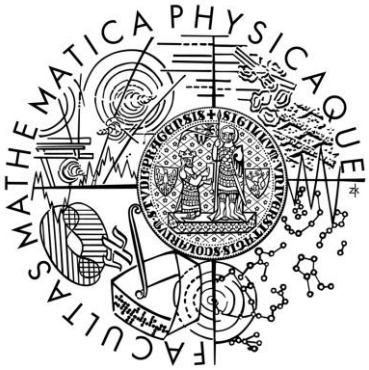
Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS



Martin Mašek

Privacy issues of the Internet of Things

Department of Distributed and Dependable Systems

Supervisor of the master thesis: doc. RNDr. Tomáš Bureš, Ph.D.

Study programme: Informatics

Specialization: Software systems

Prague 201

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In…..... date..... signature

I would like to thank to my supervisor, doc. RNDr. Tomáš Bureš, PhD., for his patience, advice and feedback. Also, I would like to thank to my girlfriend, Magdaléna Fejfarová, whose support aided me during the work on the thesis.

Title: Privacy issues of the Internet of Things

Author: Martin Mašek

Department / Institute: Department of Distributed and Dependable Systems

Supervisor of the master thesis: doc. RNDr. Tomáš Bureš, Ph.D.

Abstract: More and more devices and sensors are around us in today's world. There is an increasing tendency to connect devices and sensors to the Internet. We call such network the Internet of Things. As the cost of computing power continues to decrease, data collection and analysis becomes cheaper. Thus, we are able to get much better insight into different domains. This could be a problem in commercial sector, where we do not want to compromise proprietary know how or data. Especially big concerns are around personal data. We can now analyse behavior of people or get access to personal information such as health or spending patterns. This thesis addresses these privacy issues.

Keywords: Internet of things, privacy, privacy patterns, Azure

Contents

1 Introduction

The Internet changed the way how the world is connected. Started as a network of few connected computers it evolved to the today's form where it links billions of devices. It is a building block for the World Wide Web – the main reason we use the Internet. Humans are the main users of the Internet. Whether we send emails, access information or use web application, we are the initiators of the traffic on the Internet. However, a new group of users is emerging – physical object embedded with electronics. These physical objects are able to monitor the context they are in (such as monitoring wind turbine speed) and some are able to interact or change their state (for example smart thermostat controlling the temperature in the room). We call such network the Internet of Things.

As the cost of computing power continues to decrease, collection and analysis of IoT[1] data becomes cheaper. This brings a new set of privacy and security issues. For example we have computational power to monitor movement of phones in real time. This can be used to track the user's movement. The purpose of the thesis is to introduce these problems in more detail, find a solution and create a real implementation. At first, we will discuss how the thesis is organized. That is the goal of the following subchapter.

1.1. Thesis organization

Initial chapters are about the analysis of the problem. The first chapter introduces the problem and sets the goals of the thesis. Before diving into privacy we need to understand IoT and that is the purpose of chapter 2. Chapter 3 focuses on high level examples of IoT and privacy violations. The purpose is to give a background for the thesis. Chapter 4 talks about methods of how we can protect privacy and chapter 5 uses these methods and defines best practices and guidelines for privacy.

Subsequent chapters are about demonstration of privacy methods from previous chapters in practice. Chapter 6 introduces an idea for a framework used to prototype IoT networks. Following chapter 7 implements this framework and provides

[1] the Internet of Things

documentation. Chapter 8 then shows concrete examples where the framework is used to prototype use cases. The last chapters are dedicated to summarizing the thesis, discussing related work (chapter 10) and proposing future work (chapter 9).

1.2. Motivation

Fear from a Big Brother made of little things monitoring everything [1]. That is an opinion the public might have about the Internet of Things. IoT has been discussed in news for few years and since then we always hear privacy issues [2] and concerns next to big visions of IoT changing the world. The more data we collect, the better insight we get. The better insight we get, the better services for people. But what data are really too personal to collect? It seems that there is a tradeoff between quality of service and loss of privacy. The fact is, technology is going forward and the chances of IoT not coming to our lives are low. For that reason, we believe we need ways how to protect privacy. Assuming IoT comes and we are surrounded by devices monitoring movement, things we buy, health condition, temperature, door to our homes and much more, what are the ways how we can retain our privacy? Is it even possible? The goal of the thesis is to give methods to protect privacy and try to satisfy IoT demands in a maximum possible way so a thorough analysis of data is still possible, yet sensitive information are not revealed. We believe that governments will impose some regulations on IoT providers to protect privacy of citizens. For that reason, it make sense to look into IoT privacy issues and come up with solutions.

1.3. Problem description

This work focuses on privacy and not security. It is vital to understand the difference. Let us give an example. Suppose we want to collect an age and gender of each user. These are examples of questions we could ask when we think about privacy:

- Is it necessary to collect a precise age?
- Could we work at our application with age ranges (like 10-15, 15-20 etc.) instead of precise ages?
- If we work with age ranges, where should be the logic which transforms the precise age to the age range for the further computation?
- Is there a real need of knowing gender of the user? What happens when we find out that only 10% of our users are female? In that case it might

be possible to reveal the identity of a female user even though we use age ranges.

In contrast, these are the examples of questions which concern security:

- How to send user data to the server? Should it be encrypted?
- How do we recognize that the entity we are sending our data to is trusted?
- Is the device which monitors our data safe? Maybe there is a backdoor and data are also sent somewhere else.

However, there is a blurred line where security ends and privacy starts. An example is access to data. Suppose we send the data to a data warehouse. Access policy to the data could be classified as both security and privacy issue. To give a more clear vision, we focus on following aspects of privacy:

- What types of data are exchanged and what the data flow looks like.
- Data consumer should get only data it needs.
- Techniques how we can transform data so it is not possible to identify the producer of the data but the consumer still gets useful information.
- Techniques how we can identify privacy issues in a data flow or a system as a whole.

We work on the assumptions that:

- Connected devices are "honest", that is they do not have a backdoor. E.g. we buy an expensive mining machine equipped with sensors. The manufacturer of the machine provides a service to which we can send data from sensors to predict future failures. If we decide to create a filter and send only certain data, the manufacturer is not able to access data directly from the sensors. Instead, it is able to access only filtered data.
- Data exchange between mechanisms adding privacy is safe. When two components communicate, it is through a secure channel.

These assumptions are not far from the real world because usually we would use secure channels for communication and we would encrypt our data. Furthermore, we buy devices from trusted manufacturers.

As you can see, we are interested in privacy on a higher level (focus on data semantics). An example of solution focused on a lower level is a paper [3] where authors solve packet-tracing attack by proposing a new packet routing protocol.

Packet-tracing is a type of attack when an adversary trace packets from sensors and therefore the receiver can be located.

We would like to touch on a topic of privacy itself before we set goals of the thesis. It is hard to define what is and is not violation of privacy. Everybody is different and being more benevolent to one's privacy can actually be vital. Sharing health data could be beneficial in many ways, for example a customized diet or an instant notification of health problems for hospitals. Some people are ok with sharing data with a company in exchange for better customized services. Privacy perspective could be influenced by the culture an individual is raised in. The US laws are more benevolent than the ones in the EU in terms of privacy as experienced by Google [4].

Lastly, even though laws can be set in a way to protect an individual's privacy, companies could just prompt the individual to agree with their terms. In most cases the individual would not even bother to read through the whole EULA[2] and if the law is abided, the user will grant access to some personal data. As we see, no clear frame for privacy can be established and therefore we will adhere to our defined aspects from the beginning of this subchapter. We will set goals for the thesis in the following subchapter.

1.4. Goals of this work.

Main purpose of the thesis is to address various privacy issues, come up with generally applicable solutions and create a proof of concept in practice. The goals in more detail are:

- To present set of methods to use in a system to provide privacy protection.
- To provide guidelines for identifying and solving privacy issues in a system and using methods mentioned in the previous point.
- To provide a framework which can be used to quickly prototype IoT networks and try mentioned methods to get an insight into sent data and an overall network flow.

[2] End-user license agreement

We have set our goals and we will start with a concept of IoT in the next chapter.

2 The Internet of Things

This chapter focuses on the Internet of Things to give a reader background of the problem. We will describe IoT, look at the trends and usage in practice, explore what an architecture of IoT could look like and what data could be gathered. As we defined in chapter 1, by IoT we understand objects embedded with electronics connected to the Internet. Even though by objects we usually mean simple sensors for the monitoring of an environment, IoT is not limited by that. It could be also things like self-driving cars or drones. We will give a simple example of IoT before going further so that a reader could get an idea what we are talking about.

We will describe a so-called smart home. You have a house with a smart thermostat, which is able to control temperature in the house. Also a small device can control your window curtains. Both objects are connected to a local home network and you can use your phone to synchronize temperature and light with your alarm clock. Temperature will slowly rise 10 minutes before the alarm clocks starts and curtains open once the alarm starts ringing. You can also control these devices remotely (when you are not at home) by establishing secure connection to your local network.

2.1. Current situation and key drivers

Although IoT is still not fully adopted by industry and consumers, the number of connected things increase from year to year. According to Gartner [5], 4.9 billion of things were connected in 2015 (28% increase from 2014) with forecast 6.4 billion for 2016 (roughly 5.5 million of new things connected every day). There are many driving forces for this growth. We describe the main ones.

RFID[3] tags

Tags are small objects which could be attached to a thing. They contain information which can be read by a reader. They could be passive (tag uses energy of the reader) or active (tag has its own small battery and sends signal). Passive tags are

[3] Radio-frequency identification

the most common type. They are cheaper – their cost range between tens of cents to couple of dollars. It depends on energy and distance needed for reading. They could be very small (see Figure 1) and conveniently attached to an object. When the object is tagged, we could process information about the object automatically using a reader. Thus, we are able to better sense an environment and identify objects. Tags are used in transportation (e.g. automatic toll system, container identification and tracking), banking (credit cards), item inventory checking and many more. Being able to track and identify object could be abused to monitor a user.

Figure 1: Possible size of RFID tag (black little square). Source: http://www.rfidjournal.com/

Cost of sensors

According to [6], cost of sensors decreased to an average 60 cents from $1.30 in the past 10 years.

Cost of processing

Processing cost decreased because of Moore's law and commoditization of hardware. Popularity of cloud computing is increasing. It is possible to easily buy computing power as the demand grows. This is convenient for testing and prototyping. Together with a decreased cost of bandwidth, it makes processing of all the generated data affordable and convenient for businesses. The whole network could be smart enough for a reasonable price.

Smartphones

Smartphones and tablets became very popular. Their value increased with cheap ubiquitous internet connection. This brought a new possibility of having a device with a strong computation power at people's disposal anytime they want. A smartphone is a true mobile way to digest and work with information. For an IoT consumer market to be successful, people need to be able to interact with the IoT network on the fly. Let us demonstrate what we mean. A very popular area of IoT are self-driving cars. There is a futuristic idea of having fleet of cars driving in the city with an option for citizens to get a ride on demand. Smartphones will be used for interaction with the service.

Other

There are other technologies which helps IoT adoption. Let us name few.

- Speech recognition so users can easily interact with smart homes.
- Progress in the artificial intelligence which can be used to track user's habits and help them (e.g. reduce energy consumption by monitoring when and what devices users use).
- Smart wearable devices monitoring the user's health.
- IPv6 which enables assigning a unique IP address to an object.

2.2. Communication between devices

The purpose of this chapter is to give a little more information what an IoT network could look like and what communication patterns exists. This is not the main purpose of thesis and therefore we do not provide comprehensive information. However, we find this topic valuable since it gives an illustration of the network which is important for understanding where the privacy elements should be placed. We will start with an architecture.

Every IoT network needs one essential component – brain, hub, intelligence or a similarly named component. What makes IoT valuable is the ability to process various data streams and have a bigger picture for decision making. Position of a single car is not that useful, get positions of thousands of cars in a city and you are able to see traffic jams. We can divide IoT networks by the amount of intelligence contained in the things and in the brain.

Figure 2 describes the usual case. We see many small components (small circles) which represent simple devices (like sensors) with a single task and they send data to the central component. That component represents logic and brain of the network. Communication from simple devices is usually one-directional or with an option to send simple commands (like turn the sensor on/off). Example of this communication pattern could be temperature sensors inside a shopping mall with a central server controlling air

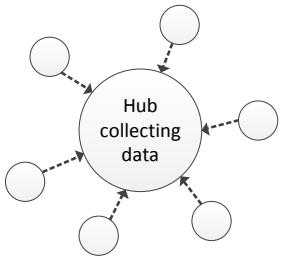


Figure 2: Central hub collecting data from sensors.

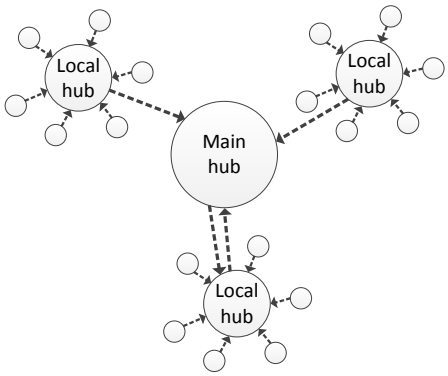


Figure 3: Regions have local hubs.

conditioning. Figure 3 shows more distributed logic. Data are first gathered in local regions and a different kind of data is sent to the central region. Local hubs can transform data for the main hub and provide more complex messages. The communication between the main hub and local ones can be bidirectional because the main hub may know more information and would provide that information to the local hubs so they can act appropriately. For example the little circles could represent sensors monitoring electricity usage. Local hubs would be then smart electric meters aggregating data for a given district and the main hub would represent a data center of an electricity retailer. If there is a sudden increase in energy demand in a region, the main hub could ask other local hubs to cut down consumption of low priority devices for a moment.

The last example of communication pattern are independent devices without any central component. Devices sense surroundings so they know what other things are present or they could be informed about other things in advance. Devices communicate with each other when they find it appropriate. Such example could be self-driving cars communicating between each other to solve local logistic issues (as we said at the beginning, we are simplifying things for demonstration purposes, the real world example would be more complicated and also contain a central component where intelligence of the system would be concentrated).

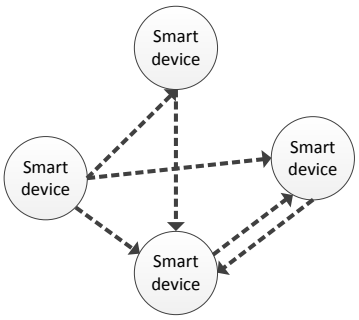


Figure 4: Connection without any central object.

