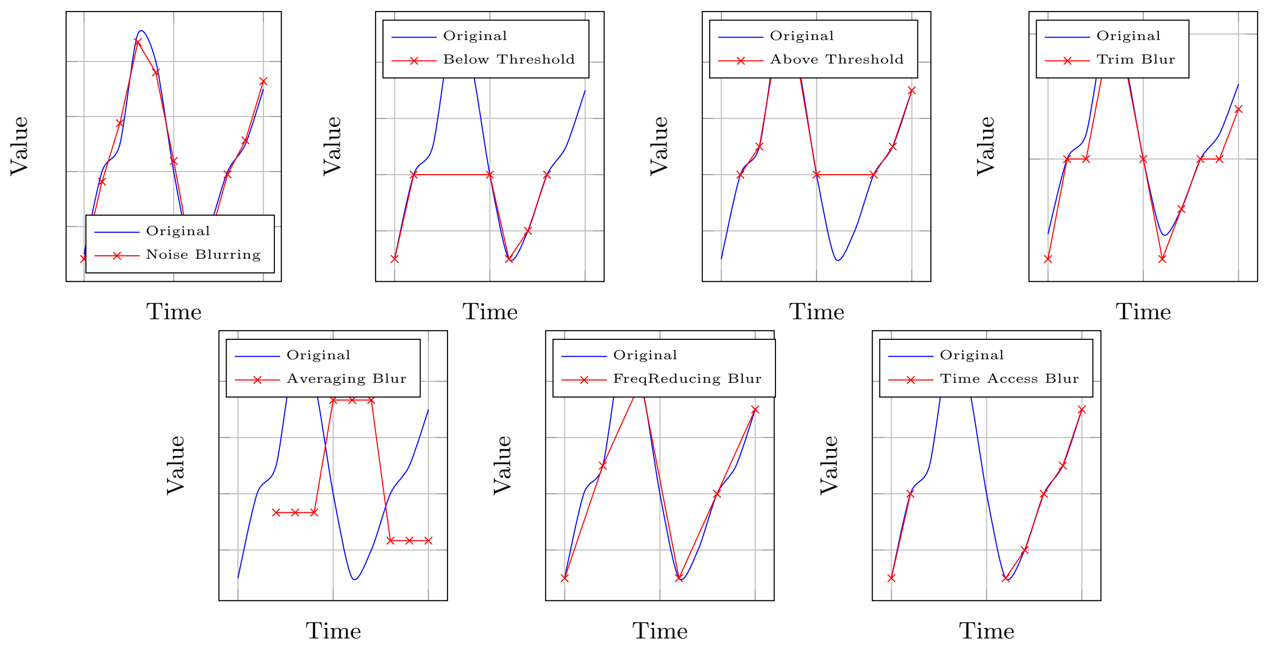


Figure 4: Different blurring techniques

4.3 Reasoning Engine

The Evolutionary Algorithm Reasoning Engine (EARE) is the core component of our platform. As depicted in the framework architecture section (section 4), it continuously monitors context changes, service requirements, and user privacy preferences. EARE leverages the reflexion ability of the Models@Run.time layer and performs adaptations through the model. Whenever a new service connects to our platform, the reasoning engine analyzes the following information: *Technical information* of the service, *e.g.* IP address, port, and user credentials. *Performance requirements* and *communication preferences*. This is inspired from Android's permission based applications [21]. *User's privacy risk tolerance* for this service or group the service belongs to and for the current context. Different *fitness functions* that represent requirements for privacy, data utility for the service, and performance. In future work we plan to integrate additional fitness functions, *e.g.* to optimize energy consumption or network quotas. Therefore, we added an *extensibility feature* to allow to provide additional fitness function implementations.

Our reasoning engine can access the architecture model representing the current running platform and modify it using Models@Run.time concepts. In order to setup a new connection for a service, the EARE first deploys a proxy and configures it with the technical configurations of the service in question. Then, a search starts to find the best path for each sensor to this proxy in terms of blurring components. The initial population is seeded with empty paths (aka all required sensors are connected directly to the proxy without blurring in the middle). At each step of the optimization a new generation of potential solutions is generated by mutating the previous generation.

