R. Rios, J. A. Onieva, R. Roman, and J. Lopez, "Personal IoT Privacy Control at the Edge", *IEEE Security & Privacy* , vol. 20, pp. 23 - 32, 2022.
http://doi.org/10.1109/MSEC.2021.3101865
NICS Lab. Publications: https://www.nics.uma.es/publications

Special Issue: Security and Privacy Issues of Home Globalization
Editors: L. Caviglione, S. Wendzel, S. Vrhovec, A. Mileva, sp1-22@computer.org

Personal IoT Privacy Control at the Edge

Ruben Rios
NICS Lab & ITIS Software, University of Malaga

Jose A. Onieva
NICS Lab, University of Malaga

Rodrigo Roman
NICS Lab & ITIS Software, University of Malaga

Javier Lopez
NICS Lab & ITIS Software, University of Malaga

***Abstract*—This article introduces a privacy manager for IoT data based on Edge Computing. This poses the advantage that privacy is enforced before data leaves the control of the user, who is provided with a tool to express data sharing preferences based on a novel context-aware privacy language.**

WITH THE ADVANCEMENT OF THE IOT, in just a few years we have populated our homes, offices and even our bodies with smart devices (smartbands, voice-controlled home assistants, etc.) that aim to bring us comfort and make our lives easier. Unsurprisingly, much of the data collected by these devices is personal data and, what is worse, in many cases belong to the intimacy of our homes. This raises serious privacy concerns as these personally sensitive data typically end up in remote cloud servers far beyond the users' control. In contrast, the GDPR regulation in the EU and the recommendations of the Federal Trade Commission in the US put the individual in control of the personal data generated by IoT devices.

One concept that has been developed to help people retain their own data is the notion of privacy managers [1]. Privacy managers are, in essence, entities that help users to express their privacy preferences and to control access to their personal information through privacy policies. As a result, not only access to personal data will be limited to authorized external entities, but also such data will be filtered and/or anonymized. Such privacy managers are usually designed to run in powerful devices, and only manage data generated by a single device.

In this paper, we take advantage of the recent development of Edge Computing to create a privacy manager that manages personal IoT devices in extended home ecosystems – where multi-

ple IoT devices are distributed across various locations. The proposed solution, called Privacy Manager based on Edge Computing (PMEC), is designed as a set of interconnected virtualized instances, or privacy manager instances (PMIs), that run on Edge Computing devices. PMIs are geographically distributed in order to store the data generated by the IoT devices that the user has at different locations, say, at home, at work, and so on. In addition, we devise a novel context-aware policy language to help data owners express their privacy preferences. These preferences define when, how and to what extent data is shared with third parties (data access privacy policies), and which edge devices can handle which data types (data management privacy policies). Clearly, this poses an advantage from a privacy point of view, since the data owners are always in control of the data collected by their devices.

PRIVACY MANAGERS

The main goal of a privacy manager is to control access to data and, if necessary, alter them in accordance with a set of policies before they are shared with third parties. A common approach to implement this is to make use of obfuscation services working as a middleware. Such services are deployed between the users' devices and external applications – either on the IoT devices themselves or as services in the Cloud [2, 3]. They receive all raw data coming from the devices, store them internally, and then perform privacy policy enforcement as well as data aggregation and obfuscation. As a result, from the point of view of external applications, privacy managers become the interface to users' IoT devices.

For the implementation of such privacy managers, there are several challenges that have to be considered. One of the main challenges is related to their availability: if the privacy manager is not available, then external applications will not be able to access the information from the IoT devices – even if these IoT devices are working correctly. Another challenge is related to the storage of information: although privacy managers should store all raw data coming from the devices in order to implement the privacy methods, due to storage limitations historic data should be delegated to external repositories. Finally, privacy managers should take into account the mobility of IoT devices, plus they should be flexible enough to include additional aggregation, filtering and anonymization features in a modular way.

There are several variations of the basic middleware approach whose goal is to tackle some of these challenges, although they have their own advantages and disadvantages. One such approach uses active data bundles [4], where an execution environment that contains a copy of all the data and the policies is sent to servers closer to the data requester. Although this strategy has some advantages, there are also some downsides – especially in terms of efficiency (e.g., when the remote request needs one-time partial data). Other approaches make use of a data pre-processing approach [5], where queries are not executed on original data but on purged data. This strategy saves space, as only purged data needs to be stored within the privacy manager. Still, it imposes stringent requirements on the privacy methods that are available to the user. In order to solve these challenges, the solution presented in this article will make use of the benefits of Edge Computing and distributed systems.

