SETHER

A model for secure transfer of patient data between healthcare systems and mobile applications

Simon Hjelmfors

Tutor: Magnus Bång
Examinator: Magnus Bång
Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Simon Hjelmfors
Abstract

Strong authentication and secure communication of sensitive medical data are crucial parts when empowering patients by giving them access to data stored in healthcare systems. Through the use of a mobile application and device patients access relevant personal health information and thus bringing more control over their diseases and empower them to, for instance, make long-lasting lifestyle changes. This thesis presents the communication model SETHER, an approach to manage secure transfer of patient data between healthcare systems and applications running on mobile devices, based on SAML and OAuth. From this model a proof of concept prototype was developed, using API Gateway to connect to the National Service Platform, an infrastructure that interconnects most healthcare systems in Sweden. The results indicate that using SETHER as a model for communication, a secure transfer of patient data between patients and caregivers is possible. The advantages of this is a secure single sign-on user experience to access patient data through a nationwide system independent on the healthcare system the caregivers are using. Future work includes developing mobile patient applications, using SETHER to connect patient with their caregivers and providing tools to empower patients in their everyday life.
Contents

1 Introduction .................................................. 1
  1.1 Motivation ................................................ 1
  1.2 Goal ..................................................... 1
  1.3 Limitations .............................................. 2

2 Background .................................................. 3
  2.1 Patient-centred applications .............................. 3
  2.2 Communication and authentication ....................... 3
    2.2.1 Authentication and password fatigue .................. 3
    2.2.2 Single Sign-On architecture .......................... 3
    2.2.3 Security Assertion Markup Language .................. 4
    2.2.4 OAuth2 .............................................. 4
    2.2.5 Nationwide Medical Information Architecture ........ 4
    2.2.6 API Gateway ........................................ 5
  2.3 Patient empowerment ..................................... 5
  2.4 Patient information ..................................... 5

3 Method ...................................................... 7
  3.1 Procedure ............................................... 7
    3.1.1 Initial planning and requirement elicitation ........ 7
    3.1.2 Planning ............................................ 8
    3.1.3 Analysis and implementation ......................... 8
    3.1.4 Testing ............................................. 8
    3.1.5 Evaluation .......................................... 9
    3.1.6 Deployment ......................................... 9

4 Results ..................................................... 10
  4.1 SETHER Model ........................................... 10
  4.2 SETHER Prototype ........................................ 11
    4.2.1 App ............................................... 12
    4.2.2 App Server .......................................... 12
    4.2.3 Service Provider - API Gateway, NTJP, Healthcare Systems 12
    4.2.4 Identity Provider - CGI ............................. 13
    4.2.5 Login - first time .................................. 13
    4.2.6 Login - subsequent times ............................ 14
    4.2.7 Fetching data ....................................... 15
  4.3 Technical test of the SETHER Prototype .................. 16

5 Discussion .................................................. 17
  5.1 Method .................................................. 17
  5.2 The work in a wider context ............................. 18
  5.3 Possible uses ........................................... 19
    5.3.1 Medication .......................................... 19
    5.3.2 Diagnosis .......................................... 19
5.3.3 Appointments ........................................ 19
5.3.4 Self care ............................................. 19
5.3.5 On the ward ......................................... 19
5.3.6 Medical record ...................................... 20
5.4 Barriers and limitations .............................. 20
  5.4.1 Hardware and connectivity ...................... 20
  5.4.2 Security ........................................... 20
  5.4.3 Legality ........................................... 20
5.5 Future work ........................................... 21

6 Conclusion .............................................. 22

References ............................................... 23
1 Introduction

This section covers the motivation, goals and limitation of the thesis.

1.1 Motivation

Many patients today call for more involvement and insight into their medical data (Batalden et al. 2015). An application for a mobile device where the patient can access their medical information, communicate with their caregivers and get an overview of their health has great potential to increase quality of life for the patient. With the remarkable advances in information and communication technology (ICT) this is technically possible. Even the healthcare, that in the past has been reluctant to share the medical data they hold, are connecting their systems to emerging services to allow patients access to their own medical data. Such a system is the National Service Platform (Swedish: Nationella Tjänsteplattformen - NTjP) that interconnects most healthcare systems in Sweden. Another such service is API Gateway that allows for patient applications to connect to NTjP (Meunier 2011).

