UPRISE-IoT July 2019

SPIoT Book

UPRISE-IoT: User-centric Privacy & Security in the IoT

Silvia GIORDANO [a,1,2], Victor MOREL [b,2], Melek ÖNEN [c,2], Mirco MUSOLESI [d,2], Davide ANDREOLETTI [a], Felipe CARDOSO [a], Alan FERRARI [a], Luca LUCERI [a], Claude CASTELLUCCIA [b], Daniel LE MÉTAYER [b], and Cédric VAN ROMPAY [c]

[a] *SUPSI, Switzerland*
[b] *Inria, France*
[c] *EURECOM, France*
[d] *UCL, U.K.*

Abstract. Data privacy concerns the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others [1]. However, most people are not aware of privacy and security risks. This is particularly relevant for the Internet of Things (IoT) which is increasingly ubiquitous and thus can become very privacy intrusive. On one side, this problem can slow-down IoT development, on the other side it is essential to make users gain control over data generated and collected by the IoT devices surrounding them, and to adopt a privacy-by-design approach for the IoT. UPRISE-IoT takes a fresh look at the IoT privacy space by considering a user-centric approach. It builds upon user behaviours and contexts to improve security and privacy, and follows the privacy by design approach. UPRISE-IoT improves data transparency and control, as users are informed about the data that are being collected in a user-friendly manner and have the option to oppose it. We raise user awareness, to ensure that their behaviour does not compromise their privacy, and we provide new tools to control data collection in the IoT.

Keywords. privacy awareness, user-centric, privacy-by-design,

1. Introduction

Privacy is an open issue, still obscure for most people, despite the current huge attention it got in relation to the recent stories of Cambridge Analytica and Facebook [2], or Google [3], the new GDPR regulation [4] as well as the discussions in the European Parliament on regulation of the digital market [5]. Even if new rules have been developed, people are still not aware of the value of their data, of how they are handled and used, and of their rights. UPRISE-IoT's goal is to make users gain control over data generated and collected by the IoT devices surrounding them by adopting a user-centric, privacy-

[1]Corresponding Author: Silvia Giordano, Institute of Informatics and Networking Systems (ISIN) - DTI, Univ. of Applied Science and Arts - SUPSI, Switzerland; E-mail: silvia.giordano@supsi.ch.
[2]Main Author

by-design approach for the IoT. The UPRISE-IoT[3] methodology consists in the design and development of algorithms and solutions to build such control, in order to put the basis for a new understanding of data privacy [6], [7], and to improve transparency and control. Users should be aware of the data collected by the IoT. Transparency helps users to make informed choices about the IoT services use. Furthermore, within the project it has been discovered that even when a user has blocked access to her personal data it is still possible to derive some sensitive information of her from data of her friends and acquaintances [8]; so, a more global awareness is needed for a real privacy. To this aim, UPRISE-IoT considers user behaviours and the user context to increase security and privacy, and realizes privacy preserving data collection and processing to that end.

Such user-centric security and privacy space tailored for IoT and for IoT users, is based on the following multi-disciplinary elements:

- *behavioural models* extracted from the large amount of data generated by IoT devices for security and privacy applications;
- *advanced functionalities and algorithms* for increasing security and privacy by design;
- *tools for empowering users* in terms of transparency and awareness;
- *open libraries* for supporting user control (preferences, filters, accounting and analytics) and transparency;
- *strategies to secure IoT devices* for M2M authentication systems and encryption libraries.

The result is a new secure space centred around the user where security solutions are either integrated within IoT devices directly (creating smart secure objects) or interfaced to the user by a powerful user-friendly app for: (i) smartifying the IoT devices which are not intrinsically secure (creating smartified secure objects), (ii) fine-tune the level of privacy; (iii) getting awareness of her behaviour for being protected from security and privacy threats, (iv) getting awareness of the value of her information. The UPRISE-IoT project has obtained very promising technical and scientific results, and also applied them to very relevant scenarios such as mobile phones and smart cities. As discussed in the next sections, we provide experimental work involving users that validate this approach.

2. Privacy Awareness and Control by Design

Privacy by design principles are fundamental in the security landscape. As UPRISE-IoT mainly considers user awareness and control over personal data, we focus on the following well-known privacy by design principles: data minimization (i.e., only data that are necessary for a given purpose can be collected), purpose limitation (i.e., the purposes for which data are collected must be specific, legitimate and unambiguously known to the final user) and data de-identification (i.e., data must not reveal the identity of its owner). All these principles require the user to be aware and in control of her data, which is a very challenging problem in the IoT ecosystems, and in particular in the mobile phone and smart city scenarios considered in the project, since human interaction is not as easy as with traditional IT systems.

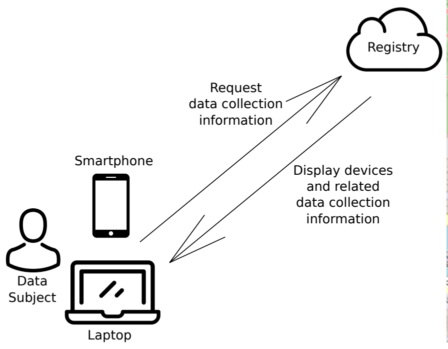
[3]http://uprise-iot.supsi.ch/

Thus, to make them applicable for user awareness and control in IoT, we have developed several experiments to assess the degree of privacy required by the data that are collected and to map the degree of privacy that the user wants for the data collected.

