Bringing Privacy Control back to Citizens

DISPEL — A Distributed Privacy Management Platform for the Internet of Things

Christoph Stach
University of Stuttgart, IPVS / AS
Stuttgart, Germany
stachch@ipvs.uni-stuttgart.de

Clémentine Gritti
NTNU
Trondheim, Norway
clementine.gritti@ntnu.no

Bernhard Mitschang
University of Stuttgart, IPVS / AS
Stuttgart, Germany
mitsch@ipvs.uni-stuttgart.de

ABSTRACT

The *Internet of Things* (*IoT*) is becoming increasingly popular. It enables a variety of novel applications. Such applications require a lot of data about their users. To this end, sensors continuously monitor various aspects of daily life. Despite the indisputable benefits of IoT applications, this is a severe privacy threat. Due to the *GDPR* coming into force, there is a need for action on the part of IoT vendors. In this paper, we therefore introduce a *Privacy by Design* approach for IoT applications called *DISPEL*. It provides a configuration method enabling users to specify globally, which application may access what data for which purpose. Privacy protection is then applied at the earliest stage possible, i. e., directly on the IoT devices generating the data. Data transmission is protected against unauthorized access and manipulation. Evaluation results show that DISPEL fulfills the requirements towards an IoT privacy system.

CCS CONCEPTS

• **Security and privacy → Privacy protections**; **Distributed systems security**; *Access control*; *Information flow control.*

KEYWORDS

Privacy, IoT, authorization concept, attribute-based access control

ACM Reference Format:
Christoph Stach, Clémentine Gritti, and Bernhard Mitschang. 2020. Bringing Privacy Control back to Citizens: DISPEL — A Distributed Privacy Management Platform for the Internet of Things. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3341105.3375754

1 INTRODUCTION

The growing popularity and usage of *Internet of Things* (*IoT*) technologies, not only in the business sector but also in the private sector, lead to an ever-increasing number of novel applications that are highly beneficial for users. For instance, consumers directly benefit from the IoT via *Smart Health* (continuous monitoring of health data by patients themselves), *Smart Traffic* (autonomous driving or intelligent traffic guidance systems), or *Smart Home* applications (home automation or surveillance in private homes) [4].

To this end, sensors permanently gather lots of data from a vast variety of domains. As single sensors are cheap, they can be deployed in large quantities to ensure a high degree of data coverage. To cut costs further, IoT devices often lack computational power. Thus, they are not able to process and analyze the large amounts of data. For this purpose, data are sent to a processing back-end, the so-called IoT platform. This platform not only processes the data, but also links data from different sensors and IoT devices and subsequently provides them to any kind of IoT application [17].

Yet, this raises numerous privacy concerns. Data owners—who, in a private setting, are most likely also the data subjects—lose control over their data as soon as they are transmitted to an IoT platform. Since IoT platforms are able to perform in-depth analyses due to their comprehensive database and their virtually limitless computing power, they can derive further knowledge. This knowledge is then possibly available to any third-party IoT application [15].

While IoT platform vendors can be considered as *honest but curious*—i. e., they process data properly and adhere to the terms of usage but are still interested in generating new insights to monetize the data—IoT application developers are unknown to users. As a result, technical protection and control measures are required to ensure privacy when dealing with IoT platforms. For a long time, this was only a wish of the users. Yet, due to the *GDPR* [6], IoT platform vendors are forced to act. The GDPR ensures data subjects transparency concerning handling of private data (Article 5) and requests that s/he is informed of, and agrees to, the processing of any private data (Article 6). In particular *Privacy by Design* (Article 25) necessitates user-friendly tools to oversee and control how IoT platforms process their data and share it with IoT applications [24].

On that account, we introduce a **dis**tributed **p**rivacy manag**e**ment p**l**atform for the IoT called *DISPEL*. DISPEL provides privacy protection at the earliest stage possible, i. e., IoT data sources such as sensors or data stores. Data owners specify globally what data an IoT platform is permitted to share with an IoT application for which purpose. In addition, data accuracy and quality can be adjusted. Since dealing with honest-but-curious IoT platform vendors, data regulation is handled distributed by the data sources and the platform only receives masked data. In addition, DISPEL secures the communication between data sources and the IoT platform to ensure confidentiality and authentication.

To this end, we make the following three contributions: **(1)** We introduce a configuration method enabling users to specify globally which application may access what data for which purpose. **(2)** We introduce privacy techniques tailored to IoT data sources. This gives users fine-grained control over their data. **(3)** We introduce an attribute-based and thus dynamic approach to encrypt and sign any data exchanged between IoT data sources and an IoT platform.

DISPEL is based on the *PMP* [20] and *CHARIOT* [8]. The PMP is a privacy system for smartphones and CHARIOT is a resource-efficient authentication mechanism for the IoT.

The remainder of this paper is as follows: Section 2 discusses state-of-the-art IoT architectures. In Section 3, a literature review reveals requirements towards an IoT privacy mechanism. Related work is discussed in Section 4. Section 5 introduces DISPEL which is subsequently assessed in Section 6. Section 7 concludes this paper.

2 GENERIC IOT ARCHITECTURE

An IoT architecture has to interconnect a large number of different IoT devices and make their data available to applications. To this end, the standard IoT architecture has three layers: In the *Perception Layer*, sensors integrated into physical objects gather data. For instance, a heartbeat sensor installed in a wristband is able to capture performance data of an athlete. These data are made available to all IoT nodes (i. e., IoT devices and applications) via the *Network Layer*.

However, the IoT is highly dynamic, i. e., new devices can be added at any time and existing devices can disappear. P2P connections between all devices are therefore not possible, as otherwise every device would have to know all other currently available devices. Therefore, it is advisable to implement the Network Layer as a central connection and distribution component which also ensures data retention. The network layer therefore has to provide not only sufficient bandwidth but also the necessary storage capacity.

