TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

LAURI LIMNELL

ENGINEERING PRIVACY AND TRUST IN SOCIAL DEVICE ENVIRONMENTS

Master of Science Thesis

Examiner: Timo Aaltonen, DSc.
Subject approved by Faculty of Computing and Electrical Engineering in Council meeting on 6.4.2016

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Diplomityö, 42 sivua
Toukokuu 2017
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
Pääaine: Verkkoprotokollat
Tarkastaja: TkT Timo Aaltonen

Avainsanat: Social Devices, Privacy, Trust, Man-In-The-Middle, OrchestratorJS

Älykkäiden laitteiden päämääränä on tehdä elämästämme sekä tehokkaampaa että tuottoisampaa. Samaan aikaan ne tuovat mukanaan mahdollisuuksia ja vaaroja. Kuitenkin yksityisyyden, mitä tietoja varjelemme ja haluamme piilottaa, ja luottamuksen, kuinka suhtaudumme muihin käyttäjiin tai itse sovelluksiin, jolloin näiden välille syntyy riippuvuussuhteita, jonka mukaan valitsemme kumpaa vaalimme enemmän. Tällä hetkellä kaikki lähestymistavat luottamuksen arviointiin pyörivät tunnettujen arvojen olemassaolosta, esimerkiksi olettamukseen, että kaikki sosiaaliseen kanssakäymiseen liittyvät applikaatiot omaavat yhtenäisen luottamustason (Facebook, Twitter, etc.). Tämä väittämä luo illuusion ja efektin, että alustoja käyttävät käyttäjät eivät pysty erottamaan muita luotettavia käyttäjiä, vaan he tekevät kaikki päätöksensä rajoitetulla tiedolla laajasta käyttäjäkunnasta, esimerkiksi muiden käyttäjien tai itse sovelluksen suosituksesta.

Tämän diplomityön tarkoituksena on arvioida Social Devices- konseptin heijastamia luottamus- ja yksityisyysongelmia. Laitteiden kerätessä läheisyystietoja muilta laitteilta, alustan vaatimukset yksityisyydelle riippuvat suositellusta yksityisyyden tasosta. Itse asiakasohjelmalla on perustoimintona automaattisesti ja asynkronisesti hyväksyä pyynnöt ja toivomuksena myös muodostaa muiden laitteiden välille yhteyksiä.

ABSTRACT

Tampere University of Technology
Master of Science Thesis, 42 pages
May 2017
Master's Degree Programme in Signal Processing and Communications Engineering
Major: Network Protocols
Examiner: Timo Aaltonen, DSc.

Keywords: Social Devices, Privacy, Trust, Man-In-The-Middle, OrchestratorJS

The purpose of smart devices is to make our lives both more efficient and effective. At the same time, this raises real opportunities and, more importantly, dangers. However, the balance between privacy and trust changes depending on the use of a selected device. For the time being, all the approaches for assessing trust revolve around the existence of known values. Typically, these are either users / peers in a trusted network providing measures for asserting that particular trust value or the assumption that all social-based applications share the same levels of trust. This triggers the illusion and the effect that users using these platforms have difficulties differentiating between trusted users. Instead, they make their own decisions based solely on their limited knowledge of the potentially wide user base or relying on recommendations set either by other users or possibly by the application itself.

The purpose of this thesis is to evaluate the projections of trust and privacy issues that have arisen with new implementations of the Social Devices concept. With devices gathering proximity information about other nearby devices, the platform's requirements for privacy vary, depending on the level of recommended protection. The client itself has a basic functionality of autonomously and asynchronously accepting requests and desirably forming connections to other devices.

.

PREFACE

This thesis has been made possible by the Tampere University of Technology and its department of Pervasive Computing. At the outset, this thesis was originally funded by Tekes as part of the Cloud program, which has then led to this particular research direction.

I would like to express my gratitude towards my supervisor, Timo Aaltonen, DSc., and. Niko Mäkitalo, MSc., who have guided me through the writing process and given me excellent advice throughout.

Most of all, I would like to thank my girlfriend Hanna, mother Erja, father Pekka, and brother Ville for their continuous and undeniably ceaseless support towards my studies throughout all these years.

Tampere, 14.10.2016

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

App	Application
AcOP	Action-Oriented Programming
API	Application Programming Interface
BS	Buddy Searcher
D2D	Device-To-Device
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IE	Isolated Execution
JSON	JavaScript Object Notation
MD	Medical Dispenser
MITM	Man-In-The-Middle
MnG	Meet and Greet
MS	Media Share
OJS	OrchestratorJS
REST	Representational State Transfer
SD	Social Devices
TP	Trusted Pathing

1. INTRODUCTION

Over the past couple of decades, smart device technologies have evolved into a more consumer-friendly environment, where consumers want immediate access into a social-network type coaction and are willing to share vast amounts of media in various categories. Naturally, this type of digital sharing extends beyond simple consumer media sharing rather quickly as options and possibilities grow [1].

As several devices are turning into smart devices, the days when only mobile phones were involved are history. Everyday household appliances such as televisions and refrigerators are seeing the new age of smart devices along with advanced automotive technology, not forgetting individual items such as smartwatches. Generally speaking, anything that is equipped with sufficient and capable hardware and feasible networking capabilities can be considered a smart device.

With smarter devices we, both as individuals and groups, have accepted web services as a crucial part of our everyday lives. We are becoming more or less addicted to social interaction as we strive to share experiences and feelings while simultaneously staying in touch with people by using sites such as Facebook and Twitter.

The Social Devices concept was set up to research multiple areas of social interaction. These were based on discoveries with a lot of recent media coverage about unintentional leaks of organizational and personal data. Certain realizations were made early on. Users expect privacy solutions. Unfortunately, privacy, as an everyday concept, is not very clearly defined.

Additionally to the research made for this thesis, another project was ongoing during the period of writing this thesis. The D2D project, better known as Device-to-Device Communications [21], was established to strive for an answer to the question of whether people could establish relevant direct connections between their devices without the explicit need for Internet connectivity. This led to the solutions described in this thesis on how successful connections could be first triggered and then established. The concept which the D2D project came up with was similar to Media Share, which is also described in this thesis.

The above-mentioned project was designed to fulfill the necessary requirements to act as a legitimate Social Device experience with capable, yet as modest devices as possible, with certain limitations; in this scenario, a specific Android version. The experiment itself was first demonstrated at Tampere University of Technology and then it progressed to other universities for stress testing.

The interest of this experiment in terms of Pervasive Computing was to measure the capabilities and functionalities of the Device-to-Device app design, which provided ideal platforming ideas for future development, enabling the design and execution of more complex applications such as the Medical Dispenser, which will most probably be implemented at some time in the future.

Furthermore, as smartphones and mobile devices have replaced traditional personal computers as the main means of accessing the web and other context-based services, a plethora of applications are already available for these devices and more become available on a daily basis. Social Devices tackle the problematics of location privacy issues, which are critical especially in such pervasive communication systems and mobile communication networks. Fortunately, both of these have recently received a lot of attention, leading to beneficial research and possible solutions. It became clear at the beginning of this research and thesis that the best way to evaluate such problems is to use real-life settings as parameters.

Figure 1.1: A use case example of a situation enriched by Social Devices interaction [2].

Social Devices as a concept was established to study and define the social aspect of new powerful smart devices that use services similar to Facebook. Social Devices proposes an alternative approach where both people and devices would form themselves into a new system, where the devices proactively participate in social situations by enriching them in various ways. [2]. For example, the devices could participate in conversations or actually provide applications that promote socializing in different situations, as seen in Figure 1.1, which demonstrates such an application; in this type of scenario a group of individuals have come together. The system recognizes the situation and suggests the users take part in an event both suggested and made possible by their devices.

In every app concept described in this thesis, a more or less obvious factor was hidden, namely, familiarity. We live and interact across a diverse set of physical spaces. The variety of how we formulate our personal meanings towards observable cues such as public-private, which was a key element in this research, is never-ending. We see people around us every day and we share this space with those same people, which dominate our perception of how we see these individuals. Whether they are friends, family, colleagues, or total strangers depends purely on how we observe them.

This first purpose of this thesis is to evaluate the projections of trust and privacy issues that have arisen with the new implementations of the Social Devices concept. With devices gathering proximity information about other nearby devices, the platform's requirements for privacy vary depending on the level of recommended protection. The client itself has the basic functionality of autonomously and asynchronously accepting requests and desirably forming connections to other devices.

The second aim of this thesis is to present parts of the related concepts in Social Devices and how they operate and interact.

The thesis is structured as follows. Section 2 is dedicated to introducing the Social Devices (SD) concept, which was originally the core idea for a variety of implementation possibilities which will be introduced below, such as OrchestratorJS (OJS) and the Action-Oriented Programming model (AcOP). This section also clarifies the implementation architecture by explicating the functions of related components.

Section 3 is designed to give an understanding of the concept of data minimization and how it affects Social Devices. Additionally, more specifically, it explains why gathering data is relevant in the first place and how proper data gathering should be initiated.

For Section 4, a privacy / trust indicator will be introduced to enlighten how privacy and trust are seen in Social Device environments and furthermore to demerge the key factors of privacy and trust.

In Section 5, the infamous Man-In-The-Middle (MITM) attack is covered in the light of several example Social Device concepts and implementations that all have the same basic functions of forming connections; but the application itself varies in a way that the required level of privacy or trust is different. Moreover, this section illustrates how privacy takes place in Social Device innovations.

Section 6 draws conclusions and summarizes the thesis.

2. SOCIAL DEVICES

This section is dedicated to introducing the concept of Social Devices with a specific prototype implementation: OrchestratorJS. Subsection 2.1 includes a general overview and requirements of Social Devices. Subsection 2.2 defines the underlying programming model of the implementations and in subsection 2.3 the needs and characteristics of the implementations are discussed. OrchestratorJS implementations and actual architecture are described in detail in subsection 2.4.

2.1 SD concept

The concept of Social Devices was introduced by Mäkitalo et al. in [2]. The actual aim of Social Devices is using technology to enrich interactions either between devices or between devices and the user. Along with the concept, devices that are proactive and potentially context-aware can form a completely new socio-digital system, where it is possible for both users and devices to start interactions with either other devices or people. Social Devices can enrich a simple device-to-device interaction by making it explicit to users, for example in face-to-face situations.

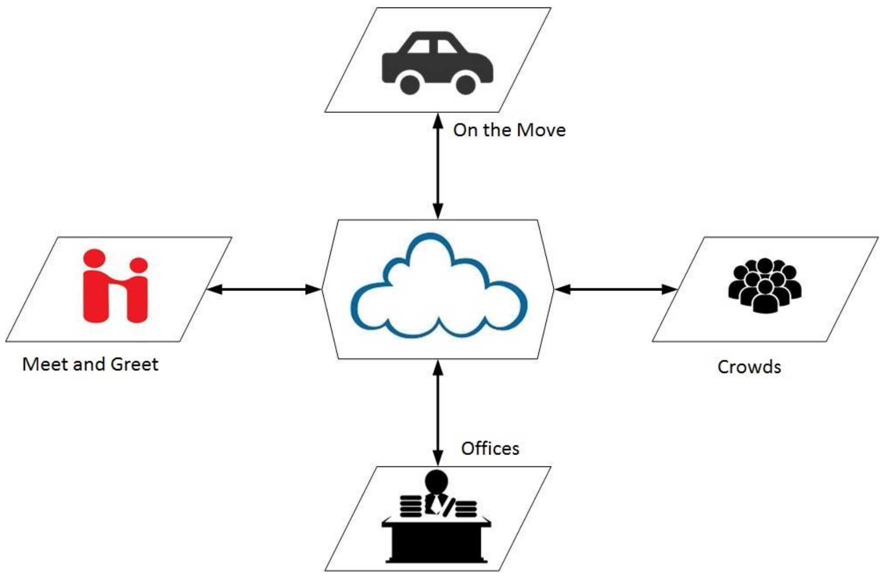


Figure 2.1: A demonstration of environments in which Social Devices can be used [2].