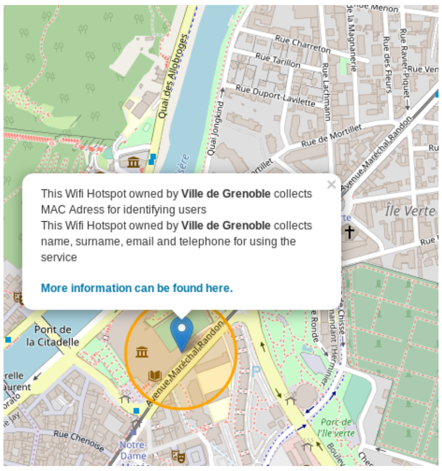
2.1. A registry for the smart city

The smart city is a paradigm which consists in fielding numerous devices in urban areas to provide new services, or to improve existing ones. Some of these devices collect personal data, and their ubiquity combined with growing data analysis techniques can lead to privacy issues [9]. Raising awareness in this context is difficult as these devices 1) do not necessarily have appropriate user interfaces to display intelligible information and 2) may lack the computational capacities or the required network interface to declare themselves. For instance, stores use Bluetooth beacons to track customers to provide targeted advertising [4] ; and municipalities often provide Wi-Fi hotspots to denizens, and these Wi-Fi access points collect personal data.

The General Data Protection Regulation (GDPR) requires data controllers to intelligibly inform data subjects of data collection and processing. Therefore, stickers or wall signs can be considered as non-compliant to the extent that they remain unnoticed from most data subjects. To tackle this transparency issue, we propose a registry for the smart city : Map of Things.



(a) Explanatory diagram



This Wifi Hotspot owned by **Ville de Grenoble** collects MAC Adress for identifying users
This Wifi Hotspot owned by **Ville de Grenoble** collects name, surname, email and telephone for using the service

More information can be found here.

(b) Screenshot of the interface

Figure 1. A registry for the smart city

2.1.1. Map of Things

The registry Map of Things raises privacy awareness by providing information about personal data collection and processing in a user-friendly manner. It consists in a website [5] on which data controllers can declare any device collecting personal data, regardless of

[4]`www.nytimes.com/interactive/2019/06/14/opinion/bluetooth-wireless-tracking-privacy.html`
[5]Available at `https://mapofthings.inrialpes.fr/`

its capacities or interfaces. The registry can be accessed by any device owned by data subjects, such as a smartphone or a laptop (see Figure 1a). It provides a summary of the privacy policies as well as a link to the full version of the privacy policy (see Figure 1b). It also provides further information which is of prime importance in ubiquitous environments such as the geolocation of devices, their range and frequency of collection. This information is available in a graphical format, and through an API for interoperability purposes. [6] The registry Map of Things has been tested in Grenoble and La Rochelle. The registry hosts Wi-Fi hotspots privacy policies communicated by the municipalities. [7]

2.2. Mobile Apps and User's Data Collection

In the past few years, smartphone applications have greatly contributed to the quality of users experience in their smart devices, allowing them to reach a set of functionalities even larger when the user's personal information is used. However, many applications fail to guarantee user privacy; this is especially true for Android smartphones where the Application Market is self-regulated. The mechanism that Android offers to increase user privacy is the use of system permissions: every time an app is downloaded from the market and the installation process is started, Android controls the app's permissions and then asks the user to allow them. If the user does not agree, the installation process is interrupted. This system permits users to know what information is used and which entity processes it, but it does not indicate when and how the information is used inside an application. For instance, if an application proposes a game that uses sensor data and secretly records it, the sensor data could be used for malicious intents without the user's awareness. Novel regulations (such as the GDPR [4]) require informing the user about the data accessed and collected by the app. However, still users are not informed enough to decide whether a data access in an app is a potential threat or not. To show this point, we made some measurements of users awareness about the potential risks of apps installed on their devices.

2.2.1. The UPRISE-IoT app

We developed an Android application, the *UPRISE-IoT app* that monitors the behaviour of the other apps installed on the users device. The UPRISE-IoT app also provides two distinct interaction with the device owner:

- It informs the owner about a potential risk a given app has, considering a specific permission.
- It asks the user if she is aware of those risks and (in case she is not) if her perception of the app is changed.

The UPRISE-IoT app collects the following data:

- The privacy level of the apps installed on the users device: we collect the permission granted to the app and classify them according to their level of risk.
- The level of user awareness: we periodically show to the user a potential risk connected to the data accessed by apps, and ask if she is aware of that and if her perception of the app is changed with this information (in terms of privacy).

[6] `https://mapofthings.inrialpes.fr/api/devices`
[7] As of July 2019.

We perform two rounds of experiments: the first one with a selected group of 17 users from our university, and a second one with 50 users recruited via internet. The two groups are both very heterogeneous in terms of gender, technological skills and age, as indicated in Table 1.

Table 1. Population distribution in the two experiments.

	Experiment1			***Experiment2***		
total population	17			50		
gender	**M**		**F**	**M**		**F**
	11		6	26		24
IT knowledge	**high**	**medium**	**low**	**high**	**medium**	**low**
	11	5	3	9	27	24
age	**18-30**	**31-40**	**>41**	**18-30**	**31-40**	**>41**
	5	8	4	20	16	14

2.2.2. Experimental Results

The permission distribution of the observed app within the two experiments is shown in Figure 2. We can see that the majority of the app requires numbers of permission between 0 and 7 with mean around 2. However, there are some outliers that requires an astonishing number of dangerous permission, permissions that could potentially affect the user's privacy or the device's normal operation, up to 16. To determine whether a user is aware of such potential risks, we provide a notification of the potential risk in the given application whenever we notice that the application is under execution. The notification contains the app name, the permission it accesses and a list of potential risks connected to this access. We then ask the user if she was aware of the app dangerous accesses, and, in case she was not, if her perception of the app has changed after being informed of such information. We also notice that in the majority of the cases (more than 60% in both experiments) the user has no idea of the risks related to such app. We further ask them if their perception of the app and of its dangerousness was changed after the notification, and we received a positive answer in more than 80% of the cases for the Experiment1, and in about 50% of the cases in the Experiment2.