We define three possible mutation operations: *1)* adding a blurring component between a sensor and the proxy, *2)* removing an existing blurring component from the chain, and *3)* tuning the intensity parameter of an existing blurring component. Using just these three mutations, we are able to generate any possible architecture of blurring components. The Java code in listing 1, shows the implementation of the third mutation operator, which randomly selects a sensor, then a blurring component attached to this sensor and mutates the corresponding intensity parameter of the blur between the range [min,max].

After the mutation step, we evaluate the generated solutions of the new population against fitness functions. We provide the following three fitness functions: First, a fitness function that uses the risk and counter-measure knowledge base to evaluate the privacy risks of a particular blurring chain. Second, a fitness function that evaluates the data consumer requirements and the utility of the data sent. And finally, a fitness function that aims to optimize the performance, thus minimize the time lag between sensors and gateways. A blurring component with long processing time will be penalized according to this fitness.

Listing 1: Example of Model-based genetic mutator

```java
class ChangeBlurSettingMutator implements
     MutationOperator<KevModel> {
  void mutate(KevoreeModel model) {
    List<ComponentInstance> ls = getSensors(model);
    //Select sensor randomly and get all the blurring
         components attached
    ComponentInstance sensor=
         ls.get(random.nextInt(ls.size()));
    List<ComponentInstance> already=
         getAttachedBlurComp(model, sensor);
    if(already.size()>0){
//Select randomly a blurring comp. and change param
      ComponentInstance cmp =
           already.get(random.nextInt(already.size()));
      double min =
           cmp.getDictionary().find("min").getValue();
      double max=
           cmp.getDictionary().find("max").getValue();
      double val = random.nextDouble()*(max-min)+min;
      cmp.getDictionary().find("value").setValue(val);
    }}}
```

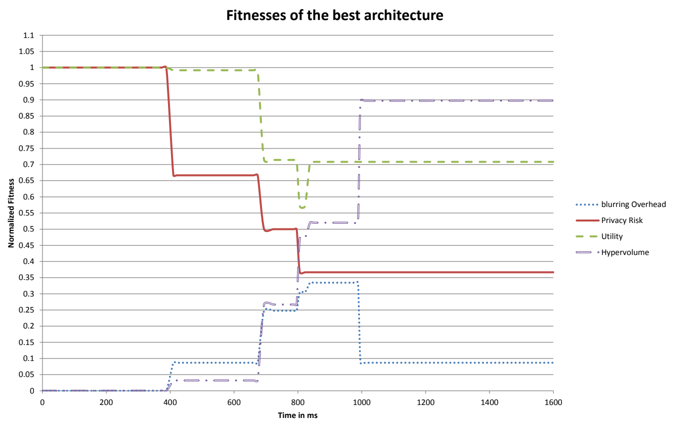


Figure 6: Evolution of the different fitness functions and the hyper-volume in time during the search

The evolution stops whenever there is a stable state of fitnesses or a maximum number of generations have been reached. At this point, several possible architectures of blurring components between sensors and services are computed. In the final surviving population each representing a different possible trade-off. Our reasoning engine selects the one with the biggest Hypervolume [23]. The last step consists of actually deploying the blurring components and the first flow of information is established from our platform to the newly connected service.

In case a new service or user connects to the platform, any user (owner) configuration changes, a knowledge base is updated, or a service preference changes the reasoning engine enters again the optimization search stages. This time, the previously deployed architecture instead of an empty one is used as initial population.

5. RESULTS AND EVALUATION

In this section we evaluate our platform on a smart home domain case study. The validation is based on the following key performance indicators: How much time does it require to connect a data consumer for a first time to the platform? How much time is needed for re-adaption in case of context or input changes? How good is the solution found by the evolutionary algorithm compared to a full search?