To realize this the sensitive medical data sent back and forth between the patient application and the healthcare systems must be secure and let no unauthorized user access. The identity of the patient needs to be strongly verified and the application should live up to the current standards of being aesthetic, responsive and easy to use. There might also be features not made available through NTjP or patients outside of Sweden wanting to have the same power. This calls for a model that can fit all these needs together.

Cambio Healthcare Systems, one of the largest providers of e-Health systems in the Nordic countries, believes strengthening patients through involving them more in their own healthcare is very important (Cambio Healthcare Systems 2016). This thesis was proposed by Cambio Healthcare Systems and carried out with their support on their premise.

1.2 Goal

The goal of this thesis is to develop a model for secure communication of data between health care systems and mobile applications and evaluate its feasibility by implementing a proof of concept prototype. Furthermore, the model shall be flexible to allow future improvements and additions. The end user experience is also an important aspect and the model should therefore support single sign-on functionality.

The following questions will be discussed in the thesis:

- How can a model for two way communication between healthcare systems and mobile devices be designed to achieve secure transfer of data?
- How generalizable is the model?
- Does the model allow for additional functionality to be added without significantly increasing the complexity?
What can be learned from developing a proof of concept prototype regarding barriers and limitations that may be helpful when designing the connected components?

1.3 Limitations

A model that supports rigorous security, good user experience and flexibility to expand has an architecture that easily becomes complex. Therefore several aspects are out of scope for this thesis and the limitations will be explained below.

This thesis’ emphasis is on the communication between a mobile device and health care systems. A web application will be used to verify that the integration to the user works. No native application on a device specific operating system will be created.

The security will rely on widely used, rigorously tested and standardised security solutions. It can be assumed that if these solutions are implemented a secure system is achieved. Therefore no verification of security will be done.

The goal is a model that can be expanded and use several service providers (SPs). However because of the effort involved and time constraints on the thesis only one SP will be used. It is assumed that, similar to the issue of security, if a widely used and standardised method is used it is most likely able to be expanded upon.

Since the data that the model is supposed to handle is mainly sensitive medical data it would be irresponsible to handle in a prototype. Therefore mock data will be used and the prototype will not be connected to an actual healthcare system but to a developer or quality assurance (QA) environment.
2 Background

This section covers a brief explanation of the concept of patient empowerment, patient-centred applications as well as a brief overview of different security and communications protocols to handle authentication and single sign-on.

2.1 Patient-centred applications

With the last decade’s advancements in information and communication technology (ICT) new possibilities have emerged in patient centred care. Mobile devices such as smart phones provides accessibility superior to that of stationary terminals making them able to fit seamlessly into the patient’s everyday life. The power of these devices can also handle user-friendly graphical user interfaces (GUI) as well as connected peripherals such as a step counters or a blood glucose sensor. These features gives a vast variation of possibilities for patient centred applications. Many such applications already exist for different groups of patients and for a diversity of purposes (Lafferty 2012). One application helps with medication to schedule (Silva, Moutham, and Saddik 2009), remind and inform about the patients medicine another is an application designed to help managing side-effects from chemotherapy (Weaver et al. 2007).

2.2 Communication and authentication

In this section a brief overview of a couple of ways of handling secure communication between users and systems over HTTPS will be presented.

2.2.1 Authentication and password fatigue

As users today often need to access several independent software systems they also need to authenticate for each system. To ensure that this authentication remains secure users are encouraged to have different passwords and IDs for each accessed system. This is a cumbersome task which leads to password fatigue where users have a problem differentiating between the different authentication processes. Which in turn leads to calls to help desks, interruption in the user’s workflow and the use of same or very similar passwords on many systems (Dennis 2013).

