bETHerCare

Nicolas Aymon (naymon@student.ethz.ch), Alexandru Dan (aledan@student.ethz.ch),
Betim Djambazi (betimd@student.ethz.ch), Johannes Gasser (jgasser@student.ethz.ch),
Yves Haberthür (yvesha@student.ethz.ch), Philip Jordan (jordanph@student.ethz.ch),
Mose Marius Müller (mosmuell@student.ethz.ch)

April 13, 2019

Contents

1 Introduction

1.1 KISS

KISS, *keep it short and simple*, is a association which provides neighbourly help for people based on time tokens. In our society, neighbourhood assistance is an important part of social integration and solidarity. KISS makes it possible to *give* and *take* the care which is needed by supporting each other in a simple and unbureaucratic way. The caregivers of KISS are rewarded with time tokens (stored in a time bank) which they can use in two different ways: Firstly, they can use these tokens for themselves as soon as they need help or care. It does not matter whether they use them directly or years later. Secondly, the caregivers should be able to give them away to anybody as a gift. The goal is to make KISS accessible for everyone easily without considering the age!

Issues

The main challenge is to save costs while improving people's health as well as their quality of life. Swiss society will have neither the human resources nor the funding to cope with the demographic development until 2050, which will bring Switzerland an annual cost increase of about 20 billion Swiss francs.

KISS is one of the answers to this challenge. It has the potential and ambition to become the "4th money-free pillar" of the Swiss old age provision system. KISS helps to cut cost and strengthens the community spirit as people can live at home longer and suffer less from loneliness. KISS members provide assistance with everyday tasks to each other (no medical care) and get time vouchers for their effort.

For the moment there are several issues which need to be improved:

1. *Regarding the organisation:*
   The Swiss political system is not familiar with time-based currency and every process is manual at the moment and takes a lot of time.

2. *Regarding the network:*
   Different organisations see each other as competitors. This fact unfortunately fragments the structure of social organisations.

3. *Regarding the funding:*
   The long-term funding of KISS is not ensured because of several factors. One of the most noteworthy factors would be the insufficient governmental subsidisation.

1.2 Time Banking

The time-based currency (time bank) of KISS should be a local agreement between the association and the caregivers for the provision of mutual benefits based on a moneyless exchange economy. It should turn into an organized form of assistance. This is realized by a credit or token system which corresponds to a time unit fulfilled by someone. The time-based currency should value the different contributions of each caregiver equally. As an example, in our system a caregiver provides a service for a service receiver and gets a certain amount of time, which corresponds to a specific amount of tokens. The amount of tokens gives the caregiver the right to claim the same number of hours he spent to receive for himself when he has a demand or, as a second option, will be able to give the tokens to anyone else by free choice.

2 Challenge and Solution

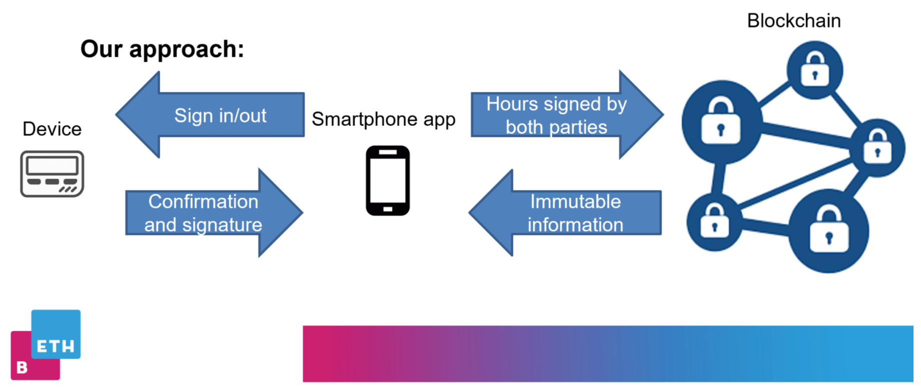
2.1 The KISS challenge

We want to design a sustainable and simple to use system like the one KISS is already promoting. We also have to provide security such that no one cheats in this system.

A further challenge is the ability to create and transfer tokens to KISS-users and to record their hours properly. It is also important to map or present the categories of the KISS-service catalogue to every user. Furthermore, our group should also balance properly between privacy and transparency of the given services.

2.2 Our Solution

Figure 2.1: Our solution consists of three parts: The backend, the mobile application and the device.



As seen in Figure 2.1, we divided our project into tho following three main parts:

The Device

We have built a device which is responsible for the time stamping. The caregiver logs on and off the service via interaction with the device. In addition, the KISS service receiver can indicate their satisfaction. We also invented a prototype badge to test our device.