EDGE COMPUTING

Edge Computing brings computing and storage devices from the network core to the network edge. This not only enables the creation of full-fledged IoT scenarios that are close to the users, but also facilitate the deployment of privacy managers [6]. In essence, edge infrastructures are geographically distributed, tiered systems organized on a continuum from the edge of the network to the cloud. This brings about several evident benefits [7], such as:

- Reduced network latency and response times,
- Minimizes data transfers to and from the network core,
- Improved scalability and reliability,
- Increased context-awareness.

These benefits themselves are key improvements to existing cloud-based solutions in terms of privacy (see for instance [8]). As pre-processing and storage of private data gets closer to the data source and can be under the user's control, no granular choice over data capture

before transmitting to the cloud is needed, and furthermore, in some use cases data might never be transferred to cloud services; and definitively, real-time data would never consume core network resources being transferred to the cloud.

Additionally, audit logs can include context info when data access policies deny third party access to data. This cannot be accomplished when data and access control is made at the cloud level. Although numerous solutions exist in the cloud to avoid data leaks, when these type of attacks occur [9], raw data from thousands of users is disclosed as a consequence of a less distributed architecture.

This distribution is achieved thanks to a set up of tiers with a number of edge nodes capable of running virtual machines or containers, thus enabling the seamless deployment and execution of different types of services. The tier closer to the network edge is in charge of collecting and processing data from IoT devices. As we move up in the infrastructure, the edge nodes are more powerful and engage in more complex computations, typically using aggregated or historical data from lower levels. At the top level is the Cloud, but the Edge should also be able to work without a stable Internet connection and access to it.

Besides storing and processing IoT data, edge nodes can provide IoT services to external entities based on these data. For example, smartwatch data can be used by fitness applications, insurance companies or physicians. For this purpose, it is necessary to make use of discovery services that allow entities to find the interface that controls access to the data of interest to them. There are already various discovery mechanisms geared to IoT networks that can be used in this context. For example, the Domain Name Service – Service Discovery (DNS-SD) allows the lookup of a given service via DNS [10]. Through DNS-SD, users can retrieve what services are available within a certain PMEC, and then obtain the IP address and port of a particular PMI instance. An alternative method is the CoRE Resource Directory [11], which is an IETF draft that proposes the use of a Resource Directory (RD) in the context of a REST-based Web model.

Edge nodes can also be user-owned, like an edge-ready home router, or controlled by cloud providers or network operators, for example, deployed in a 5G base station. Therefore, choosing to deploy services or upload data to a third-party edge node may not only have monetary costs but also privacy implications if such nodes are not adequately protected.

AN EDGE-POWERED IOT SCENARIO

Due to its features and benefits, Edge Computing can facilitate the deployment of interconnected extended homes, where several devices belonging to a single entity (e.g., a person or even a business) may be geographically distributed across different locations (e.g., the home, the office) collecting data related to the entity. Yet, this scenario also imposes additional challenges for the deployment of privacy managers as data of different nature – including highly sensitive information such as the precise location of individuals – are collected from both static and mobile devices. It is then necessary to define this edge-powered IoT scenario in detail to fully understand such challenges.

Consider, for example, a user wearing body sensors such as a smart watch for monitoring his heart rate and tracking his location during training. These data should be uploaded to a personal edge device where the privacy manager is running for controlling access to these personally sensitive data. This, in itself, is an important data privacy flow change compared to pure Cloud-based IoT scenarios: current smart watch phone apps upload all available raw data to enterprise cloud servers lacking almost any privacy feature and exposing precise location coordinates, which is prone to attacks [9].

However, the user can also own other smart devices at different locations, which also collect user-related data. For example, at home, the user has some internet-enabled appliances such as a smart TV or dishwasher. The data they collect may be subject to privacy violations: the times of day at which these devices are used reveal user preferences and habits. Similarly, the user may be in possession of some other personal devices at the office, which might leak information about the user's work shifts. Moreover, the user may carry devices from one location to another. The data being collected by all these IoT devices must be properly protected.

To this end, the privacy manager must be ge-

ographically distributed across different locations next to where the devices are collecting data. In many cases, the devices will remain static or attached to a particular location and thus having a privacy manager controlling access to their data is not convoluted. For example, the privacy manager can run in a dedicated in-house server or an edge-ready router. Furthermore, this distribution naturally reduces the risk surface from a privacy point of view, because data storage will occur closer to where the data is produced, therefore controlling privacy leaks in one domain.