2.2.2 Single Sign-On architecture

To combat this an architecture called single sign-on is used that enables the users to access several systems while logging in only once. As stated above this way of access is beneficial both for the users and the system owners as the user needs to spend less time authenticating, recovering forgotten password or memorizing new ones and the system owners get less calls to help desks and thus free up resources. However this also increases the damage of a security breach where the intruder gains access to all connected systems similarly to the
case where the user had used the same login credentials on all systems. This calls for systems designed with security in mind (Groß 2003).

2.2.3 Security Assertion Markup Language

Security Assertion Markup Language (SAML) is a solution by the OASIS Security Services Technical Committee (Madsen et al. 2005) where authentication and authorization data can be exchanged between systems and is one of the most widely used way of implementing SSO. In SAML terminology the systems providing the user with a resource or service are called a service provider (SP) and the one handling the user identities and authorization is called an identity provider (IdP). A SP trusts the IdP to handle the authentication and several SPs can decide to use the same IdP(s) for their authentication process and then also allow for a login through any of the SPs to be valid for all SPs. Furthermore SAML also allows for this kind of communication between organizations. Such collaboration is called a federation and requires mutual trust between all parties.

2.2.4 OAuth2

OAuth2 is a an open standard for authorization and used to securely delegate access. It does not handle authentication but works well in extending an authentication framework. OpenID Connect which is another way of handling SSO is built on top of OAuth2. In a similar way OAuth2 can be used in conjunction with SAML to create an easier way of integrating to the SAML framework, creating a robust way of handling SSO. The main benefit of OAuth2 is it’s simplicity and this is why it is generally used with consumer application. It is based on HTTP which makes it optimal to use for native application that cannot integrate as easily with SAML.

2.2.5 Nationwide Medical Information Architecture

In Sweden there is a nationwide architecture to facilitate secure communication between different healthcare systems and across regions called "nationell tjänsterplatform - NTjP" (english: national service platform). The goal of this architecture is to simplify the communication between different healthcare systems. It is essentially a hub where parties of the healthcare community connects their systems to achieve interoperability. This is done by standardized communication templates called "tjänstekontrakt" (english: service contracts) that each healthcare provider or consumer needs to conform to. Another part of the NTjP is an index that catalogues what healthcare system(s) a patient has information stored and thus decreases the overall network traffic required to get the desired information.

An example of a service that uses this platform is "nationell patientöversikt, NPÖ" (english: national overview of patients) which allows for caregivers with consent to access a patient’s information from any connected healthcare system. It uses several service contracts to get such information.
2.2.6 API Gateway

Another goal of this architecture is to allow for patients to access their own medical information without asking each system they are registered at. There exist a service that allows users access to this platform through third party applications. This feature is known as citizen services (swedish: ”Invånartjänster”) but will henceforth be known by its technical name - API Gateway. The communication between the third party program and API Gateway uses OAuth 2.0 and SAML 2.0 between API Gateway and NTjP. The IdP uses the Swedish electronic ID (e-ID) which is available to all Swedish citizens and is used by banks and government agencies among others. The connection to API Gateway requires a two-way authentication over HTTPS and a third party server that handles some of the OAuth access flow. This will be described in further detail in the prototype section (4.2).

2.3 Patient empowerment

There is a general consensus in the research community, backed up by a growing body of evidence, that if patients have a deeper understanding of what afflicts them they will cope with their illness better (Mougiakakou et al. 2009). By understanding the treatment, the disease’s trajectory and the cause of the disease the patient will feel in control and thus psychological distress that often accompany a diagnosis can be reduced significantly (Lemire, Sicotte, and Paré 2008). Likewise, if patients makes lifestyle changes based on their own beliefs rather than just doing what the health care personal (HCP) tells them more often make long-lasting lifestyle changes. This is the essence of patient empowerment and is believed to be a key component in the care of chronically ill patients who’s care mainly consist of self-care. A common contradiction to patient empowerment is HCPs trying to empower patients by telling them how to change their lives without the patient understanding the scope of the illness and thus being less motivated to make lifestyle changes. Resulting in HCPs often feeling frustrated by failing to persuade patients into compliance while patients may feel they cannot live up to the HCP’s expectations and feel blamed by them (Anderson and Funnell 2010).