A reader should be now familiar with the IoT network and should have an idea of patterns devices could be connected into. This is will be helpful later when we introduce privacy elements for IoT networks as it will give a better idea where to plug these elements in.

2.3. Industry adoption

We would like to close this chapter with brief discussion about IoT adoption in industry. As we said in the subchapter 2.1, number of connected devices is increasing form year to year. IoT is mostly being adopted by the business market. Sensors are used most of the time. The reasons are cost savings whether it is getting rid of bottlenecks in real-time location of containers leading to inefficiency or driverless mining trucks working 24 hours a day, 365 days in a year. The consumer market is still at the beginning and we can see proofs of concept or plans for the future. A very popular vision is a concept of smart homes equipped with sensors and devices helping the inhabitants. A very popular and probably one of the most disrupting concepts being tested are self-driving cars which should be on the market in a couple of years.

Privacy is not a big concern at the moment because IoT is a new technology and it is still not clear what the market fit will be. There is little incentive to focus on privacy at the moment. It is sort of "where there is no accuser, there is no judge" approach as technical problems must be solved first. However, with the increasing popularity privacy is addressed more. Position tracking is especially sensitive topic [7]. It is quite possible that a set of privacy standards will be required by a new legislation which would, as a result, push manufacturers and service providers to be more concerned with privacy.

We will talk about examples how IoT is used and what privacy issues it brings in the following chapter. For now, we assume a reader understands what IoT is.

3 Privacy violation in examples

This chapter shows a set of IoT solutions with their privacy issues. This will serve as a base for the next chapter 4 where we create methods how to solve these problems. We consider this approach (separation of the problem examples from the problem solutions) clearer because solutions would be then scattered across the chapter in the examples. We do not think it would serve well in the future when a reader would like to use the thesis as a reference and quickly find a set of solutions for privacy issues.

The purpose of this chapter is to demonstrate various types of privacy issues so it could be clear what problems we are solving. Each problem consists of a description (an example what IoT is used for and what benefits it brings) with a set of privacy issues. Examples are future visions of IoT or solutions which are being worked on. In these examples we will focus on parts which concern privacy. Therefore, there will be often described a perspective from the user's point of view and we avoid, for the most part, describing advantages for a business using IoT. Described concepts are more high level and give a taste of problems. In depth examples are provided in the subchapters 5.4 – 5.7 dedicated to privacy patterns.

3.1. Smart grid

A smart grid is an electrical grid which includes a variety of operational and energy measures including smart meters, smart appliances, renewable energy resources, and energy efficiency resources[4]. The grid is provided with sensors so an energy consumption could be monitored in real time. It is hard to satisfy an electricity demand and be prepared for energy consumption peaks. Therefore, appliances are equipped with a chip so a two way communication between an energy plant and the appliance could be established. That is used during peak hours where some appliances (like a boiler) could be told to wait with the energy consumption or their energy consumption could be postponed. The goal is to utilize generators of energy in a better way.

[4] Federal Energy Regulatory Commission Assessment of Demand Response & Advanced Metering – http://www.ferc.gov/legal/staff-reports/12-08-demand-response.pdf

Privacy issues:

- List of appliances in a household can be formed. Then a total price of appliances could be used by burglars to target the lucrative households.
- Real time information about energy consumption would reveal households with a high probability of nobody being inside, thus expose the households to a burglary.
- A household profile can be created from the energy consumption and the list of appliances. A better service (such as prioritized energy consumption) could be provided for lucrative profiles. The household could be wrongly classified and provided with worse services. Implicitly, providing better services for more lucrative profiles could be also seen as a discrimination.
- Providing more information than needed. The goal is to monitor energy consumption, nevertheless it could be possible to find out that nobody was at home and the heating was left turned on. That could be classified as a higher risk of fire and an insurance might increase. That is not connected with the energy consumption.

3.2. Smart shopping

Items are equipped with RFID tags (shelves are tagged instead if the goods cannot justify its cost of tagging). It is possible to use that information and navigate a customer in the shop. If combined with a smart refrigerator or other device tracking what is missing, a shopping list can be created and send to a phone. Or even better, automatic order could be send to a retailer and an order could be delivered to home or picked up in the shop. Tagged items could provide more detailed and better structured information and a phone could be used to read it and to provide better information about allergens, unhealthy substances or origin of the product (for example a product of sustainable agriculture). History of shopped items is preserved and a special tailored sales could be made.

Privacy issues:

- Different retailers have different information about users. The information they have is “this user whose identity we don’t know buys these items”. It is possible to link data from different retailer systems and identify the user across all retailers. This identification could be for example based on a combination of gender, age, frequency of shopping and the purchased products. Therefore a complete picture of the user’s shopping habit could be assembled.
- When information are linked to a concrete profile, they can be traded. Assume that a grocery retail chain knows identity of the user and history of the purchased food. A correlation between health and an amount of bought food of a given category could be created and traded to a health insurance company.
- The customer can reveal more information than necessary by purchasing specific items such as medicine. The retailer has an access to sensitive information such as poor health condition.

3.3. Smart transportation

Transportation consists of few categories. If we are able to monitor user’s movement and public means of transport, we can adjust the public transportation in a way where a waiting time will be minimized and the most people will be quickly transported. By monitoring cars people use, we can detect traffic jams and provide different routes or dynamically change signal lights. Knowing the car’s needs (target location, remaining gas/electricity etc.), we can optimize a lot of things. For example it is possible to utilize parking spots so the time required to find a place is reduced.

Privacy issues:

- Every system in which we track the user’s movement has one big privacy flaw – if we identify the user, we can use it to know his exact location in time or his past movement patterns.

3.4. Gadgets

In this example we consider a gadget to be a small personal electronic tool which can serve many purposes. From a simple one like monitoring certain variables such as a heart rate speed, weight, air quality or a sugar level to more sophisticated ones that enable interaction with an IoT environment. That could be a smart watch aggregating an overall health condition from health sensors, a NFC[5] chip equipped device for a payment or a phone with an app used for a self-driving taxi. An example could be a whole health ecosystem. An app which would tell you a workout routine and what and when to eat. It would be linked to your health sensors so it could know your progress, linking with a fridge could recommend recipes to cook or food to buy and an access to your alarm clock would find a proper time when to weak you up (based on a sleep cycle). In case of danger an ambulance could be automatically called.

Privacy issues:

- The user must provide information about himself if he wants to interact with the environment (such as providing his music list to customize music in a bar). These public information could be accessed by a third party for which the data are not meant.
- Public information from the previous point could be too specific when, in reality, more general information would suffice.

3.5. Summary of the main privacy issues

We could see the main privacy issues emerging from the previous examples. Let us sum these up:

- Identification. Privacy sensitive information could be identified from a set of anonymous data. E.g. we could identify the health condition of an 80 years old patient when there is no other 80 years old patient in the hospital and the processed anonymous health data contains age.

[5] Near-field communication. Technology evolved from RFID for communication within few centimetres.

- Localization. Position of a subject can be revealed. E.g. if we get precise coordinates of a subject, we know exactly where it is.
- Profiling. A profile is created from data and a user is assigned a group. Future actions are based on the group the user is in. E.g. create a group with lucrative customers and wrongly assigning, thus discriminating, the user.
- Revealing more information than necessary. E.g. a health condition from a history of purchased food.
- Linkage. We connect two independent data sets for a user to get a better information about the user. E.g. link systems of two different grocery retailers.

At this point, it should be clear what data could be collected and misused to breach privacy. The reader should also have an idea of different kinds of IoT use cases and their disadvantages from the privacy perspective.

We described some of the IoT use cases and we will focus how to solve their privacy issues in the next chapter.

4 Privacy protection methods

This chapter introduces methods to protect privacy. We specify what problem we address and explain what criteria we use to classify our methods. Then, we systematically go through all methods. Even though we create a set of methods how to protect privacy, the weak link can be always a user. Social engineering or other methods could be used and if the user is not concerned with his privacy, there will be a way to violate it. For example let us consider that the user has an app for his TV which tracks what shows are watched (for recommendation purposes) and the app enables rating the shows. Let us also assume that the user rates shows and shares his show interest publicly using his profile (say Facebook profile). We could link data from the TV app with the public data, get TV address and therefore know where the user lives. Another example is a device becoming smarter without the user knowing so. The typical user has no idea that an image contains metadata, which could be misused when uploading the image (if the software for uploading does not trim the metadata). In the same way, the user may buy a simple IoT device without knowing that a future versions or a new firmware update allows the device to collect more data. We are not concerned with these types of privacy issues here.

At first, we have to get an idea of what elements and techniques figure in privacy protection. We use that to create criteria which we will use to evaluate individual methods. We definitely know that we will have two entities in our system – data sources (usually data from a user) and data receivers (some third party, e.g. corporation or service). We do not want data receivers to get personal information. One method, which is quite obvious and well used, is data obfuscation or modification [8]. It is only logical to use it for our purpose. Another very popular method called k-anonymity [9] is used to anonymize data as well.

Previously mentioned methods were general ones – modifying data. To get more information, we analyzed existing papers concerning IoT privacy. Positional tracking is a very sensitive topic. It is solved in [10] and [3]. Both solutions use different methods than simply data obfuscation. In [10] proof to verify that data source does not lie is required. In [3] injecting of fake packets is used. Another research in this area is concerned with sensors detecting our position when we are close by. It introduces

mixed zones [11] – zones without a reach of any sensor. Smart grid is another big topic of IoT actively being deployed. By 2020 at least 80% of consumers should be equipped with smart meter [12]. Apart from laws and restriction, privacy in smart grid is ensured by not real time monitoring and giving rights to users to delete their data [12]. A good source of information concerning IoT privacy is [13] because of its thorough research performed by many authors. It is not as specific as previous works but it gives a good overview of IoT privacy from many perspectives which helps with our classification. As a result of our research, we can introduce the model we will work with. We introduce the criteria we use for privacy protection methods after the model.

In our model, we will have 4 following entities as shown in Figure 5:

- Data source – sends data.
- Data receiver – receives data from a source.
- Connections between sources and receivers (represented by arrows). Data are exchanged using these connections. We will call the sent piece of data a message and we will say that the message has properties – set of values with its names. E.g. age, timestamp, location etc.
- Transform elements which modifies data or flow of data. Transform elements act as both receivers and sources.

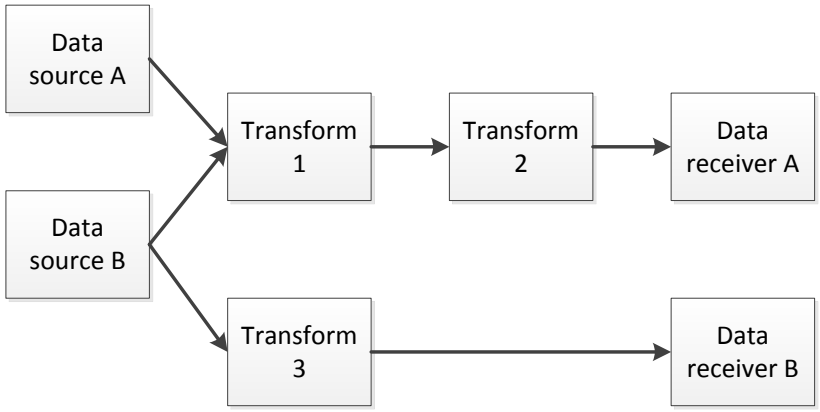


Figure 5: Demonstration of a data flow from sources to consumers. Data are transformed so privacy is preserved.

Final data receivers should not be able to get data in such a way that privacy is violated. That is a purpose of transform boxes. For each proposed method, we will:

- Write description in which we give a source if the method was inspired by some existing work; methods without source are either created by us or are inspired by some generic solution used in industry.
- Example in which we demonstrate the method. This could be just a short illustration.

Then, we will use following criteria to evaluate proposed methods:

- Type of protection. This means whether the protection is based on data modification (in other words transform boxes from Figure 5) or communication with other party (such as confirming that a receiver is authorized).
    - For data modification, we are interested whether they work with a single message or with batch of messages.
    - For communication, we are interested with whom it is and what it looks like.
- Complexity of implementation and of the very method. We establish 3 types of complexity – low, medium, high.
- Randomness. We would like to what degree the method is deterministic. Sometimes a solution needs one given response for one given input and randomness could be a bad thing. Solutions using randomness could be also seen as safer as data could not be reconstructed easily from the received message.

We have our model and criteria so we are ready to introduce privacy protection methods. Methods have to be always used by someone who has domain knowledge and who is able to tune parameters of the method for a particular solution. We will not mention this fact explicitly in each method. Parameter tuning and experimenting could be done using our framework described in chapter 7. Chapter 8 then demonstrates the framework using concrete examples. We give an overview of all methods in the last subchapter.

4.1. Hashing

Description

We hash one or more properties of a message. It is a standard way how to anonymize data (for example mentioned in [14]). By hashing we mean a cryptographic function. That is a function which maps data of arbitrary size to data of fixed size. An important property of the cryptographic hash function is that computing an inverse function is practically impossible and that the same input is hashed to the same value. A salt can be added to the hash function. The salt is a small piece of data which is hashed together with the input. Salt should be a function of some other variable so the resulting hash is different when an environment, in which is the message sent, changes. For that reason a constant is a bad salt. A good salt is:

- Time
- Position
- Random value

Example

Consider a case when we send user position data to a receiver which needs it to identify traffic jams. If we hashed user data, the receiver would know the whole path the user took. The receiver would not know the identity but the path could help and it is definitely not necessary for traffic jam identification. Therefore, a random salt is added to the hash function.

Type of protection

This type of protection modifies single sent message.

Complexity

We rate complexity of the solution as medium.

Randomness

This solution is deterministic and could be made more random by using random salt.

4.2. Trimming

Description

Also known as suppression (e.g. [15]). This is a simple but effective method – trim data, in other words delete some properties of a message. The goal is to send only data a receiver needs so no additional information is provided. The sooner trimming happens in a workflow, the better as a random attack is less likely to hit the part with untrimmed data.

Example

An image has metadata which are unnecessary for a lot of services such as location. A web page with recipes does not need a location of the uploaded recipe photo so we delete it before propagating the message further.

Type of protection

Modification of a single sent message.

Complexity

Easy.

Randomness

This solution is deterministic.