The *Application Layer* hosts all IoT applications. These third-party applications can combine and process any of the data. For instance, a fitness app for runners could consider not only their heart rate but also terrain information about their training route for monitoring their training progress (e. g., via a GPS sensor) [25].

Yet, this basic three-layer architecture is not sufficient, as many IoT devices have only limited connectivity—e. g., many smartbands can only be paired with a single device via Bluetooth while other sensors require a physical connection, such as a built-in GPS sensor. A direct data distribution to all IoT nodes is thus not possible. Therefore, an *Edge Layer* is needed, in which gateways collect all data of connected sensors and forward them to the Network Layer [11].

Figure 1 shows how such a four-layer architecture can be implemented. Some examples of sensors are shown on the left. They are characterized by the fact that they have very limited resources and are therefore hardly able to carry out any data processing. Rather, they are solely responsible for data collection. The sensors are either integrated in more powerful devices such as smartphones, Raspberry Pis, or similar single-board computers or are connected to them via close-range communication. These devices have sufficient capacities to store and preprocess these data. In addition, these devices serve as a gateway to an IoT platform. This platform must be capable of both, processing and analyzing large amounts of data in real time as well as long-term retention. Cloud computing is ideal for this purpose. Then, IoT applications running on these platforms are able to access the data in order to provide various services. They include fitness and health applications or supporting services for Smart Homes—i. e., applications for home automation

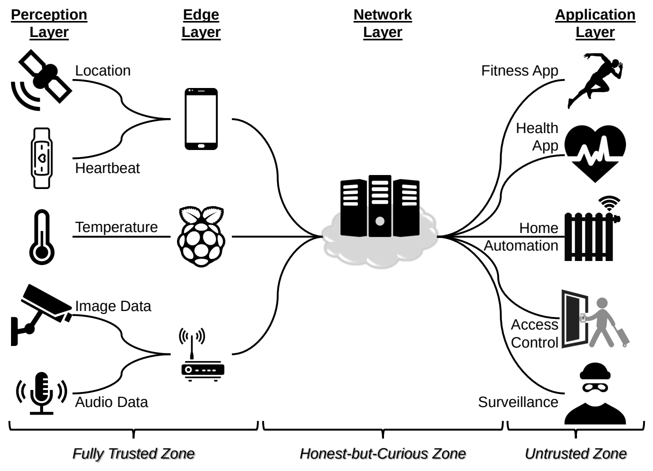


Figure 1: An End-to-End IoT Architecture.

to automatically adjust the room temperature to the user's requirements, to use a smartband to open a door when a user is close to it, or to detect home invaders via anomalies in audio or video data.

From a privacy point of view, this architecture can be divided into three zones: Sensors and gateways belong to the *Fully Trusted Zone* as the user controls these devices.

The Cloud-based IoT platform belongs to the *Honest-but-Curious Zone*. It is regarded as mostly trustworthy as its vendor can be assumed to comply with contracts made with the data owner. Nevertheless, an unrestricted data access is not recommended. The linkage of diverse data and deep data analytics generate knowledge, which can be misused, e. g., for advertising purposes. Also, the connection between gateways and the IoT platform can be compromised. On the one hand, attackers can pretend to be the IoT platform or eavesdrop the connection to gain access to private data. On the other hand, attackers can pretend to be gateways and send fake data to the IoT platform or corrupt data during transmission. Moreover, when data are passed to the IoT platform, the data owner loses physical control over it, which leads to a feeling of insecurity. While enterprise IoT environments can rely on a *private Cloud*—i. e., the enterprise hosts its own Cloud which cannot be accessed by third parties—this is not feasible for private users (e. g., in a Smart Home setting) due to infrastructural and financial reasons.

IoT applications belong to the *Untrusted Zone*. They are provided by third-party vendors about whom no assumptions can be made regarding trustworthiness and reliability. It is therefore crucial to control which application may access what data for which purpose.

Since privacy is an individual perception, the specification of such access rules has to be managed entirely by the data subjects. Furthermore, this specification should be done at a single central point and not individually for each IoT device. This could be achieved by a Privacy by Design approach for the IoT platform. Yet, as this platform is honest but curious, privacy measures should be applied as close to the data owner as possible, i. e., directly at the sensors and gateways. Also, the communication between gateways and the IoT platform has to be secured.

Next, we present requirements towards an IoT privacy approach.

3 REQUIREMENTS SPECIFICATION

A literature review shows that IoT privacy concerns are a prevalent research subject. Porambage et al. [15] study requirements towards a privacy approach for the IoT. They identify five key requirements:

R_1 **Transparency.** It has to be comprehensible to the users for what purpose their data are provided to an IoT application.

R_2 **Temporal and Location Privacy.** In particular, time series data and location data must be protected, as this kind of data is predominant in the IoT domain. Furthermore, very accurate behavior profiles can be derived from them.

R_3 **Query Privacy.** Queries towards data sources from which profiles can be derived have to be preventable.

R_4 **Interoperability.** An IoT privacy system must be able to protect data from any kind of data source adequately.

R_5 **Data Minimization.** Only the necessary amount of data must be sent to the IoT platform, i. e., leave the user's control.

They also address *security* in general as another important requirement. Lin and Bergmann [13] refine this general requirement by breaking it down into the following three sub-categories:

R_6 **Confidentiality.** Only authorized parties (i. e., IoT platforms and applications) must be able to access private data. This applies to both, data sources as well as data transfer.

R_7 **Authentication.** It has to be ensured that data are not manipulated by third parties. This applies to both, data sent to an IoT platform as well as requests sent to IoT devices.