2.4 Patient information

Patient information is an integral part of healthcare. As health care personnel (HCP) gather, analyse and act upon this information to provide care for patients. This includes all healthcare related information from the first contact occasion until a resolution and ranges a variety of different types of information. Such as lab results, diagnoses, contact information, scheduled treatments and visits, drug list, social status, responsible HCP and much more (Henwood et al. 2003). It is vital that all caregivers to a patient has the same patient information. Therefore HCP needs to have easy access to patient information in order to provide the best possible care and limit healthcare related incidents. To
achieve this the information is stored and organised in patient specific medical records and other modules that together forms the health care systems.
3 Method

This section covers the proceedings of the development of the prototype.

3.1 Procedure

During the development of the prototype the software development model used was "iterative and incremental development". Which allows for developing parts of the software while iterating through number of smaller steps in a cyclic manner. It essentially goes through the following phases: planning, implementing, testing and evaluating, see figure 1. This is particularly favourable when there are parts of the design process that is uncertain in the start and an agile approach is desired (Larman and Basili 2003).

![Development Model](image)

Figure 1: The development model used when creating the SETHER proof of concept prototype. The process starts from requirement elicitation and then enters a cycle of iterative development in steps until the prototype is ready for deployment

3.1.1 Initial planning and requirement elicitation

The goal of this step is figuring out the requirements of the proof of concept prototype and looking at previous work.

The overarching prototype requirements was decided together with the job initiator, Cambio Healthcare Systems, where secure authentication and having SSO was deemed to be essential to the app. It was also decided to investigate the possibility of using API Gateway to connect to healthcare systems. The patient app was decided to be done in HTML5 using AngularJS and the server in Node.js using javascript.

This was followed by researching other patient applications by reading scientific articles that described the application, the experiences during the development and the patient’s feedback. We were also shown previous work pertaining patient applications that had been done internally in Cambio Healthcare Systems.
In hindsight this contributed little to no information that was needed in the making of the prototype as creating a patient application was considered out of scope. However it did provide insight into the bigger picture which could be considered important to have when creating a specific component.

Since the goal of this prototype was to be a proof of concept for a sharp, real world application the judicial aspects needed to be taken into consideration as well. Therefore pertinent Swedish laws were studied where ”Patientdatalagen” -(PDL) is the most central one. In addition to this documentation of API Gateway were studied as well as the national service platform that API Gateway connects to.

3.1.2 Planning

In this stage the of the iterative loop it is decided what part of the software will be developed next. By looking at the requirements and previous evaluations design a sensible plan.

Each goal was divided into smaller sub goals if possible. And what the next goal was to be implemented was decided here. For example creating a node.js web server that connects to API Gateway through HTTPS with mutual authentication using bankID was not done in one step. Instead the first step was calling API Gateway through a simpler HTTP request without any certificates or authentication.

In the next cycle when the result of the carried out plan had been evaluated a new plan was picked. Either increasing the complexity, dividing it to even easier tasks or picking another goal to work with. Depending on what the evaluation phase had concluded.

3.1.3 Analysis and implementation

The goal of this phase is to implement the plan from the previous step. In order to do this it is sometimes needed to analyse how it should be implemented. For instance in the example above the HTTP call could be achieved with a simple CURL request from the terminal and the implementation would then be to carry out that assignment. Most often it would include coding a node.js web server or the web application in AngularJS.

3.1.4 Testing

The main goal of this stage is to measure if the design implementation yields the expected result.

The testing in this project was straight forward. Most iteration was supposed to yield a certain result and if this occurred the testing was then deemed successful. Furthermore different browsers were tested (Firefox, Chrome and Internet Explorer) and the server was run both locally and on a cloud platform (Cloud9). One great tool used to track the communication in the browser was the Firefox addon ”SAML tracer” that made following the different calls easier.
During the development process contact with a developer from Callista Enterprise, the developers behind API Gateway and the national service platform, was also formed and he provided access to their quality assurance (QA) environment and issued a certificate to be able to test the prototype using the highest security requirements. He was also able to follow the flow on their end to further acknowledge the result. At times there were issues on their end that caused our tests to fail. At those times it was invaluable to have a contact on their end.