Any interaction caused by Social Devices can be considered an *action* (explained in subsection 2.2). It is essential to understand that these actions may not functionally operate the same way but they serve the same purpose: providing information to users in various socially interactive situations such as in large crowds, meet-and-greets, on the move, or in office environments, as illustrated in Figure 2.1. As mentioned above, in situations where people meet face to face, that particular interaction could for example be an acknowledgement of a greet by audio or exchanging business information in a conference. These interactions can be made even more extensive by adding actual sound to connecting devices when certain proximity factors are met, such as mobile phones and laptops. A new design could even be used in medical care, reminding individuals to take their daily dosage of medicine, which will be covered in a later section.

Nowadays, most people own a mobile phone and carry it everywhere; they have become the perfect social devices. If the preconditions are met, other devices could be considered social as long as they have sufficient computational capacities and capabilities and the ability to interact with the surroundings and surrounding devices is ensured. For Social Devices, this means that it can be very heterogeneous and include very personal devices or the devices can be practically stationary and impersonal, located in public places like in a meeting room or a public library. This heterogeneity is shown best by the distinct resources of the devices.

Furthermore, the surroundings where Social Devices operate can be adjusted quickly as users carry out their daily routines. The arbitrary nature of social encounters could lead to situations where devices quickly and repeatedly move in and out of range of each other, causing the actions of participating devices to change constantly.

2.2 Requirements and Characteristics

Standardized requirements and characteristics are essential as the requirements for implementing applications in a Social Devices environment are derived from both of these. Devices have their own identities as users have preferences regarding their own devices, thus the identity can show which device is allowed to do what. Furthermore, identities can be used for indicating the actual proximity of the physical user. It works as follows: the more personal the device is to the user, the better it is for calculating the proximity approximation.

For instance, pinpointing a laptop connection can be much harder to calculate than a mobile device connection based more on proximity, such as Bluetooth, and here a specific registry is introduced in order to keep track of devices. However, this raises a serious privacy issue, which will be discussed later in Section 5.

As a precondition for Social Devices, certain proximity of users and their devices must be met before any social interaction can commence. Each device calculates its own proximity with respect to each device willing to participate in a desired action. The proximity information is forwarded to a centralized server that maintains the proximity information of each device. From this information, a mathematical graph is formed where nodes denote devices and edges denote the mutual distance between devices, reducing the required computing on the device end.

Additionally to registering and reporting the proximity of other devices, the devices have to provide the server with necessary information regarding action preconditions, which define what is expected from the participants of the action that is occurring. This form is also known as ‘action descriptions’ or ‘action body’ and is covered in more depth in subsection 2.3. Enabling and installing multiple interfaces and capabilities can be done by the owner of a social device, making it easy to choose and moderate interfaces, and select the privileges that the application can access.

Moreover, the system needs a means to determine which devices are able and willing to participate in an action. This can be solved by including information on the logical expression of the devices in the precondition part of the action description. As the environment is usually mostly dynamic, more information is needed from the device, which is handled by monitoring the device states. States may vary from a Boolean value to any abstract value, which indicates for example if a device is on silent mode or not.

Executing actions requires additional coordination and orchestration for the devices, which is solved by defining a platform with a built-in coordination method. As the devices do not handle the configurations and decisions themselves, even the most modest devices can act as social devices.

2.3 OrchestratorJS

The concept, characteristics, and requirements of Social Devices were introduced above. This section aims to elucidate the requirements by showing the actual architecture and implementation of OrchestratorJS, with reference to [9] and [14] unless stated otherwise.

Orchestrator.js coordination middleware is especially designed to support the Social Devices concept by offering tools for implementing interactions between devices and people and for observing situations when these interactions should be proactively triggered. Although Social Device applications differ from traditional mobile apps, the Orchestrator.js can still be used for the quick development of heterogeneous multi–device applications, as the lightweight server can easily be hosted individually for each purpose [9].

Moreover, the developer need not pay too much attention to the execution model of Social Device apps, but can instead focus on a single application, and host it on any regular computer, such as a living room media center for example. The main task of Orchestrator.js is to maintain a persistent connection to devices, store applications, store the contextual information of devices, and offer a means for triggering the applications. Moreover, Orchestrator.js (OJS) offers tools for developers to implement, test, and deploy their applications [9].

The basic operations of OJS, illustrated in Figure 2.2, are as follows: as devices connect to OJS, they measure the context with their sensors and report this to the selected Device Registry, which will be explained below. When the values of data change, meaning that new values replace the old values, the Device Registry publishes the changes through a pub / sub API.

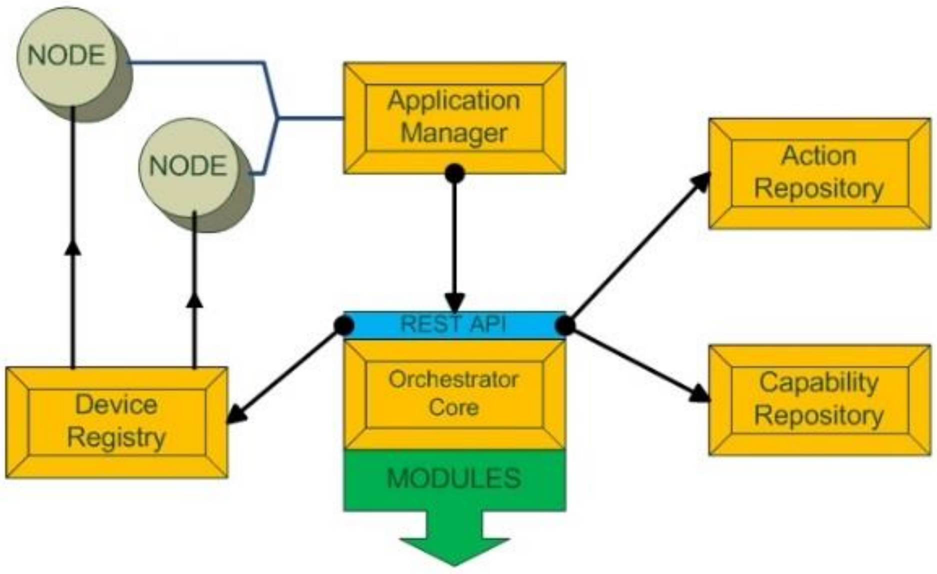


Figure 2.2: A diagram of OJS operations [9].

When the applications run in a cloud they receive events through this pub / sub API and simultaneously monitor other sources such as polling with HTTP. When OJS receives a trigger, it fetches resources from repositories. After all these steps have been carried out, OJS begins the coordination process by commanding participants with their capabilities. This is illustrated in Figure 2.2.

2.3.1 Architecture

“Orchestrator.js (OJS) offers a web-based IDE for implementing apps in a heterogeneous, multi-user, and multi-device environment.” [9]. Not only can it be used for the implementations themselves, but also for testing. With an IDE, it is relatively easy to monitor device states, read debug messages from a console, and trigger interactions for a predefined set of devices.

As OJS is based on Node.js, which is a server–side JavaScript platform, the communication itself operates through Socket.IO, a non-standardized protocol for relaying events between device and server. With this protocol stack, it becomes highly efficient to execute input / output operations between OJS and a selection of devices.

Additionally, with Orchestrator.js it is easy to implement context-aware applications for a heterogeneous set of devices. Through an API, multiple devices constantly report their state to the Orchestrator.js Device Registry, which publishes updates whenever the contextual data of the devices change.

2.3.2 AcOP Execution Model

The applications in Orchestrator.js are based on the Action-Oriented Programming Model (AcOP). This programming model includes the following three main concepts (an illustration of the concept is presented in Figure 2.3):

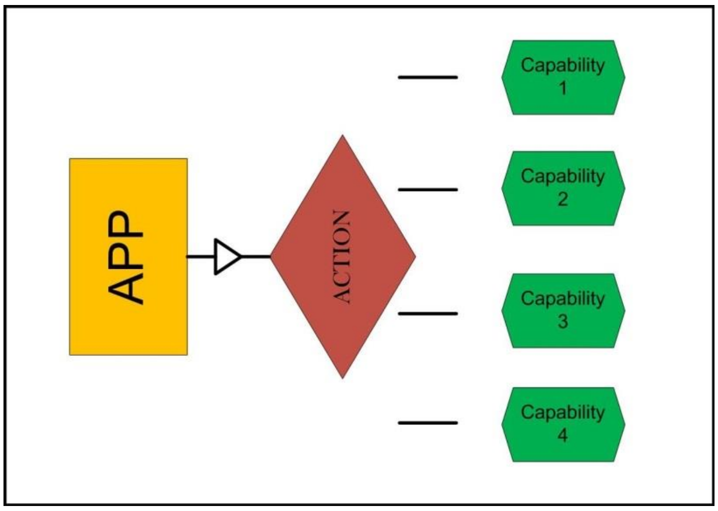


Figure 2.3: Illustration of AcOP model concepts [9].

Apps - monitor device contexts, and outside sources like social media services, such as Facebook, Twitter, and so on. Based on the observations and logic defined by the developer, the app triggers actions. Thus the app defines when and what interactions (actions) should take place.

Social Device apps are extremely modular, and thus the execution model of Social Device apps differs from traditional applications. Social Device apps mainly run inside a cloud where observer components monitor contextual data and, based on developer-defined logic, schedule interactions between users and devices. Hence, the apps are the top-level components that select when and what interactions will be scheduled and for whom.

Social Device functionality, however, is not entirely tied to apps, and interactions can be triggered by other components, like other interactions for instance. Typically, the apps are personal, meaning that users can start an instance for themselves and grant permissions to monitor their personal data.

However, developers can also implement apps that monitor data from multiple users, or data from external sources. Naturally, the users have to enable and grant permission to use their personal data with these non-user-specific apps too.

Actions - define how the devices interact with each other as well as with users. An action can be very short, like a simple greeting, or longlasting, like a game. The operation principle is based on modularity, encapsulating the interacting device-to-device behavior, and defining roles, parameters, and preconditions. Actions define the joint behavior of multiple devices, and interactions between people.

Roles determine each device’s part in an action; for instance, in a media-sharing action, the users share pieces of data with one another and the roles define which device shares what and which devices have access to that piece of data. In Social Device environments, a precondition works as part of an action, defining what is expected from each device participating in the action.

Capabilities - are used to describe how a device appears in the system, what a device can or is allowed to do, and in which interactions the device can take part. The capability can either be very generic, like a talking capability that allows the device to speak, or more specific, like the Attend capability that allows the device to participate in a co-op game, for instance.

For implementation purposes, the capabilities manifest as interfaces consisting of operations, making it a contract of operations relating to a specific device and its capabilities. As the AcOP model does not define one true interface programming language, the communication protocols between server and client are likely to differ in the programming languages they use.

The number of interfaces or the capabilities is not defined by the model, as a device can only be invoked through its interfaces, rendering useless devices that have no capability to participate in an action. With reusable capabilities, any developer can write new actions without implementing a device-specific code.

2.3.3 Invoking Web Applications through REST and NodeJS

This subsection is dedicated to creating an in-depth view and understanding of both REST (Representational State Transfer), which is an architectural style used in software engineering giving a coordinated set of constraints in designing components, and Node JS, an asynchronous event-driven framework; and, finally, more specifically how these two components are invaluable in Social Device development.

The concept of REST was introduced by Fielding in [16]. The component or rather the tool called REST will be described below since it is essential for Social Device development in order to enable swift development and interactive modules for mobile socially primed products.

REST - When RESTful services are built, certain HTTP methods are introduced to construct a more uniform interface to meet the constraints and provide counterparts for noun-based resource actions. The primary and most commonly used methods are GET, POST, PUT, and DELETE, which correspond to read, create, update, and delete operations. An overall perspective of the methods as well as a closer appraisal of how REST is connected to Social Devices will be covered in the next section.