4.3. Generalization

Description

This technique could be seen in already mentioned [15]. In many cases we do not need exact data to get insight into what is happening. Instead, we replace the exact values with surrogates both general enough, that they will provide protection, and concrete enough so we do not lose much information. Value A is replaced by value B and A is subset of B. More values different from A could be replaced by B. A way how to generalize data depends on a problem. We name a few general ones:

- Number values are replaced by ranges.

- In sequence of characters (such as phone number), some are deleted/replaced by another character (e.g. *).
- We use a specific domain knowledge to create a superset and substitute values from that (e.g. replace a concrete car model with its brand).
- Geographical places are replaced by a larger area they belong to.

Example

Assume we want to collect ages of subjects to get an insight into age distribution (for example we could use it for marketing purposes). We do not need exact ages and we can replace them with age ranges. So instead of collecting data such as 18, 20, 23 etc. we collect information in what range the subject is, such as 18-25, 25-30 etc. If we wanted to use generalized data in computation, we could replace each range by its mean. The tricky part is how general the surrogate should be. This should be based on an estimate of number of values going through the system. In our examples, if we analyze one million people, it is quite likely that we do not even need ranges and we can collect just ages. On the other hand if we analyze ages of hundred people with normal age distribution, we need to choose larger ranges for boundary ages (such as 5-12, 55-70) and smaller ranges for other values (such as 20-22, 23-25). This subject is more discussed in the next method k-anonymity.

Type of protection

Modification of a single sent message.

Complexity

We consider it as medium difficulty because of tricky part of choosing the right amount of generalization which is still useful for data analysis.

Randomness

This solution is deterministic.

4.4. K-anonymity

Description

K-anonymity uses trimming and generalization to create such a data set that when given one element, there are at least k-1 other indistinguishable elements. Creating k-anonymity set is a hard problem and is well studied. [16] gives a good thorough description of this method. The tricky part about k-anonymity is how general the generalization should be. Because it is based on type of problem, it is not easy to answer. Following points should help with that.

- A rule of thumb is the bigger K the better privacy but beware of losing too much information.
- K should be based on data records. A big number of distinct records relative to the total data size means a more rough generalization will have to be used.
- A small number of distinct records relative to the total data size does not imply easy k-anonymity. Distribution is important. For example 1, 1, 1, 1, 1, 1, 1, 1, 15, 45, 72 cannot be easily anonymized without losing too much information.
- Only important data should be used, the rest should be trimmed.
- If we know what data we have in advance, there are studies giving a heuristic for finding a good solution such as in [9]. Finding the optimal solution is NP-hard. IoT is mostly used for real-time or near real-time processing, therefore we are not that interested in k-anonymity used for data known in advance.
- If data are generated dynamically and we know a rough distribution, we can have an idea what generalization to use and we can create buckets where we hold on generalized data. Data from a bucket are data send downstream only if k-anonymity of the bucked is satisfied. We create a little artificial example to demonstrate our thought. Let us say we are going to track waiting lines for 2 different menus in a high school canteen. We use ages as an identifier of a waiter. We know that ages are distributed

more or less evenly (assuming all kids have the lunch at the same time) and they will range from 12 to 19. Therefore, we create two intervals 12-15 and 16-19 to track kids. We use 2-anonymity and two buckets – for each line one. Figure 6 represents a situation where a kid of age 12-15 goes into the line with 3 other kids. After that, the condition for 2-anonymity is met and information about number of kids waiting in the upper line will be send. When the condition is broken, information stops being sent. Number of waiting kids for the menu represented by the bottom rectangle is not sent, because 2-anonymity does not hold.

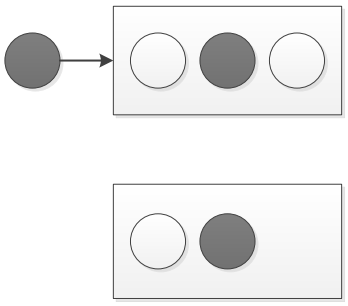


Figure 6: 2-anonymity for lunch waiting lines. The dark circle represents a kid of age 12-15 and the light one represents a kid of age 16-19.

- A case in which we get dynamically generated data, know nothing about distribution and have no idea of origin of data is unlikely. Nevertheless, we can start with a very large generalization and refine it as we get more data. Let us consider a case when we collect people's names and we would like to provide a snapshot of names collected so far. At first, we hide names completely and we publish only number of names. After some time we may decide to publish sex instead of name. When the collection becomes large enough, we may publish the first letter and so on.
- When getting data dynamically, it is important to think how the data will be propagated downstream. We could publish the whole snapshot of data or we just send incoming data further and a receiver would know that at least k-1 undistinguishable data was received by the sender before. When we choose the latter one, we must consider whether the time of sending would reveal more information which could be used to violate the privacy.

We would like to close up k-anonymity with one final thought. The idea of k-anonymity, publishing when there is enough of indistinguishable data, is very effective and a cornerstone for privacy protection. However, it may not be enough in some

cases. More information how to combine it with other techniques and counterexamples when it fails are in the next chapter 5.

Example

Figure 7 shows a left table containing data about people – the name, their city and ID. These data are not anonymized and each person can be identified. We used generalization and trimming to get the right table. We replaced the name with gender, the city with a country and we got rid of the ID. The right table is in 2-anonymity format; in other words for each record there is at least one other identical record.

Name	City	ID
Martin	Prague	1
Susan	Seattle	2
Cate	London	3
Jennifer	New York	4
John	Brno	5
Laura	Manchester	6

Gender	Country
Male	CR
Female	USA
Female	GB
Female	USA
Male	CR
Female	GB

Figure 7: The left table contains data, which are then 2-anonymized and represented in the right table.

Type of protection

Modification of a message. It can work either in batches – collect messages and output the k-anonymity result, or with single messages – saving data and letting the message through if at least k-1 same messages was seen in a given time interval.

Complexity

We consider k-anonymity as a highly complex solution as it is not easy to set *k* and there is a lot of parameters to tune.

Randomness

We consider this solution as deterministic. One modification could be implemented to make it more random. If we work with batches, batch size could be of random size. If we work with single messages, we have a time interval for each seen

message after which the message is deleted. For an incoming message, we check whether there is at least k-1 seen messages and if so, we propagate the incoming message further. This time interval could be made random.

4.5. Expiration of data

Description

In this method we have a component holding personal or sensitive historical data. The data should have expiration time after which too old data are deleted. Of course, the deletion should happen only if that makes sense (deleting historical emails may make users angry). This method makes sense if the component holding data cannot be requested by some third party to output all data or be notified every time new data arrives. In that case the third party would just save the data and no expiration in the original component would help.

Example

Let us say we want to monitor our spending habits. Every time we pay we save the item and its price. We consider this as sensitive information so after a year, we delete the information. Just as a matter of interest, we could create a more sophisticated method. After a year we could replace the item by its category (grocery, electronics etc.) and sum all transactions in a month. We are practically deleting the old information, however, we still keep some information to get an insight into our spending habits.

Type of protection

We consider this method as message modification. It works primary with single messages but could be modified to work with batches.

Complexity

We classify this as medium complexity as some forgetting mechanism based on time must be implemented.

Randomness

If the time period is constant, it is deterministic solution. It could be made random to have a stochastic solution.

4.6. Providing the latest data

Description

This method provides the latest information in a given context. Usually, it is a time interval but it could be also based on a change of state such as providing information about the last person paying certain amount of money.

Example

Say we monitor outside temperature. It is not necessary to send data every few seconds, a minute would be sufficient enough.

Type of protection

This method is not about data modification. The component communicates with itself. It has internal state (such as time interval) and provides data based on that state.

Complexity

We classify this as easy to medium complexity as using time is easy but the latest data could be based on more complicated relationships.

Randomness

We primarily consider this method as deterministic. When using random interval it is probabilistic.

4.7. Shuffling

Description

This method is useful when we send batches of data. Each consists of a sequence of values ordered by certain criteria (usually time). If a receiver is only interested in values and does not need ordered data, shuffled data should be provided instead as sequence may convey more information.

Example

For example, data set representing time for which automatic door were opened could reveal more than needed. The longer time could be linked with more people walking in/out. If we get the data unshuffled, we get an idea what the traffic in time through door looked like. For that reason, we shuffle data.

Type of protection

This method modifies batch of data.

Complexity

We classify this as easy to medium complexity. The shuffling part is easy but the size of how many data should be shuffled to provide privacy, yet not slow down the system has to be explored.

Randomness

This method is probabilistic as it uses probability to do the random shuffling.

4.8. Notification

Description

This is a technique when a data source notifies the user after certain criteria are satisfied. The user is notified and could be even offered an option whether to allow the action or not. Examples of criteria could be:

- Limited amount of requests in a given time interval (e.g. only one user's position request per minute).
- A threshold is exceeded (e.g. amount of people below a certain number).
- Access to personal data is requested.

Example

A user walks into coffee shop and a device wants to connect to his phone to get last shopped coffee to do the recommendation. User is notified and can cancel the request.

Type of protection

This method communicates with a user to notify about or prompt to approve a transaction.

Complexity

We classify this as medium complexity.

Randomness

This method is deterministic.

4.9. User profiles

Description

In many cases the app does not need to know personal data of a user but only a category which the user belongs to (such as young male, person with certain spending habits etc.). So instead of sending personal data linked to a certain profile, we use the profile as an intermediary.

Example

Assume we would like to play music in a bar which would be enjoyed by the most customers. Instead of users providing their music habits, they can send information what profile, from a set of profiles provided by the bar (rock, pop etc.), they choose.

Type of protection

This method uses a profile for communication with the target. Also, it communicates with the user to create the profile.

Complexity

We classify this as high complexity. The idea is not complicated but we consider integration with all the parties involved as hard. There is a part which has to link personal data to profile provided by the third party.

Randomness

This method is deterministic.

4.10. Computation on a client

Description

Oftentimes, a data consumer wants to have data to compute a certain thing. In that process, the consumer gets user's data which can provide insight beyond the desired target. For example, a service providing number of burned calories computes that from a travelled distance and time required. There is no need the service should access exact user's movement. Instead, the distance could be computed locally on a device and the service would receive only the necessary information – distance and time.

We face a problem when the computation is shifted to the user device – truthfulness of data. If the user benefits from true data (such as in our example with an amount of burned calories), he has no incentive to cheat. If the user benefits from false data, there is a high chance of cheating and we need to provide some kind of verification. This verification is a subject of cryptography and security and is its own standalone problem. The thesis is about privacy, therefore we give an illustration of the usage and leave more thorough research to the reader. For example, [10] describes a location privacy for cars using automatic toll system.

A good way of verifying is following. The client has a private-key[6] and signs[7] personal data. The data consumer holds all personal data with the hash resulted from the signing. The consumer also provides a public-key so everybody can send a message safely. Because the data consumer does not know private keys of the users, it cannot link data back to users. When the client makes a claim, it has to provide signed data verifying the claim to the consumer as a proof. A punishment for revealing a dishonest behavior can be added to further reinforce truthfulness.

[6] Secret key, byte array, known only to the user. It is used for encryption/decryption.

[7] Appends the key to the data and encrypts everything.

Example

Say we want to monitor food we buy for various purposes such as a refrigerator being able to make an automatic order, calories tracking or alerting when a bad ingredient is found in food we buy. A retailer would like to track food the customers buy so a special tailored discount for people living healthy lifestyle could be made for the next month. Linking what each customer buys interferes too much with privacy. It is also not needed because everything the retailer wants is just a proof that the customer buys healthy food and is therefore qualified for the discount. That leads to a following process:

- The customers will have a special app on their phone which will be used for paying. Each customer will have a private-key to sign payments.
- Every time a customer pays for a shopping, he signs the receipt and the receipt with signature is sent to the retailer and saved (receipt is represented by R character in the left part in Figure 8, hash is next to the receipt illustration). The receipt is also saved locally in the phone (represented by the right part in Figure 8).
- A discount of 10% for the next shopping on all items which are rated by the shop as healthy is offered. To get the discount, a user must buy healthy food of total price X in the current month.
- When a user wants to get his discount, his client must send signatures of receipts to the server where the total cost of all healthy food is at least X. Computation is done by the client, the signatures are chosen, encrypted by the retailer's public-key and sent to the retailer (the first three horizontal arrows in Figure 8).
- The retailer verifies that signatures exist, checks the total price of healthy food and if the condition is met, sends the discount coupon to the customer (the last three horizontal arrows in Figure 8).

- Only some of the user's receipts are revealed to the retailer, thus his privacy is safe.

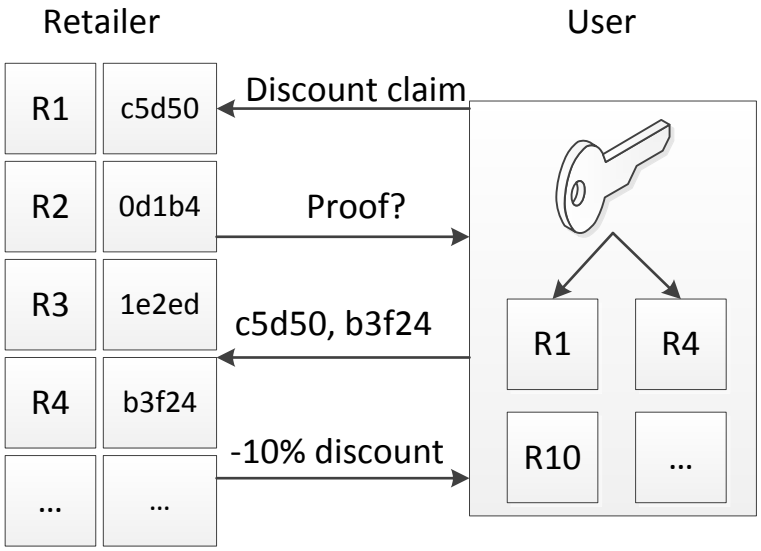


Figure 8: The left part contains the retailers saved receipts with signatures. The right part contains the client with the private-key and saved receipts. Arrows show interaction of getting the discount.

Type of protection

This method could modify data but the core is communication between a sender and a receiver. Receiver of the data verifies that the sender's claim is true.

Complexity

We classify this as high complexity.

Randomness

This method is deterministic.

4.11. Mixed zones

Description

This concept was introduced in [11] related to position privacy. It introduces areas called mixed zones which serves one purpose – to hide a person within other people. When we have places where we are registered for a callback (discount on coffee, interesting news, getting time of a next bus etc.), these places can collaborate and track our movement. Figure 9 shows a café, a restaurant and a bus stop. Each rectangle represents an area where a user is registered. If the user went for a coffee,

then bought something to eat and finished at the bus stop waiting, these places could collaborate and reveal almost all user's movement. That is privacy violation.