R_8 **Access.** Moreover, it has to be ensured that only authorized IoT devices connect to the IoT platform and that only authorized entities send valid requests to the data sources.

Felt et al. [7] also take users and their needs into account:

R_9 **User-friendly.** As users of IoT applications are often no IT experts, the privacy configuration has to be simple. This also means that no unnecessary setting options are offered.

Lastly, Yang et al. [26] consider technical limitations:

R_{10} **Lightweight Computation.** IoT devices lack of computational power and have to keep battery consumption in mind. This has to be reflected by a privacy system, i. e., only lightweight privacy techniques can be executed on an IoT device.

Next, we discuss whether related work meets these requirements.

4 RELATED WORK

In accordance with Zhou et al. [27], four task domains can be identified in an IoT privacy system. In the following, we discuss selected representatives for these domains.

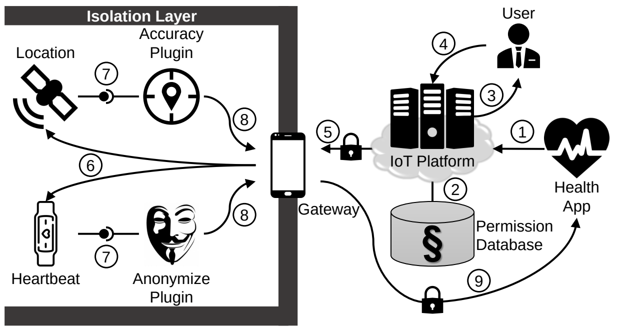
IoT-tailored Privacy Mechanisms. The *PMP* [20] introduces a permission model to specify which application may access what data for which purpose. To achieve this, the PMP provides special privacy components tailored to each data source. This enables users, e. g., to reduce the accuracy of location data. An extended version also supports any kind of Bluetooth device as data source [23]. However, the PMP focuses only on single smartphones. Each device has to be configured individually and interconnections between several devices are not considered. By contrast, Michael et al. [14] introduce a holistic privacy solution for data mining in an IoT environment. They provide several meta-models that describe, e. g., which data are available or who has access to them. A user defines policies describing his or her privacy requirements. The meta-models are then validated against the policies, and only when they are compatible, the respective mining results are released. Yet, this solution only takes effect after all data have been collected and sent to the Network Layer. Moreover, static and well-defined meta-models of the whole IoT environment and all data consumers are required. That is not feasible for arbitrary IoT applications.

Distributed Policy Deployment. *AVARE* [1] enables users to specify their privacy policies at a central hub which then deploys them to any registered IoT device. Similar to the PMP, these policies describe access rights to data sources. However, AVARE considers each device isolated. As a result, AVARE does not take into account that data from several IoT devices can be combined in order to gain more insights. *ACCESSORS* [19] addresses this issue. It can be used to model which information can be derived from which data sources. So, ACCESSORS ensures that an application cannot obtain any private information, even if this information is derived from multiple data sources. However, this requires a central monitoring component that is fully trusted, as it needs access to all data.

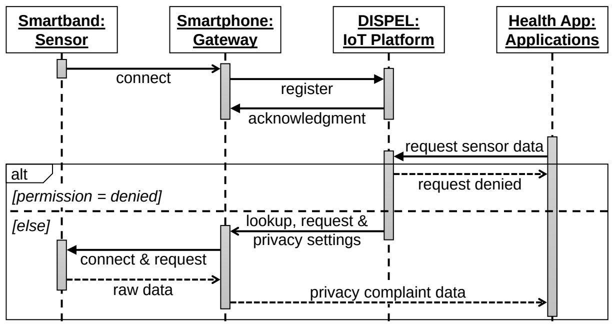
Secure Communication. *NTSA* [16] is a lightweight symmetric encryption algorithm for the IoT. Like all block ciphers, NTSA uses multiple encryption rounds. To provide a higher degree of security, NTSA varies the key in each round. Still, the encryption is fast and can be executed even on the limited resources of an IoT device. Symmetric encryption, however, requires that the master key is exchanged between all participants. In an IoT environment with a large number of nodes, this results in many key exchanges, which poses security risks. Dwivedi et al. [5] introduce an encrypted blockchain approach for data distribution. All data are stored in the blockchain and only authorized applications get access to it. Yet, this approach requires a *private blockchain*, which has to be operated by the user or a fully trusted authority. Besides, the access control is binary: It either allows full access or denies it completely.

Privacy-preserving Authentication. Amin et al. [3] introduce an authentication protocol for distributed data sources. It is based on a central authority that creates a secret key for each user which has to be stored on a smart card. Via this smart card, the user gains access to all data shared with him or her, whereby the data on the smart card do not disclose the user's identity. However, this approach not only generates a high technical and organizational overhead, but also assumes rather static data sources. To reflect the dynamic structure of the IoT, Gritti et al. [8] use an attribute-based authentication method. A device authenticates against an IoT platform via a set of its attributes. Depending on these attributes, the IoT platform can assign different permissions to the device. In this approach, the complex calculation of the proof that the device has the required attributes can be outsourced to a gateway. Furthermore, it is ensured that third parties are not able to spy on the user via these attributes [9]. However, its proving component represents a single point of failure for all assigned devices.

As no approach fully reflects the challenges in an IoT environment, we introduce our IoT privacy approach called DISPEL next.



(a) DISPEL Architecture and Component Interaction.



(b) Data Access in DISPEL Depicted as a UML Sequence Diagram.

Figure 2: Functionality of DISPEL and its Integration into the IoT Architecture.