Setup: Our platform and implementation details are available as open source[2]. We used 4 sensors from the smart home and health care domain: a temperature sensor, humidity, heart rate, and power consumption. The privacy risks and counter-measures knowledge base were filled from known risks in literature [7] (12 counter-measures involving different components were setup). We have implemented three fitness functions: the privacy risk fitness, the utility of the shared data fitness (a measure of the information in the blurred stream compared to the original stream [2]), the blurring overhead fitness (the time in ms required by the blurring components to perform their tasks). In order to evaluate our approach, we will rely on the following metrics: The first indicator is the time required to connect a new data consumer to our platform for the first time. The second indicator is the time for re-adaptation. It is defined as

[2]https://github.com/securityandtrust/pla

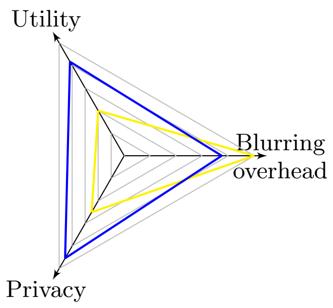


Figure 7: Trade-off between fitnesses

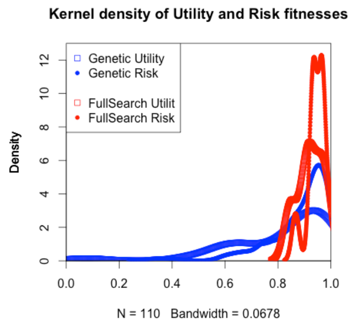
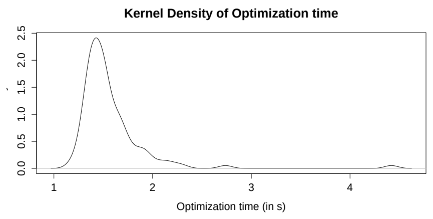


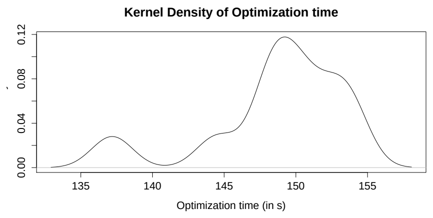
Figure 8: Effectiveness of the genetic algorithm compared to full search. On x axis, the fitnesses values (bigger is better). On y, their densities.

the time needed from the moment an input is changed (be it the context, the privacy risk knowledge base, the user's preference, or data consumer's settings) to the moment a new architecture is deployed and fully functioning. The third indicator is how well the evolutionary algorithm was able to find good trade-offs compared to a full search based approach. We perform the simulations on an Intel Core i7 2620M CPU with 8Gb of RAM.

Results: To study the performance of the platform, we consider a worst case scenario: the different consumers connected to the platform have asked all for different sensor connection requests and utility preferences. We consider as well that they all have different privacy risk tolerances by the owner. In real-life, consumers may be grouped in privacy groups, thus may have the same privacy settings. Moreover, if they have the same hardware terminals, or want to run the same service, they will require a similar utility level from similar sensors, thus reducing the load on the reasoning engine. After running 100 random tests simulating different user preferences, the average time to connect a data consumer to the platform for the first time was 1.44 seconds (varying between 1 to 2 seconds per data consumer per sensor requested). The density distribution of the optimization time is plotted in figure 9a. The average time of re-adaptation when an input changes was noticeably smaller (1 second per sensor). The evolutionary search found 80% of the time more than one solution that satisfies the privacy risk requirement within a 5% margin and 90% at least a solution within a 10% margin. By average, 4 architecture solutions were found for each problem. Each architecture represents a different trade-off between the three requirements. The solution with the highest hyper-volume is the