GET - This particular method is one used for reading or retrieving a physical representation of a chosen resource. With paths such as GET http://www.socialdevices.com/users/Limnell - the method returns a representation in either XML or JSON format with an HTTP response code, informing whether the execution was successful or not. Furthermore, as a GET request is only used for reading data and not actually changing it, these requests are considered to be safe to call without the risk of data modification or corruption. With GET, whether it is called once or ten times, the effect stays the same, as it is idempotent, meaning multiple identical requests cause the same results.

POST - The POST verb or method is mostly utilized for creating new accessible resources. In particular, POST is used for generating subordinate resources, meaning they are subordinate to some other parent resource. When creating new resources, the associations of the new resource with the particular parent object are taken care of by the service by assigning an ID, e.g., a new resource URI, to the parent.

PUT - the PUT method is most often utilized for its updating capabilities. By using PUT in a chosen original resource, the representation is then updated into a new form of representation, differing from the original URI request body. Additionally, PUT also enables the creation of resources in situations where the resource ID is chosen by the client rather than by the server.

DELETE - DELETE is the last of the four most commonly used HTTP methods. As its name denotes, the DELETE method deletes a resource identified by a URI. When deleting resources, it is recommended to add a visual display of an HTTP status acknowledgement confirming that a specific resource has been removed with, for example, a wrapped response.

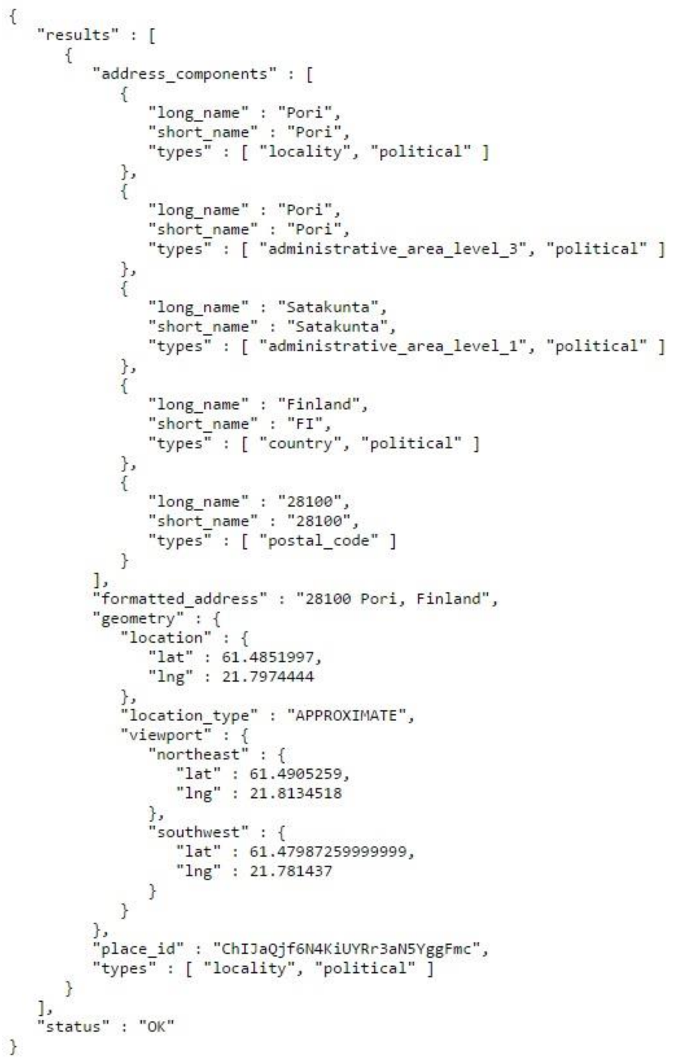
In terms of HTTP, DELETE operations are considered idempotent, meaning that if a resource is deleted, then it is permanently removed. Alternating = swapping / Altering = changing) the number of DELETE executions on a resource has no visible effects; the resource is removed after one single run.

In addition, if the operation is meant to decrement a counter within a resource, the operation no longer remains idempotent. Arguably if an operation returns a not found status, the end-state of the resource is still the same so whether the DELETE is truly idempotent or non-idempotent is a matter of preference.

Finally, by understanding the principles of HTTP methods and how they correlate with RESTful environments, new architectures with a stateless nature can be introduced and used to offer more versatile web services with fewer restrictions and easy utilization. By simply modifying the executable code, servers can temporarily extend or customize the functionalities of a client.

As clients ordinarily cannot tell whether they are connected to the end servers directly or to an intermediary, load balancing can be implemented to share caches, thus enabling the true nature of REST, which is discussed in the next paragraph.

As mentioned earlier, in order to increase overall performance and scalability, REST was introduced with the intention and vision of being implemented and used in distributed hypermedia systems, leading to higher performing and more maintainable architecture. The principles of REST and how it operates usually, if not always, resemble the operative functions of a website, as the features act similarly to making a call from a client to a server, which then sends data back over the HTTP protocol. The requests look similar, but the responses are completely different when an API is accessed directly over the web, as Figure 2.4 indicates.



```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "Pori",
               "short_name" : "Pori",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Pori",
               "short_name" : "Pori",
               "types" : [ "administrative_area_level_3", "political" ]
            },
            {
               "long_name" : "Satakunta",
               "short_name" : "Satakunta",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "Finland",
               "short_name" : "FI",
               "types" : [ "country", "political" ]
            },
            {
               "long_name" : "28100",
               "short_name" : "28100",
               "types" : [ "postal_code" ]
            }
         ],
         "formatted_address" : "28100 Pori, Finland",
         "geometry" : {
            "location" : {
               "lat" : 61.4851997,
               "lng" : 21.7974444
            },
            "location_type" : "APPROXIMATE",
            "viewport" : {
               "northeast" : {
                  "lat" : 61.4905259,
                  "lng" : 21.8134518
               },
               "southwest" : {
                  "lat" : 61.47987259999999,
                  "lng" : 21.781437
               }
            }
         },
         "place_id" : "ChIJaQjf6N4KiUYRr3aN5YggFmc",
         "types" : [ "locality", "political" ]
      }
   ],
   "status" : "OK"
}
```

Figure 2.4: A live capture of a direct API request to Google Maps Geo.

The results illustrated in Figure 2.4 appear to be exceedingly complex and at first sight they seem to have no informative ground. However, the results are represented as a JavaScript object notation or JSON formatted data. The structured data are displayed by organizing the queries of key-value pairs or parameters, such as */maps/api/geocode/json*, which were used in Figure 2.4, and restricting the results to *address=Pori&sensor=false*. This type of JSON can also be implemented in such applications as Buddy Searcher, which will be introduced in Section 5.

From the Social Device point of view, all the exposed web services available for access or, in other words, online APIs, are executable but in different web service points. From a more in-depth perspective, Social Devices publish web services that allow other developers to call them with their own code, thus consuming the available services. The part played by NodeJS in the Social Devices concept will be introduced next in order to link together how these concepts work in web application development.

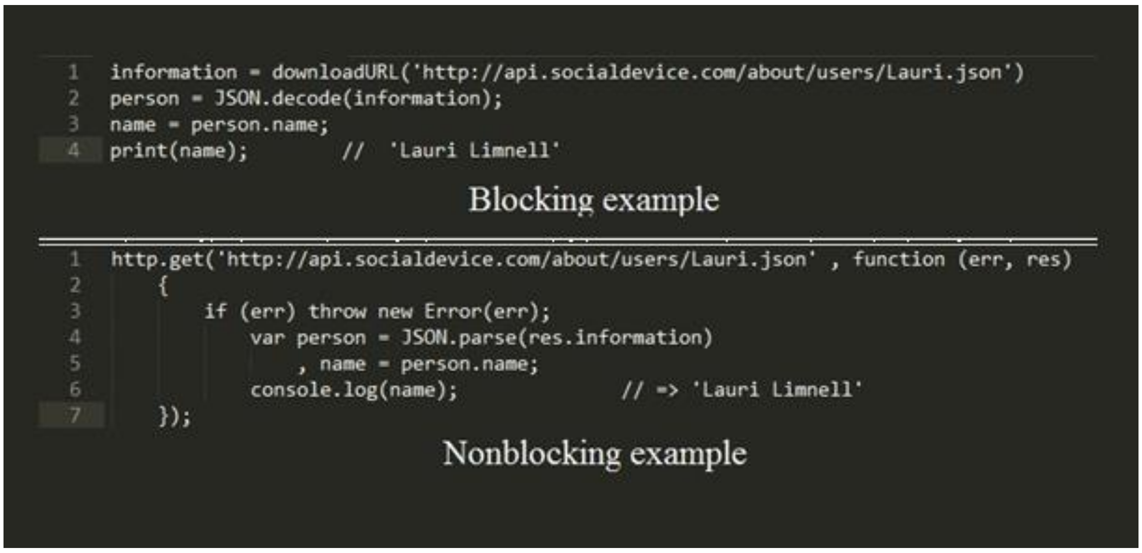
2.3.4 NodeJS as Operative in Social Devices

Node was mentioned in subsection 2.3.1, where the overall architectures were introduced. In more detail, the core principle of Node is to enable JavaScript, which used to be confined to browsers, and successfully allow it to run on a device. As JavaScript runs in the browser, it means that it can only access a webpage. However, if JavaScript were to be introduced into an environment where it could be run, a whole new world would open up. With Node it is now possible to access the files on devices, which it would normally be impossible to do with JavaScript [4]. It is also possible to listen to network traffic or the specific HTTP requests received by any device used and if chosen, to send a file back. The ultimate change is that with Node, databases can be accessed directly.

Node.JS is used for two main purposes: the first aim is to take advantage of certain utilities which can be developed later on a device, mainly for building JavaScript files, making it possible to listen to file system changes, and do live reloads. The second aim for Node is mainly to build web servers, or in other words, a web application with Node.

Figure 2.5 shows the basic principle difference between blocking and non-blocking connections. The blocking pseudo code for dynamically displaying a chosen name after which a person’s data would be downloaded, decodes the data and finally displays the name. While the download function reaches to the web to run it, the actual program is completely blocked. Although this has always been the most common way to download data from a server, NodeJs offers an alternative, a non-blocking solution.

As mentioned in the description, Node uses an event-driven model, making it extremely light and efficient. This particular model makes Node perfect for real-time applications running across distributed devices, especially if the environment is data-intensive [4]. Furthermore, as Node is considered event-based and non-blocking, users do not have to wait for results before moving forward. This is covered in more detail in Figure 2.5 and by the following example.



```
information = downloadURL('http://api.socialdevice.com/about/users/Lauri.json')
person = JSON.decode(information);
name = person.name;
print(name);        //  'Lauri Limnell'
```

Blocking example

```
http.get('http://api.socialdevice.com/about/users/Lauri.json' , function (err, res)
    {
        if (err) throw new Error(err);
            var person = JSON.parse(res.information)
                , name = person.name;
            console.log(name);              // => 'Lauri Limnell'
    });
```

Nonblocking example

Figure 2.5: Demonstration of code executions for both blocking and nonblocking variations in NodeJS.

Although the Node code appears confusing when applied to a server, it allows the use of event loops, creating an “always on” concurrency model, which in simpler terms means that JavaScript waits for an event to happen, and whenever that particular event occurs, a call-back function is executed. This can be translated into an everyday example such as a mailman delivering mail:

Every single piece of mail represents an event of its own in the event loop, which in this scenario, is represented by the mailman. Each of these events is to be delivered in a certain order. For each of the letters, the mailman walks a certain distance to deliver that letter using a particular route, which now represents the call-back function assigned to the event. Furthermore, the mailman can only walk one single code path at once.

However, there is a possibility that while the mailman is walking the path, he can be given a new letter. This is because the call-back function that the mailman is coded to walk has emitted another letter. Should this happen, the mailman will deliver the newest letter immediately due to the fact that the letter was handed to him directly instead of at the post office, making the new letter a priority. By diverging from the current code path, the mailman will deliver the prioritized letter following the newly formulated path, after which the original route that emitted the event can then be continued.