5 DISPEL

In order to address the shortcomings of current privacy solutions for the IoT, we adopt a distributed privacy approach in DISPEL. Figure 2a shows the architecture and functionality of DISPEL. When an IoT application is started, it fetches all of its data from the IoT Platform on which it is hosted ①. The IoT platform checks whether the application has the required permissions, i. e., whether there is a corresponding privacy rule in the DISPEL permission database ②. Unlike state-of-the-art approaches, in which usually *access control lists* (*ACLs*)—i. e., a list of permissions for each application—or *role-based access control* (*RBAC*)—i. e., permissions are assigned to roles, e. g., health application—are used for this purpose, we apply *attribute-based access control* (*ABAC*), which is significantly more dynamic. In ABAC, permissions are assigned to certain attributes of an application. As a result, e. g., the context in which the application is executed can also be taken into account when assigning permissions. This is particularly well suited for the IoT [2].

As DISPEL adopts a *Privacy by Default* approach, the permission database is organized as a *white list*, i. e., an application initially has no permissions. DISPEL informs the user if an application needs additional permissions in order to process a certain task ③. The user decides whether this task is relevant for him or her, i. e., the user is able to specify for which functionality s/he is willing to share what data ④. A respective privacy rule is then added to the database, which is effective until it is revoked by the user ②. A privacy rule not only specifies whether data access is permitted but may also entail constraints. A constraint could be, e. g., that an application has access to mock data only, that certain values are not shared with an application, or that the accuracy of the data is reduced by inserting noise before sharing them with an application.

If an application is authorized to access the data, the request and the privacy constraints are forwarded to the respective gateway that can provide the requested data ⑤. The gateway does a lookup whether the required data source is currently available and whether the constraints can be applied to it ⑥. For instance, it is not possible to reduce the accuracy of arbitrary plain text as random noise would impair data quality too severely. Therefore, in DISPEL each data source provides privacy plugins for all supported constraint types. These plugins are able to retrieve the required data from the source and apply the corresponding privacy techniques to them ⑦.

The gateway selects the appropriate plugin and binds it to the data source ⑧. Via the plugin, the gateway then accesses the requested data. Subsequently, the revised data—in terms of sensitive information—are forwarded to the requesting application ⑨. The necessary communication—starting from the request of an application and ending with the return of results—is detailed in Figure 2b.

For the implementation of this concept, we partially rely on existing approaches, namely the PMP [20] and CHARIOT [8].

Next, we describe how the global configuration of privacy rules is realized (Section 5.1)[1], which privacy techniques we introduce specifically for the IoT (Section 5.2), and how we secure the communication between the IoT devices and the IoT platform (Section 5.3).

5.1 Privacy Settings

To define the privacy rules in DISPEL, we use a simplified version of ACCESSORS [19] which we have adapted for a distributed deployment. The data model is shown in Figure 3 in ER notation.

An IoT application specifies different access purposes—each of them representing a certain functionality of this application which requires some kind of information about the user. An example of this is a heart monitor provided by a health application that requires access to heartbeat data. Our data model reflects from which data sources these data can originate. In the heartbeat data example, they could be derived from a smartband or a chest strap. In the model, it is also possible to describe that information is generated by joining several data sources. Thus, a privacy rule consists of two key components: a certain information and an access purpose.

Furthermore, each privacy rule can be enriched by an additional constraint. A constraint could be that noise has to be added to the heartbeat data or that its sample rate, i. e., the quantity of shared data sets, is reduced. If no constraint is specified, the application gets unrestricted access to the information. However, if no privacy rule is specified for an access purpose / information pair, the application does not get any access at all due to our Privacy by Default approach.

Another option for limiting data access is the activation context of a rule. It can be described either by the user's context—e. g., as a spatio-temporal description to express that access is only permitted at a certain location or at a certain time—or by the attributes of an application—e. g., access is only permitted when the application

[1]Nota bene, although these rules are stored centrally, they can be configured anywhere.

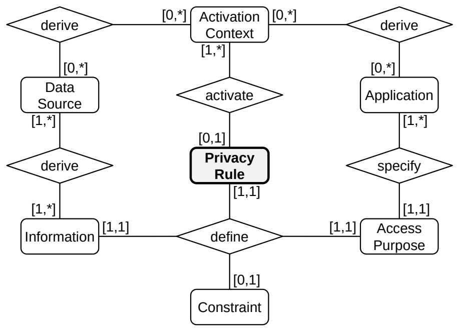


Figure 3: Data Model of a DISPEL Privacy Rule (cf. [19]).

runs in a specific execution environment. If no context is specified, the corresponding privacy rule is valid unconditionally.

Due to this data model it is possible to define privacy rules in two different ways: A user can either select functionalities s/he wants to use and DISPEL shows the required permissions or s/he selects the data sources s/he is willing to share with an application (including constraints) and DISPEL shows which functionalities are available with these permissions. Functionalities that do not get sufficient data (i. e., which were not granted the required permissions) are not provided by the application and are skipped during execution.

For this purpose, applications must have a modular structure, i. e., they must be divided into functional units. The developer specifies for these units which permissions in terms of data accesses are required during execution. This specification is used by DISPEL to prompt the user whether s/he grants the permission in order to receive the associated functionality. If a required permission is not stated, the corresponding data access is not possible. It is therefore in the developer's interest to fully specify the relationships between functionality and data access. This also provides a transparent purpose description for every data access. Developers are obligated by the GDPR to provide this kind of information anyway when dealing with personal data (Article 12). Thus, DISPEL does not cause any additional overhead from the developer's point of view.

Yet, for a user our Privacy by Default approach causes an overhead. S/he has to define for every application what information s/he is willing to reveal for which purpose. However, DISPEL can analyze his or her privacy rules, i. e., his or her user behavior, and derive privacy setting recommendations for new applications from them. DISPEL also has access to the privacy rules of other users, which can also be taken into account for the recommendations as well. Stach and Steimle [22] introduce a recommendation system for privacy settings based on *collaborative filtering*, which can be applied to DISPEL in order to reduce the effort for the user.