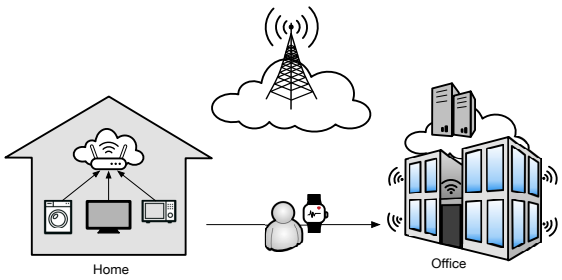


Figure 1: User moves with some device(s) to another location.

Still, there will be personal IoT devices that change from one location to another. This situation is depicted in Figure 1, where there are several static devices at home, and the data generated by them are being collected by a privacy manager instance installed at the edge-ready home router. The user's smart watch may also send data to the same privacy manager while at home; but when the user leaves, the smart watch should be able to continue collecting data and making it available to third parties according to the preferences of the user. Consequently, a privacy manager must be accessible near the smart watch, deployed on a 5G antenna. In addition, all queries addressed to this device will be handled directly by the privacy manager, which may only be in possession of the most recent data generated by the device. Eventually, the user arrives at the office and he may want another privacy manager running on an on-premise edge platform to take over in order to avoid overspending due to the use of third-party edge platforms. At the end of the day, the privacy manager at home will take control back again.

Privacy Manager Requirements

Here we summarize the requirements that a privacy manager must meet in order to provide an adequate service in edge-powered IoT scenarios. These requirements take into consideration not only the general challenges of privacy managers but also the specific challenges of this scenario. Note that we do not highlight key requirements that are ensured by the edge architecture itself like reduced latency, increased computational power and real-time provision of services. They are as follows:

- IoT data collection: the privacy manager must offer southbound interfaces for allowing the interaction with different types of devices using typical IoT communication protocols.
- Ease of deployment: the privacy manager must be based on virtualization technologies in order to facilitate its deployment in diverse scenarios regardless of the edge nodes' hardware features.
- Mobility support: the IoT devices must be able to move from one location to another without affecting data availability. This process should be triggered automatically but also provide control to the user.
- Offline operation: the privacy manager should always be available to protect the data collected by IoT devices even if there is no Internet connection available. This implies that the privacy manager does not completely depend on the Cloud.
- User control: the privacy manager must enable data owners to define their own privacy preferences for controlling access to data. Privacy preferences allow the users to configure when, how and to what extent data is shared with others.
- Data filtering: the privacy manager must be able to incorporate any data filtering mechanism to fulfil the requirement of sharing data with different levels of granularity and to translate quantitative data into qualitative data.
- Scalable storage: The data needs to be stored and managed by the privacy manager according to data freshness or any other condition established in the scenario. Following Edge Computing principles, real-time and fresher data should be located close to the sources

for fast computation and access with minimum latency.

PRIVACY MANAGER ARCHITECTURE

In order to fulfill the requirements of this edge-powered IoT scenario, we need a distributed architecture that facilitates the management of all IoT devices, regardless of their location. Such architecture is our Privacy Manager based on Edge Computing (PMEC), shown in Figure 2.

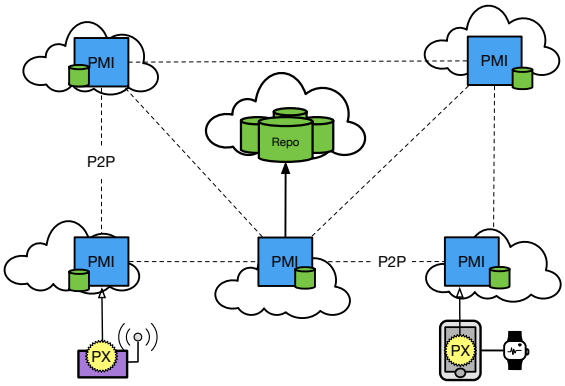


Figure 2: Elements of the PMEC system

The PMEC system is comprised of several *privacy manager instances* (PMI) connected to each other through a *P2P network*, a common *edge repository* (Repo), and auxiliary elements, known as *privacy proxies* (PX) that facilitate the interconnection between IoT devices and the PMIs.

PMI

The privacy manager instances are deployed on edge nodes at different locations, typically in the vicinity of the IoT devices belonging to a user or business, in order to bring computation, storage and services close to the data source. Each PMI takes care of the data collected by one or more IoT devices, enforcing the privacy policies defined by their users. When a PMI is in charge of protecting the data from a particular device, we say it is the *primary* for this device. For example, a PMI may be primary of a smart watch the user wears, and at the same time store the data collected by a smart meter installed at the user's home. If the primary PMI is close to IoT devices, privacy can be enforced as close as possible to where data is produced.

