# Architecture design

**BBW**
Jasper Hu, yjhu, 4356241
Naqib Zarin, nzarin, 4384474
Ashay Somai, asomai, 4366220
Ymte Jan Broekhuizen, yjbroekhuizen, 4246586
Luat Nguyen, tlnguyen, 4467574

June 2, 2017

# Contents

**1 Introduction** **3**
- 1.1 Design goals . . . 3
- 1.2 Availability . . . 3
- 1.3 Manageability . . . 3
- 1.4 Performance . . . 3
- 1.5 Reliability . . . 3
- 1.6 Scalability . . . 3
- 1.7 Securability . . . 4

**2 Software architecture views** **4**
- 2.1 Subsystem decomposition . . . 4
- 2.2 Hardware/software mapping (mapping of sub-systems to processes and computers, communication between computers) . . . 6
- 2.3 Persistent data management (file/ database, database design) . . . 6
- 2.4 Concurrency (processes, shared resources, communication between processes, deadlocks prevention) . . . 6

**3 GUI** **6**

**4 Glossary** **7**

# 1 Introduction

This document provides a high-level description of the final product and the system it uses. The end product exists out of a mobile application in the Android environment, which is developed in the Java programming language. Using the Android environment and the programming language Java, we are developing a blockchain-based web-of-trust. The idea is that you use blockchain to verify the authenticity of the user. Blockchain itself is like the name says a chain of blocks and each block contains the information to verify a user, normally this is a public key.
The Android environment is an open-source platform and operating system, which is developed by Google. This operating system is available for many devices. However, we are focusing on the mobile devices.

Hereafter, the paper will be composed of the following sections. The next section will be the design goals of this project. In the third part, there will be more information about the software architecture views of the project. Specifically, these views are Subsystem decomposition, Hardware/ software mapping, Persistent data management and Concurrency.

## 1.1 Design goals

Since the concept design goals are too large, it is easier to split it in six different aspects: availability, manageability, performance, reliability, scalability, and securability. Consequently, these six aspects are elaborated in the following sections.

## 1.2 Availability

The idea is that there will be a working version after every sprint week. This way, the product owner can see the features, we are working on, every week. So in case, we are going the wrong way, we can still turn to go the right way during the meeting.

## 1.3 Manageability

To manage our program, we use the version control program Git, and it is stored on GitHub. In case there is an available update, you can just pull the working release version from the master and use that one instead. Our working or developing branch is the branch 'develop'.

## 1.4 Performance

The program should be able to be processed on a mobile device, so the blockchains that the program is using should not be too large. It should even be limited to a specific capacity, which could be a part of the capacity of the device. Since most of the user would like to use their mobile phone for other applications as well, it would be good to let the application not exceed about 2GBs of memory.

## 1.5 Reliability

To ensure the reliability of our program, we use several techniques. The first technique that we are using is Test Driven Development. This technique increase the probability that we completely understand what a specific feature should do. Since it improves the understanding of the feature, we can develop it more reliable and more efficiently. Testing is done using unit tests, integration tests and regression tests.
The next technique that we are using is Travis Continuous Integration. After every commit to our program, or in other words, after every update to our program, the program is automatically tested. Consequently, it will let you know, whether the update contains any errors.
Testing is important because it helps to find bugs and preventing unwanted behaviour.

## 1.6 Scalability

The whole program is decentralized to a mobile application. The persisted data is saved on the server and the server is saved on the device as well. The calculation of some hashes only require

little processing power, so there won't be a bottleneck on the processing power. The only problem is that you have to save the blockchain to your device. So the only limit would be storage capacity of the device. However, a single block uses less than 1 MB of storage space so that the limit would be colossal for the current phones. There are no other bottlenecks, so the program scales pretty well. The program runs smoothly on multiple devices, like Chinese and Google Android devices.