3.1.5 Evaluation

In the last phase of the cycle the test results are analysed to show what can be improved in the next iteration.

The evaluation phase usually pertained reading through debug messages, logs or documentation and some searches online for other developers experiences with the same issues.

When the prototype lived up to all the requirements the next step was deployment.

3.1.6 Deployment

The deployment phase is the last part of the development process and is not a part of the iterative loop. The software is handed over to the customer together with necessary documentation.

The deployment phase consisted of a demonstration for the supervisors at Cambio Healthcare Systems and presentations for other employees of the company.
4 Results

In this section the design of the model will be presented as well as the prototype created as a proof of concept.

4.1 SETHER Model

The SETHER(SEcure Transfer of HEalth Records) model consists of four components groups: Apps, App Server, SPs and IdPs. The communication between the apps and the SP is done with OAuth 2.0 and uses the App Server as an intermediate. The communication between the SPs and the IdPs is done with SAML 2.0.

Figure 2: An overview of the SETHER Model and the components internal relations. To the left is a description of the number of entities in each category of components. Any number of apps (of any device) connects to a single entity called App server who in turn connect to any number of service providers (SP) that each is connected to at least one identity provider (IdP). All SP and IdP are members of the same SAML Federation to allow the apps single-sign on. From the apps to a SP, through the app server OAuth is utilized.
There can be N number of apps all connecting to the same App Server and through it reach N number of service providers that is connected to M number of IdPs. However to achieve true SSO all SPs and IdPs needs to be in the same SAML federation, see figure 2.

The model is designed for communication between a patient application on a mobile device and healthcare system(s) in mind but is more generalizable than that. As the communication between app and App server is HTTPS it could also be implemented on a desktop computer. Likewise the model has no requirement of the SP to be connected to a healthcare system, it could as easily be a bank or a government agency.

The use of the national service platform (NTjP) is a clever way of connecting all Swedish healthcare systems but is not a requirement for the model to work. This model could also support that a specific healthcare system could be connected to a SP without relying on the NTjP to have a service contract for a certain feature.

The model also supports the notion of apps sending data to the SPs. The model supports two-way communication.

4.2 SETHER Prototype

The SETHER proof of concept prototype is an instantiation of the SETHER model, see figure 3. It uses only one app and one SP but as stated in the limitations section fully proving the model would be a time consuming endeavour and thus out of scope for this thesis.

To ensure the data is confidential all communication is encrypted using mutual SSL/TLS over HTTPS. The certificate was provided by an API Gateway representative.
Figure 3: Overview of the communication of the SETHER proof of concept prototype. The flow depicts a scenario where the end user of the App is not authenticated, requests data, authenticates using e-ID and receives requested data fetched from Healthcare Systems. The dotted arrows represents a HTTPS redirects.

4.2.1 App

In the prototype the App category is represented as a single web application written in HTML5 and JavaScript using the framework AngularJS and runs on Firefox, Chrome and Internet Explorer (only browsers tested).

4.2.2 App Server

The App Server is represented as a web server written in JavaScript using Node.js and the framework express.js and can be run both locally and remotely (tested on Cloud9, a cloud platform).

4.2.3 Service Provider - API Gateway, NTjP, Healthcare Systems

The prototype has one SP and it is represented as API Gateway, National Service Platform (NTjP) and Swedish Healthcare Systems as mentioned in the
limitations section using real healthcare systems, infrastructure and medical data would be irresponsible. Therefore the mentioned components is mock representations of these.

4.2.4 Identity Provider - CGI

The IdP is provided by CGI and is set up to be used through API Gateway. Multiple e-ID is allowed by the IdP but the prototype uses bankID which is the most commonly used e-ID in Sweden. It is used by government agencies, banks and companies and can be used as a mobile application. It has test users that was downloaded and installed locally and used throughout the project.

4.2.5 Login - first time

In this section a detailed description of the communication flow of the SETHER prototype is presented, see figure 3.