The Application

We also designed an interface and developed a mobile application for the interaction between KISS-members. The application should be able to perform a new member registration. Furthermore, the clearly arranged interface should show the different kinds of offers which will be provided, just as in the KISS-catalogue.

Backend and Contracts

The third and last part of our project should realize the balance between privacy and transparency. On one hand, we want to keep activities secret from third parties in order to guard user data from exploitation. Therefore, we identify users only by their public key and we just log the time transactions onto the blockchain. On the other hand, the number of supplied hours should be accessible for everyone. This activity will be logged onto the blockchain and, by combining these guarantees with public key cryptography, we are able to provide authentication, non-repudiation and anonymity.

3 Design

3.1 Overview

Our design builds on three main principles: sustainability, usability and trust. We create a sustainable solution by providing independence from third parties via the distributed ledger technology. The simplicity of our solution is rooted into the simple application interfaces provided for our KISS members and the intuitive hardware design for measuring time. Finally, we provide bidirectional trust between the donors and receivers by using public-key cryptography in combination with a distributed ledger and IoT devices.

Trust is a main concern both for our users and the KISS project backend. On one hand, any caregiver needs to be able to verify that his work hours have been correctly and durably registered. On the other hand, the project coordinators at KISS need to make sure that a caregiver has done the advertised work. We discuss this aspects in section 3.3.

The next goal we defined was usability. We present an application that can be used by caregivers to both register user preferences regarding the assigned chores and to visualize time transactions in real-time. This is the first step towards reducing the paperwork needed in the association's pipeline by automation of the currently manual bookkeeping process of chores. The details regarding this application are discussed in section 3.2.

Finally, the interaction of the service receiver with administrative tasks should be both minimal and pleasant. We do this via an intuitive device that will be installed in the house of the service receiver. The purpose of this device is both providing trust in our application and allowing the service receivers to give feedback. We discuss the implementation of our device in section 3.4.

3.2 Application

3.2.1 Motivation

Currently, the registration and time management of KISS is done manually which naturally comes with a lot of paperwork.

To cope with that, we have tried to set up a user interface that on the one hand handles registration as well as administration of the services the user wants to provide as well as time banking, and on the other hand is easy to use. Our first idea was to build a website as a prototype. However, since we also wanted to use a device which measures the time (as described in section 3.4) we decided to approach our goal by building an app which can interact with this devices via a smartphone.

3.2.2 Approach

We decided to use the *React Native* framework for building the application. One reason is that React Native makes it easy to simultaneously develop for both *Android* and *iOS*. Another reason is that the applications can be built using only *JavaScript* and *React*.

Another objective was the ability of the application to present the services provided in a given period and the total amount of time the user has spent to help other people (comparable to *online banking*). The user specifies a period over which he wants to have an overview of the completed "transactions" (services rendered). The time period and the user ID is then sent as a request to the backend, which returns the information about duration and nature of the activity as well as name of the person who was helped of every transaction that took place during this time.

In order to be able to interact with the device of the person who is being helped, we also wanted to integrate a wireless connection tool into the app which can connect the smartphone to the device (*NFC*, *Bluetooth*, *WLAN*, etc.). This connection will be needed to authenticate the transaction described above: The application creates packets containing the information of the rendered service, but before putting it on the blockchain, it has to be authenticated by the device (this is explained in more detail in section 3.3.2).

Finally, we wanted to integrate the registration process into the app. Each user is assigned a unique ID, which is used to assign the transactions to the users and store them on the blockchain. This will allow the user to verify the transactions made and therefore create trust.

3.3 Backend and Contracts

3.3.1 Backend

The backend component sits between the application and the blockchain. Let us motivate the need for a backend in a decentralized application.

Firstly, the KISS project already contains a classical backend service which function both for administrative purposes and derivative data analysis. We observe that the analytics functions together with parts of the administrative data(e.g. personal details, affiliations, chore preferences, etc.) are still well-suited for storage in a centralized manner both due to cost and privacy reasons.

Secondly, providing a gateway interface hides the user from the need of interacting directly with the blockchain. This may be desirable since transaction costs would have to be supported by the client, the client needs a wallet, etc. Furthermore, via the cryptographic primitives described in subsection 3.3.2 we can achieve the exact same guarantees as if the client created a separate account.