Our experiments demonstrate that there is no much user awareness and control on the data accessed by the mobile apps, even if it improved during the last years. Users tend to accept all the access requests from an app during the installation, and then they do not control it anymore. We demonstrated that, if we provide awareness to the users, they are taking care of that. Therefore, in the next release, we want to provide them with some guide about how to control such permissions, and a direct link to the android app permission for performing some modifications, if they are interested.

3. Privacy policy and Consent management

This section presents a generic framework for information and consent in the IoT. The framework is generic in the sense that the proposition relies on few technical require-

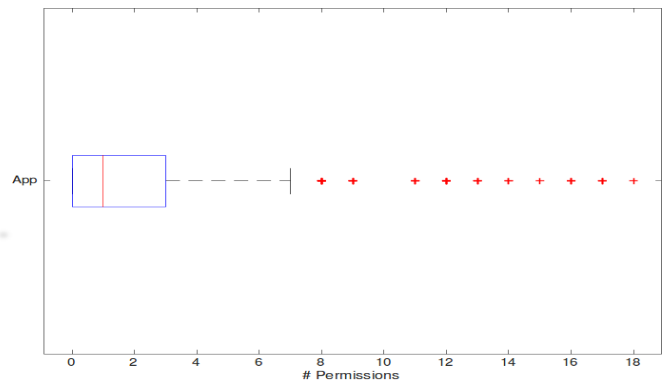


Figure 2. Distribution of the permissions required by the installed users apps (591 distinct apps)

ments — we do not consider protocols used, types of devices, nor fielding configurations — it can therefore be adapted to numerous contexts. The framework uses privacy policies to provide information and manage consent. The framework consists in: PPNP, a protocol to communicate these privacy policies (Section 3.2), an agent to act on behalf of users that we denote Personal Data Custodian (PDC), and a set of requirements to demonstrate the obtention of consent. The framework refers to previous work [10], and the interested reader will find descriptions on the PDC and the proof of consent in [**?**].

We will set out by presenting the hypotheses of the framework in Section 3.1.

3.1. Hypotheses

The framework proposed in this section relies on certain hypotheses: it requires **devices** possessing certain **features**, which communicate by means of specific **messages**.

3.1.1. Devices

Data controllers need a Data Controller Gateway (DSG), a device able to communicate information (the content of which will be described thereafter) and to retrieve consent; one or many Data Controller Devices (DCG), devices owned by a data controller and collecting personal data; and a Consent Storage Server (CSS), a machine on which consents are stored. Data subjects (DS) need a DSG (a device able to communicate information and to communicate consent), and possibly Data Subject Devices (DSD), devices owned by a DS and potentially subject to data collection. Figure 3 provides an explanatory diagram of the different entities and of their possible interactions.

3.1.2. Features

A DSG must possess a Data Subject Policy (DSP), as well as a DCG must possess a Data Controller Policy (DCP). The privacy policies must be machine-readable, *i.e.*, in a structured format interpretable by a machine, and it should be possible to perform the operations described thereafter.

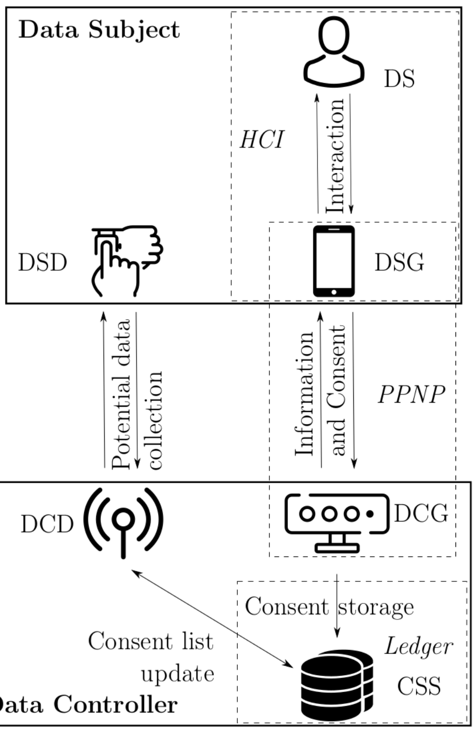


Figure 3. Explanatory diagram of the framework. Entities and their interactions are denoted with normal font, components of the framework are denoted in *italic*, and the two sides of the framework are denoted in **bold**.

A DSG must be able to hash files, to store cryptographic keys, and to sign messages with the keys. Similarly, a CSS must be able to hash files, to store cryptographic keys, and to sign messages with the keys.

A DSG must have an appropriate interface. We mean by appropriate interface: *a touchscreen or the combination of a screen and a keyboard, allowing for the human-computer interactions described in [**?**]* (consultation of privacy policies, management of DSPs, and notifications to the DS).

3.1.3. Messages

Privacy policies As described in Section 3.1, DSGs and DCGs must respectively possess a DSP and a DCP. A DSP can be seen as a set of *a priori* consents. Privacy policies must be machine-readable, and therefore be expressed in a privacy language. A suitable privacy language for this facet should meet requirements over the syntax (the content expressible), and over the operations permitted. Any language meeting these requirements could be used in the framework.

Required content Privacy policies can be represented as a set of *rules*:

$policy ::= (rule_1, rule_2 \ldots rule_i)$

The mandatory part for a rule in this framework is the type of data, the purpose of collection, the retention time (in days), as well as the DC concerned.