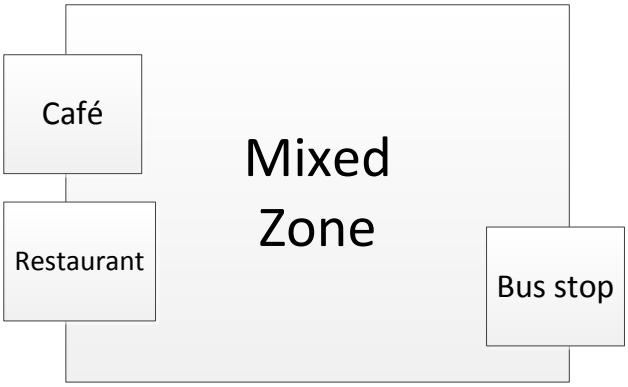


Figure 9: A mixed zone.

The purpose of mixed zones is to act as a place which people are emerging from and to. If the user is given a new identification in the mixed zone, the collaborative places would not be able to reveal him. This, of course, works only if the user is not registered and the places tracking user are of general purpose (not needing sign in). Unfortunately, this method will not work when few people are in the mixed zone. When a person leaves one place, exits the mixed zone, appears in the second location and the time between entering the mixed zone and exiting from the mixed zone matches the time it would take a person to walk form the first to the second, it is possible the person is the same, even though he has a different identifier.

We can solve the problem of few people in the mixed zone by introducing mediator – intermediary element which is explained next – to the architecture. The places would not be able to interact directly with a person, but they would communicate with a mediator. The mediator would listen for nearby people and notify the place. In this model, we would introduce virtual people – mediators would randomly create a fake customer (if there were not enough real) and the places would not know whether the customer is real. Therefore, they would not be able to track people. However, this would work only in a case when the process the places require is very simple and we can fake the customer.

Example

Example was provided with description of the method.

Type of protection

Protection is based on communication between devices emerging from mixed zones and the devices sensing them. Thou identifier of the device in the mixed zone is changed, we would not classify this method as data change.

Complexity

We classify this as medium to high. Simple implementation of mixed zone (change of the identifier) can be quite straightforward, however, creating virtual agents and having a mediator is complex.

Randomness

This method could be random based or deterministic. It all depends what method we use to change the identifier of a device. Introducing virtual agents makes this method more random based.

4.12. Mediator

Description

A mediator is component inserted between a sender of data and a receiver. It acts on user's behalf and the purpose is to provide an abstraction layer between the communicators and shield the user. A sender sends data to the mediator and a receivers receives data from the mediator. Thus, the mediator is effectively hiding both parties involved in the communication.

Example

One example is used in the last part of the previous method. Other example could be demonstrated by automatic food delivery. We use a mediator as a queue for food orders. The mediator collects all orders, say in an hour, and makes request to a grocery shop. It then delivers required goods, mediator notifies users who, in response, pick up their orders. Both parties are effectively hidden.

Type of protection

Protection is based on communication between a mediator, a receiver and a data source.

Complexity

We consider this solution as highly complex.

Randomness

This solution is deterministic.

4.13. Aggregate functions

Description

Oftentimes, data are provided so an aggregate function could be executed. Therefore, instead of sending data, we can send aggregated value. The only thing we need to know is the time window for aggregation and we send the result of aggregation after that time window.

Example

Consider a smart grid. Households are equipped with smart meters measuring energy consumption so the household could be billed properly. Sum of all energy consumption in an hour can be provided instead of real time data sending.

Type of protection

Protection is based on data modification and needs batch of data as its input.

Complexity

Easy.

Randomness

Solution is deterministic.

4.14. Scatter proxy

Description

This method is well suited when we send private sensitive data in order to get a good useful response. We make more requests, one with true data, the rest with different fake data. The receiver is not able to distinguish what data are false or right. We get the responses and choose the right one.

Example

A good example is sending a position to get nearby restaurants. We would not like to reveal our exact position, therefore we choose 5 other positions, send all requests and choose the right response (as we know our current position).

Type of protection

Protection is based on communication between a data source and a receiver.

Complexity

Easy.

Randomness

We consider this solution as probabilistic because using random believable fake data does not follow any hard pattern.

4.15. Delay

Description

Sometimes a lot of privacy information is hidden in the very time the information is send. So the delay can act as a good protection in cases when the information does not need to be processed immediately.

Example

For example position information has the biggest value at the time it is created if we want to monitor real time things. If we delay this information by small amount (for example tens of seconds), our exact position cannot be revealed.

Type of protection

Protection is based on communication with an internal state of the data source. This state determines what delay is used for a message.

Complexity

Easy.

Randomness

This solution could be both deterministic and probabilistic. It depends on whether the delay is constant or random.

4.16. Summary

We created a table of all methods to give an overview. The columns “more info” provides more information based on protection type as mentioned at the beginning of this chapter. When two options are possible, fore slash (/) is used. When two options are possible but one is preferred, the one **not** preferred is in round brackets.

Method	Protection type	More info	Complexity	Randomness
Hashing	Data modification	Single message	Medium	Deterministic (random)
Trimming	Data modification	Single message	Easy	Deterministic
Generalization	Data modification	Single message	Medium	Deterministic
K-anonymity	Data modification	Single/batch message	High	Deterministic (random)
Expiration	Data modification	Single (batch) message	Medium	Deterministic/random
Latest data	Communication	Self	Easy – medium	Deterministic/random

Shuffling	Data modification	Batch message	Easy – medium	Random
Notification	Communication	User	Medium	Deterministic
User profiles	Communication	Data source, receiver	High	Deterministic
Client computation	Communication	Sender, receiver	High	Deterministic
Mixed zones	Communication	Data sources, (mediator), receiver	Medium – high	Deterministic/random
Mediator	Communication	Data source, mediator, receiver	High	Deterministic
Aggregate functions	Data modification	Batch data	Easy	Deterministic
Scatter proxy	Communication	Data source, receiver	Easy	Random
Delay	Communication	Self	Easy	Deterministic/random

At this point it should be clear what different types of protection we can use to protect privacy. We showed a comprehensive list of methods. In the following chapter, we will describe how to combine these methods.

5 Privacy patterns

We provided a comprehensive list of methods used to protect privacy in the previous chapter. However, it may not be clear where to begin when implementing a privacy protection for a system. This is exactly the purpose of this chapter. We will look into following:

- Explain how to think about privacy in a system (what are privacy methods for and where they should be implemented).
- Show examples of how an outside knowledge can break privacy even though we implement the protection.
- Show a workflow of implementing the privacy protection.
- Show patterns for typical problems.

We consider the last two points as privacy patterns. As we can see, the purpose of this chapter is to satisfy the goal from the chapter 1 – to provide guidelines for identifying and solving privacy issues in a system. Let us step back and talk about privacy patterns in general before continuing. Today's privacy issues and solutions concern mostly the web because it is a place where users spend a lot of time, fill their privacy information use web services. Thus such collected data could be misused to breach privacy. Effort was put into these issues and the resulted methods and recommendations of how to protect privacy are called privacy patterns. Privacy patterns could be a little vague in a form of recommendations such as showing disclosure information, an option to delete data, giving reasons why the information must be collected, encrypting messages and so on (we took examples from [17]). Some patterns are what we describe as one of methods in chapter 4. For example Anonymity set in [18] could be seen as k-anonymity or generalization method or morphed representation, which changes data so incoming cannot be linked with outgoing, is basically our mediator with profile.

As we mentioned in 1.3, studying users' privacy is difficult and can be hardly quantified. There is a blurry line between what we call privacy protection method (chapter 4) and privacy patterns described here. Though methods from the previous chapter could be seen as privacy patterns, we consider patterns as something composed

from more methods with defined IoT use case (e.g. difference between a data shuffling method vs problem of collecting a lot of data and what to do with it). Also, we are looking for more specific patterns used for IoT and not for privacy protection in general. We identified 4 main interaction patterns in IoT which could threaten privacy and we describe these patterns later in this chapter. Now, we start by description of privacy elements in a system.

5.1. Privacy elements in a system

As we mentioned in subchapter 1.3, we focus on methods how to transform data and not security, access right, encrypted communication etc. We will call a part of a system, which transforms data or modifies data flow in order to protect privacy, filters. The purpose of filters is to apply data modification techniques from the chapter 4. We explain where the filters could be implemented and how they should be connected in this subchapter. Later subchapter 5.4 – 5.7 explain what types of filters should be used.

The first important thing to realize is where the filters should be implemented. It could be right in the data source or at the end of the data pipeline in the data consumer. The rule of thumb is the closer to the source, the better privacy we get. The reason behind is that the data source has access to privacy sensitive data and the sooner we transform it to a safe form, the less likely it is for some third party to get sensitive information (by purpose or by accident). Unfortunately, we often cannot choose and we are constrained by the architecture. If we have smarter devices, which are able to run code, we can implement some of the filters there. Figure 10 shows an example, where some of the filtering is done in a device. Area A represents the device and two types of data it is sending. We see that Filter 1 and Filter 3 are implemented in the device. Data from Filter 1 are transformed by Filter 2, which resides in an intermediary represented as B. Finally, the area C represents the data consumer which consumes

two types of data from the device. This architecture could represent a smart programmable wearable device (A). It sends transformed data for monitoring (such as time being turned on) directly to the manufacturer (C) and more personal data (e.g. heart rate) to the phone (B) via Bluetooth. Phone transforms data to appropriate format and sends them to the manufacturer.

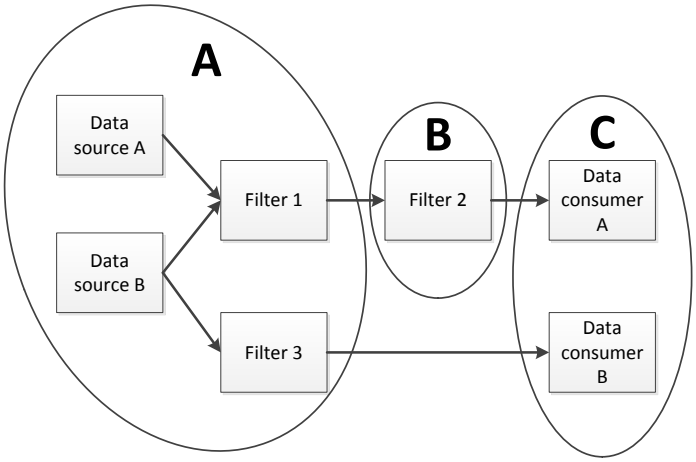


Figure 10: Data flow for a smart device which is able to transform some of its data locally.

If we do not use smart devices and use simple sensors, we must collect the data and transform it in a standalone component. The fewer component, the better, as more components are worse to maintain. As mentioned, we should position components as close to the sensors as possible. If we have a house, a local server could gather and filter data from sensors before sending them further. Figure 11 shows a popular way of collecting data (A) to a cloud (B) and then sending them to the third party.

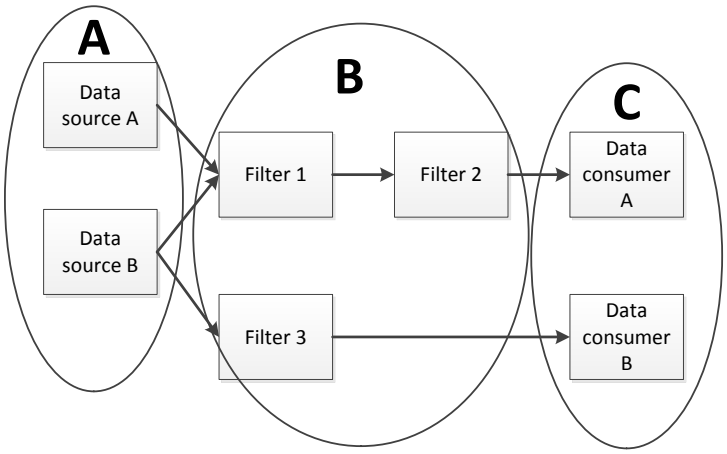


Figure 11: Data are collected and transformed in a cloud and send to a consumer.

We would like to close this subchapter with a recommendation how the filters should be connected together. It starts with devices. If the device is capable of sending various data, we should look at each of data as a separate data source. Also, if the device sends very extensive data, we should separate the data to reasonable groups based on requirements of data consumers and look at these groups as sources of data. We do so, because it is clearer what data goes through the pipeline, thus it is less risky that the unnecessary data, leading to a privacy exposure, leaks.

The next thing is not to reuse filters in more data flow paths. This is not true for components doing some sort of aggregation or monitoring of the whole system (basically it is the very purpose of these components). If one component is present in at least two different data paths, it has more information about the system and can make use of that knowledge. In our example represented by Figure 10 (or Figure 11 respectively), the filter 1 has two inputs and the only justified reason for that is that it aggregates data or needs the other's data source information to function properly.

5.2. Outside knowledge to breach privacy

In this subchapter, we show how outside information could be used to breach privacy. The purpose is to demonstrate that no system is perfect. As we mentioned in chapter 1, we chose to focus on data anonymization and data workflow to protect privacy. However, security and privacy is a big area with problems on many levels, from technical issues like secure communication, encryption or data anonymization, to legal and ethical questions, to self-awareness of the users and their proper habits. Protected data is a powerful method and it is even better when part of a bigger system.

For example, in an extreme case, kidnapping or spying on a target brings a lot of data and counters heavily our methods. Even though solving these issues is not the purpose of the thesis, we would like to briefly introduce how outside knowledge can harm privacy so a reader is aware. This awareness leads to a better privacy implementation as the one who implements the solution has more information about the problem domain than this thesis can provide.

Being part of a set is compromising

This problem arises when we have anonymized data, some of these data contain sensitive information (such as health condition) and we are able to link a person to the data. We necessary do not know the exact record but we get some information. Let us demonstrate using simple example. Assume we have anonymized data in 3-anonymity format (exact number is not important for the example). These data contain age range and a disease (labeled by letters). Figure 12 illustrates this data set.

Age	16-20	16-20	21-25	21-25	21-25	21-25	16-20	21-25	21-25
Disease	A	A	B	A	A	B	A	A	B

Figure 12: Data set of ages and diseases of patients.

We have two problems here.