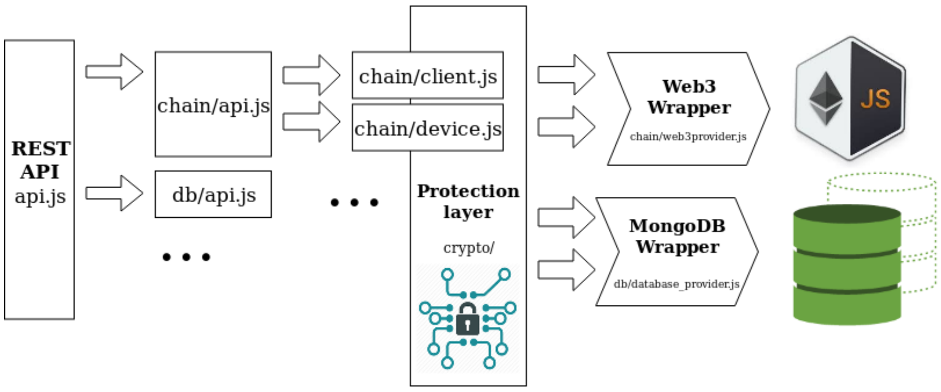


Figure 3.1: Backend components

Now, let us discuss the design of the backend. By taking a glance at the picture above, one can observe that the data flows from left to right. This simple principle allows us to decouple services and add intermediate layer processing components, such as the protection layer that is realized via cryptographic primitives.

The mobile application can interact with the backend via a RESTful API. The calls made to this API are then redirected towards more specific APIs. The components get registered in the upper-level API, so that redirection can take place:

```
export default class Api {
  register() {
    [...]

    new DBApi().register(this.server);
    new ChainApi().register(this.server);
  }
}
```

As an example, if a client wants to read the time balance from the blockchain, the request would be redirected to `chain/api.js`. The chain API would redirect the request to the particular file in which the handler sits, namely `chain/client.js`:

```
export default class ChainClient extends Web3Provider {
  [...]
  handleBalance(conn, publicKey) {
    this.handleCall(conn, publicKey, 'getBalance', parseInt);
  }
}
```

As we will see in the cryptogrphic section, each client is identified by its public key. What happens in the code above is that we want to call a method on the blockchain, namely `getBalance`, and then parse the result we get and write it back to the supplied connection `conn`.

```
export default class Web3Provider {
  [...]
  handleCall(conn, publicKey, method, ret) {
    let theId = `0x${new Crypto().hash(publicKey).toUpperCase()}`;

    this.owner.then(owner => {
      this.contract.methods[method](theId).call().then(ret).then(x =>
          {
          return { ans : x }
      }).then(x =>
        conn.send(x)
      );
    });
  }
}
```

The code above shows a part of the actual implementation of the `Web3 Wrapper` that facilitates communication with the *Ethereum* blockchain. The purpose of the example was to notice a few characteristics of our backend design:

- each class talks only to the lower and upper layers, making the design easily extensible
- each API component can filter a request if needed without damaging the message flow, making the design easy to secure
- the API request handlers form a tree, making the system easy to shard, hence scalable
- each upper-layer component can call methods on any low-level component (e.g. *MongoDB*), making the low-level component transparent, hence easily replaceable

A major advantage of our approach is the opportunity for short-circuiting. That is, the backend can short-circuit the calls to reading the blockchain, using the *MongoDB* component as a cache. The service is much faster than reading from blockchain, being able to provide the latest data. This means that the users have a much more interactive experience, being able to see their balance and latest transactions instantly.

The final point we make is that the backend design is easily adaptable to new features. As an example, we discuss how KISS could implement an opt-in service, that is, a user could make an *Ethereum* account and interact with the blockchain directly. Since each service is self contained, we only need to create a thread on the server reads the most recent calls on the contract and updates the information kept in the backend. This can be done only by using the components `Web3 Wrapper` and `MongoDB Wrapper`.

3.3.2 Cryptography

The cryptographic part is a central component of our service. The main goal of it is providing bidirectional trust between the server and the client by using the device. The workflow of the process uses public-key cryptography together with the blockchain's guarantees and is described in the figure below.