Operations Privacy policies are required to allow for certain operations.

A DSP is required to be comparable to a DCP, and the result is required to be deterministic. For instance suppose a DSP and a DCP. Then either $DSP \geq DCP$ or $DSP \ngeq DCP$. It is said that the DCP matches the DCP if and only if a $DSP \geq DCP$, *i.e.* the DSP is less restrictive than or equal to the DCP.

Two privacy policies are required to be intersectable. We denote the intersection operation $\cap$. In other words, given two policies DSP and DCP, the result of $DSP \cap DCP$ is DCP_2 such as $(DSP \geq DCP_2) \wedge (DCP \geq DCP_2)$. The result is a DCP. If $DSP \geq DCP$, then the intersection $DCP_2 = DCP$.

Consent A consent is the authorization from a DS to a data controller to collect data and use it according to a DCP. A consent consists in: a hash of a DCP, the set of identifiers concerned by data collection, and a cryptographic signature for the authentication. A consent should be communicated in both plain text and signed — *i.e.* encrypted with the DS cryptographic private key.
Dissent A DS can also withdraw a consent by communicating a *dissent*. A dissent is similar to a consent except that it is related to a *nil* privacy policy.

Other messages The protocol described in the next section uses other messages, that we present here. Devices can also communicate the following messages:

Refusal It is a message sent by a DSG to a DCG to inform that no negotiation is possible
Deny It is a message sent by a DC to a DCG to inform that the intersection is denied
Accept It is a message sent by a DC to a DCG to inform that the intersection is accepted

3.2. Privacy Policies Negotiation Protocol

The protocol consists in two phases: **information**, and **consent**. What we refer below to *negotiation* is part of the communication of consent. PPNP ensures that a consent can be communicated if and only if 1) the DSG has received the DCP, and 2) the DSP matches the DCP.

3.2.1. Informal description

We provide an informal description of the protocol to intuitively illustrate its functioning.

Information The DCG initiates the communication by providing the DCP. The DCP is received by the DSG.

Consent and negotiation Upon reception, the DSG compares the DCP and the DSP and issues a consent message if they match.

Otherwise, the DSG prompts the DS, sets a timer, and awaits for a new DSP from the DS after having requested it.

After the input of the new DSP from the DS (or the expiration of the timer) the DSG checks whether the DSP matches the DCP:

- The policies match (*i.e.*, in other words, the DSP is less restrictive than the DCP), in that case a **consent** for the DCP is issued.
- The policies do not match and their intersection is null (*i.e.*, the DC and the DS cannot agree on any data collection according to their current privacy policies), the DSG issues a message of **refusal**.

- The policies do not match but their intersection is not null (*i.e.*, they can agree on terms of data collection and processing), in that case the DSG sends the **DSP**.

The DCG, after having sent the DCP, awaits for answers from the DSG.

- If the DCG receives a consent, this consent is forwarded to the CSS.
- If the DCG receives a refusal, it continues providing the DCP.
- If the DCG receives a DSP, it requests its DC for permission to use the intersection policy between the DCP and the DSP, and sets a timer. The DC can either **accept** or **deny** this intersection policy. If it is accepted, the intersection is sent to the DSG, and the DCG awaits for a consent, a refusal, or a new DSP. Otherwise (a denial or the expiration of the timer), the DCG goes back to providing the DCP.

The DS can withdraw a consent at any moment by sending a dissent message to the DCG. The CSS accepts only two messages: consent and dissent.

A DCD should query the CSS regularly — or it can be updated by the CSS — to maintain an up-to-date list of consents.

3.2.2. State diagrams

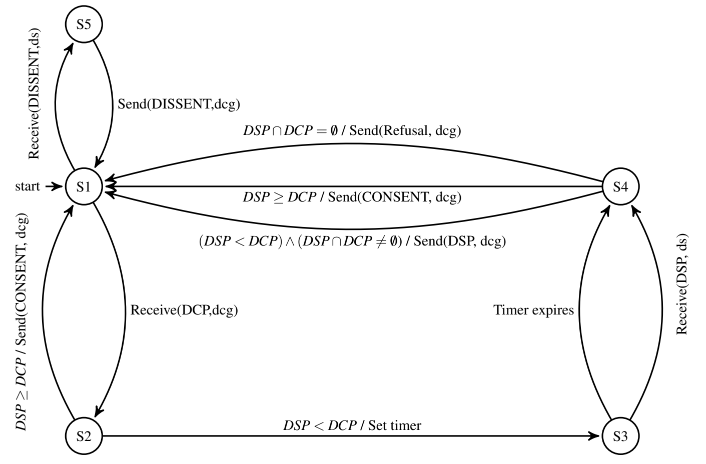

S5
Receive(DISSENT,ds)
Send(DISSENT,dcg)
$DSP \cap DCP = \emptyset$ / Send(Refusal, dcg)
start
S1
S4
$DSP \geq DCP$ / Send(CONSENT, dcg)
$(DSP < DCP) \wedge (DSP \cap DCP \neq \emptyset)$ / Send(DSP, dcg)
$DSP \geq DCP$ / Send(CONSENT, dcg)
Receive(DCP,dcg)
Timer expires
Receive(DSP, ds)
S2
$DSP < DCP$ / Set timer
S3

Figure 4. DSG state diagram

This section presents the state diagrams of the different entities. Entities are denoted with **lower cases**. Variables are denoted with **UPPER CASES**, other messages with regular **Title Cases**. We first describe the states and transitions used by the state diagrams; then the state diagram of the DSG (Figure 4) and the DCG (Figure 5) as directed graphs.