1. The chain of communication starts by App initializing a HTTPS request from App to App Server that contains what resources it requests. App Server then creates a unique string as id (uuid) and saves it in a cookie to identify the App user in the future. The state is saved with the cookie id as key on App Server (only in memory in the prototype).

2. App Server builds a call with the following parameters:
   - its redirect URI (for a later part of the communication)
   - a unique random string called "state" to prevent Cross-Site Request Forgery (CSRF) attacks
   - the different resources being requested by App (provided by in the first step)
   - a client-id provided by API Gateway, stored on the server

   The app is then provided with the cookie and redirected with the parameters above to the API Gateway.

3. App is redirected again to the CGI IdP by API Gateway for authentication.

4. This is the first thing the user sees in the login process and now gets to choose what kind of e-ID method to use. As we use bankID the user picks that and is then prompted to enter the credentials for the bankID test user.

5. BankID sends the credentials back to the CGI identity provider that acknowledge the successful authentication.
6. After a successful authentication App is redirected back to API Gateway. Parallel to this CGI provides API Gateway with a authorization token that is saved by API Gateway. The user is prompted to allow API Gateway to gather the resources specified in the first request.

7. When this is approved API Gateway will redirect the call back to App Server (with the URI provided in the first call) together with an authorization code. App Server will check if the state received in the call is the same it sent with the first redirect by looking up the cookie id.

8. App Server make a HTTPS call to the API Gateway with the auth code to be verified.

9. API Gateway creates an OAuth token and send it back to App Server who then saves it with the cookie id as key. The token has an expiration date when it will no longer be valid and be rejected by API Gateway and the login process must be redone.

10. After this App is informed that the login was successful and are now allowed to request the data resources it specified in the first request (in separate calls).

4.2.6 Login - subsequent times

In subsequent login attempts App Server checks if the cookie id already has a valid OAuth token and if so the App is told it is already logged in. No further actions needs to be performed, see figure 4.
Figure 4: Overview of the communication of the SETHER proof of concept prototype. The flow depicts a scenario where the end user of the App is already authenticated, requests data and receives requested data fetched from Healthcare Systems.

4.2.7 Fetching data

11. When assured by App Server that it is authenticated App makes another HTTPS request to App Server for a resource (ie. careDocumentation).

12. App Server fetches this information for App by requesting the resource from API Gateway together with the saved token.

13. If the token has not expired and is valid for this resource API Gateway then requests the information from the National service platform together with the saved authentication token (from CGI).

14. NTjP looks up an index, called ”Engagemangsindex”, with the patient id to see in what healthcare system the patient has saved data with the type of resource that was requested. Then it fetches the data from the matching healthcare system(s).

15. NTjP sends the fetched data back to API Gateway.
16. API Gateway de-identifies the information and removes unnecessary fields and then returns the information to the App Server.

17. As API Gateway uses SOAP and the App Server and App uses REST the App Server must convert the XML to JSON before sending it to the App.

The app can then parse the JSON and present the data for the user. In the prototype this was done by showing the raw data in a text field and a list below with selected data fields.

4.3 Technical test of the SETHER Prototype

The prototype was tested by running it towards a quality assurance (QA) environment hosted by Callista Enterprise that mimics their implemented system (API Gateway, NTJP) in order to test the prototype without exposing any sensitive data. The security measures implemented on the QA environment was as rigorous as on their live environment. The result was a successful retrieval and presentation of the mock data. All the steps described in the sections above, see figure 3, was monitored through logs, debug messages and network monitoring tools. This was also monitored by a developer from Callista Enterprise that could verify the steps on their end proceeded as expected.
5 Discussion

One of the main strengths of the SETHER model is its generalizability. It is unspecific when it comes to the implementation of the specific components but has an emphasis on the crucial parts: Federated service providers to provide single-sign on, identity provider(s) to authenticate the user, a single hub in the form of an app server to allow for many different users and devices to access the same system while increasing the security.

Another aspect is the constantly evolving digital landscape, having a more general model may not become obsolete as quickly when it is based on widely used, open and standardised communication processes such as OAuth and SAML. This notion of being generalizable also allows for additional services to be added but often the danger in doing so is greatly increasing the complexity of the system. This is probably always true to some extent but by using standards that every component must conform to this modular approach should scale well.