5.2 Privacy Techniques

To enforce the privacy rules described above, DISPEL applies a technique similar to the one used in the PMP. Data are only accessible via a special driver, the so-called privacy plugin. Each data source can offer several plugins to support various privacy techniques, which are used to mask its data. The appropriate plugin—i. e., the fitting privacy technique—is selected by the gateway according to the constraints of a data request. This plugin is then bound to the data source (similar to the binding of Bluetooth devices to the PMP [23]). Thereby, data protection at source level is enabled.

By default, a data source offers at least two plugins: One plugin provides access to the authentic data from the source. This is used if access was granted without constraints. The other plugin provides only mock data—i. e., random values. However, it is ensured that the generated data are *realistic* so that the requesting application is able to process them. For instance, the value range of the mock data has to correspond to one of the authentic data and if value distributions are known for a certain data source, the plugin developer should consider this when generating the mock data as well. Therefore, there is no generic mock plugin that can be used for all sources, as plugins need to be tailored to the respective data sources.

The plugin developer defines which further constraints are supported by the data source. For string-based data types, a certain prefix can be filtered out, e. g., a phone book resource should not display numbers with a certain area code. A threshold level can be specified for enumerable data types, e. g., a continuous blood glucose meter should only transmit values with priority orange or higher. For numerical data types, data accuracy can be adjusted, e. g., noise is added to the data of an IoT-enabled personal scale.

In addition, further constraints can be defined depending on the respective data source. In the following, we introduce three of DISPEL's privacy plugins which are relevant in an IoT context.

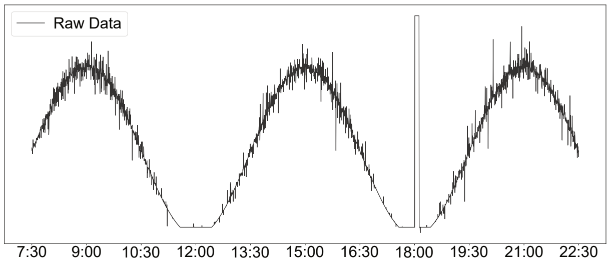
Location Data. Geolocations provided by GPS sensors are very dangerous from a privacy point of view, as they enable to capture motion profiles. By combining these profiles with maps, a lot of insights about a user's activities and preferences can be obtained.

DISPEL therefore uses a privacy plugin for devices with GPS sensors, which returns a location in the user's surrounding instead of his or her actual location. The user defines via a constraint how far away from his or her current location this mock location can be (*max*). Depending on the application, very coarse location data are often sufficient. For instance, to set the correct time zone, it is sufficient to recognize in which country the user is located.

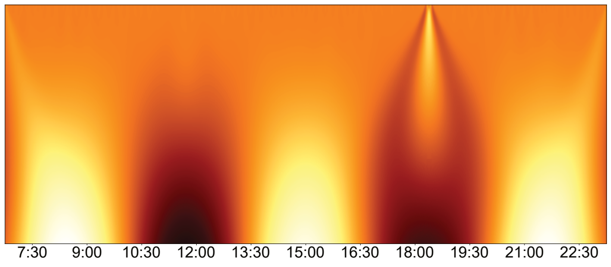
As in Alpers et al. [1], we initially choose a random angle α ($0° \leq \alpha < 360°$). Then, the user's location is shifted in this direction by a random distance d ($0 \leq d \leq max$). For each subsequent location request, another random angle α' is chosen. Yet, the shifting distance d is no longer chosen randomly, but it is set to the distance d' that the user has actually moved since the last location request. Then, the location is shifted by distance d' in direction α'. Thereby, an application is still able to determine the speed of travel and the traveled distance without having access to any real data.

It is also possible to filter the values by time, i. e., the location is only updated every x minutes. As a result, a user can no longer be tracked exactly as an application cannot record what s/he does between two updates. This method requires less resources than the aforementioned one and can therefore be executed on any IoT device. Yet, the speed of travel and the traveled distance are not determinable that way. To this end, temporal filtering can also be combined with the more advanced technique described above.

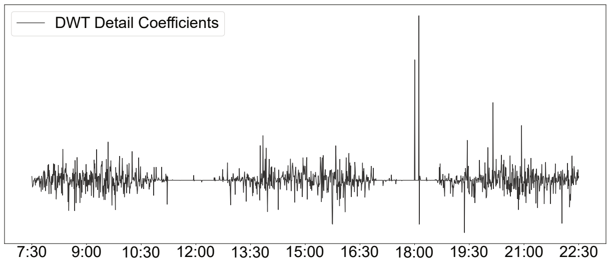
Secure Data Stores. Yet, a privacy plugin can provide more functionality than just data access and masking. One issue for IoT applications is that IoT devices, especially at the Perception Layer, are not permanently available and therefore can temporarily not



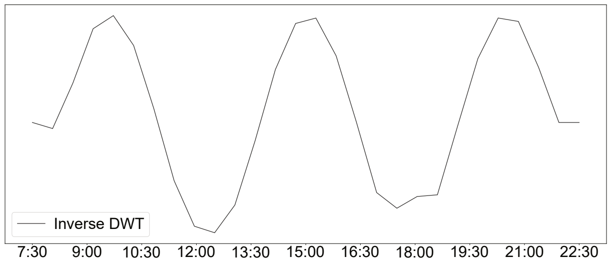
(a) Continuous Glucose Monitoring Data.



(b) Continuous Wavelet Transform.



(c) Discrete Wavelet Transform.