States and transitions States are configurations holding information. They can hold the following information: **DCP**, **DSP**, CONSENT, and DISSENT. Transitions can be triggered by events, or by meeting conditions. Transitions are represented by arrows going from a state to another.

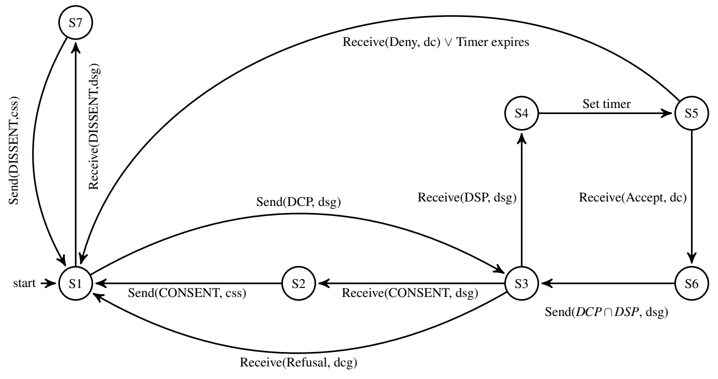


Figure 5. DCG state diagram

Conditions Conditions are boolean tests over the information held by the state. An action is performed if the test evaluates to *true*. Our systems are deterministic in the sense that: given a state, there always is only one transition such that its condition evaluates to *true*. Conditions are denoted before the / character. If no condition is stated, then the system only considers actions. A timer will eventually expire in the case where no action may be performed.
Events Our state diagrams consider two types of events: internal and external. Internal events trigger external events.

Some events are internally triggered by the entity

- Send(A,b) where A is a information communicated to another entity b. Send() triggers a Receive() in another entity
- Set timer. A timer set will eventually expire, triggering a passive function

Some events are triggered by external entities, or by the expiration of a timer

- Receive(A,b) where A is the information received by another entity b. It implicitly changes the configuration of the entity according to the information
- Timer expires

4. Privacy preserving data management

With the interconnection of large numbers of IoT devices and the explosion of the amount of (often personal) data exchanges between these devices and other systems, the need for data privacy solutions become crucial. Such data can originate from some cameras feeds or from smart meters and therefore can be considered as highly sensitive. During the data exchanges, control by individuals can easily be lost because users usually cannot trace how data are flowing among devices or are even not aware of the processing of data. Hence, the IoT technnology might be slowed down if privacy is not examined as an essential criterion in designed protocols. UPRISE-IoT hence adopts a privacy-preserving data management approach for the data collected from the IoT devices and processed afterwards. The key challenge raised by this new environment is the fact that the massive amount of data usually needs to be processed in real time. When analysed it provides valuable information for stakeholders who could further improve their services. Hence, one of the key functionalities of an IoT platform is the incorporation of data analytics solutions. Unfortunately, as previously mentioned critical privacy issues are raised when analyzing and extracting some information about this sensitive data. Existing privacy-preserving data analytics solutions either are not scalable or unfortunately leak some information to centralized entities. Therefore, there is a strong need for novel privacy mechanisms that would help end-users gain back the control over their data. Another challenge that has to be taken into consideration is the multiplicity of data sources and data queriers. Indeed, in the IoT environment, all devices become data producers and these data can be queried by multiple stakeholders that we could name data queriers. In Uprise-IoT, we focus on one of the most used analytics operations that is the word search operation and we describe privacy preserving variants of search operations that allow third parties to process search queries without having access to the content of the data. This section presents the problem of multi-user searchable encryption that is more suitable to the IoT environment and then overviews the state-of-the-art and two newly developed solutions.

4.1. Multi-User Searchable encryption

UPRISE-IoT aims to develop new privacy primitives that allow the outsourcing and the search of data while being encrypted. Searchable Encryption (SE) [15] enables searching on data hosted on a third-party server (such as a cloud server), while protecting the privacy of the data and the queries that are executed against the potentially malicious server. More precisely, a SE protocol involves a user (the data owner) who encrypts her data with a key that is only known to her, then uploads the encrypted data to a potentially malicious cloud server. The user can later request the cloud server to perform some search operations by sending privacy preserving search queries. The main privacy guarantees that a SE scheme should provide are mainly the privacy of the data (and keywords) and the privacy of the queries. Ideally, a SE should not leak any information about the search pattern (i.e. the information about whether any two queries are generated from the same keyword or not) and access pattern (i.e. the information about which data segments contain the queried keyword for each of the user queries).

In an IoT environment, we are in the context of a multi-user setting [16] whereby each querier (data consumer) may have access to a set of encrypted data segments stored by a number of different owners (IoT devices). The quantity of collected data have been persistently increasing while IoT devices' storage and computational capacities are very limited, making storing and processing data difficult on devices' side. Outsourcing storage and computation of data to a third party (i.e. a Cloud Service Provider (CSP)) with much greater storage, computational and network capacities is made possible with cloud computing. In order to ensure data privacy while delegating the storage and search operations to the CSP, Multi-User Searchable Encryption (MUSE) appears to be the solution. A multi-user searchable encryption scheme enables a user to search keywords in multiple data based on some authorization rights granted by the owners of those data segments. MUSE can involve a large number of participants who can have the two following roles:

- Writers (IoT devices, e.g.) who upload data segments to the CSP and grant some parties to search over their data;
- Readers who send their search queries are authorized by writers to do so.

4.2. Existing solutions and their privacy guarantees

MUSE is a recent but active research topic, and some solutions already exist ([16,17], etc.). However it seems very difficult to reconcile security and efficiency in MUSE. Therefore prior to the paper of Popa and Zeldovich [17], solutions on MUSE were giving priority on performance optimization and were assuming that the adversary only had control over the server, implicitly assuming that all users were fully trusted.