(a) Execution duration distribution for our reasonning engine



(b) Execution duration distribution for a full search

Figure 9: Execution duration comparison between our approach and a full search

one we actually deploy. However, the reasoning engine performed 15% less than a full search in optimizing the utility and privacy risk fitnesses (fig 8), but the full search took much more time (150 seconds per consumer per sensor, fig. 9b). Figure 6 shows one run of the evolution in time of the different fitness values of the best architecture model. For this example, we set the tolerated privacy risk level of the new user to 0.4 and the utility requested to 0.7. Initially, the evolutionary search starts with an empty architecture (with no blurring components). Thus, the initial risk fitness is 1, as well as the initial utility of the data (no blurring), and the initial blurring overhead is 0 (no performance delays caused by the non-existence of blurring components). After each generation, the risk fitness function is driven to the requested value of 0.35, which means the addition of new blurring components in the architecture. This reduces the utility of data and increases the blurring overhead. In figure 7, we plot the normalized values (between 0 and 1, higher is better, 1 means that the fitness has reached the desired target) of the 3 fitnesses at the initial condition (in yellow) and of the chosen architecture with the best hyper-volume (in blue). We can notice that the blurring overhead fitness has decreased from 1 (no blurring overhead, best performance) but both utility and privacy fitnesses have improved to reached their targets within 10%.

Discussion: The performance of our platform and its real utility, usefulness, and value depends on the privacy risks and counter-measures knowledge base and the fitness functions provided to the evolutionary algorithm. At one extreme, if this knowledge base is empty, the genetic algorithm will not deploy any blurring between any sensor to any data consumer. On the other hand, the more complex the knowledge base, the more time the evolutionary search engine will need to find a suitable architecture. This can be problematic in case of an emergency situation, to take some time before re-adapting. A simple solution is to pre-calculate and store the architectures to deploy for emergency cases. This dramatically drops the needed time for re-adaptation. Besides, the current privacy risk and counter-measure model treats each sensor independently: each sensor has its own set of possible privacy risks independently of the other sensors. We did not yet include privacy risks that occur when sharing information with a data consumer from two or more sensors at the same time. This will be included in a future work.

6. RELATED WORK

Blurring techniques, generalization, and obfuscation-based privacy have been used in literature, mostly to protect the anonymity of a user inside a group. K-anonymity [20], l-diversity [13], and t-closeness [12] are the most famous and widely known approaches in this area. These approaches protect single user data by averaging or aggregating information with other users in groups of a minimum size k. Our platform has a different aim. Our target is not to preserve the anonymity of the user towards services. Both can be well identified to each other. Instead, we target to protect against privacy risks such as: inventorying, profiling...

Many researches have developed blurring components and proposed them as privacy preservation solutions. In location-privacy domain, the Casper platform [16] contains a component called “Location anonymizer”. This component blurs the users’ exact location information into cloaked spatial regions based on user-specified privacy requirements. In the smart grid domain, Rajagopalan et al. [18] proposed a framework that abstracts both the privacy and the utility requirements of smart meters. They defined a power frequency region where the trade-off between utility and privacy exists. In video surveillance, Wickramasuriya et al. [22] presented a tool that processes video at realtime and decides when to share the video stream with the surveillance system and when to share at lower resolutions, in order to preserve privacy. These previous works enforce our main idea that privacy preservation tools in ubiquitous environments can be modelled as blurring components. Our work is highly inspired by these techniques, and all of these blurring components can be seamlessly integrated into our platform. The main focus in this paper is the dynamic aspect of mobile ubiquitous services and therefore how to propose a dynamic software platform and combine blurring techniques to dynamically adapt the level of data privacy.

In pervasive environments and ambient assisted living domain, the CONNECT platform [1] presents a context-sensitive privacy management middleware. This platform is the closest to our work but it is based on defining a privacy policy. Their approach is influenced by the scheme adopted in the XACML architecture [19]. Although our platform also offers a context-sensitive privacy management framework, it is different in the core that it uses an evolutionary algorithm based on a reasoning engine as a main privacy decision point and not predefined privacy policies. We use a multi-objective optimization solution where privacy is not the only requirement to take into account. In addition we leverage what we called a proportional access control, which enables more security levels of sensors data by using various blurring techniques to decrease the quality before sharing it.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented a concept of proportional access based on blurring techniques in order to dynamically balance privacy and utility for ubiquitous platform services. We showed that this generic concept can be used in various domains, such as ambient assisted living and demonstrated its usefulness for sensor communication platforms. Our platform applies Models@Run.time as a main paradigm to enable a dynamic reconfiguration whenever the context, configuration, or requirements changes. A new architecture model is derived using an evolutionary multi-objective reasoning engine, which searches the most appropriate solution of blurring components and parameters that best satisfies the requirements regarding privacy, data utility, and performance. This allows sensor platform owners to share their data in a controlled manner with data consumers.