(d) Inverse Discrete Wavelet Transform.

Figure 4: Exemplary Application of a Wavelet Transform (the x-axis shows the time and the y-axis the blood glucose level).

provide data. An external data store in which different sensors can deposit their data solves this issue, since it enables applications to receive data from the data store when an IoT device is not available.

In DISPEL, such a data store can be realized as a privacy plugin. Each gateway operates its own plugin which can be fed with data by any connected data source due to its customizable data model. If an application requests data that is available in the store, the gateway binds the store plugin if the actual data source is not available. The application cannot tell whether it receives live or cached data.

For privacy protection, various filters can be defined as constraints. For instance, a horizontal filter corresponds to a *selection* (i. e., entire tuples are concealed) while a vertical filter corresponds to a *projection* (i. e., attributes are concealed). These relational algebra operators are automatically inserted into the incoming query via *query rewriting*. In addition to these basic filters, there are also approaches to ensure, e. g., *differential privacy* using query rewriting [10]. Such approaches can also be integrated into our plugin.

Additionally, the data store has to be secured against unauthorized data access. For this, the concepts of the *SDC* [21], namely full database encryption and secure deletion, can be applied. These security concepts can be applied to both, SQL databases as well as NoSQL databases in order to meet both kinds of demands [18].

Time Series Data. Time series data are important in an IoT context. Sensors continuously gather data and add a time stamp to it. This enables time series analyses, e. g., to give insights into the progression or to predict future trends. Such analyses have to face two problems: On the one hand, the amount of data is growing rapidly as sensors continuously generate new data. Therefore, data reduction techniques are needed. On the other hand, data are often noisy. This requires denoise techniques that do not tamper with the characteristics in the data. Wavelet transforms solve these issues. We, however, use wavelet transforms for privacy reasons.

Figure 4a shows the glucose progression of a diabetes patient. There is a reading every 15 seconds, i. e., the depicted time window consists of 3600 data points. A normal progression can be seen in which the blood glucose rises after a meal (e. g., at 12 pm) until insulin is released (e. g., at 3 pm). In addition, minor spikes are visible due to the high measuring coverage. At 6 pm, there is a sudden peak, which indicates the consumption of sweets. Figure 4b shows the result of a *continuous wavelet transform* (*CWT*) using a *Gaussian derivative mother wavelet.* On the y axis the frequencies are plotted. Thereby, besides the extreme values (light and dark areas) also the sudden rise at 6 pm can be identified clearly, whereas the minor spikes disappear. On the contrary, a *discrete wavelet transform* (*DWT*) can emphasize them (see Figure 4c, using a *Daubechies 4-tap mother wavelet*). The first level of the DWT operates as a high-pass filter. Thereby, not only the periodic changes are concealed, but the data set is additionally reduced to 1801 detail coefficients. To conceal the sudden rise, the high frequency detail levels of the DTW can be suppressed for an *inverse discrete wavelet transform* (see Figure 4d). This graph consists of less than 1% of the original sampling points, Yet, the glucose progression can still be diagnosed.

A privacy plugin for time series data sources enables users to use constraints in order to provide applications with abnormalities in the data progression via CWT or to apply a high-pass or low-pass filter to the data, both via DWT. In the process, the number of sampling points is reduced, which further increases privacy.

The computational overhead depends on the applied privacy technique and the amount of data. To increase the performance, masked data which are frequently requested and have a low velocity—i. e., data that have long-term validity—can also be stored in a DISPEL secure data storage plugin. Subsequent, requests can be answered partly by this storage. Only data that were newly added to the data source have to be preprocessed by its privacy plugin.

5.3 Signing and Encryption

The DISPEL privacy rules, which are specified on the IoT platform and then deployed to the IoT devices, and the DISPEL privacy plugins, which enforce these rules by masking private data, ensure that applications do not get access to private information. However, this is utterly ineffective if it is not guaranteed that an attacker cannot intercept (e. g., to access data from a data source without having the necessary permissions) or corrupt (e. g., to strip the constraints from a data request) the communication between the IoT platform and the IoT devices. An asymmetric cryptosystem is able to accomplish this via a private key for signing and decrypting messages and a public key for verifying and encrypting messages.

However, a single key pair for each application is not adequate for the IoT. An application might have different permissions depending on its context. For instance, if an application is running in an untrusted execution environment, it should not be able to access the same messages as if it is running in a secure execution environment. Therefore, in DISPEL we focus on attribute-based encryption in order to reflect such aspects. That is, our private keys correspond to the attributes of an application. As a result, decryption and verification are only possible if the key's attributes are compatible with the current attributes of the application. Such an approach is particularly suitable for Cloud applications [12].

Attribute-based encryption algorithms, however, require a lot of computational effort. Yet, IoT devices have substantially less resources than Clouds. This is why CHARIOT [8] introduces an attribute-based signature technique that is tailored to IoT environments. In CHARIOT, the computation is partly outsourced to a Cloud component. However, by doing so, this component gains insight into the attributes used for signing. These attributes can reveal private information about a user, e. g., the location where s/he uses the application. Gritti et al. [9] prove that this outsourcing can be done in a privacy-aware way by prefiltering the attributes.

Figure 5 shows how the concepts of CHARIOT are applied in DISPEL. The authentication process consists of three stages:

Key Generation and Deployment. Initially, a trusted authority (e. g., a federal data protection authority) creates a public / private key pair for each sensor and application (the latter is not shown in Figure 5 for simplicity's sake). The authority then distributes the generated keys to the components of DISPEL ❶. Let τ be a set of a sensor's identifying attributes, then all attributes in τ are reflected by the *full keys* (depicted in black). These keys are assigned to the sensor in question. The trusted authority also generates *delegated keys* for the gateways and the IoT platform (depicted in white). They are generated using only a subset of the sensor attributes $\tau' \subsetneq \tau$.