Another reflection that can be drawn from this is that the endgoal of the SETHER Model, secure transfer between healthcare systems and patient applications, is highly dependant on all services being correctly implemented and available. If the healthcare systems are not fully connected or blocks certain communications the overall use of the system is diminished. In a similar fashion if not all the service providers can trust eachother and form a SAML federation true SSO will not be possible.

Implementing the proof of concept prototype was a smooth process and the original plan was the one fulfilled in the end, few alteration was needed along the way. However it is important to remember that this was only a prototype that used no sensitive data and thus was not target of the thorough scrutiny these products usually is exposed to. This can slow down implementations, especially considering the possibility of interconnecting many systems. Although this is the harsh reality of the field the SETHER model could help by being a central nexus for different applications to utilize the system as one.

The programming languages used is most likely of little importance and the model has no such requirements. Node.js has a large, stable community with many useful extension that made creating the prototype smoother. Also using JavaScript in both the front-end and the back-end awarded a good experience, albeit a minor one.

5.1 Method

Developing the prototype incrementally meant making problems seem less and also being able to take advantage of any experience gain on the way. Whether this had an effect on the result or not is hard to say. If the final prototype would be implemented at the get-go it could probably work, especially for an experienced developer. However an experienced developer would complete the smaller steps quicker as well resulting in little difference. Doing the work in small increments also helps when there were some uncertainties whether the
services were compatible or not. Any issues would have been discovered early leaving a pivoting action as an option and little work done in vain.

This reasoning is also applicable on the research done before the actual coding of the prototype. It was possible to be done without some of it but a greater insight into the problem and the technology used increases the certainty of the design and the readiness to combat unexpected issues.

As mentioned earlier in the thesis it is also important to understand the whole system in order to implement the correct service. For a communication model between a patient application and a healthcare system it is probably advantageous to have some medical knowledge or know how that kind of data is stored and used. I found that my medical education I got when studying to become a nurse was of great help in this. It also helped when reading through scientific articles about patient involvement and other patient centred applications. Having practised as a nurse in the healthcare allowed me to have a better sense of the healthcare system. And if there were still anything unclear about the healthcare system I could ask employees of Cambio Healthcare System, the largest producer of healthcare systems in Sweden, where I resided during my work on the prototype.

5.2 The work in a wider context

The work of this thesis aims to empower patients by allowing them an easy way to access their medical data. The notion of patient empowerment through medical data is backed by research and patients but is still met by a high degree of scepticism by healthcare professionals. There is an ongoing debate about whether or not it is dangerous to the patient and if the benefits outweighs any risks such system would bring. Politicians, other decision makers and the continuing digitalization of the healthcare is pushing for this to further realize this system where patients can access their medical information and communicate with the healthcare. The ethical issue is mainly concerning if patients overall are capable of handling such information that the healthcare professionals today control.

It could then be argued that patients impotence to handle their medical information stems from an old system where they were not privileged to such information and thus have not had the chance to learn. The problem would then not belong to the new system where the patients are more involved but to the old system that denied them this power. It is, in my opinion, sad that an old system with a paternalistic view of patients has blocked the progress and digital evolution of patient involvement for so long but then it heartens me even more to see that despite all this it is becoming a reality. There are a plethora of possibilities that can vastly improve the patient’s power over their disease through digital means, such as this thesis has worked to implement. Some of these I will present in the next section.
5.3 Possible uses

Some of these examples are partly implemented in already existing application but their connection to medical records are often limited or non-existent. There is also potential in having several of these features in the same application working together.

5.3.1 Medication

A list of the patient’s prescriptions is retrieved from the healthcare system into the app. Information connected to the medication is then displayed in the app. Such as names of medicines together with their prescribed dosage, instruction for taking them, remaining amount on the prescription and when to renew it. The app could also include functions to show the patient when to take the medicine, give reminders and have a check-box for the patient to check when the medicine has been taken. Increasing the quality of life for the user (the patient).