P2P network

Due to the distributed nature of the PMEC architecture, all PMI instances have the possibility to communicate and cooperate with each other through an underlying peer-to-peer network that takes the proximity of the nodes into consideration, such as the overlay networks in [12]. This paradigm fits this architecture, as it provides a fault-tolerant and scalable means for interconnecting all entities. In fact, there are already various libraries such as libp2p (https://libp2p.io/) that provide the high-level P2P services required by our architecture, including i) peer discovery (for connecting all PMI instances), ii) direct and broadcast secure messaging (for negotiation and maintenance – heartbeat – purposes), and iii) file sharing (for storing and caching configuration and policies). Also, for this architecture, we assume that PMI instances are deployed in trusted edge nodes that either belong to the user's trusted domain or are managed by trusted telecommunication companies (e.g., 5G-powered edge nodes). Therefore, there is no need to make use of complex byzantine consensus algorithms for decision making.

Data repositories

Having several PMI instances can result in data from an IoT device being scattered across different locations. Thus, the primary PMI will store the real-time data generated by its designated IoT devices, and will periodically (or driven by events) offload data to the common edge repository, which is expected to have sufficient capacity for storing these data. Moreover, a bigger and highly-available (possibly aggregated) cloud repository can be used to store both historical and backup data. This creates a 3-tier architecture for storing data, where real-time and fresh data will always be closely located to the IoT device. In addition, all data that leaves the personal domain will always be stored in an encrypted manner.

IoT proxy

From an implementation point of view, one theoretical downside of the PMEC architecture is that it would require IoT devices to become aware of the PMEC network, as they must a) negotiate various parameters with candidate PMIs during the primary election process, and b) change their configuration automatically to connect to the ad-

dress of its new primary. This is not feasible in most cases, as many IoT devices are black boxes whose functionality cannot be changed. Moreover, in case of a primary PMI change within a cellular network, an orphan IoT device cannot be contacted directly by the elected primary PMI due to current limitations of such networks – including security concerns and lack of available IP addresses.

Nevertheless, these issues are easily solved by deploying an element known as *IoT proxy* (PX). IoT proxies are simple software components that can be deployed within the IoT devices themselves – in case they allow the installation of external components – or in closely located external devices (e.g., smartphones). IoT devices are then configured to connect to these proxies as if they were their external data servers. Such proxies will simply tunnel any commands from the IoT device to the primary PMI, and vice versa. In addition, the PX monitors whether its primary is offline, and if so, accesses a previously configured location to request information from the PMEC network. Moreover, all PX will incorporate additional functionality required by the PMEC network.

PMI ARCHITECTURE

The architecture of the Privacy Manager Instance, as well as its core components and their interactions, are depicted in Figure 3.

The PMI acts as a broker for all entities querying, pulling, pushing or updating data from or to the infrastructure, enforcing privacy policies before the data is released or stored. The implementation of the PMI is modular – based on lightweight containers – so that the architecture integrates different components that are replaceable (e.g., IoT interfaces, data filtering mechanisms). This facilitates the provisioning of partial updates of the system as well as new anonymization primitives. The main components of the PMI architecture are as follows:

- **IoT Interface**: This corresponds to the service protocol component that allows the different IoT devices to push data into the architecture. Here, we follow the idea of sensor drivers (analogous to printer drivers), as different type of sensors use proprietary protocols and formats that need to be translated to a common format used in the Privacy Manager.
- **Data Filtering Module**: This module integrates the operations the manager needs to perform in order to preserve data privacy. Due to the modular nature of this architecture, various privacy-enhancing technologies can be integrated, such as anonymization/de-anonymization, data obfuscation, de-identification, selective deletion, summarization or inference, among others. In addition, the PMI can make use of additional Edge resources to implement the most resource-consuming filtering mechanisms.
- **AuthN Module**: This component is in charge of implementing authentication and authorization decisions matching the credentials provided by data consumers/producers and the policies stored in the PMI. This component can also query existing edge services from the edge node where it is deployed in order to gather additional decision information (e.g., context-inferred information).
- **Cloud Interface**: This interface allows for (edge or remote) cloud service data operations access like large backups or complex big data analytics.
- **PMEC Interface**: This interface connects the PMI instance with others and allows to run functions implemented over the P2P layer.
- **Data**: It is a repository for storing real-time and recent IoT data. If the information requested by external entities is not available here, it will be retrieved from external repositories. This is the first level of the 3-tier storage architecture considered by PMEC.
- **Policies**: It is a repository for policy files. When a PMI gets a policy update, a network P2P file sync flow allows almost instant policy updates on all candidate PMIs nearby. This architecture allows for contextual privacy policies, i.e., all IoT devices controlled by the same edge node may share privacy rules.
- **Query and Store**: These are background processes in charge of handling (i.e., queuing, processing, etc.) data requests and storage operations once the end-points have been authenticated and authorized.