For Social Devices, Node creates virtually endless possibilities, such as writing server-side JavaScript to handle database connections or as in the upcoming use cases, to connect to web APIs. Each Social Device application or program can be considered as a node of its own in a larger network and can be called by other nodes in other parts of the network or, with the correct network bridging, even from an outside network if so chosen. This allows for easy reuse and rapid evolution of code, making Node one of the most powerful languages in Social Device development.

3. SAFE DATA PRACTICES AND LIMITATIONS IN SOCIAL DEVICES

After decades of mobile application development a vast selection particularly of social applications that allow some kind of sharing of information have appeared. The main issue with these ingenious social applications is that it is dauntingly easy and routine to locate people using them [13].

Unfortunately, in the case of smartphones, we are willingly carrying a location-aware wiretap in our pockets every day. Equipping a wiretap with a camera itself is bad enough, but our phones are capable of documenting so much more: our movement patterns, basic comings and goings, and who we come across either just passing by or engaging in conversation [15].

Our mobile devices could be used as a tool of destruction but fortunately all the pressure can be off-loaded onto application developers to solve this problem, regarding our potential sensing social devices. Luckily for us, smartphones offer a vast variety of benefits too, such as being the number one communication tool in the world, supporting social networking, stimulating productivity and innovation, which means that as our smartphones evolve, technology evolve with them.

Concerns about privacy and whether it has been breached is a topical question. There is no one true solution for preserving our privacy due to the simple fact that our operating patterns and behaviors are always different. The burgeoning online media expose a considerable amount of information about the owners themselves, the subjects, and the content of the viral media itself. It is interesting to see and study how users make their privacy decisions when they are dealing with online content, and whether there are certain patterns when using these social media sharing environments. One more important question to be asked is whether people even care about location disclosure, or exposing themselves, because there are no guarantees that one’s location would be safe in the first place.

The understanding of what the ownership of data actually stands for varies considerably. The common opinions are that it either refers to the plain possession of data, or manifesting true responsibility while producing and maintaining data. In Social Device environments, the true identity of ownership is both of these. The core identities of Social Devices do not mean merely having the ability to access data. Instead, the core revolves around creating, modifying, storing, and deriving benefits. Furthermore, the core ultimately bestows privileges on individuals using these applications or participating in Social Device environments. This section is dedicated to analyzing the nature of data possession and how it coexists with Social Devices.

Evidently, every piece of gathered data has essential value as a by-product of information processing. Social Devices form a multi-platform component environment, no matter what momentary use case may occur; the user base expects privacy in all types of interaction. The users start creating bonds and strong relationships, which later evolve into relationships of trust. While receiving access to personal data within gray boundaries depends on both the concept and the desired outcomes, the actual acceptance required by both parties involved in the interaction is commonly taken for granted.

Furthermore, connecting both user and device in a new device-to-device-to-users environment automatically generates extensive amounts of data, as illustrated in Figure 3.1. This generated data can be gathered and analyzed later on. Maintaining privacy in such environments creates enormous benefits as well as unpredictable challenges. For example, referring to the Medical Dispenser concept introduced later in Section 4, allowing users with serious health conditions to work with their personal doctors to create an environment where all manner of objects have their own digital presence and the ability to communicate with other objects and / or people.

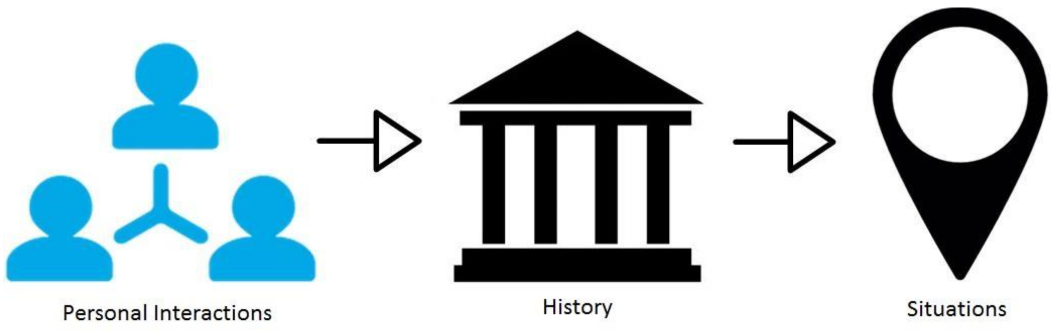


Figure 3.1: An illustration of situational data gathered from daily routines [17].

Additionally, as Social Devices thrive from creating situations where data are exchanged, the more data given to the systems that obtain data while interacting with users, arguably the more the line of privacy will fade.

In general, preserving privacy is valid in every situation where data can be mined or transmitted. As the trend of sharing increases day by day, so does the concern that only a few pieces of data, which can be mined from either one source or multiple sources, may create a digital fingerprint that is highly traceable. Nowadays, users are creating limitless amounts of personal data with all their connected devices. These devices may either be in the possession of end users with mainly personal products or the devices may also be multiple inter-connected devices in one big environment [7].

Along with the facts mentioned above and referring to Figure 3.1, the potential risks of interacting enable unauthorized access and misuse of personal information, facilitate attacks on other systems, and finally create general safety risks. These occurrences can be considered more of a security issue but this interaction acts as a transition towards the general concerns of privacy and covering data [19].

In Social Device environments there are two types of risks. Firstly, the risks can be considered direct when there is a leak of a collection of personal data or, depending on the purpose of use, even of confidential financial information. Secondly, the risks can be regarded as delicate or sensitive, such as account information; the general volume of data created by social devices can vary from small fractions to large scale sensitive data.

With sensing devices sharing information, leaking collections of data such as personal information, habits, locations, and moreover physical conditions may, without proper privacy allocations, allow an outside entity that has not directly collected sensitive information, to infer it.

Multiple different types of scenarios can be affected by situational data, which is demonstrated in Figure 3.1. Whether it is our daily personal interactions with family, friends, or co-workers, these situations all create data. As we propagate our routines, more situational data are generated. Imagine a scenario no more complex than grocery shopping or a quick visit to the ATM. Both of these quick interactions leave a stamp of when and where these occurrences took place. Pieces of user history are generated and can be mined to form patterns.

In addition, situational data are formulated on the basis of situations. In this particular case, situations refer to locations, time of day, or situations altered by a variable. For example, at a certain time of day, an individual would walk to a coffee shop near his / her office and purchase a beverage. The situational data would now compose of a time stamp, should it be morning coffee, the location in which the coffee was purchased, and a variable could be concerned such as a friend appearing and altering the choice of coffee shop (altering all of the above-mentioned scenarios) or simply the weather, as for example, on a rainy day, the daily stamp would not occur.

Moreover, in environments where Social Devices exist, the massive amounts of granular data will eventually allow those who have access to the data to perform analyses, which are impossible to do at the beginning state while the data sets are still quite poor. In general, while providing beneficial services to users is mandatory to satisfy them, the possibility of misuse where the enabling of sensitive collections such as behavior patterns arises as unauthorized use or use by unauthorized individuals.

A valid approach for preserving the integrity of Social Devices lies in data minimization. The concept of data minimization implies that companies limit the amount of data gathered. Data will inevitably be gathered and companies should dispose of it after it has fulfilled its purpose. As long as there exist estimations of which pieces of data are crucial, the minimization of risks and the maximization of the potential benefits and innovations for Social Devices can also guarantee and preserve the privacy of partaking users [8].

Restricting data collection also has the power to severely limit the potential of beneficial uses in certain situations. If certain minimization rules are set too strictly, prohibiting sets of data from being gathered or restricting them, it could lead to the loss of potentially useful data sets and the lack of future development for the application in question. This demonstrates the difficulties in finding the happy medium between user safety and application development [20].

Furthermore from Social Devices' perspective, it was also suggested that restricting background data could possibly limit the amount of unwanted information to be leaked device-wise. As the tested devices were Android based devices (in the beginning of the project) limiting access would be straightforward by enforcing the OS settings. This could be done for example by enabling the developer mode. By doing this, an active application would no longer be able to decide whether to observe its own accessing rules. In other words: that selected device running a particular Social Devices application would no more let that said application access the network unless it is running in the foreground. One thing that makes this difficult to test is proximity. WiFi will become one of the dominant methods of connecting (or possibly Bluetooth). The problem being; while using WiFi, it overruns these settings and allows connections to be made freely, making this data leak limiting hard to prove to be worthwhile.

In conclusion, whether we as individuals act as data creators or consumers, we have the responsibility to participate in delimiting the appropriate ground for gathering data. We do not want to hoard our own data but instead to share the data to promote innovations. The aim of Social Devices from the start has been to inquire what the users want and protect the parties involved in any interactions. Even though privacy has become the new appraisal criterion for individuals and larger entities, it must be seen as an opportunity to assess the benefits and costs in an efficient way. By developing new standards for applications (like AcOP) and keeping things simple yet safe, both the users and developers will obtain the desired outcome. Social Devices meet the balance between providing innovative features and ways to mitigate against the growing concerns of ‘dataholics’ by means of standardization, regulations, and simplification.

4. TRUST AND PRIVACY PRIORITIZATION

As the concept of smart devices is to make our lives both more efficient and effective, the real opportunities and, more importantly, the dangers have to come into consideration. However, there will be a fluctuation between privacy and trust depending on the use of a selected device after a certain adequate base “security” has been established. This section is dedicated to analyzing the privacy-trust representation illustrated in Figure 4.1 (page 22) in the light of the upcoming use cases, followed by an overall analysis of trust and privacy in Social Devices.

4.1 Understanding Privacy and Trust in Social Device Environments

One of the most critical observations regarding privacy and trust implementation is that it is often non-trivial for programmers to enforce certain privacy policies. As applications tend to emphasize information flows, separately implemented policies for ensuring and protecting sensitive values of information from other functionalities are a rising concern [18].

For the time being, all the approaches for asserting trust revolve around the propagation of known values of trust. A trust value is assigned by the users themselves in trusted networks, as they use the application and evaluate its pros and cons. Their conclusions provide an assessment of their level of trust in the app, whether it is worth using and how much personal data have to be shared to use it.

A secondary value of trust is the assumption that all social-based applications share the same levels of trust. This triggers the illusion and the effect that users using these platforms have difficulties differentiating between trusted users. These assumptions cause issues. Users making their own decisions based solely on their limited knowledge of the potentially wide user base, relying on recommendations set by either other users or possibly the application itself causes so-called “tunnel vision”, which slowly erodes the potential trust bonds.

As all the application examples in this thesis are centered on dependencies on social interactions, continuously reading or sharing content with connected devices, the above-mentioned point of certain assumptions being made solely on beliefs and hunches is substantial to a certain extent. In real life, we do not trust everybody in the same way and moreover we continuously make trust judgements about other people / users by relying on our intuition.

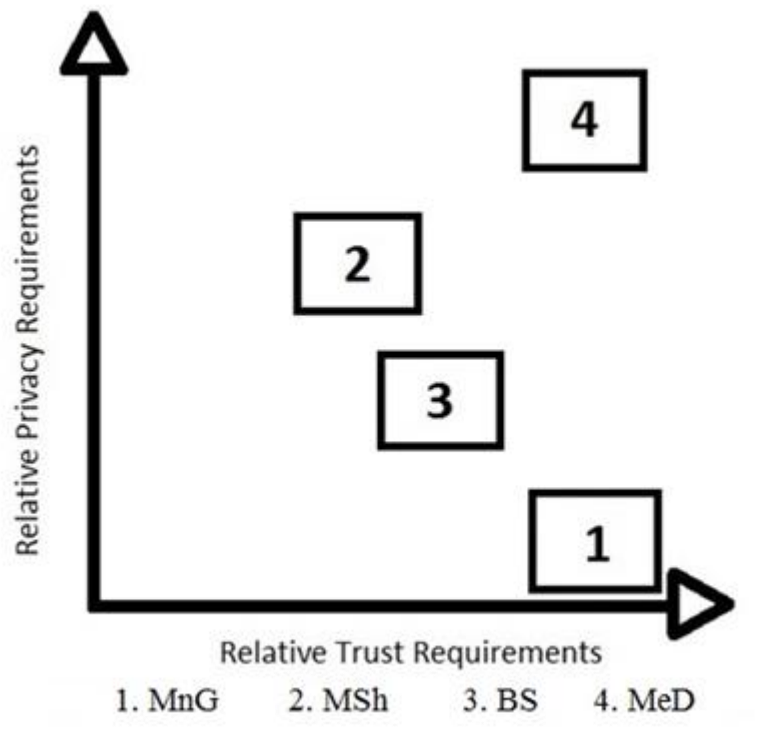


Figure 4.1: Illustration of a use case privacy-trust graph.