Due to the increasing number of readers and writers, considering the presence of some corrupted users colluding with the server in the threat model of MUSE is a necessary step. Unfortunately, the large majority of existing MUSE protocols cannot protect the privacy of data and queries in the face of even a small number of corrupted users. On the other hand, we have discovered a design pattern that is shared among all these MUSE protocols and which is the origin of their weaknesses against user corruption and hence against user-server collusions.

This design pattern in MUSE protocols is studied in [18] and is called the *iterative testing structure*. The paper describes its consequences on the privacy properties of these protocols in the face of collusions. In iterative testing, the server applies a query on an encrypted record by testing encrypted keywords one by one. The server is thus able to see when two queries originating from different readers correspond to the same keyword if these two queries match the same record. The server does not see the plaintext immediately, but rather relations between various queries and records. Then, after the server processed sufficiently many queries, the information from even a single corrupted user is sufficient, due to these relations known to the server, to recover large amounts of plaintext across the whole database, as well as to recover a large number of queries. The amount of information leaked in these MUSE protocols, and the consequences of such leakage on privacy remain very important. Recently, several attacks [19] were proposed against Single-user Searchable Encryption (SSE) systems that combine the information leaked during the execution of the protocol with some extra information the adversary would have obtained through some other means. A first step towards a systematic study of these attacks named leakage abuse attacks is proposed by Cash et al. [19]. The authors suggest to classify SSE using a notion called *leakage profile* that captures the amount of information they leak to the cloud service provider (CSP). They define 4 leakage profiles from L1 to L4 where L1 corresponds to schemes leaking the least information and L4 to the schemes leaking the most.

Additionally, in [19] Cash et al. present new leakage-abuse attacks against some of the defined leakage profiles that outperform the existing ones: They present an attack named count attack which targets L1 profiles and, given the probability of co-occurrence of keywords and the number of indices containing each keyword, is able to recover the content of some queries. They also present a simple attack against the L3 profile where the CSP, knowing the content of some indexes, is able to recover part of the content of other indexes. We present a very simple leakage-abuse attack against MUSE schemes that affects almost all existing solutions [18], for MUSE and have very serious consequences on privacy. This attack is made possible by the iterative testing structure. This structure consists in encrypting each keyword in an index separately, and in testing the presence of a keyword in an encrypted index by applying the trapdoor against each encrypted keyword separately. While very intuitive, this approach lets the CSP see a lot of information, and this leads to the leakage-abuse attack we present. Indeed, not only the iterative testing structure lets the CSP sees which index matches the query (an information that is commonly referred to as the access pattern), but it also lets the CSP sees which encrypted keyword in the index was the one that matched. The CSP is then able to tell that some encrypted keywords from different indexes owned by different writers actually represent the same keyword. This information alone is not sufficient to reveal the content of indexes or queries to the CSP; but as soon as some users collude with the CSP, these known relations between encrypted keywords let the CSP deduce the content of some indexes and queries the colluding users did not have access to. In a system where a large number of queries have been made, the CSP has a lot of information about which encrypted keywords are similar to each other, and we show through statistical simulations that even a small amount of data revealed through collusion may lead to a very large privacy breach.

We then suggest ways to avoid this kind of privacy issues for future MUSE schemes. The first two suggestions we make are of course to add user-CSP collusions in the threat model of MUSE, as well as to find an alternative to the iterative testing structure; but we also extend the classification of Cash et al. [3] in order to capture the newly identified threat and to provide a more systematic way to prevent this kind of issue. We define a new leakage profile named keyword access pattern or **LKWAP**. Let $\{W_i\}$ be the set of uploaded indices and w a given keyword in a query. The **LKWAP** leakage profile can be formalized as follows:

$$\{(i,l) : i \in Auth(r)?W_i[\sigma_i(l)] = w\}$$

Each σ_i is some permutation and "$i \in Auth(r)$" means that reader r is authorized to search the index of writer i. This profile captures the leakage of iterative-testing based schemes in the sense that any MUSE scheme based on iterative testing leaks at least the information described by this profile. While this leakage profile was commonly considered as equivalent to the L1 profile of Cash et al., we show that at least in the multi-user setting it is not the case and the LKWAP profile is in fact closer to a query-revealed version of the L3 profile of Cash et al. We show these claims with the following statistical experiments: we simulate an attack against a system with a L1 profile, and an equivalent attack against a system with a LKWAP profile with the same data, and we measure a loss of privacy that is much greater in the case of the LKWAP profile than in the case of the L1 one. We then perform a similar experiment with LKWAP and L3 profiles and find similar privacy losses. To summarize, almost all existing MUSE schemes provide close to no privacy as soon as even a few users collude with the CSP, which greatly weakens their privacy guarantees. Given the scalability issues of the MUSE scheme, there is a need for new solutions for MUSE that are both secure and scalable. The new guidelines and tools we provide should help avoid similar privacy issues in the design of future MUSE schemes.

4.3. New designs of Multi-User Searchable Encryption (MUSE)

In Uprise-IoT we have designed 2 MUSE protocols each of them having different tradeoffs between privacy (leakage) and efficiency and hence suiting different situations. The main idea that is common to these two solutions is the use of two servers that are assumed not to collude. This idea is illustrated in figure 6. The additional server also called *proxy* will mainly help MUSE protocols protect against user-server collusions. We also introduce the notion of *data preparation* where for each reader being authorized to search a data, a copy of the data is created by the server that will solely be used for queries coming from this reader. This copy is sent to the second server (called the *proxy*) which is in charge of query application.