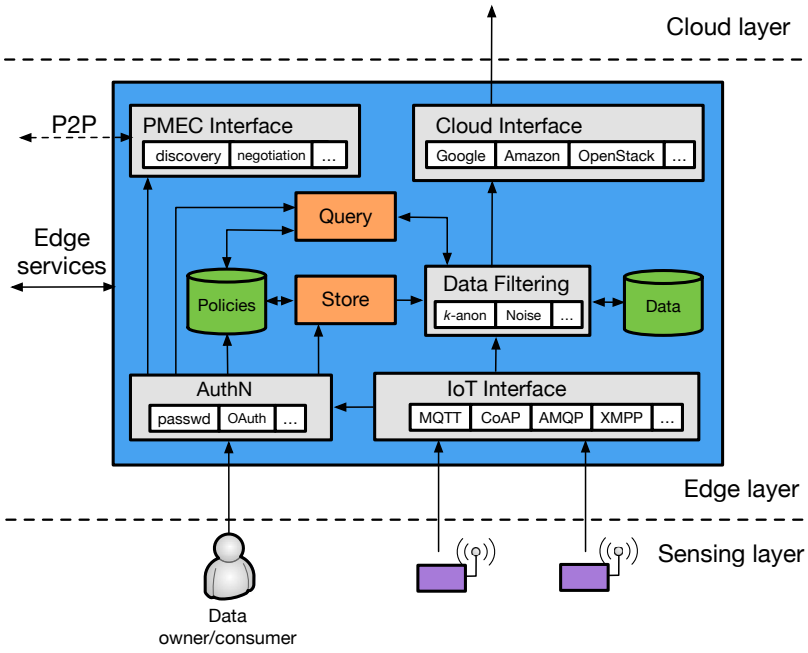


Figure 3: Privacy Manager Architecture

PMI Lifecycle

Although the PMI's architecture is identical for all PMIs, a PMI goes through different states in which it will perform different operations. This is referred to as the PMI life cycle.

During the **Initialization state**, prior to deployment, the PMI is pre-loaded with proper administrator credentials and default policies, which allows the data owner to configure the system once deployed. After its deployment, the owner generates new credentials for all the IoT devices, plus data access privacy policies for each type of data. In addition, the owner will provide the location of the common edge repository – which will be initialized if no repository exists. This information will be used to connect this instance with the rest of the network using the P2P PMEC interface. Finally, the PMI will be chosen as primary for all previously configured IoT devices that are marked as such, which in turn will update the naming system for IoT devices.

Once the PMI is initialized and configured, it is now on its **Active state**, and can provide its services to external users. One of such services is the provisioning of information: when a data requester wants to retrieve data from a particular IoT device, it first needs to query the naming system (e.g., DNS-SD), which will then provide the address of the primary PMI associated with this device. In case the data being requested is recent, it can be served directly. If not, the primary PMI will access the edge repository and retrieve the data that the requester is entitled to access. These data will be temporarily cached in the primary PMI in order to speed up future queries.

Another external service that is available at this stage is the PMI configuration update. Using this service, the PMEC owner can change the configuration of a particular PMI – including its privacy policies and IoT credentials. Once the update is finished, the new configuration will be pushed to all other PMI instances and saved in the common edge repository. This way, the latest configuration will be available even in case of a hard shutdown of the PMI.

If a PMI can no longer behave as the primary PMI of a certain IoT device, the **Primary Change state** will be activated. This state – which runs in parallel to the active state – will be triggered whenever a) the PMI detects that one of its IoT devices is moving far away from it due to factors such as increasing latency, b) a certain primary PMI is deemed unreachable by a heartbeat mechanism integrated in all nodes of the PMEC network, or c) the user explicitly asks the PMI

to "refresh" the primary PMI of a particular IoT device, e.g., by sending an input through the IoT proxy interface installed on his smartphone. The goal of this state is to execute the primary election process – where a new primary PMI that complies with the user-defined data management policies is selected from all existing PMI instances.

At the beginning of the primary election process, if the original primary PMI is not available, all members of the PMEC network will choose a temporary leader through a P2P leader election algorithm. This leader retrieves the data management policy of the old primary PMI from its cache or from the common edge repository, and creates an initial set of potential PMI candidates that complies with such policy (e.g., edge nodes located within close proximity to the IoT device). Once this initial set has been chosen, the election leader contacts the members of this set using the P2P interface, and request from them additional information that involves the IoT device (e.g., latency of the communications between the device and the PMI). This information is then used to select the most optimal primary PMI, which will in turn update the PMI interface address of the previously mentioned discovery mechanisms – so that external users can always obtain the most current PMI interface of a particular IoT device. Due to this election process, there is no need for PMIs to migrate from one edge node to another.