All of the use cases described below can be seen as concepts with devices that sense their surroundings, thus gaining a real voice and utility. Possessing multiple devices capturing specific yet completely different data and communicating either through or about the data, truly contributes to contextualizing heterogeneous systems. It is essential to acknowledge how to assess the point where those miniscule pieces of collected data turn into a true unit of information, which represents the real value and in the worst cases, the real danger.

Meet and Greet – The Meet and Greet application is an extremely straightforward one. The required privacy and trust represent the minimum level of user requirements in the light of the fanfare use case.

Additionally, even if the use case were slightly different, say a contact information swapping application operating in a conference where contact information was shared between the participants when proximity requirements were met, the operating principle would not change much. At this particular point of the digital era, acquiring phone numbers and names is remarkably easy. As long as the privacy side of the application is guaranteed, the users will automatically assume a certain level of trust where the contact information will not be leaked by any of the other users.

Media Share – Sharing various pieces of data is an everyday occurrence for the majority of mobile device users worldwide. As for the voluntary nature of this occurrence, on the other hand, it is not always clear. Ensuring the privacy of users who are sharing any given media is always a high priority. Whether it is photographs or videos, the old saying holds true: a picture is worth a thousand words. As we share data, it quickly evolves into information and that very information creates context. Users demand that very same context when operating any application and, with applications such as Media Share, the data involved automatically communicates with other pieces of data in order to have a working meaning.

Let us imagine a scenario where the users compile a story by taking either photographs or short video clips that will be chained together to create a video story. As long as everything is protected and managed by a more or less unbiased party according to the users, they do not really concern themselves with trust towards other users. This argument is not one hundred percent accurate as some of the users will still have a valid concern about who has access to their photos or videos, as illustrated in 5.1. Arguably, at the same time, this concern can be seen as invalid as the users have agreed to the user agreements whem using that particular application, which guarantees unwavering privacy and uniform trust.

Buddy Searcher – This socially charged application model thrives on situational data. By analyzing the locations a user “logs into” through Social Devices, a stamp could be produced indicating the current time and location, e.g., a restaurant or a bar and sharing it with a group of friends, should this be the intended and desired operation. Furthermore, a search field could be added to locate your friends as some sort of user check-in history feature.

Additionally, an action could be triggered by entering the premises of a friend. An indicator would pop up letting the user know someone is close by without doing a search, if this person is on a friends list. This feature could also function as a reminder or a notification, announcing that a group of friends have got together at a bar nearby and the user should also go.

Taking into consideration the operating principle of this application, it is important to notice that the trust factor acts as a more relevant indicator than privacy. As mentioned previously, similar adaptations of this application are already in use and, surprisingly, individuals using these applications are not particularly concerned about leaked information. This creates a problem for privacy.

Every tag that is made by a user creates a little piece of the story of their everyday life. The concerns that arise because it remains a mystery what happens to those stories after they have been told. Managing such information should not be taken lightly, as what happens to the information collected by the taggers and who actually controls it is still unclear.

Depending on the situation, meaning whether the users are family or a group of friends or just semi acquaintances, the trust factor fluctuates in various directions. As users we are mainly concerned about who has access to our data and what is done with it, but at the same time it is easily overlooked if we have a trusted bond with someone. Moreover, the trust factor can fluctuate the other way around where certain activities are more suitable for sharing with a friend rather than a family member, e.g., paying a visit to a marriage counsellor.

Furthermore, applications that have the ability to monitor movements also have the capability of calculating our preferences and actually suggesting various activities, depending on our current location, for example, from an automated coffee maker to much bigger components, which will be discussed later.

It is safe to say that the stories that are told keep evolving constantly and how they actually evolve is up to the users, who have to be aware of what is “put out there” while using this type of application. Collecting data has gone beyond identifying information and individuals as it slowly progresses to depicting our lives and movements from no-value content into an actual narration of the lives of users who choose to use the service.

Medical Dispenser – As technology moves forward and the population ages, certain changes to the state of one’s health can be perceived differently. Soon it will become commonplace to be able to “visit the doctor” from home. The Medical Dispenser acts as a medication intake reminder device. It is specifically custom designed for individuals with daily medications such as blood pressure medication or even for patients with Alzheimer’s. The Dispenser recognizes the counter-part that a patient always carries with them at home, e.g., a watch, a phone, or in the future some sort of bio-tag implanted under the skin. When a certain physical proximity occurs between the counter-part and the Dispenser, it runs a true value check to ensure that the daily dosage has already been taken and informs the patient.

The operation can also work vice versa, meaning that the Dispenser reminds the patient to take the dosage if it has not yet been taken. The reminder can be a simple sound notification enforced with a display on the Dispenser, stating the results of the value check.

The Medical Dispenser represents the high-end requirements for both privacy and trust, yet there are situations where both of these can be overlooked. The Medical Dispenser exemplifies the maximum level of both privacy and trust in a sensitive medication situation. With applications such as medicine reminders, the actual value that is on the line is extremely valuable to the individual using the device. It not only represents a record of what is happening at home but an actual bio profile / record, generated by a sensor, which is later analyzed by another party, in this case the medical staff in charge of the medication and treatment. In situations like this, the users have the right to know what data are being collected and for what purpose but unfortunately there is no real possibility to prevent the collection of the data.

Furthermore, situations where medicine dispensing is still sensitive but the people around the individuals using the application have a comfortable bond with the user change the need for privacy and trust. Medication that could be embarrassing for the user, e.g., for STDs, is a highly personal matter but in circumstances of a very close group of individuals, disclosing that particular information with an aural reminder from the OJS device might only cause laughter without any other repercussions.

As shown in all of the use cases, trust and our willingness to share do not always automatically correlate with privacy. Whether we are making contacts, sharing media and creating stories, or leaving digital fingerprints behind, the desire to be creative and inspirational to generate actual insight for the future is real. As long as we can protect and manage our own stories in conventional ways to satisfy individual desires to ensure adequate privacy, we will always create content in spite of the data created and surrounded by our conversations, whether they are traditional or prompted by Social Devices.

Vasudevan and Owusu state in [11] that as the post-PC era is coming to an end, we as users and developers are placing a great deal of dependence on our mobile devices and, in all honesty, these devices are still insecure.

In an optimal Social Device scenario, we have to analyze the environment into which we are about to release our more or less finished product. The responsibility and understanding toward mobile device platforming, which is currently dealing with access and providing an opportunity for high-end development, give us the leverage for innovation and future research.

In Social Device environments, an enormous amount of trust has been placed in mobile devices. With the vast possibilities for application development suggested in this section, more or less sensitive data and highly confidential information are at stake, and need to be protected. As a side note it can be stated that currently, even though we have surpassed the PC era, mobile operating systems are untrustworthy compared to desktop operating systems. Even though mobile devices in principle try to be more secure, in practice they are still highly vulnerable, mainly due to their complexity.

4.2 Protecting Trust in Social Devices

The points that are raised in this section exemplify the main point of this entire thesis; namely, it is important to comprehend that users are both the source and the problem of creating reliable trust in Social Device environments. This is mainly due to difficulties in understanding the concept of data discretion, which applies to both personal and collaborative users, depending on the purposes of a selected application. For example, in the Media Share app the operation is quite simple. A problem could arise if a user in a collaborative environment creates inappropriate content and shares it or abuses other user information, thus spoiling the user experience for others. In turn, personal applications like the MeD app have a high trust component as the processed data is extremely personal. In this scenario, the user already knows how delicate the data is and trusts the other party, e.g. the physician, without question.

Earlier in this section, privacy and trust relations were described by means of various use cases for Social Devices. As certain assumptions were made, not all the speculations are definite. For instance, using the Meet and Greet application can lead to awkward and potentially dramatic situations. The basic principle is simple as the application is only meant to acknowledge proximity parameters and the acknowledgement itself is a predetermined sound. Whether this sound is a whistle or a formal greeting, the trust parameter remains immutable.

However, with crafty users, if the acknowledgement is changed to a possibly racist sound or exclamation, depending on the situation and the location, the consequences will become unpredictable, which will tip the trust scale. As the scale tips, the privacy-trust gauge shifts correspondingly. This example illustrates the fact that it is impossible to substantiate how privacy and trust react until every possible scenario has been covered in real time.

The main problem as mentioned previously is users with a lack of data discretion. For the sake of humor or harm, every user can potentially be considered ‘dangerous’ even if the application itself is safe privacy-wise.

Moreover, in Social Device environments, on some occasions it is possible to have certain space allocations with an unknown quantity of collaborating devices, sensors, or non-determined gadgets. This means that data will be constantly generated, transmitted, and stored. In contrast to all the transmissions and interactions, the amount of sensitive data is a concern for users, especially with trust issues.

The reasons for these issues are many, but in terms of Social Devices, certain assumptions can be made. For instance, with a variety of application possibilities, the environment itself is often unfamiliar to the users. In addition, the trust relationships that users come across in Social Device environments differ in that, occasionally, the users do not have a trust relationship with the owners of the environment in such a way that they might have with their local administrators for private information handling, e.g., network service operators in which a more or less genuine relationship has been established.

With regard to the application opportunities for Social Devices, users' rights for accessing certain functions and interactive modules will dynamically change, with their relationship with that particular mechanism generating data in environments which might not even be predetermined.

As there are no particular reasons for users not to form their own ad hoc groups, referring to Media Share, and using cameras to record interactions in a conference room for example, the camera action is certainly administered by the environment even though it might appear that the application is in charge of it.

Moreover, the users should not have access to the produced video outside of the meeting period. As the environment also adapts as it is being associated with pieces of information, limitations for users accessing the pieces of information generated in the meeting have to be laid down.

The use cases introduced in this thesis are decentralized, meaning that there can be multiple sources and outputs for the data. Users must have guarantees that there will be no caching or data replication over the required amount as there is usually no single point where access and control can be enforced. When possibly large groups of users are involved, all protection schemes must allow efficient sharing among legitimate users without endangering said users. This is how trust is enforced.

Duan and Canny introduce an interesting claim in their paper [10]: "Unless there is a clearly stated protection policy and scheme in place, together with provisions for convincing verification, users will not be comfortable trusting the infrastructure with their data. And this in turn will hinder the acceptance and limit the usefulness of such systems." However, for Social Devices this means that the generated data are relatively easy to protect even though the data can be generated by another user or party, which is not the case. This complicates the trust bonding as users are required to trust the systems and infrastructures to begin with to a certain degree.

Blind trust cannot be forced upon users. Instead, by performing certain operations, the infrastructures and environments can be developed in a way that promotes trust and confidence amongst users even before any major commissioning has occurred.

4.3 Privacy-Oriented Social Devices

Social Devices is heavily focused on collecting data, as stated earlier. Much of the collected data used for further development may be extremely sensitive. As the data streams can also be heavy, they have to be dealt with in real-time. This can be done by protecting the privacy aspect of Social Devices.

It could even be said that privacy, in the sense of pure coding, offers the solutions for effective control of access data, especially in ubiquitous environments. The coding is also responsible for ensuring that no minor or major disorders that enable unauthorized breaches or open backdoors are a possibility. Should these disorders become valid, it can lead to privacy violations and in the more severe consequences described in a later section.

Regarding privacy, the desired function is to enable secure execution without any major issues during interactions. Especially with mobile devices, as the wish list of features increases, problems tend to occur, and how privacy for users can be offered includes a multitude of guaranteed aspects.