Any change in the requirements will trigger the evolutionary search to dynamically re-adapt the platform and the blurring components (at runtime) to match the new requirements. The privacy risk evaluation in our platform relies on a privacy risk and counter-measure knowledge base. Many previous research results in this domain can be integrated in our knowledge base. For instance, privacy in video surveillance [22], [17], smart metering [18], and location data [16].

The evaluation of our platform showed that the overhead introduced by our approach is around 1.5 seconds per user connected and per sensor. The data utility and the blurring overhead were optimized to around 85% of a full search algorithm but is 100 times faster.

In future work, we plan to extend our platform to evaluate the potential privacy risk for combinations of several sensors connected to a data consumer (mutual information). Additionally, we plan to integrate more complex risk reduction profiles than the ones we currently provide (linear, exponential, and logarithmic) and see how we can integrate additional fitness functions, *e.g.* to optimize energy consumption or network quotas.

8. REFERENCES

[1] S. Alcalde Bagues, J. Mitic, and E.-A. Emberger. The connect platform: An architecture for context-aware privacy in pervasive environments. In *Security and Privacy in Communications Networks and the Workshops. SecureComm 2007. Third International Conference on*, pages 117–126. IEEE, 2007.

[2] L. Bhuvanagiri and S. Ganguly. Estimating entropy over data streams. In *Algorithms–ESA 2006*, pages 148–159. Springer, 2006.

[3] L. BROWN, Y.-L. TIAN, A. EKIN, C. F. SHU, and M. LU. Enabling video privacy through computer vision. 2005.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[5] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006*, pages 486–503.

[6] F. Fouquet, O. Barais, N. Plouzeau, J.-M. Jézéquel, B. Morin, and F. Fleurey. A Dynamic Component Model for Cyber Physical Systems. In *15th International Symposium on Component Based Software Engineering*, Italy, July 2012.

[7] M. Friedewald, E. Vildjiounaite, Y. Punie, and D. Wright. Privacy, identity and security in ambient intelligence: A scenario analysis. *Telematics and Informatics*, 24(1):15–29, 2007.

[8] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2008.

[9] A. Juels. Rfid security and privacy: A research survey. *Journal of Selected Areas in Communication*, 2006.

[10] G. Kalogridis, Z. Fan, and S. Basutkar. Affordable privacy for home smart meters. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 77–84. IEEE, 2011.

[11] D. A. Kelly. Disaggregating smart meter readings using device signatures. 2011.

[12] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, volume 7, pages 106–115, 2007.

[13] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[14] M. Madden. Privacy management on social media sites. *Pew Internet Report*, pages 1–20, 2012.

[15] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security and Privacy*, 7(3):75–77, May 2009.

[16] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[17] C. Neustaedter, S. Greenberg, and M. Boyle. Balancing privacy and awareness for telecommuters using blur filtration. Technical report, Report 2003-719-22, Department of Computer Science, University of Calgary, 2003.

[18] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor. Smart meter privacy: A utility-privacy framework. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 190–195. IEEE, 2011.

[19] O. Standard. extensible access control markup language (xacml) version 2.0, 2005.

[20] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[21] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 31–40. ACM, 2012.

[22] J. Wickramasuriya, M. Alhazzazi, M. Datt, S. Mehrotra, and N. Venkatasubramanian. Privacy-protecting video surveillance. In *Electronic Imaging 2005*, pages 64–75. International Society for Optics and Photonics, 2005.

[23] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876. Springer, 2007.