Finally, when a PMI gracefully shutdowns (either by a request from the user or by a local decision, e.g., low battery), it executes various processes before entering its **Shutdown state**. First, it will store a backup of all relevant data within the common edge repository. This backup will not include configuration information, as it was already shared and backed up in previous phases. After this, it will inform about its intention to shutdown to all other PMIs in the PMEC network. Moreover, in case it is a primary PMI, it will run the primary election process. Finally, once the new primary PMI is elected, it will shutdown.

A PMI can also reach the shutdown state from any other state after a hard shutdown. In this case, the PMI is unable to perform the actions it would have taken if the shutdown had been controlled. Nonetheless, this is not a big issue, as PMIs backup their configuration occasionally or after a user configuration change. In addition, other PMIs will detect its absence and initiate the primary election process if necessary. Moreover, a PMI that is powered on after a shutdown can retrieve the configuration either from its own local storage or from other PMI instances via the P2P network. In such a case, the PMI will directly move to the active state.

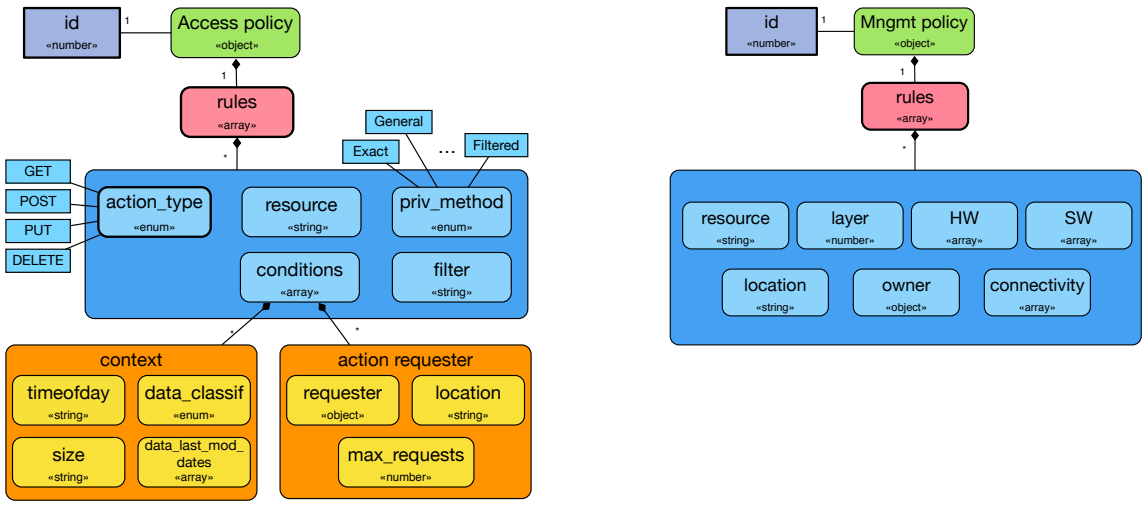
PRIVACY POLICIES

Privacy policies are at the core of any privacy manager, as they describe how the system manages data. Although numerous privacy policy languages exist [13], these are not specifically tailored for Edge Computing and IoT scenarios, and there is no clear consensus on the most suitable language for representing the information collected by IoT devices.

Based on previous experiences, we have defined our own schema for representing privacy policies. This schema is available on https://github.com/nicslabdev/privacy-manager, and it is expressed in JSON using a particular syntax, which for readability reasons, is depicted in Figure 4. PMEC uses two types of privacy policies to define i) when, how and to what extent information can be accessed by others (i.e., *data access privacy policies*), and ii) when, how and to what extent another PMI can take over as primary to protect data (i.e., *data management privacy policies*). This is achieved by means of a set of rules that impose restrictions, such as which data operations are granted to data requesters, or under what conditions a new PMI becomes primary of a certain data type.

A **data access privacy policy** (see Figure 4a) consists of a number of rules. Each rule controls a given *action type* performed by an actor over a *resource*. We represent a resource with an SQL-like query to be applied over the database. The actions considered here are `GET`, `POST`, `PUT` and `DELETE`, which represent a data access request, data uploading, data update and data deletion, respectively. These actions can then be invoked through a RESTful API.