### 1.7 Securability

Since the persistent data is stored in an SQLite file on the device itself, the first layer of security is the security layer of the operating system itself. The second layer is the password protected SQLite file. The third layer of protection is that the (valuable) information, like IBAN or contact information of the owner of the blockchain, in a block is saved using the AES-CBC encryption protocol. This encryption protocol ensures that the data is not readable directly, but should be decrypted first, when read out of the database. Likewise, the data should be encrypted first, before it is stored in the database. This is not a impenetrable layer, but will make it much harder to read out the data.
If all layers are penetrated, there is still an option to revoke a block. This way the block, containing information about another user will be invalid. Next to that, every block contains a trust value, representing the trustworthiness of a user. All actions on a block influence the trust value, like verifying your IBAN, successful/ failed transactions and revoking your own block. The trust value represents the trustworthiness of a block and is initialized to 20; this is an integer between 0 and 100. This value will decrease with 10, when you have a failed transaction or will be set to 0 when you revoke a block. Consequently, this value will increase with 10 upon a successful transaction and set to a high initial value of 50 if the IBAN is verified. Revoking a block can be done using the interface, where you select your block and press revoke afterwards. If you revoke a block, in the future it will send a message that the block will be invalid, so the others can revoke it as well. The trustvalues for initalization and mutation have been chosen on random and still have to be calibrated.
Finally, you can verify a block that is sent by someone else using backtracking. Using the previous hash of the sender, you can backtrack to the original owner to verify its legitimacy.

## 2 Software architecture views

The program is interconnected using different parts and the software architecture views elaborate on them. In the subsystem decomposition, it shows how the system is divided into subsystems. Secondly, in the hardware/software mapping, it illustrates how the hardware subsystems are connected to the software subsystems. As third, in the persistent data management, it shows how and where the data is persisted. And finally, in the concurrency part, it shows how the resources, processes are shared and how the communication works between them.

### 2.1 Subsystem decomposition

The client uses the Mobile Application to handle all requests. Or in other words, communicates with the server using the mobile application. This mobile application also processes all requests. The mobile application consequently communicates with the database using the Android SDK. Since the mobile application processes all requests by itself, there is no other communication between the database and the mobile application.

- Models
  All persisted data is represented by a class or multiple classes. It is not efficient to continuously make requests to the database to get a block or the whole blockchain. Our code exists of the following models:
  - Block
    The Block class represents the data that is saved in the database. This data includes:
    * Owner of the block and thus the blockchain
    * The sequence number of the block
    * Its own hash value

    * The hash value of the previous block in the chain and if it is the first block, it is an empty string
    * The hash value of the previous sender and if it is the first block, it is an empty string
    * The public key of the block
    * The IBAN of the block
    * Trust value, which represents the trustworthiness of that block.
    * Block type, which indicates the block type: genesis, add key, or revoke key.
  - Subclasses of block
    The three block types all have their corresponding class: GenesisBlock, AddKeyBlock and RevokeBlock. They all check their fields for correctness in debug mode: GenesisBlock always has sequence number one and previous hashes "N/A" (not available) for example.
  - BlockData
    Since block has many fields so that the constructor became large and unwieldy, an intermediate class BlockData has been introduced which acts as the data storage for blocks. You can set any field in peace, and when you are done simply pass the BlockData object to the blockFactory to instantiate a Block.
  - BlockFactory
    The BlockFactory class is responsible for instantiating blocks of the correct subclass on basis of a block type and some block data.
  - BlockController
    The block controller is the core class of our block chain. It provides an interface for adding or revoking keys without going through the hassle of talking with the back-end database. In earlier designs you had to create the block and add it manually, while the current create methods automatically fill in correct values for the block so that it is inserted correctly inside the block chain.
  - DatabaseHandler
    This layer converts requests within the functionality to requests to the database. Every action to the persisted data requires a request to the database and thus also to the database handler to able to communicate with the database.
  - User
    The User class represents any user inside the system. It stores the name and IBAN of yourself and contacts and can be passed around to be stored inside of a block. At this time it is constructed when necessary from a name and IBAN, while in the future it will be inserted in the database as to keep track of a consistent user collection.

- Graphical User Interface
  This is the layer that the user sees. He interacts with the program using this layer. For example, he can add a block, revoke a block, or see its blockchain using buttons, which are implemented in this layer. This layer itself does not contain any functionality; it only links the interface layer with the functionality. There are different views for adding, revoking a block, displaying a blockchain. The view that controls those views is the MainActivity.

- Controllers
  The controller handles the requests from the Graphical User Interface (GUI) layer and processes it. For example, if there are requests from the GUI, it will process the request itself or send it to another controller. If the request makes use of persisted data, it will handle this as well. The controller, that is responsible for all actions on a block, is the BlockController. And the controller that is responsible for the calculation of the hashkey is the ConversionController.

- Database
  Since the data needs to be persisted, a database has been used. The client uses the persisted data from the database to handle any other requests.