Full Authentication. To authenticate that a message originates from a sensor and has not been tampered with, the sensor signs it with its full key ❷. The gateway verifies the signature against an authentication policy ρ ❸. ρ describes the attributes which are mandatory for the sensor. If the attributes in τ satisfy ρ, then the signature is valid, and the integrity of the message is ensured.

Delegated Authentication. The gateway then modifies the signature using its delegated key, i. e., it filters out all attributes $\tau \setminus \tau'$ ❹. This simplified signature satisfies a reduced authentication policy ρ'. An application can then use ρ' to verify the authenticity of a message without gaining access to the all sensor attributes τ. This is handled by the IoT platform which hosts the application ❺.

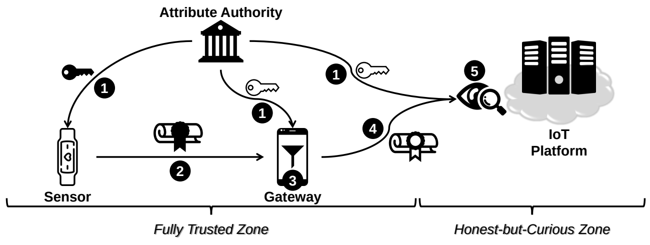


Figure 5: Privacy-aware Attribute-based Signature (cf. [9]).

For instance, a smartband signs its data with its device number and an id of its current user (i. e., the data subject). The user's smartphone verifies whether the smartband may send data about that user to an IoT platform. If that is the case, the smartphone signs the data with all attributes except the user id. This reduced signature is sufficient for the IoT platform to verify the smartband.

CHARIOT supports only signing messages. Yet, the processing steps for encrypting messages are virtually identical—in these cases, public keys are used instead of private keys [12]. Thus, CHARIOT can be extended to support lightweight encryption as well.

6 ASSESSMENT

In conclusion, we evaluate whether DISPEL fulfills the requirements towards a privacy system for the IoT as specified in Section 3.

The data model used in DISPEL describes which private information can be derived from which data sources. In addition, we apply a Privacy by Default approach, i. e., the user has to authorize each access to a data source at least once. As a result, s/he is always informed which applications access what data for which purpose and in particular which information can be derived on this basis. That is, *transparency* is achieved in DISPEL ($\mathbf{R_1}$).

In DISPEL, we provide privacy plugins for different types of data and use cases. In particular, we provide a plugin for location data that conceals the user's current location, but still preserves other characteristics in the data if required—e. g., the traveled distance or the direction of movement—and a plugin for time series data that conceals certain temporal aspects in the data. For instance, it is possible to share only extreme values, trends, or anomalies, while an application does not get any other insights from the data. Thus, DISPEL ensures *temporal and location privacy* ($\mathbf{R_2}$).

The third plugin discussed in this paper enables *query privacy*. In the basic plugin, only certain attributes or tuples can be filtered out. However, there are approaches to ensure differential privacy in database queries, which can be integrated into our plugin ($\mathbf{R_3}$).

The privacy plugins can be tailored to every IoT device and to the characteristics of their data, e. g., to preserve certain information intentionally, while concealing others. This ensures *interoperability* between DISPEL and different IoT devices ($\mathbf{R_4}$).

DISPEL is executed distributed, i. e., data are masked directly on the IoT devices. Therefore, the user never loses control over his or her private data. Furthermore, s/he can use the constraints to specify on a fine granular level which data are shared with an application. That way, *data minimization* is ensured ($\mathbf{R_5}$).

Due to the asymmetric cryptography in DISPEL, it is possible to sign and encrypt all messages. This ensures both *confidentiality* and *authentication*, as only authorized applications have access to private data, which cannot be tampered with by third parties (**R$_6$** & **R$_7$**). Moreover, this also ensures that no unauthorized entities have *access* to the IoT platform and send requests or data to it (**R$_8$**).

There are two ways to specify privacy rules: A user describes either which features of an application s/he wants to use, or which information must not be disclosed. This enables users to describe their privacy requirements in a straightforward manner. In addition, there are approaches to recommend appropriate privacy settings tailored to the respective user. This makes DISPEL *user-friendly* (**R$_9$**).

DISPEL causes a performance overhead for the configuration and deployment of its privacy rules. However, these tasks are handled by the Cloud-based IoT platform, i. e., additional resources can be allocated if required. The privacy plugins also cause an overhead on the IoT devices. These plugins are, however, tailored to the respective IoT devices and their limitations. The attribute-based encryption causes an overhead as well. However, as shown for signing, most of these steps can be outsourced to the IoT platform and thus to the Cloud, i. e., this overhead is also negligible. Thus, DISPEL is suitable for the IoT as it supports *lightweight computation* (**R$_{10}$**).

Thus, DISPEL fulfills all requirements towards a privacy system for the IoT. So, an IoT Platform complies with the Privacy by Design principle enshrined in the GDPR (Article 25), by integrating DISPEL. That is, DISPEL ensures that private data are processed in compliance with the GDPR (Article 5) and that data access is only permitted if the user has given his or her consent (Article 6).

7 CONCLUSION

The IoT has become an integral part in our daily lives. More and more everyday objects are equipped with sensors to monitor their surroundings and have connectivity options. An IoT platform provides third-party applications with the thereby gathered data. This enables a variety of innovative application scenarios, e. g., Smart Health applications. Yet, such applications also represent a significant privacy risk, as IoT devices reveal insights into our private lives. Reports about data misuse are increasingly worrying users.