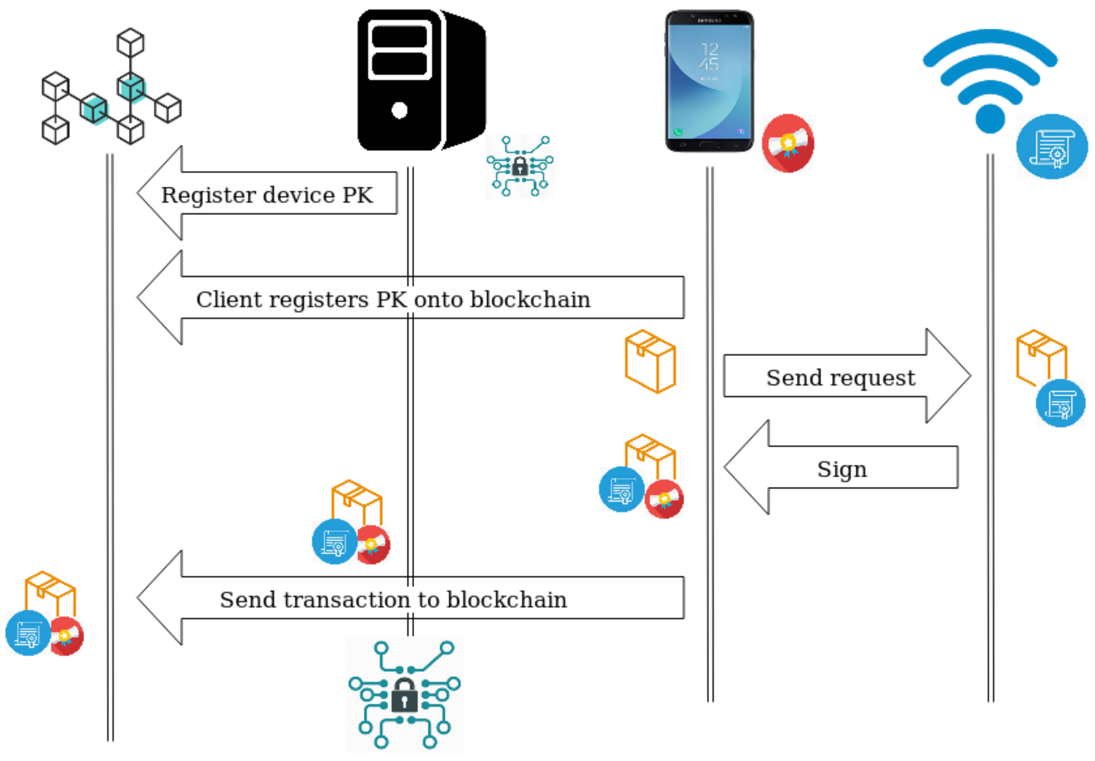

Register device PK
Client registers PK onto blockchain
Send request
Sign
Send transaction to blockchain

Figure 3.2: Cryptographic workflow

In the first step of the process, the device is registered onto the blockchain before being given to the service receiver. Now, each KISS member needs to register its public key with the backend, so that the contracts only keeps information relevant to the KISS project. These steps are represented by the first two arrows in the diagram above. Devices and clients are identified by a public key(denoted PK on the diragram) from a public-private key pair. Note that the key pair is generated at the respective client/device, hence the server does not know any of the private keys.

In the next step, the client (that is the Android application of the user) creates a packet containing its public key to send to the device for signing. The communication path can be any wireless channel. When the device receives the packet, it will wait until a button is pressed for finishing the session. At this point in time, the volunteering session has finished and the packet is augmented with the amount of time and a counter. The augmented packet(referred to as packet from now on) will be signed by the device and sent back to the mobile application. This is represented by the third arrow in the diagram.

At this point, the client has all the necessary information for publishing the transaction onto the blockchain. He also has the guarantee that the device did the correct job since its own public key is contained in the packet. Finally, to publish onto the blockchain the client will sign the packet as well and send it to the blockchain either thorough the backend(current implementation) or directly. The final data stored onto the blockchain is:

```
(clientPK, devicePK, amount, clientSignature, deviceSignature,
    counter)
```

The verification of the transaction can be done by anyone, and to reduce the data kept onto the blockchain the backend will verify the transaction before publishing. We mention that we also have a variant of the smart contract that verifies the transaction, but the initial contract(and the one we use now in the implementation) allows anyone to store a transaction onto the blockchain. However, all the needed data is onto the blockchain, hence all transactions can be verified by any 3rd party.