For some actions, it is also possible to apply a particular *privacy method* before the data is released or stored locally. Examples of privacy methods are the following: (i) `Noise`, which introduces three levels of noise (min, med or max)



(a) Data access policy syntax

(b) Data management policy syntax

Figure 4: Graphical representation of the policy syntax

to the data; the amount of noise for each level depends on the data type to which it is applied, (ii) `Generalization`, which takes entries from the database with similar attributes and creates k-anonymous groups, (iii) `Encrypted`, which encrypts the data output, and (iv) `Filtered`, which applies a *filter* to the query received by the privacy manager (similar to those already existing in NoSQL databases).

Precisely, there are various libraries that are used to provide data filtering functionality. The k-AnonML library [14] provides various k-anonymization algorithms, such as Optimal Lattice Anonymization (OLA), Mondrian, Top-Down Greedy Anonymisation (TDG) and k-NN Clustering-Based (CB) Anonymization. There are also some libraries, like ARX (https://arx.deidentifier.org/), that provide support for different privacy models beyond k-anonymity. ARX supports various privacy models (e.g., k-anonymity, l-diversity, t-closeness, differential privacy), methods for transforming data (e.g., generalization, suppression, aggregation) and methods for analyzing the usefulness of output data based on quantification of the information loss (e.g., attribute coverage, mutual entropy, ambiguity).

All actions associated with policy rules will be applied under certain *conditions*. Examples of such conditions are the following:

- On the *action requester*, to control requester available features like:
  - A *requester* authentication token for granting the operation.
  - Requester *location* from which the operation is initiated.
  - Maximum number of operations (*max_request*) to attend per time-window. This helps with DoS (Denial of Service) attacks and/or performance tuning.
- On the *context*, to control data context features like:
  - *time of day* to prevent data requests at particular hours or time frames.
  - *data classification* level over which the operation requested is to be performed. Various privacy levels can be defined in order to generalize different data privacy categories (personal identifiable information, sensitive, confidential, internal-only, etc.) that may exist in different business areas.
  - *data modification dates*, in order to control stored and served data freshness.
  - *size* in order to limit the amount of data shared or uploaded.

A **data management privacy policy** is used to help with PMI lifecycle state transitions. As shown in Figure 4b, this type of policy has only one type of rule, which is used to specify the

features an edge device should have to become the primary PMI for a given *resource* or type of data. These features include available hardware, software, location, node's owner, connectivity or layer. The layer field allows to control where a new primary PMI can be instantiated and under which conditions. For instance, a layer value of zero disables a PMI takeover, a value of one enables it for the same edge layer as the original PMI and so on in the edge to cloud continuum.

With these privacy policies we can model complex data sharing practices in edge-powered IoT scenarios, giving users more power over the management of their data. For example, using the policy syntax described in Figure 4a, we can describe a scenario where a user is willing to share his heart rate reserve data or HRR (difference between the fastest and slowest heart rate), only in 1KB chunks, with entities proving to be an authorized medical doctor but only when outside home. In addition, we can define for this scenario a data management policy that follows the syntax described in Figure 4b, which establishes that if a new PMI takes over, the node in which it runs must have a TPM (Trusted Platform Module), be in close proximity and have 5G connectivity.

As for the definition of the policies, we assume a privacy by default approach in which IoT objects provide users with a privacy policy template. Users can then modify these policies according to their privacy needs. When the policy changes, the system can make use of already stored data within the PMIs in order to provide the user with an overview of the information that will be shared with others. This will allow the user to refine these policies. Note that the creation of user-friendly interfaces for the definition of privacy policies is a broad area of research, as it is highly dependent on the type of data (e.g., location information [15]). The results provided by such research could be integrated into our platform due to its modular nature.

CONCLUSIONS

We have devised a distributed privacy manager architecture, called PMEC, that exploits the inherent features of Edge Computing. This architecture gives solution to several challenges of privacy managers comprised of multiple IoT devices, such as the absence of a stable Internet connection, the need for a scalable data storage with real-time data access, and the coordination between the PMIs conforming the PMEC architecture. As a matter of fact, the proposed architecture helps users to retain control of their personal data in edge-powered IoT scenarios and beyond.

Note that there are several ways in which this architecture can be improved. For example, although we assume that 5G nodes are properly secured by telecommunication companies, it is necessary to take into account that certain edge nodes might be "honest, but curious". In addition, the architecture could also integrate user-friendly mechanisms to allow users to faithfully represent their privacy preferences with policies.

Finally, a proof of concept implementation of a standalone PMI without the P2P protocols for negotiation and synchronization is available at https://github.com/nicslabdev/privacy-manager.

ACKNOWLEDGMENT

This work has been partially supported by the EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu) and by the Spanish Ministry of Science and Innovation through the SecureEDGE project (PID2019-110565RB-I00).