On that account, we introduce a **dis**tributed **p**rivacy manag**e**ment p**l**atform for the IoT to *DISPEL* their concerns. DISPEL's decentralized approach ensures that the privacy techniques are applied close to the data sources. Unlike centralized approaches, in which data are initially sent to the IoT platform, which is then solely responsible for privacy protection, our approach retains sensitive data within the reach of a data owner. S/he controls, what data are shared with which application for which purpose. To achieve this, we make three contributions in this paper: **(1)** We introduce a method to globally specify privacy rules that are then deployed on IoT devices. **(2)** We introduce various privacy techniques tailored to specific data sources. In particular, we address location data sources, a secure data store, and time series data sources. **(3)** We introduce a method to secure communication in IoT environments by encrypting and signing all messages. Evaluation results attest, that DISPEL not only fulfills the technical and functional requirements towards a privacy system for the IoT, but that it also complies with the legal demand by the GDPR for Privacy by Design approaches.

REFERENCES

[1] Sascha Alpers et al. 2018. Citizen Empowerment by a Technical Approach for Privacy Enforcement. In *Proceedings of CLOSER '18*, 589–595.

[2] Asma Alshehri and Ravi Sandhu. 2017. Access Control Models for Virtual Object Communication in Cloud-Enabled IoT. In *Proceedings of IRI '17*, 16–25.

[3] Ruhul Amin et al. 2018. A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment. *Future Generation Computer Systems*, 78, 1005–1019.

[4] Shanzhi Chen et al. 2014. A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Internet of Things Journal*, 1, 4, 349–359.

[5] Ashutosh Dhar Dwivedi et al. 2019. A Decentralized Privacy-Preserving Healthcare Blockchain for IoT. *Sensors*, 19, 2, 326:1–326:17.

[6] European Parliament and Council of the European Union. 2016. Regulation 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 (Data Protection Directive). Legislative acts L119. Official Journal of the European Union, (Apr. 27, 2016).

[7] Adrienne Porter Felt, Serge Egelman, and David Wagner. 2012. I've Got 99 Problems, but Vibration Ain't One: A Survey of Smartphone Users' Concerns. In *Proceedings of SPSM '12*, 33–44.

[8] Clémentine Gritti, Melek Önen, and Refik Molva. 2018. CHARIOT: Cloud-Assisted Access Control for the Internet of Things. In *Proceedings of PST '18*, 1–6.

[9] Clémentine Gritti, Melek Önen, and Refik Molva. 2019. Privacy-preserving Delegable Authentication in the Internet of Things. In *Proceedings of SAC '19*, 861–869.

[10] Hannes Grunert and Andreas Heuer. 2017. Rewriting Complex Queries from Cloud to Fog under Capability Constraints to Protect the Users' Privacy. *Open Journal of Internet Of Things*, 3, 1, 31–45.

[11] Jayavardhana Gubbi et al. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29, 7, 1645–1660.

[12] N. Saravana Kumar, G. V. Rajya Lakshmi, and B. Balamurugan. 2015. Enhanced Attribute Based Encryption for Cloud Computing. *Procedia Computer Science*, 46, 689–696.

[13] Huichen Lin and Neil W. Bergmann. 2016. IoT Privacy and Security Challenges for Smart Home Environments. *Information*, 7, 3, 44:1–44:15.

[14] Judith Michael et al. 2019. User-Centered and Privacy-Driven Process Mining System Design for IoT. In *Proceedings of CAiSE '19*, 194–206.

[15] Pawani Porambage et al. 2016. The Quest for Privacy in the Internet of Things. *IEEE Cloud Computing*, 3, 2, 36–45.

[16] Sreeja Rajesh et al. 2019. A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices. *Symmetry*, 11, 2, 293:1–293:21.

[17] Ted Saarikko, Ulrika H. Westergren, and Tomas Blomquist. 2017. The Internet of Things: Are you ready for what's coming? *Business Horizons*, 60, 5, 667–676.

[18] Christoph Stach and Bernhard Mitschang. 2018. Curator — A Secure Shared Object Store: Design, Implementation, and Evaluation of a Manageable, Secure, and Performant Data Exchange Mechanism for Smart Devices. In *Proceedings of SAC '18*, 533–540.

[19] Christoph Stach and Bernhard Mitschang. 2019. Elicitation of Privacy Requirements for the Internet of Things Using ACCESSORS. In *Information Systems Security and Privacy*. Paolo Mori, Steven Furnell, and Olivier Camp, (Eds.) Springer, 40–65.

[20] Christoph Stach and Bernhard Mitschang. 2013. Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In *Proceedings of MDM '13*, 305–313.

[21] Christoph Stach and Bernhard Mitschang. 2016. The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security. In *Proceedings of MDM '16*, 292–297.

[22] Christoph Stach and Frank Steimle. 2019. Recommender-based Privacy Requirements Elicitation – EPICUREAN: An Approach to Simplify Privacy Settings in IoT Applications with Respect to the GDPR. In *Proceedings of SAC '19*, 1500–1507.

[23] Christoph Stach, Frank Steimle, and Ana Cristina Franco da Silva. 2017. TIROL: The Extensible Interconnectivity Layer for mHealth Applications. In *Proceedings of ICIST '17*, 190–202.

[24] Sandra Wachter. 2018. Normative challenges of identification in the Internet of Things: Privacy, profiling, discrimination, and the GDPR. *Computer Law & Security Review*, 34, 3, 436–449.

[25] Miao Wu et al. 2010. Research on the architecture of Internet of Things. In *Proceedings of ICACTE '10*, 484–487.

[26] Yuchen Yang et al. 2017. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4, 5, 1250–1258.

[27] Jun Zhou et al. 2017. Security and Privacy for Cloud-Based IoT: Challenges. *IEEE Communications Magazine*, 55, 1, 26–33.