- If we know someone has a disease B, we know his age is 21-25. Assume we have four people of ages 17, 18, 20 and 21. If we know that someone of them has a disease A, we know it is the 21 year old.
- If both A and B are very sensitive information (such as HIV) and we know someone of age 21-25 is present in this data set, we know he has disease A or B. It does not matter we are not sure about the disease as the fact of having such disease is sensitive enough.

In both cases, the problem was not the wrong format of the data. They are perfectly fine and anonymized but the fact that a person is part of that data set is the key information. This problem is solved by having large and varying enough data. To be clearer, we define it more precisely. Data from k-anonymity set could be illustrated by having representatives from each "bucket". In our case [16-20, A], [21-25, A] and [21-25, B]. Each representative has properties, in our case [Age, Disease]. We create

a graph where vertices are values of properties. Duplicate values are represented by one vertex. In our case, we have 4 vertices – 16-20, 21-25, A, B. The final part is undirected edges. We insert an edge between vertices A and B if there is a representative containing values A and B. Notice that it does not matter if we have representatives with more than 2 properties. Figure 13 shows the graph for our example. Vertices with a small degree are indication of problems mentioned earlier.

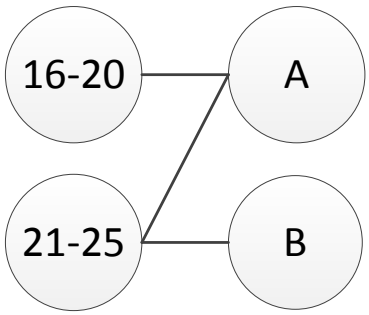


Figure 13: Graph representing 2-anonymity for our example.

Domain knowledge

This is a problem where an attacked knows more about a context in which the data are situated. It could be well demonstrated using positional data. Say we track position of a subject whose identity we do not know (we know only id). If we track the subject thorough the whole day and we know he lives somewhere in the area he is the given day, we could take a place where he spends most of the time (or night) and assume it is the subject's home. That is the reason why hashing method with a salt is used, described in subchapter 0.

Patterns in data representing habits

In this problem, we identify a user from a set of anonymous data based on patterns he follows. A good example for demonstration is shopping. People do not shop randomly. Usually, we tend to shop a set of regular items and we do not shop at random time and place. Generally, personal data about items bought are hidden within huge data of other customers. Assume a person has a very specific habit to shop very unique items at not so common time. If we find the pattern, we can track some of the person's bought items. This problem should be hard to spot in a large data set so the user should be relatively safe if he does not follow a very specific pattern.

5.3. Privacy patterns

This chapter describes steps to take when implementing privacy for a system. A reader must be familiarized with the content of chapter 4 as we will reference methods there without further explanation. We show step by step process and we recommend

to follow it when implementing privacy. After that we show how to solve typical use cases in IoT.

1. We want to determine what devices we have and what data they generate.
2. The next step is to check what data consumers we have and their requirements. We should look at:
    a. What data they need (format, like age, weight, temperature etc.) for computation.
    b. How often the data should be sent. Here we aim for a maximum possible interval.
    c. What are the data used for. We want to know the real reason so we can decide whether the data we sent are too detailed or not. For example, if the requirement is to send electricity consumption every hour in order to bill a user for a month, we do not need to send data hourly and we can aggregated result for a month (or day if month is too long).
3. After that, we create a data source for each of consumers use cases (as mentioned in subchapter 5.1). We trim data the devices generate so only the needed parts are sent.
4. The next thing is where the filters should be positioned. We want to implement filters as close to devices as possible. If we cannot (too expensive or not possible), we implement it in a trusted environment (cloud provider, own server etc.) and from this component, we send data to consumers. We should not implement the filters in the data consumers (for example in form of a library gathering data from devices) because incoming data could be gathered by eavesdropping and library's protection would effectively disappear.
5. The last part is data transformation. What transformation to use depends on type of a problem. We describe these use cases next. In general, generalization and k-anonymity should be used.

Now, we describe typical privacy patterns which should clarify the above mentioned, especially point 5. We would like to point out that we are interested in filters used and we assume proper data sources are created and only needed data sent to consumers (using trimming). Also, even though we provide patterns, there can and probably are use cases, which do not fit exactly to one of the patterns. Each pattern consists from following parts:

- Where to use. This should give more information about a context in which is the pattern used.
- Methods used. These are methods from chapter 4. We distinguish 2 categories – core methods which have to be used and additional methods which are not necessary but they help to protect privacy more. Additional methods will be in round brackets; main methods for pattern are without brackets.
- Definition of a method.
- Demonstration using a practical example.
- Benefits of a pattern.
- Consequences of using a patter.

5.4. Provide data over a period of time

Where to use

An IoT network with a goal to process data in a given time interval. An example could number of sold items in an hour long interval.

Methods used

Generalization, aggregation, (shuffling), (hashing), (computation on a client)

Description

Batch of data is provided. There is a high probability that it is provided so aggregation could be done. In that case we should do the aggregation ourselves. The time interval, over which the aggregation should be made, is based on domain of the

problem. If more complex functionality should be executed, we should try computation on a client. If it is either complicated or not possible, we have to provide the batch. However, data in the batch should be generalized and, if ordering is not a problem, shuffled. The last thing is hashing. That is used when aggregated data have to be linked back to source. We hash the source so identification is hidden yet it is clear what data belongs to what source.

Demonstration

We demonstrate the pattern in a smart grid. Assume we have 4 houses in one block and all are equipped with smart meters measuring actual energy consumption. If we send batches of data with the actual consumption, appliances load could be used to monitor behavior of users as stated in [19]. The purpose of measuring is to bill the households appropriately. For that reason, generalization could not be used as different consumption would be sent. We need to use sum as our aggregation function. The second reason for measuring is a peak forecasting. That leads us to choose a smaller interval for aggregation than let us say 1 day (which would protect privacy well). We decided to choose 1 hour. We also need to identify old data for a household to be able to learn the model used for the forecasting, thus identification has to be used. Identification is a hash of combination of a customer identification and a full street address so it cannot be easily cracked. For example, if we knew that has is based on street name, we could hash all street names and link the resulting hashes with the hashes we get from the households.

Benefits

Apart from privacy protection, less data can be transferred if aggregation is used which leads to smaller overhead of the receivers.

Consequences

Used techniques may be good at a given time. It is possible that in the future a new analysis of data is created. New data could be gathered for this new analysis by changing parameters of used methods but the old ones are not in a good format.

5.5. Provide data for real time monitoring

Where to use

When we want to access data from the IoT network in real time. That includes sensing of environment, monitoring and real time insight.

Methods used

Providing the latest data, generalization, (computation on client), (notification), (k-anonymity)

Description

At first, we should think whether it is really important to monitor in real time. If not, previous method of providing data over a period of time can be used. If real time is needed, we choose a small interval, still proper for solution, and provide only the latest data for the interval. Provided data should be generalized and, if possible, we should do computation on a client and provide only the result. In real time monitoring, the sooner we notify a user when something happens, the better so when it makes sense, notification should be used. If we are ok with not monitoring everything, instead, monitoring only larger samples, k-anonymity can be used.

Demonstration

We use health monitoring as an example. We are interested in a remote monitoring, a concept suggested in [20]. We will monitor insulin levels in blood. This should help people with diabetes[8] to notify them and automatically call ambulance when health is in danger. Providing information every few seconds is not only unnecessary, but would expose insulin levels over time in a very detailed way. Thus we will create an interval of 1 minute and will provide only the latest data. The sent insulin level is generalized by rounding to the nearest whole number so health endangering trend could be monitored and precise sensitive levels are not exposed. As a safe mechanism notification is implemented. When the insulin level rise above a

[8] https://www.diabetes.org.uk/Guide-to-diabetes/What-is-diabetes/

given threshold, a hospital gets this information, the user is notified and have 1 minute to cancel the false alarm; otherwise ambulance rides out.

If we had more complicated example (we would monitor more variables), we could do client computation and sent only a desired feature.

Benefits

Not all sensitive data are exposed. In addition, sent data are generalized and that should provide more safety.

Consequences

Super precise picture of the monitored subject in time is not provided.

5.6. Provide positional data

Where to use

When the IoT network collects positional data or has access to the position of users because of position tracking sensors or other technologies.

Methods used

Generalization, k-anonymity, delay or latest data, (computation on a client), (scatter proxy), (aggregate function)

Description

Let us start with two special cases.

- If we provide data in order to get a personalized response, use scatter proxy.
- If the positional data are provided for a given interval, the first patter Provide data over a period of time should be used.

Now, we can continue with the usual case which methods used to filter incoming positional data so a receiver cannot breach privacy. First, we need to use generalization. For that, we want to create a grid for an area from which we will be

getting positional data. Grid cells must not be necessary squares, they could be of any shape. Instead of sending coordinates, we send what cell we are part of. Cells could be small (few square meters) to large (tens or hundreds of meters). The out coming positional data are generalized and we should create an interval for which only one data is sent. For example, we can choose to send the location every minute. For that we use delay or the latest data technique.

Then, we use k-anonymity. We send the cell we are part of only if there is at least k-1 other subjects in the cell. That is static k-anonymity, we can use dynamic one. In that case we find the group of cells which are connected, satisfy k-anonymity and return this group instead. Note that not all cells, where the subject is presents, should be on the boundary of returned group of cells as it could reveal position. In extreme case imagine 2-anonymity and two cells connected by a segment as shown in Figure 14. It does not provide anonymity, because we can clearly identify the positions if we know the segment is used. Sure we cannot tell whether the A or C is on the right end but having an information that there is position where A or C is present is sensitive. So for example using a rectangle would be better.

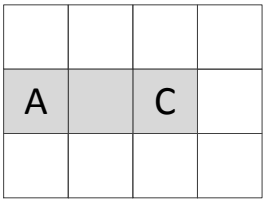


Figure 14: 2-anonymity for subject A and B. The gray area is the returner result and provides no protection if we know subjects are on the edges.

The final point is structure of data. If possible, all identification should be trimmed or hashed so it is not possible to infer the identity from the positions of the target.

If the positional data are used for the computation, we should try to shift the computation to the device or our trusted component instead of the third party. Such example could be a fitness app tracking our running and computing burned calories locally instead of sending data to a third party. If we do not use client computation, we could sent aggregated data (such as total number of entities in a region) instead of sending individual positons.

Demonstration

We choose public traffic optimization in the city based on density of people in areas as our example. Let us say that a city decided to track movement of its citizens to optimize public transport. Trams and busses will be deployed in real time more to

places with high density of people. To protect privacy of the people, mentioned techniques will be used. First, we define a 2 minute interval and we will send positional data after these 2 minutes. The premise is that 2 minutes is a reasonable interval to move away from a place but at the same time it is still short enough that we will get an insight into concentration of people. We use the latest data technique as we want to have a precise snapshot in time and we do not want to have delayed data.

Next, we have to decide what areas to use. We are interested in locations near bus and tram stops so we create a circle with a radius of 80 meters. 80 meters is an average walking distance a person travels in 2 minutes. Location from the messages are used to choose what area the user is the part of. So far, a person sends what area he is part of every two minutes. Only needed data are send, thus only the area is send further the pipeline.

The last part is k-anonymity filter. We will use dynamic k-anonymity. We will hold number of incoming messages (= information that somebody is there) for every area. This number would grow as more people came over time. For that reason, every time we get a new message, we will queue a task which decrements the number of messages after 2 minutes. We set k to 20 as we think that 20 people is large enough to blend in and the data are anonymized already. The last part is output of k-anonymity which is the number of people in area. This serves as an aggregate function because the city does not need positional data; it needs only number of people in areas.

Benefits

Positional data are anonymized and if aggregation is used, the receiver does not even receive coordinates. K-anonymity is a strong mechanism how to hide data sources which are not surrounded by other sources and for that reason, would be more vulnerable to identification.

Consequences

We will not receive all the positional data as sparse areas are filtered out by k-anonymity. In some cases that could be a problem as we would not be sure whether somebody is in an area or not.

5.7. Provide data to customize services

Where to use

This is the case in which we send information in order to get better services. It could be discount based on shopped items, automatic temperature adjustment at home or damage prevention of industrial machinery.

Methods used

Notification, mediator, client computation, user profiles, (expiration), (generalization)

Description

Services should use profiles and users should presents themselves under profiles. The client should compute what profile to use and the consumer should check that the claim is valid. Mediator could be used as an intermediary to shield user or as a component where the profile is computed. Notification has to be in place so the service cannot access user data without notice.

If the consumer needs information and profile does not suffice, generalized information should be provided. Historical data should not be accessed by receivers; if possible, data should have expiration and after that deleted or made private.

Demonstration

We will use a simple yet useful example. Let us say that user's device tracks what movies he watches and temperature he has at his home. Let us consider a new service of a cinema: provide more comfortable environment by adjusting temperature and also by more relevant movie trailers before the main movie. User's data should be used for that purpose. Just aimlessly providing these information to the cinema is privacy breach (though data might not seem that personal). Therefore, the cinema creates profiles. For our simple example it is just comedy/action for genre of a movie and cold/hot for temperature. So after having profiles of users coming to the cinema, temperature and movie trailers could be adjusted in favor of majority.

So how does the profile creation look like? A user comes to the cinema and gets notification that the cinema needs his personal information to create a profile. It also gets information about what profiles are available. He then, if he finds it beneficial, creates appropriate profile and sends that to the cinema.

Mediator could have been used instead of client computation of profile. It would get last 10 movies watched in 6 months (expiration is put in place) and one of temperature ranges the user has at home (16-20, 20-22, 22-24, 24-26 °C, this represents generalization).

Benefits

Using profiles hides all information and still lets the service do data analysis. Users can also switch easily between profiles when they find appropriate.

Consequences

The receiver has to verify the user has right profile or it has to trust him. Also, received data are not as detailed as without profiles. Well selected profiles should remove that problem though.

At this point we are finished with the first two goals we set in the first chapter. In the next chapters we will concern ourselves with a framework used to test and prototype IoT privacy solutions.

6 IoT privacy experimentation framework

When implementing privacy, it is hard to know whether we did the right job or not. It is hard to draw an exact line of where the privacy starts, thus we would welcome a way to prototype and try various solutions. We would like to have a framework, in which we would form IoT network with privacy filters. We could then generate data and see what they look like after going through the network. This chapter introduces a vision for such framework and shows high level requirement based on the knowledge we got from the previous chapters. We start by describing what we want to create, then a list requirements for usability we find important is introduced and, lastly, we discuss a message format. We use terminology from previous chapters, especially from chapters 4 and 5.