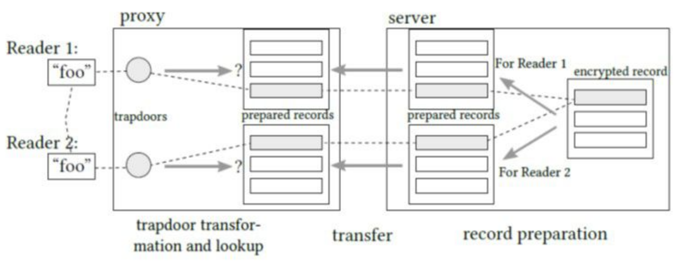


Figure 6. An illustration of our use of two non-colluding servers.
In this figure, two trapdoors for the same keyword are sent by different readers. The proxy sees which prepared keywords these trapdoor match but does not see that these prepared keywords originate from the same encrypted keyword. As to the server, it sees this common origin but does not see the trapdoors and where they match.

The first protocol named DH-AP-MUSE [20], is the most efficient one. It is also the one that leaks the greatest amount of information, although the amount of information it leaks is much lower than the one of any existing. In this solution, a writer encrypts her data with some secret key which she further sends to the proxy. In a symmetric fashion, a reader will use some secret key for encrypting her queries and this secret key is sent to the other server. The server uses the reader keys it receives to create "transformed" copies of the encrypted data (*data preparation*) which the reader sends to the proxy. Symmetrically, the proxy transforms the encrypted queries it receives using the data encryption keys it received. As a result, the proxy obtains transformed queries that it can search for in the prepared data that were sent by the server. This protocol is similar to a Diffie-Hellman Key Exchange protocol [22] and the fact that two queries from different readers will never be applied to the same (prepared) data ensures the protection against user-server collusions.

The second protocol described in [21] leaks much less information than DH-AP-MUSE but can be considered as more costly as well. More precisely, while the users' workload is almost the same as in DH-AP-MUSE, the server workload is more important as it relies on Private Information Retrieval (PIR) protocols. Intuitively, in this solution, prepared data are kept by the server. To search these prepared data, the proxy uses a newly developed privacy-preserving protocol: This protocol assures that the proxy gets no information beyond the number of records that matched the query, while the server gets no information at all. The solution uses two existing primitives, namely Garbled Bloom Filters (GBFs) and Additively-Homomorphic Encryption (AHE) based Oblivious Transfer (OT).

GBFs are a variant of Bloom Filters where the buckets corresponding to an element reveal nothing beyond the presence or absence of the element in the encoded set. OT allows a receiver to retrieve an item in a database hosted by a sender without the sender learning what item was retrieved, and without the receiver learning anything about the rest of the database. Intuitively, the protocol consists in the server encoding the prepared records as Zero GBFs and the proxy looking up these Zero-GBF using the OT protocol. As a result the only information the proxy can learn from a response is whether the response is positive or negative.

4.4. Summary

In this section we have studied the problem of data privacy for IoT environments whereby stakeholders may perform search queries on data produced by various IoT devices. We have shown that the threat model that was used by many early MUSE solutions did not include the users as part of the threat and this was not realistic. We have identified the common design pattern that causes all existing MUSE protocols to be vulnerable to collusions. Consequently a new definition of security model is given. Two solutions were further proposed each offering a different combination of privacy and performance levels. Depending on the actual application and functional requirements one of these two solutions can be used.

5. Data protection of your location: privacy and location tracking

Another very relevant element to be considered is the problem of location tracking. In the past it has been shown that it is very easy to identify users considering their location data, both in case of discrete information, such as cellular logs [23] and location-based social network data [26,28], and continuous information, such as GPS data [29]. Several solutions for dealing with the problem of privacy with respect to location tracking, in particular as far as data obfuscation and coarse-graining are concerned, have been proposed. However, in general, there is always a clear trade-off between the level of obfuscation that is applied to the data and the actual utility that can be extracted from them. In fact, for example, if noise is added to a dataset or the location is coarse-grained [27], it might not be possibile to use it for location prediction or analysis of behavioural patterns of individuals. In other words, there is indeed an inescapable trade-off between the actual utility of a location dataset and the level of privacy for individual users.

One of the most interesting aspects that has attracted the attention of researchers is the problem of interpretability [24]. In particular, in [30] a framework for interpretable privacy-preserving pervasive systems is presented. The key idea of this work is that, by means of instrumentation of the underlying identity inference algorithms, it is possible to devise solutions for suggesting users possible countermeasures for dealing with privacy risks. In particular, the authors discuss the general architecture of the framework and a potential implementation, together with the detailed analysis of a case study.

Finally, given the fact that we are considering Internet of Things systems, which are inherently multi-tiered, a solution to the problem of location privacy might reside in the distribution of computation across different computing devices. More specifically, most of the computation might take place on the devices themselves or on “edge” hosts. By doing so, identifiable personal information might not be sent to back-end servers, but analyzed locally. This solution is not possible in presence of computation that involves different users (e.g., analysis of co-location patterns), but it can be used for several applications, such as location prediction and inference based only on data from the owner of a given mobile or wearable device. But there are several situations in which population data have to be collected in order to use them as input to machine learning algorithms. A promising solution for dealing with these situations is federated learning [25].