Guarantees

The protocol we described provides several guarantees:

- privacy - each client is identified by its public key, hence only KISS knows client's name
- authentication - only the client (the app) can claim the tokens since only the client knows the private key

- non-repudiation - only data signed by the device provides the integrity guarantee, fact verifiable by anyone
- freshness - time transactions can't be replayed since the device generates a new counter value for each transaction
- immutability - from the blockchain guarantees
- tamper-evident - from the blockchain guarantees and signatures

Possible attacks

We discuss a series of attacks our algorithm is resilient against:

- client replays transaction: not possible since the device is tamperproof and transactions are fresh
- server replays transaction: not possible since server does not have a device signature for the next counter
- server uses smart device to replay transaction: not possible since server does not have the client signature for the next counter
- rollback attacks: not possible since the counter only increases on the blockchain
- server excludes transaction: this is possible, but detectable from by the log of transactions stored onto the blockchain

However, our protocol is not protected against more powerful attackers:

- attacker has access to device, user and backend: a possible mitigation is employees not being able to make an account
- attacker that can tamper with device: a possible mitigation is making the device tamper evident

3.3.3 Blockchain

The blockchain part of our application is used as a tracker of *TimeTokens*. It should store the amount of *TimeTokens* for each person participating and provide a list of transactions that contain a proof mechanism such that both donor and receiver have accepted and validated this transaction. Additionally we want to store as less data as possible on it, since space on the blockchain is limited and we don't want to store too much data publicly because much of the data is sensitive and could be easily abused for criminal purposes. An other criteria is that organisations or even people directly could participate in the system and validate all the transactions without the help of KISS or any other organisations such that we build trust between all parties.

To solve these requirements we built a smart contract that contains the following data structures:

- A mapping from an address to a client object. The client object contains the TimeToken balance of the client and the counter used to validate that the same transaction is not made multiple times. Donors and receiver of services are clients in this data structure.
- A list of transaction objects. A transaction object is equal to a service a donor provided to a receiver and contains the addresses of the donor and the receiver as well as the amount of time, the counter we mentioned above and a signature from both the donor and the receiver. The signatures are made over a string that contains the donor of the service(his public key), the amount of time he spent and the counter.

The functionalities the contract has to provide are, the normal functions an ERC-20 token should provide such that something can be done with the token in the future, registration of clients, a function that adds and automatically validates transactions and normal getter functions that let users read the stored data.

The signatures now can be used as a proof mechanism such that everybody can always check whether a transaction is valid. An insurance company or other parties interested in this system for can easily verify that that both parties agreed that a service was done.

What the blockchain doesn't do is provide names or other identification for donors or receivers of services. In my opinion this wouldn't be a good idea since this data could easily be abused.

3.4 Device

3.4.1 Purpose

When it was clear which challenge we were into, we talked to the staff of KISS. They wanted an easy time banking system where they could track and measure time easily. It had to be simple to use for service receivers with no knowledge of smartphones etc. so the idea of a «easy to use device» came up. We decided that this device should measure the time and include a feature for the service receiver to give a feedback for the received service. To keep it simple, we wanted a badge system where the caregiver can badge in to start the time measurement. After the caregiver provided the service, he just needs to badge out again and the service receiver can press a button with a smiley face according to the quality of the service. This data – time and feedback – and a signature should be sent to the smartphone of the caregiver and from there onto the blockchain.

3.4.2 Build

After we had an idea what the KISS staff wanted, we came together and decided what features our device needs to fulfill these requirements. So building the device consisted of 3 challenges: First, we had to make the check in/check out function, then include a feedback option and finally get the data on the PC(later smartphone). We decided to use the following materials: an Arduino nano, 3 Push Buttons, one LED, Bluetooth module, *NFC* reader, *NFC* chip and some resistors.

Already in the first challenge we faced a problem: There were no *NFC* chips nor *NFC* readers so we decided to simulate this check in / check out procedure with a magnet and a hall sensor. Hall sensors recognize a near magnetic field so when we held the magnet near the device the sensor sends a signal to the micro controller.

The Push buttons were pretty easy to realize. We could attach them directly to the Arduino nano so whenever they get pushed, they send a signal directly to the micro controller. The LED was also really easy to implement. In series with a resistor we could attach it to the Arduino nano.

The *Bluetooth* module was another challenge we faced. It did not work properly, no matter what we tried. Normally you turn these modules on and they should be able to connect to devices such as smartphones but the modules we got were all broken or just very unintuitiv to use. We wasted almost a day with trying to get it to work but we had no luck. Later after the BETH challenge we tried to connect a different *Bluetooth* module and it worked on the first try. However, since the *Bluetooth* connectivity seemed not to be working we decided to use a USB cable connection to transmit the data to the PC.