6.1. Vision

Our vision is a software in which a user can choose from filters introduced in chapter 4. These filters would be assembled into a network which is then used to transform data. The data could be generated by another components which could get the data from a file (in a specific format such as XML or CSV) or could be configured directly in the framework. One could imagine the data generators as sources which start a flow through the network.

The reason for having such tool is simple. We could prototype and test various methods connected together and look at the output data. We could then compare these results and evaluate privacy. For that reason, an important part of the framework must be a method of looking at the data in various parts of the network. This could be seen as logging of incoming and out coming data. That is important because we can then look at the data in various stages and observe what is happening. The possibility of being able to construct the whole chain of transformation from the resulting data helps to understand what happened and whether it is what we desired. We will describe our vision more formally and we will look into high level ideas how it could look like.

6.2. Goals

We aim to produce a framework which provides a convenient way to create an environment in which we can prototype and try methods mentioned in previous

chapters. This framework will be in form of a library distributed to one or more machines. The library requirements are in the next subchapter, here we focus on overall picture.

We want the framework to be able to work in a distributed environment so a large simulation could be made. Also, users of the library might want to test their systems in our framework. As we mentioned in subchapter 2.2, IoT networks contain edge components which collect and send data and one or more central components where the intelligence of the whole system is concentrated. As a trend of cloud computing continues to grow (see Amazon [21] or Microsoft cloud revenue [22]), it is only logical to have an option to integrate with clouds if needed.

The next thing are data sources. It should be easy to create a new data source generating data. On top of that, we want to be able to integrate the framework with existing solutions generating data when needed so the environment for testing can work with our data sources (which can be the one used in production).

Lastly, because the purpose of the framework is prototyping and trying new methods, we want to be able to get outputs from different filters as they process messages. This logging functionality should be extensible to customize logged messages, and base logging (such as create log files) should be included in the framework ready to use. There should be also a way how to display logged messages in the real time.

To sum it up, we have following goals:

- Create a framework which enables easy prototyping and simulating of different privacy methods.
- We want to develop the framework in a way that is easy to integrate it with cloud, middleware or generally use it in distributed environment.
- The framework should be able to work with various data sources which are not too complex such as CSV files, XML files or JSON files. These sources can be also other programs sending data in some simple format such as mentioned in the previous sentence.

- We want logging functionality so the sent messages could be examined afterward or on the fly.

6.3. Library usage requirements

Our user is a programmer using the library to create classes for the framework to prototype different privacy methods. Work with the library is composed from three parts:

1. Definitions of components generating data.
2. Definition of filters which will be connected to the data generators. We will configure a way how they will be connected and with what parameters. We also define logging of messages.
3. Execution of the workflow and observing results.

We would like to use Fluent API for filter configuration because we think it is very convenient and natural way. We can define Fluent API as a syntax for calling of functions in which we preserve order in which are function executed. For example, let us say we have two functions add, which adds an integer value to its parameter, and multiply, which multiplies a parameter by an integer value. If we wanted to add 2 to 5 and then multiply everything with 10, we would write `Multiply(Add(5, 2), 10)`. In Fluent API terminology, it would be rewritten as `5.Add(2).Multiply(10)`. So with filter example we imagine something like this: `dataSource.AddFilter(trimFilter).AddFilter(generalizeFilter).Log(./tmpFile.txt);`

Execution of a workflow should be with a minimal overhead with an option to customize it to the user needs. It should be easy to add a new custom filter. We believe this is a better approach than optimizing beforehand and building a functionality that is either not needed or needed only occasionally. Making the system extendable leaves the door open to future needs. Additionally, we would like to have a uniform system for exception handling across the system. Lastly, because the system could be distributed, we would like to an easy way how to use the library in the distributed environment.

6.4. Message format

We were vague about the data sent. We will call data, which are exchanged between components, messages. We have following requirements for a message:

- It should be very simple with a minimal functionality so it could be used in many different scenarios. The main purpose is to hold data.
- It should be transferable between different environments. The format should be easy to implement in different languages or platforms.
- It should allow future iterations. We cannot be quite sure with the future of the library. This is also the reason for the first point – keep it simple.

We decided usage of a set of string key-value pairs as a message satisfies all points mentioned above. Filters will use these key-value pairs and we add an optional field which can be filled with a byte array for part of messages not needing privacy filtering. We think it is the best solution from opinions we considered. Let us present these options.

1. Pure binary format. This is the most performant option but obviously the most restricting. Therefore we do not find it appropriate.
2. A base class with option to inherit and implement own properties. This is a very comfortable way for a programmer. It is not transferable as we would like but it could be solved by serializing into platform independent format. The bigger problem is with filters as they need to modify a message and to do so, they need to know what field to modify. Because we want the filters to be used generally with various message types, their input must be unified. In programming terminology, the input would be a base class or an interface, thus the properties of the custom message classes would be hidden for the filters. We would need to use reflection or complicated design structure and in the end we would need a name of the property to transform. A good example for demonstration is a filter which deletes some fields from the message and passes that message further. Having that functionality for an arbitrary class, which inherits from some base class or implements some interface, is very complicated when compared to a message composed from a set of key-value pairs.

3. Using key-value pairs with binary values. This is almost the same as the solution we came up with. The idea is that using binary values would be faster than string ones. It is not necessary true as a string could be interpreted as a byte array. Apart from that, filters would have to know binary format of the field to be able to modify it. For example, if we wanted a filter which appends a value to an arbitrary field, it is easy to do that with the string format but not with the binary as we have to understand the binary one if we want to use it. This contradicts our requirements. It is true though, that sometimes we would like to encode information to the byte array and we do not need to use it in filters (it is up to the final receiver to interpret it). For that reason we added the binary array field to the message.

As for the message size, there is no need to give restrictions unless the message has some obscure size like couple of megabytes. The only limitation is when we integrate the solution with existing industry technology. For example, when using Azure IoT Hub[9], it is 256 Kb as defined in documentation:

> *"Like Event Hubs events, device-to-cloud messages can be at most 256Kb in size, and can be grouped in batches to optimize sends. Batches can be at most 256Kb, and at most 500 messages."*

The last thing we require is the ability of a message to clone itself – creating a copy. This is important as copies of an incoming message can be created, modified and passed further for various receivers. Because objects are passed by reference, when having two receivers receiving the message, they would modify the same object without knowing so. Therefore each receiver is able to create a copy of incoming message and work with the copy without affecting others.

By describing the message, we have closed this chapter. At this point it should be clear what requirements the framework should meet. We describe documentation of our implementation in the next chapter.

[9] https://azure.microsoft.com/en-us/services/iot-hub/

7 Implementation documentation

Purpose of this chapter is to use output of the previous chapter 6 and provide information for implemented solution. We call our library, which implements the mentioned concepts, Nuntius – a Latin word for a messenger. We start by describing technology choice and similar solutions. Then structure of the solution, explaining main classes, interfaces and the whole concept of component interaction is showed. We continue with a separate subchapter about exception handling, a short demonstration of communication with IoT Hub and receiving messages on the opposite end and we close the chapter with an explanation of how a new filter can be implemented and how to generate data. The following chapter 8 is about framework demonstration using the Nuntius library and should be read if the purpose is to see it in action. We would like to point out that when we talk about a user, we mean a programmer using our library.

7.1. Technology choice

We considered Java and C# as a language choice because we feel comfortable using them and they are well supported. We see C# as more suitable because we are more fluent with the language, events are first class citizens in C# and asynchronous programming model in a new version of C# is better. To elaborate more, we think events are important because of the nature of IoT (devices fire events). What we mean by a better asynchronous model are `async` and `await`[10] keywords with the `Task` class which make programming much easier. On the other hand, by not choosing Java, we are giving up a better multiplatform development (different OS[11]) and much bigger community and ecosystem. However, Microsoft is officially trying to make .NET multi-platform as proven with .NET Core [23]. We think it will lead to a better adoption of C# as a multi-platform language. Therefore we will target .NET framework.

[10] https://msdn.microsoft.com/en-us/library/hh191443.aspx

[11] Operating systems

7.2. Similar solutions

Before describing implementation we show similar solutions which we could reuse. There is a library called TPL Dataflow Library[12]. This library is very similar to what we want to do. It uses blocks, connects these blocks together and data are passed through the block network. It is a good inspiration for our problem, however, we will not use it from following reasons:

- It is not build with distributed execution (across machines) in mind. There has to be confirmation of sent messages and a sender explicitly knows about receivers. We would prefer subscribe model in which we have data source notifying about messages with subscribed listeners.
- Data consumer must pull a message instead of a pushing data.
- Messages have to be confirmed.
- It is not build for IoT and therefore it could be a weak link in the future.

Another similar solution is Windows Workflow Foundation[13] (WF). WF creates a state machine for user defined activities. Activities can be sequential processes, if condition, sending message etc. However, it is more suited for business processes than real time simulation of processing of large amounts of sensor data.

7.3. Structure of the solution

A CD with the code of implementation is attached with the thesis. The folder **Nuntius** contains a visual studio solution file called **Nuntius.sln**. It was implemented using Visual Studio 2015 and therefore should be opened using that version. The solution is quite simple. It contains 3 projects:

Nuntius.csproj

This is the main project with the functionality described in the thesis. It defines privacy filters, exception handling, library configuration, messages and their

[12] https://msdn.microsoft.com/en-us/library/hh228603(v=vs.110).aspx

[13] https://msdn.microsoft.com/en-us/vstudio/jj684582.aspx

propagation. This is the library which provides privacy protection and is environment agnostic. Integration with IoT Hub or other solutions have to build on top of it.

Nuntius.Azure.Integration.csproj

This is a small project which builds on top of **Nuntius**. Its purpose, as the name suggests, is to integrate with Azure IoT Hub. It extends interfaces `IEventSource` and `IEventTarget` from Nuntius library so messages could be passed to and consumed from IoT Hub. It is not part of the original project so a different integration could be created without pollution the Nuntius

Nuntius.Tests

This is the test project, which uses behavior driven testing. The subject of tests is the Nuntius project.

7.4. Main components and interaction

Base interfaces

A cornerstone of the whole system are the following three interfaces: `IEventSource`, `IEventTarget` and `IEventPropagator` which combines previous two together. These interfaces exchange instances of `NuntiusMessage` class which represent messages. So basically we have instances implementing `IEventSource` which sends messages. Objects implementing `IEventTarget` interface then consumes these messages and objects implementing `IEventPropagator` then serves as an intermediary which both consumes and produces an instance of `NuntiusMessage`. As we can see in Figure 15, `IEventSource` sends messages through an event. Consumer of this event must return an instance of `Task`. It is more useful than void because the consumer of the event can return whether the message is being processed, was successfully

```
public interface IEventSource
{
    event Func<NuntiusMessage,Task> Send;

    event Action End;
}

public interface IEventTarget
{
    Task ProcessMessage(NuntiusMessage message);

    void EndProcessing();
}
```

Figure 15: IEventSource and IEventTarget interfaces. Comments removed for clarity.

processed or an error occurred. This feedback is then used by exception handling which is explained in subchapter 7.5. We thought about passing data back through `Task<NuntiusMessage>` in event handler. This could be handy, however, in cases in which we have more than one listener the things would get complicated very quickly. For an instance of `IEventPropagator`, it would be not clear what message to return when we get all messages from our listeners. Or in what order. We decided that this added complexity and overhead is not worth the effort of implementing because benefits do not outweigh the cost. The next thing we see is `End` event which signalizes that the `IEventSource` stops sending messages. This event have to be called after we are done with sending so the further components in the pipeline gets the information and can act accordingly such as freeing unmanaged resources. `IEventTarget` is explained in the following paragraph.

Connecting together

We see that a method `IEventTarget.ProcessMessage` has the same signature as `IEventSource.Send` listener. That is on purpose so instances of classes implementing interfaces can be connected easily. By connecting we mean registering `ProcessMessage` method as a listener of `Send` event. The method doing registration is called `LinkTo` and we have two options where to implement it.

- Add it to `IEventSource` interface. Disadvantage is that it would have to be implemented every time someone implements `IEventSource` interface and we do not want that. We could create a base class which would implement the registration and which would serve as a base class form which would every class implementing `IEventSource` inherit. We do not want that either because C# does not have multiple inheritance and we would restrict programmers this way.
- Have an extension method for `IEventSource` interface. This acts as a default interface method implementation in a sense.

We decided the second option is better. Therefore there is a static class `EventExtensions` which does all the registration. This approach creates a nice programming model for users as we can write code similar to this one:

`source.LinkTo(filter1).LinkTo(filter2).LinkTo(consumer);` which is what we wanted as described in subchapter 6.3.

Interaction between components

The interaction looks as following. One or more data sources implementing

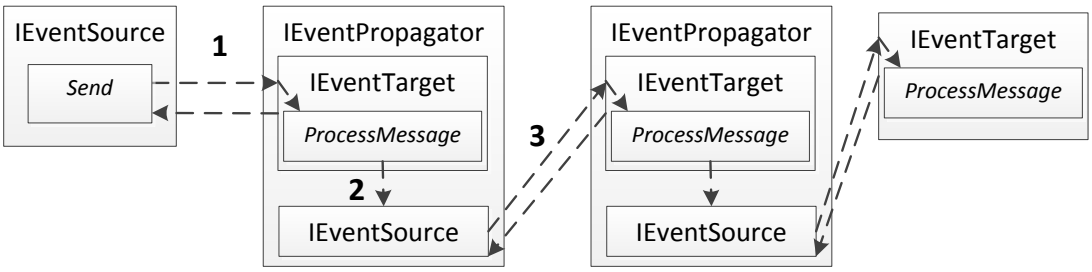


Figure 16: Schema of how the message is sent. 1 – Send event is fired and thread handling event calls registered IEventTarget. That starts asynchronous in a separate thread and returns. 2 – After the processing is done, IEventSource triggers the Send event. 3 – Whole process starts again.

`IEventSource` interface are linked to various filters implementing `IEventPropagator`. These filters modify sent messages and are connected to other filters or instances of object implementing `IEventTarget` where the workflow ends. Every time the `IEventSource` wants to send a message, `Send` event is triggered and asynchronously processed by `IEventTarget.ProcessMessage`. No order of messages or delivery confirmation is trigged by default. Figure 16 shows the workflow.