The below-mentioned concepts are mainly to be used as desired guidelines rather than factual implementations that work in every situation. The principles of secured data streams and safe storing are not new by any means; however, the illustrated concepts can be used in new ways that are yet to be developed to enable true privacy with application development. The aberrations and privacy breaches that occur are usually manmade. These breaches may be either intentional or unintentional, depending on what type of breach has been made. Coding errors are an everyday incident that can lead to leaked data or unsafe environments for a vast amount of devices, yet most of the time these incidents are unintentional, although impossible to avoid.

Moreover, intentional privacy breaches are solely meant to harm the environment, the users, and the integrity of the data at stake. These breaches usually occur more rarely but usually the damage dealt is greater than from unintentional coding errors. To avoid such breaches in Social Device environments and also in general, the following concepts might offer the solutions.

Isolated Execution – “Isolated Execution (IE) is a software reference implementation of the ‘security through isolation’ concept.” [11] This can easily be interpreted mainly as a security issue; however, this also offers an opportunity for measuring and improving privacy and even trust. While this execution enables the possibility to run pieces of software in complete isolation, it provides both secrecy and integrity for code and data run in a certain module of an application in this case. The only problem with IE is that currently the operating systems in mobile devices provide isolation for applications and other resources. Moreover, this protection can be bypassed when the operating system itself is compromised. Should this be fixed by, for example, running a parallel environment within an existing environment yet still acting as a separate coprocessor, this could severely reduce the scope of human error initiating unwanted processes or events [11].

IE would allow the user to decide whether to open or share files depending on the confidence the user has in the file origin. Similar to sandboxing, this type of concept could be implemented within Social Devices in some desired fashion in order to limit the ‘damage’ to isolated environments, leaving the users unharmed.

Trusted Pathing – Trusted Pathing (TP) is designed to protect communication authenticity and, optionally, secrecy and availability between software and a peripheral, in this case a mobile device. When used with human-interface devices, trusted pathing allows a user to ascertain precisely the application with which that particular user is currently interacting [11].

For Social Devices, the concept of trusted pathing is crucial if the intention is to embark above the novelty level of applications. With fully trusted path support, malicious applications spoofing the originals, by creating user interfaces that look identical to the legitimate applications, will conceivably become virtually ineffective and useless. Should this trusted path be absent, the consequences could be horrific. The only issue with Trusted Paths is that building them to be secure can be a challenging task, depending on the desired function and application, especially when the amount of devices multiplies even to a handful of devices, let alone dozens or even hundreds of devices.

4.4 Trust from the Users’ Perspective

Social Devices thrive on protecting the anonymity of the users in communicating environments where data are captured. Apart from the tracking application introduced in this thesis, designated pieces of information will be essential for the functionalities of the applications. Users will expose themselves and their identity to parts of these systems, but the locations will be concealed. With control and feedback, privacy is easier to preserve, although it is nearly impossible to camouflage from personal information derived from a variety of sources. As interactions are shared between collaborative users, these collected pieces of information can be filtered. Users may also have several pseudonyms and as they act through their unique action patterns, even with multiple tags, pinpointing or fingerprinting users is possible. However, attempts to make a connection between a tag and a user’s real-life identity are unlikely.

The principles for pursuing and fulfilling privacy follow the considerations of Marc Langheinrich [12], who suggested six guidelines for privacy-aware system design:

- **Notice** – Users should always be aware of what data are being collected.
- **Choice and Consent** – Users should be able to choose whether the collected data are used or not.
- **Anonymity** – Pseudonymity should apply when identity is not needed.
- **Meeting Expectations** – Systems should mimic real-world norms.
- **Security** – Different amounts of protection depending on the situation.
- **Access and Recourse** – Users should have access to data about them.

These principles summarize the fact that user data protection is related to privacy. Not only can certain techniques be used for protecting and ensuring privacy, but privacy considerations can and will influence protection policies and implementations. Privacy will not be compromised until the schemes in use suffer from dishonesty, meaning that both the servers and fragments of collected data are assumed to be honest and safe, unless detected as contaminated after all. Finally, these principles reinforce the fact that unauthorized monitoring must be detected, but at a reasonable cost, without endangering users and then again by not caging them with brute force, creating a ‘Big Brother’ type of scenario.

5. PRIVACY BREACHES IN SOCIAL DEVICES

In the current world, there are certain individuals who will always take up the challenge of breaking applications in some form, turning them against other users either just to cause harm for the sake of it or to gather delicate user information, which is not meant to be leaked anywhere. It is safe to say that every piece of equipment enabling some sort of cellular phone and laptop activity amongst users causes a different level of damage, depending on how we, as users, regard the harm.

Users tend to neglect privacy concerns and almost blindly believe the guarantees given by certain application developers about privacy and how users should be and feel safe. Unknowingly, these same users feel empowered by these applications and interact as they are supposed to, based on their attitudes and behavior. Too often the application user interface contains technologies for specifying certain rules and machine learning that subliminally refine user behavior. We adopt these applications differently, we see the usage purposes how we want to see them, and it is the purposes that cause the problems. These problems will be covered in more detail in this section.

5.1 Meet and Greet Case (MnG)

For the purpose of this example, picture a plausible scenario where either complete strangers or friends meet face to face in an elevator, for example, with mobile phones equipped with hardware capable of running OJS or other SD services on their own devices and specific software. The software running in the background will acknowledge both of these devices through Bluetooth or an actual mobile network. When a certain proximity requirement is met, the device will establish a short period connection, as shown in Figure 5.1, and generate the sound of a fanfare with applause, letting both of the individuals know that they are using the same of piece of software.

For an arbitrary use case such as this, there are hardly any privacy issues, even considering the anonymity of the users. In order for a device to perform an action of this magnitude, no greater effort is needed as the software itself is extremely light and the performed action requires no extra effort.

It is possible for a perpetrator to gain access to the established connection; however, there would not be much benefit from it. The intruder would only gain the information that two, or in larger crowds several, devices established a short period HTTP connection without leaking any personal information. The odds of these same devices establishing another connection in a short period of time would be unlikely, thus making the effort not worth the value gained.

Figure 5.1: A successful Meet and Greet acknowledgement.

Additionally, in order to gain something concrete, the devices would have to be in the same wireless network, and the perpetrator would have had to set up a scanner to monitor the activities in the network and then poison the routing protocols, which is impossible without knowing the server address. In terms of privacy, Meet and Greets are on the lowest level of privacy requirements.

5.2 Media Share Case (MSh)

In this use case, the desired action is to share a piece of media between selected users. The mechanics of media sharing is as follows: two individuals using software dedicated to sharing, e.g., video clips, with one another are in a certain proximity measured by an OJS, as shown in Figure 5.2.

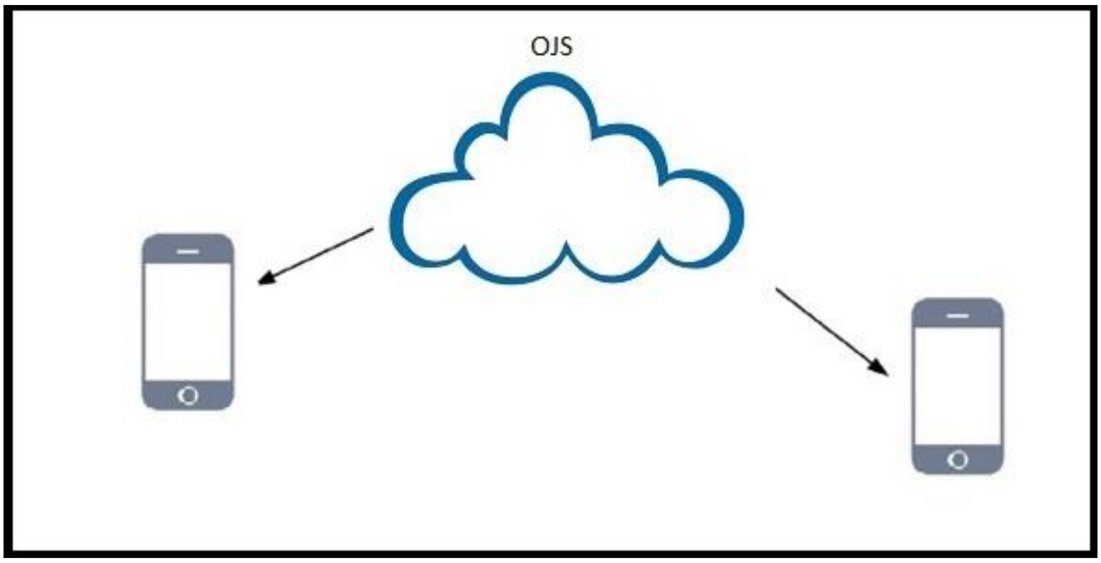


Figure 5.2: Illustration of media share principles.

The server acknowledges the individual’s desire to share pieces of information in this open type of environment and the devices will then establish a connection and commence sharing video clips with one another.

When assessing the privacy of such an implementation, the number one concern is an MITM attack. As the devices use HTTPS to secure their connections, there is a small window of time where the devices are vulnerable. As the server acknowledges two devices willing to share data, it is possible for a third party individual to poison the routing tables of the devices. In order for this to work properly, the address space has to be calculable and the server must be centralized.

Additionally, for the intruder to perform the attacks successfully, the correct time to strike is before the devices establish their HTTPS connections. All the devices will acknowledge the server when they are close enough to trigger the proximity radar. At this particular moment, all the traffic will be done as HTTP before the server commands the devices to use HTTPS. If an intruder manages to hijack the connections at this particular time, it is possible to start rerouting the devices’ traffic through the intruder’s device, which now appears to be the server for the rest of the devices, as illustrated in Figure 5.3.

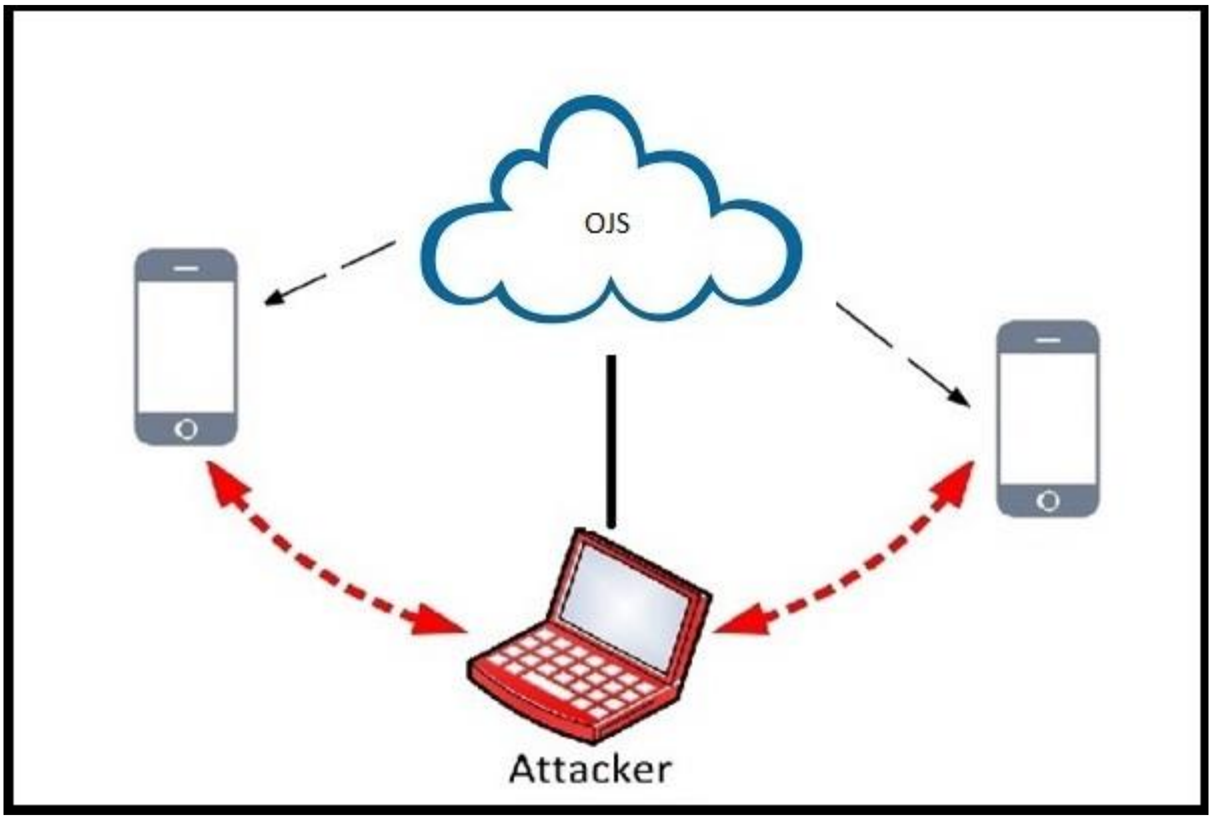


Figure 5.3: Illustration of a successful MITM rerouting attack.