5.3.2 Diagnosis

Codes to the patient’s diagnoses will be downloaded into the app and related articles from a reliable medical knowledge portal will be presented in the app. In this way the patient can read up on diagnoses in a time of his/her choice. Information about an illness could lead to better understanding and in turn coping of said illness that in turn leads to better health in the patient.

5.3.3 Appointments

Some patient groups have many appointments of different kinds to keep in mind. Such as appointment with HCP, treatments, scans, tests and scheduled self care. With these downloaded to the app it will be available to the patient to view or to integrate into their own calendar. Helping the patient get a better overview of their appointments.

5.3.4 Self care

In care, especially of the chronically ill, a large portion is self care. This might be to measure weight, blood-glucose levels or blood-pressure every day and record it somewhere. An app could help with this to remind to do the measuring and also be a place to save the data. The app could also send some or all of the collected data to the HCP. Another feature could be a journal to take notes during an illness to help remember it when giving a history to a physician.

5.3.5 On the ward

During a stay in the hospital an app could be of extra use. It display the activities that the patient will experience such as a planned CT scan, PT visit and meals. Together with the activities could be an explanation of what it
entails or further information, as for the meal it could display today’s menu. This would help the patient know more of it’s care and feel more participating.

Another feature could be that of sending messages to the nurses station. If the patient has a question or something to address a nurse that is not urgent a message could be sent from the device to the nurses station. This could help to lessen the number of sounded alarms in the ward and also help the HCP to send the appropriate staff or equipment for the call. This would hopefully increase the communication between HCP and patient while leaving the human-to-human contact intact.

5.3.6 Medical record

The patient’s own medical record would be partly or wholly visible in the app. This would help the patient to remember what was said during the last appointment with the physician, to see lab results when they come in and show close kin what afflicts them.

5.4 Barriers and limitations

The main barrier could be argued to be the fact that it is only a piece of a bigger system and thus is dependent on the other parts’ implementations. As mentioned several times previously the model, and in extension the prototype, uses standardised ways to handle communication and security and thus is easier to interconnect to other systems. There are also other aspects to consider and a few of them will be mentioned below.

5.4.1 Hardware and connectivity

As mentioned before the medical information is confidential and thus needs to be treated securely. This means encryptions which requires computational power on the mobile device and the server. This should not be a problem on modern devices but it is still worth considering. Another demand is a stable network connection during the transfer of data. Depending on the implementation of the application it could rely on being connected to the server every time you want to view data. It could also gather all data and save it locally which would lessen the requirement for being always connected but increase in memory demand.

5.4.2 Security

Using HTTPS with mutual SSL/TLS requires certificates. These are purchased and needs to be renewed regularly. If a SAML federation is used there needs to be trust between the parties and metadata needs to be shared.

5.4.3 Legality

There are laws concerning the handling of personal data which needs to be taken into consideration as well as other laws concerning access to medical data and
systems. The Swedish laws are, in my opinion, not designed for the digital world that is the reality now and thus is ambiguous at times. This could prove to be a frustrating barrier if it would be an issue.

The product would also most likely be classed as a medical device which would require it to have a CE mark.

5.5 Future work

The future work would entail to further implement the model with more service providers, setting up a SAML federation if needed. Connect and run it towards a real healthcare system to fully prove that the model works. The communication part of the prototype would also need to be further developed as more services are connected. For the system to have the desired use, empower patients, a mobile application needs to be implemented.
6 Conclusion

This thesis presented SETHER, a model for secure communication between patients and caregivers. It facilitates a highly sought after access and overview of patients’ own data by providing a secure conduit between a personal mobile device and healthcare systems.

The utilization of widely used and standardised protocols gives the user and the providers the benefits of single sign-on and enable the system to scale to incorporate more systems. This, however, comes with a challenge to keep the system modular and and rely on standardised ways of communication to keep down the complexity of the product.

The implemented proof-of-concept gives an indication that the SETHER model will be able to fulfill the need when developing future patient apps.

Future work entails expanding on what work has already been done in the prototype by connecting more systems to provide more users a larger set of tools that empowers patients in their everyday struggle with their diseases.
References