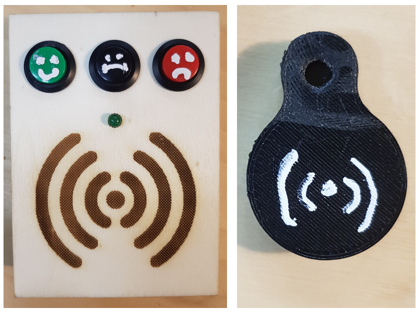


Figure 3.3: The device on the left and the small badge on the right

As soon as it was more or less obvious how the internal of our device should look like, some members of our team got to designing the case for our device. We designed it in Siemens NX and manufactured it with laser cutter out of 4mm Plywood. The badge was simply a magnet in a 3d printed case.

Writing the code was not a very big problem. The Code for the micro controller was written in the Arduino IDE. The received data from the Arduino nano was processed with the program «Processing 3» in which we could display the data(name of the service receiver, time passed and feedback) in a graphical user interface.

4 Evaluation

4.1 Discussion

4.1.1 Disruption Potential

In the following section, let me discuss and evaluate the disruption potential of the KISS idea implemented in our way as described earlier. In the economic sense, a disruptive idea is generally known as an innovation that creates a new market by applying new techniques or pursuing a novel design philosophy which then lead to the disruptive idea effortlessly outperforming the conventional approach. In recent times, leveraging new technologies arising from computer science, especially the internet, has often led to such disruptive businesses.

The blockchain, or more generally distributed ledger technology, is yet another technological invention that seems to bear great potential for giving rise to many disruptive ideas in all kinds of fields. Let me now come to the reason why this is also true for the field of health care and especially the idea of time banking.

If we think about what characteristics would constitute a well designed old-age provision system, two of the most important and somewhat related objectives that immediately come up are trustworthiness and independence from a central controlling entity. When trying to solve these problems using technology these days, a distributed ledger seems to be almost the obvious choice, since its very purpose lies in enabling a trusted way of decentralizing systems and still being able to achieve consensus among the parties involved.

Old-age provision can basically be seen as a agreement among society that younger people are obligated to support elderly people who rely on help. When the younger ones are old themselves it's up to them to profit from the support of the next generation. As we see this basic description does not include any controlling third party. Hence in an ideal implementation of such a system, it would be natural not to introduce any third party as well. Of course up to now, some central party has been necessary to ensure that caregivers are contributing their part to the agreement and not acting selfishly. Our approach using the blockchain as a tool providing trust has the potential to truly turn old-age provision into a contract just between generations as it is naturally meant to be.

The KISS idea implemented using blockchain technology thus has indeed high disruption potential, although this potential is not primarily in the economic sense as first described but rather disruption in terms of democratizing society by removing its need to rely on a third party to enforce its internal agreements.

4.1.2 Negative Aspects and Open Challenges

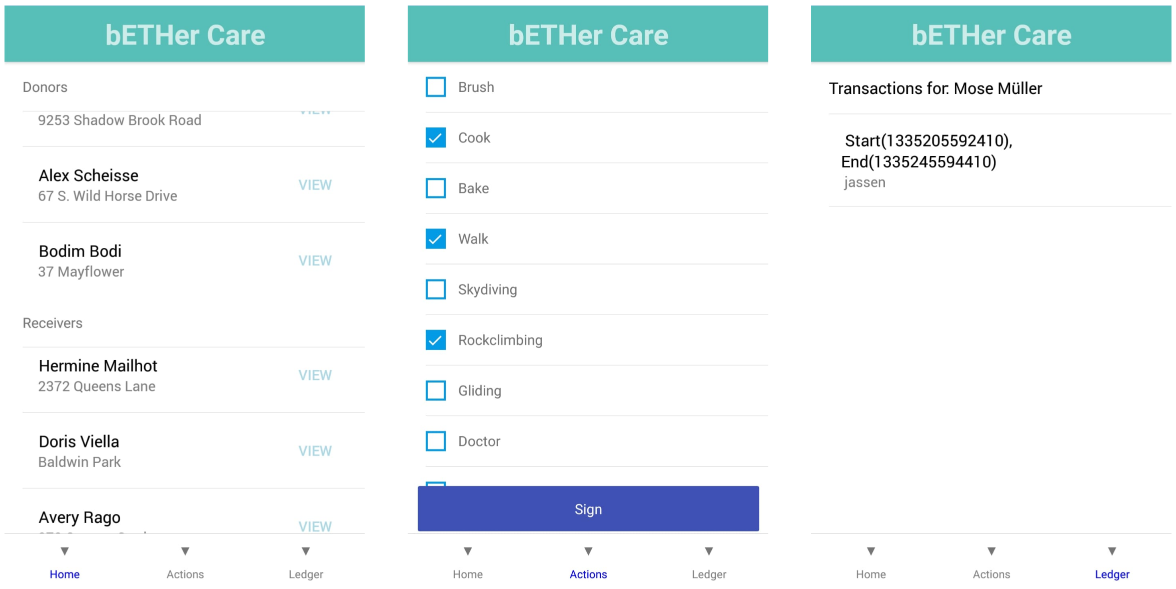
- The Source of trust is guaranteed but not easily understandable by the average person without a background in technology (especially hard for elderly people).
- Trust is only ensured on a high level, making sure that what gets recorded on the blockchain is what actually happens in the real world is a different story.
- In the current state of distributed ledger technology, transaction costs are relatively high. However this might improve with time since it's still a young, but actively researched, technology and could mature in the near future.