This type of attack is true poison for privacy. In this example, the locations and the number of participating devices are limited, but it is possible to perform these attacks on a much larger scale, as they are extremely hard to discover when done properly. Imagine a situation where the attacker starts to modify either messages or the data in transit. This type of situation can be described with a modified real life example: Imagine sending a simple letter to someone as the MITM attack commences. The attacker could modify that message to include anything imaginable, maybe just a favor asking for money. The text itself might not appear identical, but it could match the additional request word for word. As long as the attacker was doing this the entire time, the recipient would not notice the difference.

Additionally, should the recipient write back and include some money, the attacker could once again alter the letter, keep the money, and send the letter forward. This would require slightly more work in the so-called offline world, but the tricks are endorsable and much easier to do online, where all this could be automated by software or made possible through an advertised free Wi-Fi router which would later infect all the connections.

5.3 Buddy Searcher Case (BS)

This next example could be considered as some sort of local search and discovery service application, specifically for mobile phones. The operation of the application is somewhat similar to an existing application called Foursquare, which is commonly used within social networking sites such as Facebook. Foursquare operates by taking into account the places a user visits and forwarding that information to other users that allow a certain level of trust with other users. This enables the function to recommend places around a user's current location. For this use case, the principle of the function is slightly different.

This search and locator resembles a GPS, as shown in Figure 5.4. It is mainly meant for use amongst trusted individuals, close friends or family, allowing them to see which places a user has visited at which hour and possibly the date of the occurrence if so selected.

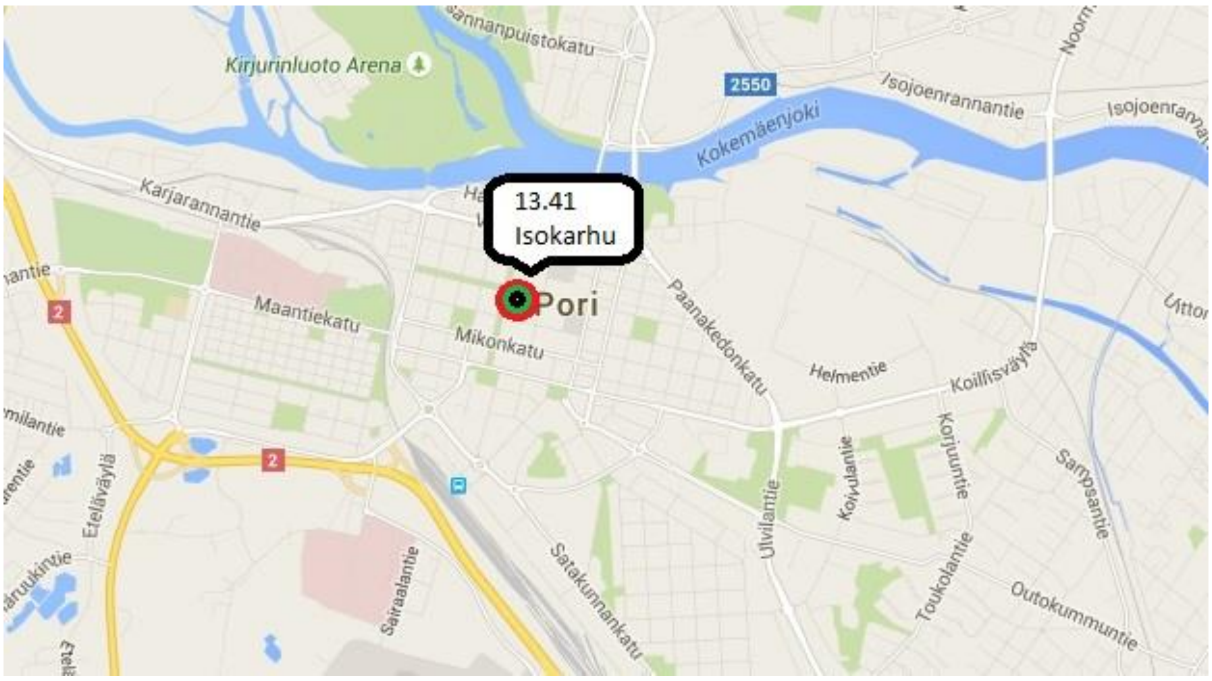


Figure 5.4: Illustration of a location stamp.

The application could even have a function that allows the user to set an ongoing current location that is tracked or a built-in application with physical proximity scanners. For example, in Figure 5.4, the user could set their current location and the tracker would keep the status open unless the individual leaves the proximity, after which the application adds a tag that informs other users that the individual they have a trusted bond with has left that particular area at a certain time of the day and is once again on the move.

At first this type of application sounds highly risky, which indeed it is in the event of an MITM attack. A scenario is presented in Figure 5.5 where an individual has used the particular tagging application for a day or even several days. With digital fingerprints, calculating routines on a daily basis becomes much more probable. Surprisingly, even a small amount of location data could project the current whereabouts of an individual, should a perpetrator somehow manage to infect all the scanners that a user has been connected to or get hold of the information that the application collects and afterwards shares with trusted users.

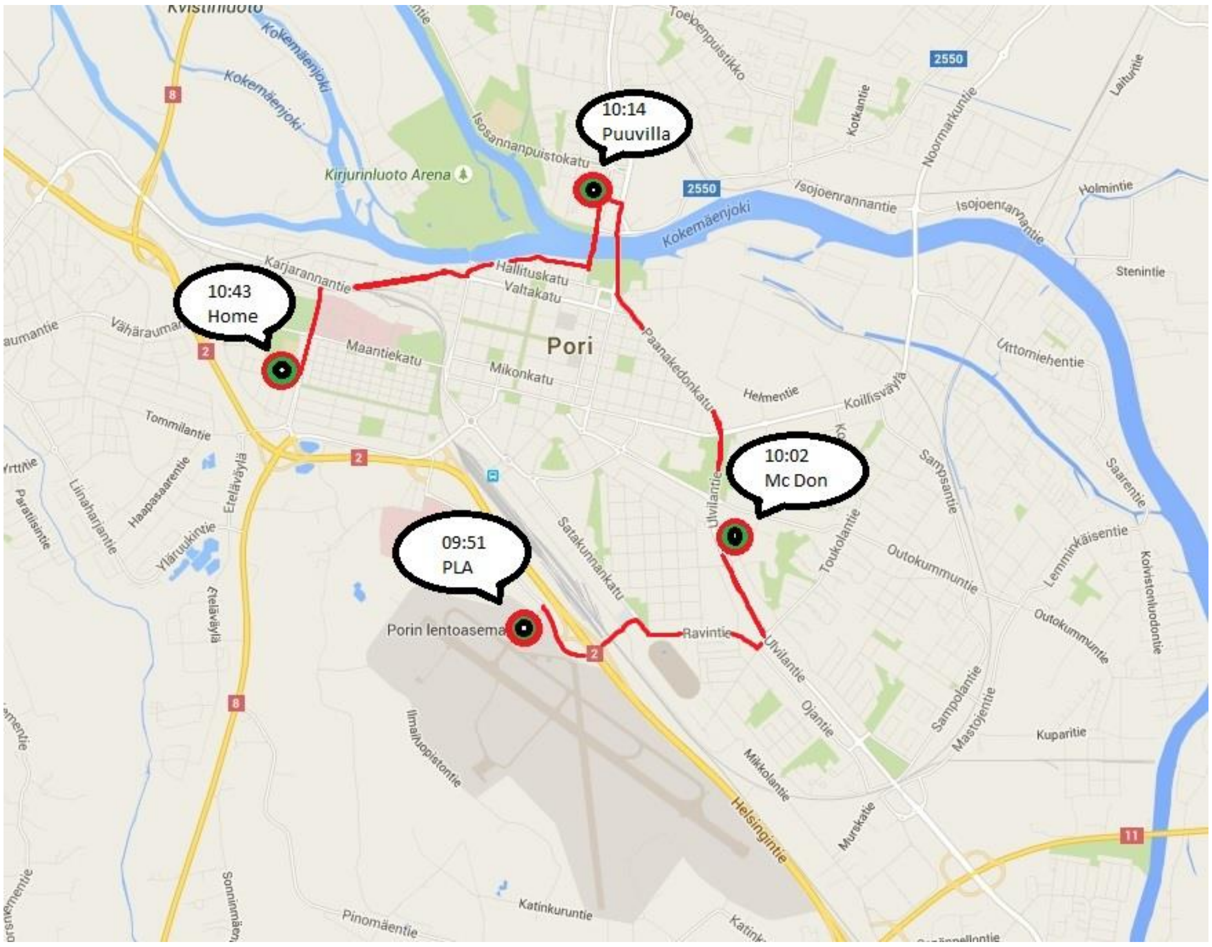


Figure 5.5: An illustration of a user’s calculated daily routine.

As the application stands, both the privacy and the trust components are highly valuable to the user no matter the situation. Should this kind of motion information be leaked, it would act as the worst kind of GPS possible. Even in the situation where the application only collected anonymous information, it could be considered highly perilous and with a vast amount of both gathered and processed data it is even possible to isolate the tags to deduct who the actual user is, exposing them to zero privacy subsequently.

5.4 Medical Dispenser Case (MeD)

For the purpose of this use case, imagine the scenario of an elderly person who has to take certain doses of medicine on a daily basis, e.g., for high blood pressure. The way the application works, shown in Figure 5.6, is that the OJS keeps track of the medicine taken and informs the user if they have already exceeded the daily limit or if they still have to take their medicine.

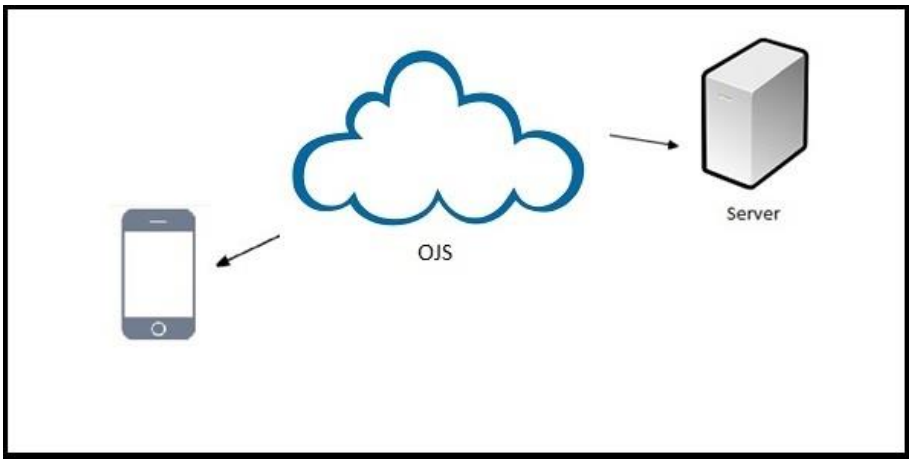


Figure 5.6: Illustration of the Medical Dispenser operation.