7.5. Exception handling

An exception can occur during the `End` or the `Send` event. Exceptions are handled by objects implementing `IEventSource` as we find it logical, that a caller notifies about exceptions originated in a callee. It is not easy to propagate the exception in our asynchronous event model and therefore there is a static class `NuntiusConfiguration` with a static event `Exception`. This event is called when an exception should be thrown in a workflow. A user can register his callback for this event and act appropriately. Origin of the message is represented by `CommunicationExceptionOrigin` enum so it is clear in what part of the workflow the exception originated. The reason for this concept will be clear after finishing this chapter. We start with `End` event.

If the `End` event is invoked, the results of how the callbacks end is awaited and if an exception is thrown, it is caught and distributed through `NuntiusConfiguration`. We wait for the callbacks in separate thread so the component is not blocked.

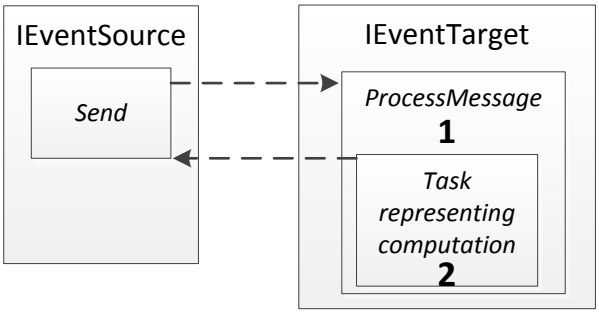


Figure 17: Representation of interaction between IEventSource and IEventTarget and where an exception can occur.

The more complicated situation is with `Send` event. The exception can originate in one of two places as described in Figure 17. When `Send` event is fired, the thread firing the event calls the first of the registered callback and the thread transfers to the `ProcessMessage` method and some computation, before returning `Task`, resulting to exception can happen. This is represented by number 1 in Figure 17. After the `Task` representing the main computation is started, the thread returns back to the `IEventSource`. Then, an exception can occur in the `Task`. That is represented by number 2. The important thing here is that `Tasks` swallow exception by default and we have to either await them or call `Wait` method to exception being thrown. For that reason, we created a class `EventSourceBase` which implements `IEventSource` and accepts `EventSourceCallbackMonitoringOptions` which signals whether the tasks should be awaited, thus exception is thrown, or not. `EventSourceBase` can be then conveniently used by every class which wants to implement `IEventSource` and does not want to do the handling itself.

7.6. Communicating over machine boundary

Let us explain first what we mean. As we explained at the beginning of this chapter, we will communicate with IoT Hub. Therefore, we need a way how to send data to IoT Hub and a way how to consume it. We do this is by implementing `IEventTarget` for device to cloud messaging and `IEventSource` for consuming cloud messages. Both implementation, `DeviceToCloudEndpoint` and `CloudToDeviceEndpoint`, are in a separate project Nuntius.Azure.Integration. We have a separate project so Nuntius project is not polluted by platform specific implementation. By using `IEventTarget` and `IEventSource` interfaces, we do not to

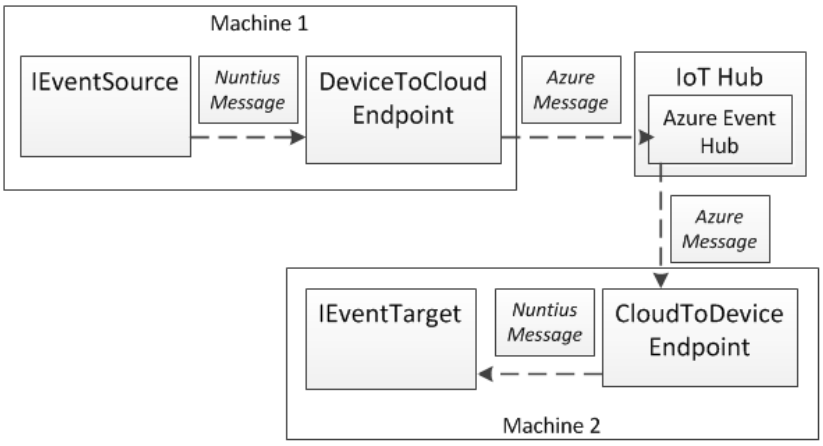


Figure 18: Communication between Nuntius and IoT Hub. Messages are transformed appropriately.

change our workflow between devices and backend as we send `NuntiusMessage` to `DeviceToCloudEndpoint` and consume it on the other end using instance of `CloudToDeviceEndpoint`. Figure 18 demonstrates this concept.

The implementation of `DeviceToCloudEndpoint` and `CloudToDeviceEndpoint` is simple as we need to demonstrate `Nuntius` in practice and we do not need a high throughput.

7.7. How to implement a new filter

We describe basic things for creation of a new filter. More detailed description is in subchapter 8.2 where we implement an example using the framework.

The filter have to implement `IEventPropagator` interface and should inherit from `EventSourceBase`. `EventSourceBase` provides `SafelyInvokeSendEvent` and

`SafelyInvokeEndEvent`, both take care of invoking the proper event with exception handling. The filter must implement `ProcessMessage` method and the main computation should be in a separate `Task` which is then returned by the method. The reason behind is that `IEventSource.Send` event, which is handled by the filter, is handled in a quick asynchronous way and `IEventSource` is not blocked. The folder Privacy contains some of the privacy methods mentioned in chapter 4.

7.8. Logging

Logging is in the folder Logging and is represented by interface `ILogger` which defines to simple methods `Log` and `EndLogging`. The class `LoggingExtensions` then contains extension methods which can be used to add logging to the linked filters. File logger is represented by the class `FileLogger` is provided out of the box. Because `FileLogger` implements both interfaces `ILogger` and `IEventPropagator` we have two options of connecting it.

- We can use LinkTo method and connect it like any other filter. E.g. `filter1.LinkTo(new FileLogger(…)).LinkTo(filter2)`
- We can use `Log` or `LogToFile` (which is a convenient wrapper around `FileLogger`) extension methods like this `filter1.Log(new FileLogger(…)).LinkTo(filter2)`.

Though both methods look the same, there is a difference. `LinkTo` extension needs a complex (in a way of properly implementing) `IEventPropagator` implementation whereas `Log` needs simple `ILogger` interface. Not only that, `Log` method is overloaded and we can pass a simple lambda for logging such as `filter1.Log(message => Console.WriteLine(message))`. Also, when using `LinkTo`, the logger is looked at as `IEventPropagator`, thus another `IEventSource` is able to listen to the `Send` event. On the other hand, the `Log` functions returns the same

`IEventSource` respective `IEventPropagator` it is extending, therefore the next filter in the chain listens to the original `IEventSource` respective `IEventPropagator` and knows nothing about the logger. The difference is shown in Figure 19. The upper picture uses `LinkTo` method, the lower `Log` method.

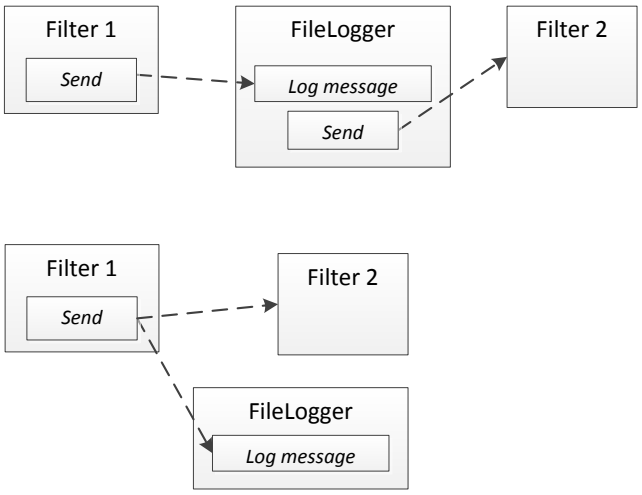


Figure 19: The difference between using LinkTo or Log method. The upper picture shows LinkTo method.

7.9. Data generation

We briefly describe how to generate data (concrete example is in the next chapter). We have to implement `IDeviceSourceEndpoint` interface. We should not implement the interface but rather inherit from `EventSourceBase` (sending has to be implemented) or `PeriodicEventDeviceSource` (to send messages periodically after a time interval). Then, we just instantiate the class and link it to filters. We can pass parameters to the instance and these parameters will be used to send messages. For example we can use some of online data generators[14] and export the data to csv file. Then, we create a source which inherits from `PeriodicEventDeviceSource` and which gets the path to this file, parses and loads it and then periodically sends the records.

[14] For example http://www.generatedata.com/

8 Practical demonstration

The goal of this chapter is to take the framework and implement examples from the first and third privacy patterns – a smart grid and a traffic optimization in a city. The attached CD contains a folder named Nuntius-example. The folder contains a visual studio solution file called **Nuntius-example.sln**. It was implemented using Visual Studio 2015 and therefore should be opened using that version. Nuntius-example solution references the **Nuntius.csproj** instead of compiled library (.dll) so easier debugging and testing can be made. We start with the smart grid; it is easier example.

8.1. Smart grid example

The project **SmartGridExample.csproj** represents a smart meter problem as described in subchapter 5.4. The demonstration is quite straightforward and does not need too much explanation. This example is basically demonstration of main capabilities of the framework and more thorough example is in the next subchapter. We decided to demonstrate the problem using 4 houses. Each house is equipped with a smart meter, thus is generating messages representing actual energy consumption. Class `Household` represents a single house which periodically generates messages with consumption. Notice how we extended `PeriodicEventDeviceSource` class which handles periodic invocation for us. The class is passed an array representing consumption and id of a household. The array it cyclically iterated through and messages with consumption are sent. A message with the consumption is sent every second (we consider 1 second of simulation as 10 minutes in the real world).

The simulation is in the class `Program`. The major part of the simulation is in `LinkEverythingTogether` method. We have 4 households generating consumption and each household passes its identification. The goal is to hide the real identity of a household and to send aggregated consumption in 30 minutes. Before linking filters, we log original messages sent to ./Original/household_X.txt, where X is a household id, so we can see original and result messages. After that we use hashing of the id. As mentioned in in subchapter 5.4, we use street name as a salt so the household id is hashed together with the street name appended. The last part is aggregation. We

aggregate all messages in 6 seconds (which represents 60 minutes in the real world) and send the aggregated result to the last part which is just a file logger logging the message to the .\After_privacy\household_X.txt file. Aggregation is done using `IntervalAggregateFilter`. To see `IntervalAggregateFilter` just look at the class definition and public methods; everything is commented. Figure 20 demonstrates the whole process using the first household.

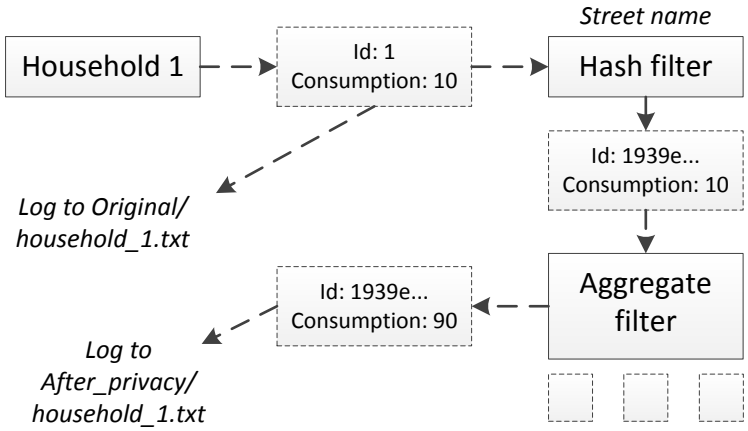


Figure 20: Demonstration of the workflow for the first household.

As we see we anonymized data. Originally, we had data generated in a small interval with specific consumption. The data contained id of the households. After the process, we have data aggregated in an interval with a hashed id as shown in Figure 21.

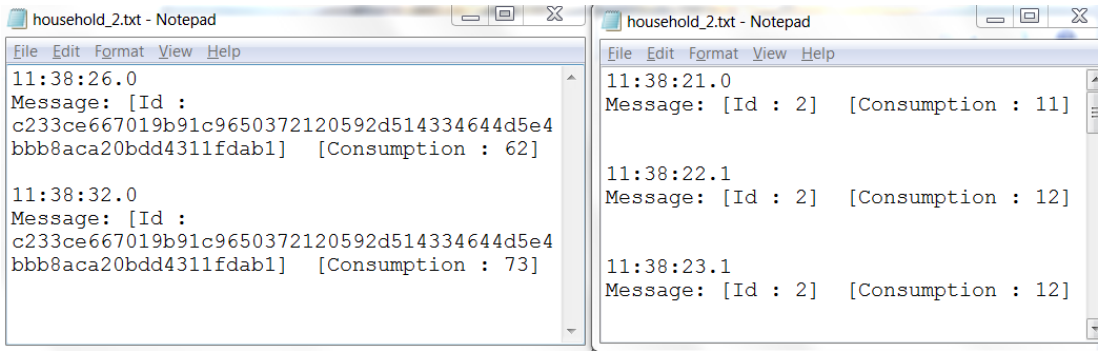


Figure 21: The left notepad shows the data after anonymization, the right shows a sample of data before.

8.2. Traffic optimization example

This example is more complicated and we assume a reader read the previous example so we can focus more on the simulation and techniques. The simulation is represented by the project **TrafficOptimizationExample.csproj** and demonstrates the example from subchapter 5.6. We assume we have 5 mobile phones which send positional data of their users. The phones are represented by `MobilePhone` class which is implemented in a same way as `Household` class from previous example – it periodically generates positional messages. We use four regions in which the users will move. We call these regions A, B, C and D. For the sake of clarity we use integer coordinates and values which are far from real but demonstrates our thought. Area A is represented by Longitude 0 – 50 and Latitude of 0 – 50. Other areas are represented in a similar way. Everything is shown in Figure 22.

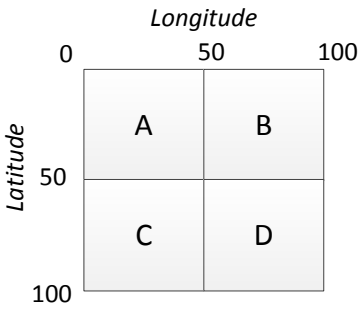


Figure 22: Regions where the simulation happens.