6. Conclusion and Future Work

We presented some works produced by the UPRISE-IoT CHIST-ERA project to give awareness and control to users over their privacy. Such activity, which is fundamental with the increasingly raise of IoT, demonstrates that it is possible to have good privacy level while not decreasing the quality of services. Further work is required to test and improve the interface of the PDC proposed in Section 3. User studies must be conducted to determine to which extent the presentation of information can influence DS choices regarding consent. It is also to be noted that the consent management framework is very relevant to the ongoing discussions about the future ePrivacy Regulation [13], the current draft of which, according to the WP29 [14], "gives the impression that organisations may collect information emitted by terminal equipment to track the physical movements of individuals (such as Wi-Fi-tracking or Bluetooth-tracking) without the consent of the individual concerned."

Regarding the processing of the data, in this chapter we mainly focused on the search operations. As for future work, it would be interesting to study more advanced analytics operations such as machine learning techniques and develop privacy preserving variants of these.

Further, this project was seminal to many other activities related to creating awareness around user data, especially with activities for broad public and teenagers. Other ongoing activities include more advanced algorithms and procedure for securing the data, as well as strategies for obfuscating the data to make useless their leakage.

References

[1] A. Westin. Privacy And Freedom. Atheneum, pg. 7, New York, 1967

[2] Wikipedia. Facebook-Cambridge Analytica data scandal. https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal

[3] EUROPEAN COMMISSION - Press release. Antitrust: Commission fines Google €4.34 billion for illegal practices regarding Android mobile devices to strengthen dominance of Google's search engine. http://europa.eu/rapid/press-release_IP-18-4581_en.htm

[4] EUROPEAN COMMISSION Regulation (EU) 2016/679 of the EUROPEAN PARLIAMENT and of the COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Brussels, 2016.

[5] EUROPEAN COMMISSION Directive of the EUROPEAN PARLIAMENT and of the COUNCIL on copyright in the Digital Single Market. COM(2016) 593 final. 2016/0280(COD). Brussels, 2016.

[6] Alan Ferrari and Silvia Giordano. A study on users' privacy perception with smart devices. airXiv preprint, https://arxiv.org/abs/1809.00392 2018.

[7] Luca Luceri, Davide Andreoletti, Massimo Tornatore, Torsten Braun, Silvia Giordano. Awareness and Control of Geo-Location Privacy on Twitter. Submitted to IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, 2019.

[8] Davide Andreoletti, Luca Luceri, Felipe Cardoso, Silvia Giordano. Infringement of Tweets Geo-Location Privacy: an approach based on Graph Convolutional Neural Networks. airXiv preprint, https://arxiv.org/abs/1903.11206 2018.

[9] Castelluccia C., Cunche M., Le Métayer D., Morel V. Enhancing Transparency and Consent in the IoT In: International Workshop on Privacy Engineering (IWPE2018)

[10] Cunche M., Le Métayer D., Morel V. A Generic Information and Consent Framework for the IoT. In: 18^{th} TRUSTCOM19, 2019.

[11] Working Party 29 Guidelines on consent under Regulation 2016/679, 2017

[12] Working Party 29 Opinion 8/2014 on Recent Developments on the Internet of Things, 2014

[13] EUROPEAN COMMISSION Proposal for a Regulation of the European Parliament and of the Council concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Communications) January 2017

[14] Working Party 29 Opinion 01/2017 on the Proposed Regulation for the ePrivacy Regulation (2002/58/EC), December 2017

[15] Bösch, C., Hartel, P.,Jonker, W. and Peter, A. A Survey of Provably Secure Searchable Encryption. ACM Computing Surveys, 2017.

[16] Bao, F., Deng, R. H., Ding, X. and Yang, Y. Private Query on Encrypted Data in Multi-user Settings. Information Security Practice and Experience, 4th International Conf. ISPEC 2008, Sydney, Australia,

[17] Popa, R. A. and Zeldovich, N. Multi-Key Searchable Encryption. IACR Cryptology ePrint Arch., 2013

[18] Van Rompay, C., Molva, R. and Önen, M. A leakage abuse attack against multi-user searchable encryption. PETS 2017, Minneapolis, USA

[19] Cash, D., Grubbs, P., Perry, J. and Ristenpart, T. Leakage-Abuse Attacks Against Searchable Encryption. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015

[20] Van Rompay, C., Molva, R. and Önen, M. Fast Two-Servers Multi-User Searchable Encryption with Strict Access Pattern Leakage. ICICS 2018, Lille, France

[21] Van Rompay, C., Molva, R. and Önen, M. Secure and scalable multi-user searchable encryption. SCC 2018, Songdo, Incheon, Korea.

[22] Diffie W. and Hellman M. New Directions in Cryptography. In: Journal IEEE Transactions on Information Theory. Volume 22, Issue 6, Pages 644-654, Nov. 1976.

[23] De Montjoye, Y.-A., C. A. Hidalgo, M. Verleysen, and V. D. Blondel, Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports 3*, 1376, 2013.

[24] Doshi-Velez, F. and B. Kim, Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017

[25] Konecný, J., H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[26] Rossi, L. and M. Musolesi, It's the Way You Check-in: Identifying Users in Location-based Social Networks. In *Proceedings of COSN'14*, NY, pp. 215–226. ACM 2014.

[27] Shokri, R., G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of CCS'12*, pp. 617627, 2012.

[28] Rossi, L., M. J. Williams, C. Stich, and M. Musolesi, Privacy and the City: User Identification and Location Semantics in Location-Based Social Networks. In *Proceedings of AAAI ICWSM'15*, 2015.

[29] Rossi, L. and M. Musolesi, Spatio-temporal Techniques for User Identification by means of GPS Mobility Data. *EPJ Data Science 4*(11), 2015.

[30] Baron, B. and M. Musolesi, Interpretable machine learning for privacy-preserving pervasive systems. *IEEE Pervasive Computing*. To Appear.