REFERENCES

1. Bjorn Schwarzbach, Bogdan Franczyk, Lucas Petrich, Arkadius Schier, and Michael Ten Hompel. Cloud Based Privacy Preserving Collaborative Business Process Management. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, dec 2016.
2. Anupam Das, Martin Degeling, Daniel Smullen, and Norman Sadeh. Personalized Privacy Assistants for the Internet of Things: Providing Users with Notice and Choice. *IEEE Pervasive Computing*, 17(3):35–46, jul 2018.
3. Patrícia R. Sousa, Rolando Martins, and Luís Antunes. Empowering Users Through a Privacy Middleware Watchdog. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12395 LNCS, pages 156–170. Springer

Science and Business Media Deutschland GmbH, 2020.

4. Abduljaleel Al-Hasnawi and Leszek Lilien. Pushing Data Privacy Control to the Edge in IoT using Policy Enforcement Fog Module. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing - UCC '17 Companion*. ACM Press, 2017.
5. Christoph Stach, Rebecca Eichler, Corinna Giebler, Julia Bräcker, and Clémentine Gritti. How to Provide High-Utility Time Series Data in a Privacy-Aware Manner: A VAULT to Manage Time Series Data. *International Journal on Advances in Security Volume 13, Number 3 & 4, 2020*, 2020.
6. Nigel Davies, Nina Taft, Mahadev Satyanarayanan, Sarah Clinch, and Brandon Amos. Privacy Mediators: Helping IoT Cross the Chasm. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications - HotMobile '16*. ACM Press, 2016.
7. Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, aug 2019.
8. Maribel Fernandez, Jenjira Jaimunk, and Bhavani Thuraisingham. Privacy-Preserving Architecture for Cloud-IoT Platforms. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 11–19, 2019.
9. Garmin press release. Garmin issues statement on recent outage. https://newsroom.garmin.com/newsroom/press-release-details/2020/Garmin-issues-statement-on-recent-outage/default.aspx.
10. S. Cheshire and M. Krochmal. DNS-Based Service Discovery. RFC 6763, RFC Editor, February 2013. http://www.rfc-editor.org/rfc/rfc6763.txt.
11. Christian Amsuess, Zach Shelby, Michael Koster, Carsten Bormann, and Peter van der Stok. CoRE Resource Directory. Internet-Draft draft-ietf-core-resource-directory-28, IETF Secretariat, March 2021. http://www.ietf.org/internet-drafts/draft-ietf-core-resource-directory-28.txt.
12. Yingwu Zhu and Yiming Hu. Efficient, Proximity-aware Load Balancing for DHT-based P2P Systems. *IEEE Transactions on parallel and distributed systems*, 16(4):349–361, 2005.
13. Saffija Kasem-Madani and Michael Meier. Security and Privacy Policy Languages: A Survey, Categorization and Gap Identification. *CoRR*, abs/1512.00201, 2015.
14. Djordje Slijepčević, Maximilian Henzl, Lukas Daniel Klausner, Tobias Dam, Peter Kieseberg, and Matthias Zeppelzauer. k-anonymity in Practice: How Generalisation and Suppression Affect Machine Learning Classifiers, 2021.
15. Mehrnaz Ataei, Auriol Degbelo, and Christian Kray. Privacy theory in practice: designing a user interface for managing location privacy on mobile devices. *Journal of Location Based Services*, 12(3-4):141–178, 2018.

Ruben Rios is a postdoctoral researcher in NICS Lab at the University of Malaga, Malaga, 29071, Spain. His research interests are mainly focused on the development of privacy solutions for emerging technologies. He received research fellowships from the Spanish Ministry of Education and later from the University of Malaga. Contact him at ruben@lcc.uma.es

Jose A. Onieva is an associate professor at the University of Malaga, Malaga, 29071, Spain. His research interests include security services for Edge Computing, Malware Analysis and Intrusion Detection and Response. He has been involved in European and National funded research projects. He has authored several international journal and conferences in the field of information security. Contact him at onieva@lcc.uma.es.

Rodrigo Roman is an assistant professor at the University of Malaga, Malaga, 29071, Spain. His research interests include the security of paradigms such as the Internet of Things and Edge Computing. Since 2005, he has been involved in several research projects financed by both the Spanish Government and the European Community. He is a member of IEEE. Contact him at roman@lcc.uma.es.

Javier Lopez is a full professor and head of the Network, Information and Computer Security (NICS) Lab at the University of Malaga, Malaga, 29071, Spain. His research activities are mainly focused on network security, security protocols, and critical information

infrastructures, leading a number of national and international research projects in those areas. He is Senior Member of IEEE. Contact him at jlm@lcc.uma.es.