4.2 Current State

4.2.1 The App

The application has a graphical user interface which can be seen in Figure 4.1. Different features are organized via a simple menu at the bottom. The user can select and save activities he or she wants to provide (menu item "Actions"). This data is saved on the mobile phone and not connected to the database yet.

In addition, we have implemented a listing of caregivers and service receivers. These are displayed with the name as well as address (menu item "Home") and retrieved from the backend via an API request.


bETHer Care

Donors

9253 Shadow Brook Road

Alex Scheisse
67 S. Wild Horse Drive
VIEW

Bodim Bodi
37 Mayflower
VIEW

Receivers

Hermine Mailhot
2372 Queens Lane
VIEW

Doris Viella
Baldwin Park
VIEW

Avery Rago
VIEW

Home Actions Ledger

bETHer Care

☐ Brush
☑ Cook
☐ Bake
☑ Walk
☐ Skydiving
☑ Rockclimbing
☐ Gliding
☐ Doctor

Sign

Home Actions Ledger

bETHer Care

Transactions for: Mose Müller

Start(1335205592410),
End(1335245594410)
jassen

Home Actions Ledger

Figure 4.1: The graphical user interface of the application. The first menu tab lists all donors and receivers, the second tab enables to specify the services one wants to provide. The last tab lists all the rendered activities (duration and kind of activity).

We also implemented a view of the services one has provided in a given period. They can be accessed via the menu item "Ledger". We could further improve this by adding a form where the user can specify the period of time for which he or she wants to retrieve the information and in addition, display the total amount of time he or she has spent helping others.

In the attempt to connect the caregiver's smartphone to the service receiver's device, we implemented a *Bluetooth* interface into the app. However, since the *Bluetooth* module did not work properly on the device, we stopped working on this feature and later removed it from the app.

Unfortunately, realizing that we would not have enough time to get it working properly, we did not work on the registration part.

4.2.2 The Backend

For the backend, we built a first, working smart contract that provides some of the functionality we wanted to implement. What is still missing, is the automated validation of transactions signatures and the ERC-20 token functionalities. At the moment the contract is built such that we need an owner who does all the administrative part like adding transactions and users. In the future it should obviously be possible for each client to register and add transactions himself. Generally this first version of the contract can still be much improved.

Additionally we provided a version two of the contract which improves some of the faulty design we made in the first contract. We also tried to add the automatic validation of the transactions which we unfortunately couldn't finish in time.

Some parts of the interaction between the backend and the blockchain were implemented but haven't been tested in the end because we focused more on reworking the smart contract since the faulty design provided many problems to the interaction.

A large part of the cryptographic operations have been implemented into the backend API. Even though not all the operations have been implemented in the backend API, we provide fully-functional cryptographic prototypes in the `server/crypto` folder. The same primitives have been deployed onto the Android application for signing and interacting with the device.

4.2.3 The Device

The device has been designed to be as simple as possible, so that people with no technical knowledge can use it without being reliant on others. At the moment it is possible to get the information out of the device to the PC with a USB-port-connection. The idea was to get the information to a smart phone via *Bluetooth*, so that the service receiver don't need to do anything. The problem we had was, that we hadn't a *Bluetooth* that worked with our setup. We had to choose another way to show our prototype so we put a sensor that recognize magnetic field. As soon as a magnet goes through the magnetic field, the sensor send a signal to the control module and the time starts respectively the time ends counting.

The device is meant to be for checking in or out the helper and to measure the time he spend and for giving them an evaluation. The evaluations had also to be as simple as possible, so we decided to design a device with tree buttons. The green button means that the care was good, so they would like to meet them again .The black one means that the care was okay (neutral opinion) and the red one means that the care was bad, so they don't necessarily want to meet them again.