The OJS receives information by scanning a trusted device worn by the user such as a mobile device or even a smartwatch, which makes wearing the device much more comfortable and less forgettable. After processing the gathered information, the OJS forwards the information to an outside server maintained by the hospital, where the information can then be accessed by the doctor in charge of the medication.

Privacy-wise, this type of application represents the end-level of privacy. Such sensitive information should not be leaked outside trusted individuals, which makes this application's need for trust as high as the need for privacy. However, depending on the medication, the level of privacy and trust might fluctuate.

Moreover, this type of application is extremely vulnerable to MITM attacks. If in some way, an attacker could set up a server between the OJS and the hospital's server and be able to reroute all the traffic through the fake server, the damage would be immeasurable – as illustrated in Figure 5.7.

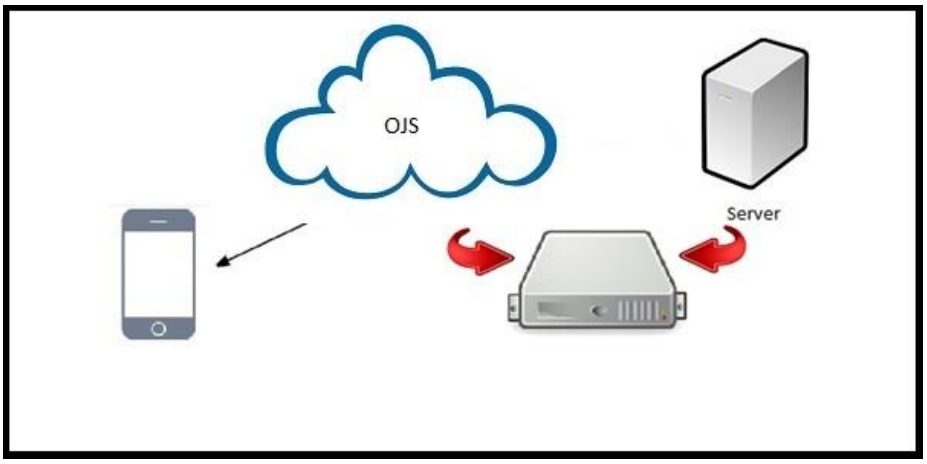


Figure 5.7: An illustration of an MITM attack using a Medical Dispenser.

The problem that arises with this hijacking is that as the fake server logs in for the user, captures their details and forwards the information, everything may seem to be operating normally and it is difficult if not impossible to detect the server sitting in the middle, forwarding the data back and forth and eavesdropping or actually hijacking the sensitive information.

The complexity of this kind of attack is that if the devices used are equipped with efficient hardware to operate on another level besides just sending to or reading a reminder from an OJS, the actual infected device could be the mobile phone or the smartwatch itself. Being infected by some sort of malware hidden in the background of the device or the application itself introduces the possibility of insertion between the device and the OJS. Additionally, there could be a scenario where the developer chooses to implement malware of this kind in order to gather sensitive information and even sell it forward.

Furthermore, the attacker could alter the reminders as they please. By tampering with the information, the user might end up taking lethal doses of medicine or even miss all the reminders throughout the day, making it just as lethal. Everything would seem fine on the server end as long as the device still kept sending acknowledgements, without actually reminding the user to take their medicine.

5.5 Unraveling Privacy in Social Devices

From the perspective of developers, our abilities to understand, choose, or even control what personal information we share and with whom is a real challenge. With an unknown amount of components in play, for instance identity and granularity, the traces of data should be well hidden so as to be practically untraceable. Traces can be mined and extremely difficult if not near impossible to retract once they have been shared. Needless to say, sharing such revealing data has a number of risks. As shown in this thesis, incidents involving stalkers or thieves are more than possible when using social networking applications. Some of these threats are visible, some are not. Since data are no longer gathered solely by large corporations and organizations, individuals and small community groups may create sensing applications and begin collecting valuable, personal data.

As developers, we have the duty to understand the responsibilities and tasks that come into the equation at this point. There are ways to contribute towards individual and small group data protection. Especially in this sensing social device environment, we must ensure that we have created workable standards in our environment for our user community. The task generally speaking is very simple: we must actively engage individuals in their own decision making, since the data are so granular and personal. The aim of improving user abilities to sense and regulate their privacy is to implement a system to negotiate decisions based on who is asking, what they are asking, what for, and so forth.

To a certain extent it is crucial to give users or participants as much control over their own location data as possible when talking about GPS traces, as in the Buddy Searcher example shown earlier. Participants must have the possibility to either make or revoke decisions about sharing their data with any third-party applications. Thus, analytically speaking, the participants using the applications are not only subjects for gathering data but they act like investigators when they take part in a small or large research initiative. As it stands, the participants should have their own input on how data are collected, processed, stored, and discarded.

Should developers hesitate to tailor unique access controls and data management to meet the requirements of the participants, developers must at least ensure privacy by limiting the amount of raw data that a participant is required to share. With privacy, more data is not always better.

6. CONCLUSIONS

As this Master’s thesis was aimed at examining the issues arising with privacy and trust regarding Social Devices, some clear main issues were detected immediately. For overall innovative development in the future it is essential to grasp the reality, namely, that where trust is concerned, users are both the source and the problem for reliable trust creation, because in Social Device environments, understanding the concept of data discretion applies to and supports personal as well as collaborative users.

For the sake of humor or harm, every user can potentially be considered ‘dangerous,’ even if the application itself is safe in terms of privacy. Especially with mobile devices, as the wish list of features increases, problems tend to occur, and how user privacy can be offered includes a multitude of guaranteed aspects.

The happy medium for success revolves of course not only around proper development but also by giving the users some leeway depending on the application, for instance adapting the guidelines by Langheinrich in [12]. The trust relationships that users come across in Social Device environments differ, in that occasionally users do not have a trust relationship with the owners of the environments. In regard to the application opportunities for Social Devices, users’ rights to access certain functions and interactive modules will dynamically change with their relationship with that particular mechanism that generates data in the environments.

Additionally, as the research progressed, certain realizations were made, such as the function of SD implementations as an abstraction layer for all devices, making it possible to take advantage of a diversity of devices, not simply limiting the use to mobile devices. As regards performance, whether mobile devices are the best way to go is still under research, as for most applications performance is not an issue; but when envisaging a heavy application that requires real-time responsiveness, certain mobile devices do not have the capabilities or capacity to function fast enough, hindering user experience.

The concept of Social Devices was introduced at the beginning of the thesis. Social Devices functions as a unique sociological system, uniting humans and devices. Also, additional implementations of Social Devices were presented. Both of these implementations, the Action-Oriented Programming model and OrchestratorJS, try to fulfill the requirements for Social Devices by, for instance keeping track of the closeness of available devices and finding a suitable configuration for said devices in the proximity of each other, allowing them to execute operative functions with one another.

Moreover, as all the application examples in this thesis revolved around dependencies on social interactions, which continuously read or share content with connected devices, the above-mentioned point of certain assumptions being made solely on beliefs and hunches is substantial to a certain extent.

Our new, in most cases wireless, upcoming, and existing technologies are rapidly transforming our relationships to other people and places around the globe. This means that the constraints that used to be valid some years ago are now simply temporal, and our society is stretching this line and reshaping it fundamentally to increase the capacity of information flows.

In real life, we do not trust everyone in the same way but instead we continuously make trust judgements about other people / users by relying on our intuition. Based on all these observations, it can be stated that this research gives valuable insight into future development, which was also one of the purposes of this thesis.

REFERENCES

[1] Analog Devices Inc., Device-Based Social Networking [WWW]. [Referenced: 2.10.2015]. Available at: http://www.analog.com/media/en/technical-documentation/white-papers/101819526Device_based_Social_Networking_White_Paper.pdf.

[2] T. Aaltonen, T. Leppänen, T. Mikkonen, V.Myllyniemi, N. Mäkitalo, T. Männisto, J. Pääkkö and M. Raatikainen. Social Devices: collaborative co-located interactions in a mobile cloud. In E. Rukzio (ed.), Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia, Ulm, Germany, ACM, 2012.

[3] T.Aaltonen, V. Myllärniemi, N. Mäkitalo, J. Pääkkö, M. Raatikainen. An Action-Oriented Programming Model for Pervasive Computing in a Device Cloud. The 20th Asia-Pacific Software Engineering Conference, APSEC 2013, Bangkok, Thailand, December 2-5, 2013, IEEE, pp. 467-475.

[4] NodeJS, About Node.js [WWW]. [Referenced: 2.10.2015]. Available at: https://nodejs.org/en/about/

[5] Wikipedia, Representational state transfer [WWW]. [Referenced: 3.10.2015]. Available at: https://en.wikipedia.org/wiki/Representational_state_transfer.

[6] RestApiTutorial, Using HTTP Methods for RESTful Services [WWW]. [Referenced: 5.10.2015]. Available at: http://www.restapitutorial.com/lessons/httpmethods.html

[7] D. Guinard, As the Internet of Things Grows, Is Privacy Possible?. IOT Journal, [WWW]. [Referenced: 8.5.2015]. Available at: http://www.iotjournal.com/articles/view?12604.

[8] European Data Protection Supervisor, EDPS: Data Minimization [WWW]. [Referenced: 28.10.2025]. Available at: https://secure.edps.europa.eu/EDPSWEB/edps/EDPS/Dataprotection/Glossary/pid/74.

[9] N. Mäkitalo. Building and programming ubiquitous social devices. Proceedings of the 12th ACM international symposium on Mobility management and wireless access, MobiWac 2014, Montreal, QC, Canada, September 21-26, 2014. New York, NY: ACM, 2014, pp. 99-108 (ACM international symposium on mobility management and wireless access).

[10] J. Canny, Y. Duan. Protecting User Data in Ubiquitous Computing: Towards Trustworthy Environments, Privacy Enhancing Technologies. 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004.

[11] E. Owusu, A. Vasudevan. Trustworthy Execution on Mobile Devices: What security properties can my mobile platform give me? Trust and Trustworthy Computing. 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012.

[12] M. Langheinrich. Privacy by design – principles of privacy-aware ubiquitous systems. . Springer Berlin Heidelberg, City, 2001, pp. 273–291.

[13] R. Latham., T. Merrill, D. Navetta, K. Santalesa. Social Media: The Business Benefits May Be Enormous, But Can the Risks -- Reputational, Legal, Operational -- Be Mitigated? ACE Limited, 2011. [WWW]. Available at: http://old.findinet.nl/~uploads/newsModule/ace_socialmedia111118.pdf

[14] OrchestratorJS , Orchestrator.js for developers [WWW]. [Referenced: 26.01.2016]. Available at: http://orchestratorjs.org/#/develop.

[15] L. Cranor, J. Hong, N. Sadeh, Understanding and Capturing Peoples' Privacy Policies in a Mobile Social Networking Application. Carnegie Mellon University Research, 2007.

[16] R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD Thesis, University of California, Irvine [WWW]. [Referenced: 3.10.2015]. Available at: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[17] Tickto. Predictive Intelligence: Notify, Expand Conversion, Boost Brand Loyalty [WWW]. [Referenced: 31.3.2016]. Available at: http://tickto.com/predictive-intelligence-notify-expand-conversion-boost-brand-loyalty/.

[18] J. Yang. Jeeves [WWW]. [Referenced: 31.1.2016]. Available at: https://projects.csail.mit.edu/jeeves/

[19] FTC Staff Report. Internet of Things – Privacy & Security in a Connected World [WWW]. [Referenced: 26.01.2016]. Available at: https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf, 2015.

[20] D. Castro. The "Internet of Things" Requires New Thinking on Data [WWW]. [Referenced 13.04.2016]. Available at: http://www.innovationfiles.org/the-internet-of-things-requires-new-thinking-on-data/

[21] Device to Device (D2D) Communications, Project introduction [WWW]. [Referenced 21.7.2016]. Available at: http://www.cs.tut.fi/projects/d2d/.