The first thing we will look at is a generalization filter which translates positional data to a region. The filter is implemented by `CoordinatesGeneralizationFilter` and is quite straightforward. When creating a new filter class, we need to implement `IEventPropagator` interface. Though it defines only two methods, it could be tricky to implement them properly, therefore we inherit from `EventSourceBase` class which provides convenient methods `SafelyInvokeSendEvent` and `SafelyInvokeEndEvent`. These two methods properly handles exceptions thrown in the process. `EventSourceBase` class has one another advantage. It has overloaded constructor which accepts `EventSourceCallbackMonitoringOptions` enum which could be set to either monitor or not monitor success state of event handler's returned `Task` instances. We do not need that configuration in our example as we are ok with not checking `Task` instances. The implementation of very generalization is simple. At first, we clone the message so no other component could modify it, then we use latitude and longitude to get a region, add a new Region field to the message and remove latitude and longitude. The last step is invoking the `SafelyInvokeSendEvent` method.

The next thing is k-anonymity. We have to decide what k to use and implement an `IKanonymitySet` interface for our set. K set to 3 is a reasonable value. Let us continue with the set. Nuntius framework contains `CountAnonymitySet` which tracks number of messages received (and each message has a life span after which its inclusion is removed). `CountAnonymitySet` implements `OfferMessage` method of `IKanonymitySet` as a virtual method and thus can be overridden. Therefore we have a class `CountForRegionSet` which inherits from `CountAnonymitySet`, overrides its `OfferMessage` method to add a region which it is part of. For each region, we have an instance of `CountForRegionSet` which tracks number of received messages in the last 2 seconds which represents 2 minutes in the real world. To demonstrate what we mean, assume following times in milliseconds of messages sent to an instance of CountForRegionSet: 100, 200, 500, 1000, 2000, 2400 and 2600. At 2nd second, the set outputs 4 or 5 messages received (depends on a timer and latency of the last message). At 4th second, it outputs 2 or 3 (the same argument as previous).

With the set ready we are able to implement the whole solution. Method `LinkEverythingTogether` of `Program` class contains the main parts of the solution. We have 5 mobile phones generating messages with their location every 500 milliseconds which represents 30 seconds in the real world. To protect privacy, the id is removed from the message using instance of `TrimMessageFilter` and then only the

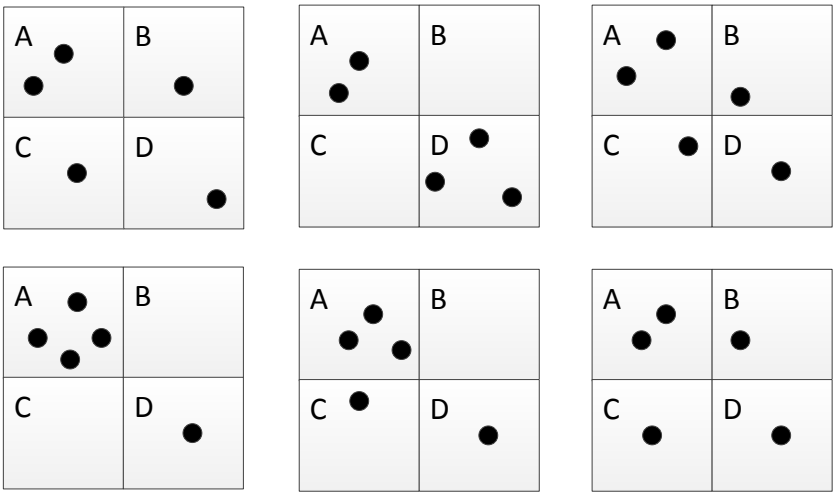


Figure 23: Movement of the phones through regions. Black dots represent phones.

last message from 2 seconds interval (that represents 2 minutes in the real world) is sent using `LastFilter`. To demonstrate k-anonymity Figure 23 represents simulated movements of the phones. 3 phones stay in their region and 2 are moved around. We can see output in console showed in Figure 24 because we used a lambda method to

print the result. As with the first example, messages sent are logged to the files. The structure is .\folder\phone_X.txt where X is order of the phone and folder is Original, Delayed or Generalized (depending in what part of the pipeline is the message logged). The whole result is in .\Anonymized\result.txt.

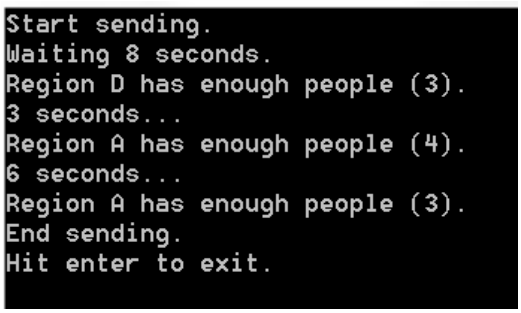


Figure 24: Results in console when running the simulation.

The original data contained personal coordinates and were generated frequently. The new data, however, contain no user information per se. Instead, the resulted data contains regions with a number of users inside and only if the number is high. Therefore we can see the difference between anonymized and non-anonymized data.

By this example, we finished everything we set as our goal for the thesis. We sum up everything and sketch the future work in the following chapter.

9 Related work

In this chapter we compare our work with existing similar solutions and topics. Our solution concerns privacy in the Internet of Things which is a narrow topic. Because of that we compare our work with papers focusing on privacy in IoT or papers focusing on those general privacy patterns which are suitable for comparison.

9.1. Work focusing on IoT privacy

We would like to start with [10]. It introduces a way how to protect against location privacy violation when monitoring vehicles. To protect privacy, each of generated positional data contains different id. Therefore, the server cannot identify the car and the car computes its path (used for example for billing) at the end of month on its own. To validate client data, random spot checks of cars are made (pictures from camera or police cars). Then, the data from the check are used by server to validate that the client does not lie. We inspired by this technique and made one of our methods – computation on a client. [10] does not go too much in detail what happens when the server is misbehaving and assumes it is trusted entity whereas we try to focus to provide data in such form, that even the misbehaving entity is not able to use them.

Another example of work interested in one particular problem is [24]. While we focus on a broader topic, this work focuses on smart energy. This work provides 3 techniques to protect privacy in smart energy. One technique is basically our mediator and the second one is to provide information on sensors and devices collecting data – we are not interested in this one as it does not work with data or communication between components. The last one is data anonymization. Whereas it is mentioned as one method in [24] and does not explain exactly how to achieve it, it is the main topic of our work. On the note of smart energy we would like to mention [25] which concerns with 3 concrete appliances at home and their security issues. However, they solve the problem at the network level which is at the low level of abstraction which we do not want.

Work [26] focuses more on communication and relationship between IoT object and its user. The relationship (e.g. personal items or objects sharing the same goal), is used to define something called bubbles. Bubbles are items with the same set of

security and privacy policies. This is all then used for access control whereas our work defines access control only as one of methods and we focus heavily on exchanged data and how to transform them. Also, the work does not define set of patterns, instead, it focuses on trust and identity management between components.

9.2. Works focusing on privacy patterns

We consider two works suitable for comparison. The first [17] is focusing on online privacy and a tradeoff of gathering information for customized services provided by a web page and privacy of the visitors. It is focusing on a user to give him options to work with gathered data or at least inform him about reasons why the data are collected in the first place. It also brings design of a web site to attention (e.g. prompting user or giving him information). Therefore it is not as technical as our work which, from a big part, focuses on data modification and structure. Some of our solutions for a problem of sending data to the third party focuses on providing anonymized data which is also suggested in [17]. We, on the other hand, build on topic of anonymization much more than [17].

The second work [18] is more similar to ours with the collection of patterns for privacy protection. If focuses on big data and its misuse for privacy breach. Big data is a topic strongly connected with IoT as IoT generates a lot of data from various sources. [18] has a similar structure to ours of introducing methods for anonymization and then patterns. However, the methods are introduced briefly and more emphasis is put on the pattern. Thus, these patterns correspond to methods from chapter 4 and adds a more detailed description. Our patterns are classified based on use cases whereas patterns in [18] use method of protection for classification.

10 Conclusion and future work

We explored privacy in the Internet of Things. We started with introducing IoT and its privacy issues. Then we analyzed the issues, existing work and created a set of methods which can be used to protect privacy. These methods either change the data to a form in which they are safe to send or used communication between components to protect privacy for example by creating a profile or using an intermediary. We then showed that privacy is a tricky topic and it can be violated even when using methods to protect it. We showed 4 patterns to use to protect privacy in IoT and introduced a framework which could be used to prototype IoT network, plug in privacy filters and see what data flow through the network. We provided an implementation of this framework and showed it in practice using two examples.

We see couple of options to continue in this work. We thoroughly described methods used for privacy and we introduced architecture for the framework used to test these methods. Therefore, a more work on the experimenting part of the privacy should be made. We see two main paths:

- Create GUI application in which the networks with filters can be created. Data sources should work out of the box with some industry adopted data generators. Whole execution should offer a functionality to step through the flow or insert conditional actions.
- The framework should be tested in a distributed environment under heavy load to see what the capabilities are. Extension should be made to easily deploy and test the framework on cluster of machines.

11 Bibliography

1. **Bradbury, Danny.** How can privacy survive in the era of the internet of things? *The Guardian.* [Online] 2015. http://www.theguardian.com/technology/2015/apr/07/how-can-privacy-survive-the-internet-of-things.

2. **Tanner, Adam.** Forbes. *World's Top Privacy Experts Worry About Internet Of Things.* [Online] 2014. http://www.forbes.com/sites/adamtanner/2014/10/20/worlds-top-privacy-experts-worry-about-internet-of-things/#2f3aa8c16249.

3. *Protecting Receiver-Location Privacy in Wireless.* **Ying, Jian, et al.** s.l. : IEEE, 2007. 1-4244-1047-9.

4. **Commission, European.** Justice. *European Commision.* [Online] 2014. http://ec.europa.eu/justice/data-protection/files/factsheets/factsheet_data_protection_en.pdf.

5. **Gartner.** [Online] 2015. http://www.gartner.com/newsroom/id/3165317.

6. **Jankowski, Simona, et al.** The Internet of Things: Making sense of the next mega-trend. [Online] Goldman sachs, 2014. http://www.goldmansachs.com/our-thinking/outlook/internet-of-things/iot-report.pdf.

7. **Roermund, Timo van.** Security And Privacy Standards Are Critical To The Success Of Connected Cars. *Techcrunch.* [Online] 2016. http://techcrunch.com/2016/01/28/security-and-privacy-standards-are-critical-to-the-success-of-connected-cars/.

8. **Oracle.** Data Masking best practices. *Oracle.* [Online] 2013. http://www.oracle.com/us/products/database/data-masking-best-practices-161213.pdf.

9. **Byun, Ji-Won, et al.** *Efficient k-Anonymization Using Clustering Techniques.* s.l. : Springer Berlin Heidelberg, 2007. 978-3-540-71703-4.

10. **Popa, Ada, Balakrishnan, Hari and Blumberg, Andrew J.** VPriv: Protecting Privacy in Location-Based Vehicular Services. *Standford.* [Online] http://math.stanford.edu/~blumberg/traffic/vpriv.pdf.

11. *Location Privacy in Pervasive Computing.* **Beresford, Alastair and Stajano, Frank.** s.l. : IEEE Computer Society, 2003.

12. **Trans Atlantic Consumer Dialogue.** Resolution on Privacy and Security Related to Smart meters. *TACD.* [Online] 2011. https://epic.org/privacy/smartgrid/Smart_Meter_TACD_Resolution_FINAL.pdf.

13. **European commission.** IoT Governance, Privacy and Security Issues. *European research cluster on the Internet of Things.* [Online] 2015. http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Governance_Privacy_Security_Final.pdf .

14. *The Mobile Data Challenge: Big Data for Mobile Computing Research.* **Laurila, Luha K., et al.** s.l. : Infoscience, 2012.

15. *ACHIEVING k-ANONYMITY PRIVACY PROTECTION USING GENERALIZATION AND SUPPRESSION.* **Sweeney, Latanya.** 05, s.l. : International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, Vol. 10. 1793-6411.

16. *k-anonymity: a model for protecting privacy.* **Sweeney, Latanya.** 05, s.l. : International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, Vol. 10. 1793-6411.

17. *Privacy patterns for online interactions.* **Romanovsky, Sasha, et al.** s.l. : ACM New York, 2006. 978-1-60558-372-3.

18. *A collection of Privacy Design Patterns.* **Hafiz, Munawar.** s.l. : ACM New York, 2006. 978-1-60558-372-3.

19. **SkyVision Solutions.** Privacy & Security Problem: Smart Meters Keep Track. *SkyVision Solutions.* [Online] 2014. https://skyvisionsolutions.files.wordpress.com/2014/08/utility-smart-meters-invade-privacy-22-aug-2014.pdf.

20. **Niewolny, David.** How the Internet of Things Is Revolutionizing Healthcare. *Freescale.* [Online] 2013. https://cache.freescale.com/files/corporate/doc/white_paper/IOTREVHEALCARWP.pdf.

21. **Amazon.** 10-Q for Quarter Ended June 30, 2015. *Amazon Quarterly Results.* [Online] Amazon, 2015. http://phx.corporate-

ir.net/External.File?item=UGFyZW50SUQ9NTg4MTIyfENoaWxkSUQ9Mjk2MjczfFR5cGU9MQ.

22. **Microsoft.** Earnings Release FY15 Q4. *MicrosoftInvestor Relations.* [Online] Microsoft, 2015. https://www.microsoft.com/investor/EarningsAndFinancials/Earnings/PressReleaseAndWebcast/FY15/Q4/default.aspx.

23. —. .NET Core A general purpose managed framework. *.NET Core.* [Online] http://dotnet.github.io/.

24. *Common privacy patterns in video surveillance and smart energy.* **Bier, Christoph and Krempel, Erik.** s.l. : IEEE, 2012. 978-1-4673-0894-6.

25. **Notra, Sukhvir, et al.** An experimental study of security and privacy risks with emerging household appliances. *IEEE Xplore.* [Online] 2014. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6997469&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6997469.

26. **Bernabe, Jorge Bernal, et al.** *Privacy-Preserving Security Framework for a Social-aware Internet of Things.* s.l. : Springer International Publishing, 2014. 978-3-319-13101-6.

27. **Microsoft.** https://azure.microsoft.com/en-us/documentation/services/iot-hub/. *Microsoft Azure.* [Online]

12 List of abbreviations

IoT – the Internet of Things

RFID – Radio-frequency identification

Attachments

CD is attached to the thesis. It has following structure:

- */doc* – Contains documentation and PDF version of this document
- */scr* – Contains source files of the framework
    - */Nuntius* – contains the Nuntius framework
    - */Nuntius-example* – contains the example of usage of Nuntius framework
- */readme.txt* – Contains instruction for the CD folder structure