Currently the device doesn't work the way we wanted because of the limited hardware we could use, but the principle idea works. It's possible to check in and to check out and also to measure the time they spend together. The evaluation buttons work too. The only thing that does not work at the moment is the connection with the smartphone.

4.3 Issues

4.3.1 Mobile Application

At the first day of the hackathon, we decided to use the *React Native* framework for building mobile apps using *JavaScript* and *React*. This seemed to be a reasonable decision, since it not only supports cross-platform compatibility (*iOS* and *Android*), but it would also allow us to use *JavaScript* as the primary language throughout the whole project. Also *React Native* should make the design and implementation of a simple user interface for the app easier, because of the *HTML*-like declarative language and the large amount of predefined UI components available.

In hindsight however, going for a native *Android* app might have been the better choice because none of us had worked with *React Native* before. Setting it up for the first time was quite a mess and took us much longer than we though. Fortunately, we still managed to create the most important parts of the interface and could eventually even send and receive content from the backend and have it show up in the app.

4.3.2 Backend

The main issues we had in the backend as well as anywhere in the project was the lack of experience we had in the group. Nobody ever worked with one of the technologies we used in this project so we had to use much of our time to get to know the concepts behind the technologies such as the blockchain and smart contracts. Because of this lack of knowledge we had to discard many ideas or concepts we had because they were fundamentally wrong or just to time-consuming or complex to implement. This also reflects in our code. Most of the code we wrote was very messy because we tried to do something but then realized somewhere in the middle or even after we finished something, that it's not the right way to do it.

A good example for this would be the smart contract. The differences between the first and the second version are already huge and we could certainly still improve a lot on the second one.

Most trouble came from working with the `web3.js` library and from the interaction between backend and frontend. Since we had almost no blockchain experience in the group, we had trouble to grasp the concepts behind it, how the accounts, tokens, currencies, etc. really work and how we could build the interaction between our application and the blockchain. The issues we had with the interaction between the frontend and backend could be solved in the end but were still very time consuming for us.

4.3.3 Device

As it was mentioned before we had a problem with setting the device up with a *Bluetooth* or wireless. We bought a *Bluetooth* in project house, but for some reasons we couldn't connect it with Arduino no matter what we tried. So we tried with wireless and we had the same problem. The *Bluetooth* and wireless we had, were unknown, so we couldn't find anything in internet that could help us solving the problem. We also asked the people in student project house for some help but unfortunately they couldn't help us at all. The result was that we decided to evade this problem and did with magnetic field and a magnet.

To get the data to the smart phone was also a problem. We didn't waste a lot of time at this problem, because in the future it's not meant to get the data from the device in a computer and then from the computer to a smartphone. The device should be able to send the data directly to the smart phone via *Bluetooth* or wireless. The other problems we had were mostly not so big and we could solve them easily.

4.4 Future Improvements

In summary one can say that many parts of the intended system's skeleton could be built, like the backend API connecting to the blockchain and providing data to the application that also works and features a basic interface for viewing a time banking account and one's recently provided services. Additionally we managed to build a easy-to-use device that can be placed in an service receiver's home and is used to verify that services were actually provided. What is missing in this basic skeleton is the connection of app and the just described device via *Bluetooth* or *NFC* as well as some of the cryptographic functionality and the whole system could never be tested on an actual blockchain. Adding those parts would make it a complete proof of concept.

Apart from that, this is of course just the basic structure and for an actual deployable system, many more functions would have to be implemented, for example a registration form in the app plus corresponding functionality on the API provided by the backend.

5 Conclusion

In summary, one may conclude that the general idea of using the blockchain as the underlying technology to base a modern old-age provisions system on seems promising. Eliminating the need to rely on a trusted third party to control the whole process and manage the "time bank" blends well into the concept of old-age provisions as a contract between generations. Detaching agreements within society from a powerful, controlling unit would definitely make the world a *bETHer* place.

It does not come as a surprise that turning these abstract ideas into a working prototype during a hackathon while still learning how the technology actually works is not an easy thing to do. As described earlier, we have been faced with all kinds of issues when implementing the app and the backend with its cryptographic parts and the interaction with the blockchain, as well as when building the physical device as an easy-to-use interface. Besides that, we still achieved some solid results, since we were able to build a working subset of the described system including communication between our app and the backend. And if nothing else, we all learned a lot from this intense week, because almost none of us had ever worked with any of the tools we have been using before.

To sum it all up, what starts with a promising idea in theory, as always turns out to be harder than expected to implement in practice. However, this project did result in a basic working skeleton for the described system and can, in our eyes, be seen as a solid basis for future work and improvements.