- Api
  As a service to other groups we have decided to create a simple Api for our application. The Api lets you view keys from a certain user and add and revoke keys at will,

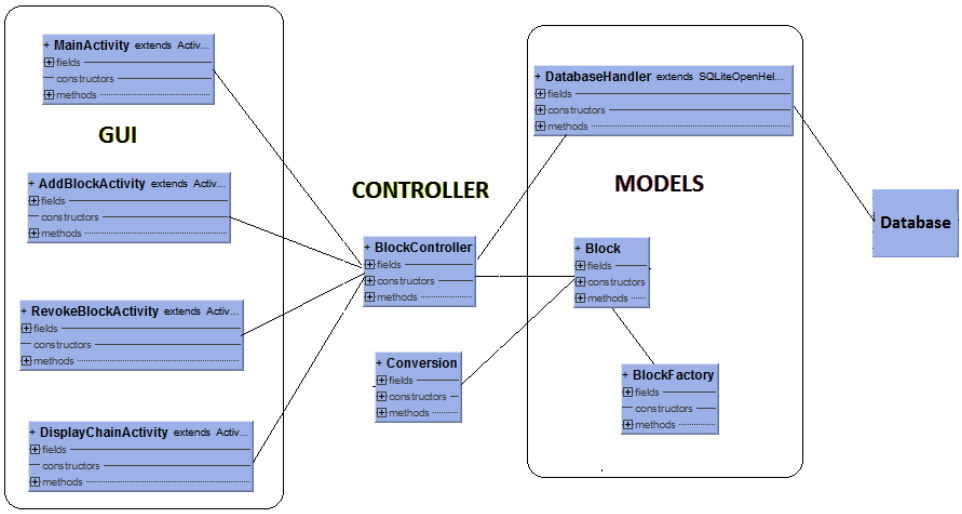


Figure 1: A graph of the interaction between the client and the database.

### 2.2 Hardware/software mapping (mapping of sub-systems to processes and computers, communication between computers)

The mobile application runs on a mobile device, which has to use the Android operating system. The mobile application makes use of the Android SDK to connect to the database. The database also runs on the mobile device itself. The Android SDK uses the SQL language to communicate with the database.

### 2.3 Persistent data management (file/ database, database design)

The blockchain of a user is persisted in the SQLite database since we do not want to lose the data upon closing the application. The blockchain consists of blocks, and since the blockchain is persisted, the blocks are persisted too. Next, to that, The general information about the user, that we need for later, is saved in the database as well.

### 2.4 Concurrency (processes, shared resources, communication between processes, deadlocks prevention)

The users of the blockchain will have to communicate with each other, using a communication protocol. Currently, we are simulating a communication protocol, but in the future we will use a Bluetooth protocol. Every user keeps their blockchain, and every other user can see this blockchain. The other user now can add the initial user to his own blockchain, and extend his chain. The users should also be aware of their own keys. In case these get stolen, the user should revoke it.

As for the shared resources, the idea is to keep everything decentralised. This means that there is no central server which keeps track of the blockchain; the central server is the device itself. Every user keeps track of their own blockchain, and as stated earlier, the communication between users gives users the opportunity to know about the blockchain of others.

## 3 GUI

When we open the app we will enter the so called main page. In this page we will have three options. First, we can go to the contacts page where a list of contacts and some more information will be displayed. The second option is to pair with a new friend. The third (temporary, for testing purposes) option is to clear the database. When we click on the pair button, we will see an overview of contacts that we are able to pair with (see figure 3). When we click on a person, say

TestSubject1, we go to the Friends Page (see figure 2). The Friends Page will contain the name of the person you just connected with, his or her public key and his or her iban. These values will be retrieved once those blocks are made (when clicked on that person in the Pair Page) and passed on to the Friends Page. Also his trustworthy rating will be shown and finally, you will have the option to add this person to your contact list or visit his contact list.

Figure 2: A display of the Friend Page. Figure 3: A display of the Pair Page.

# 4 Glossary

- AES
  AES is an abbreviation for Advanced Encryption Standard and is an encryption protocol.
- Android
  Android is an open-source operating system, which is developed by Google.
- Android SDK
  Android SDK is the software development kit, which is used for creating applications for the Android environment.
- Block
  A block contains information about another user. This information could be hashed.
- Blockchain
  A blockchain is a chain of blocks, of which a block contains the hash of the previous block.
- Bluetooth
  Bluetooth is an open standard for wireless connections between devices on a short distance.
- CBC
  CBC is an abbreviation for Cipher Block Chain. This protocol ensures that the data is divided into several blocks using a blockchain structure with a XOR (exclusive or) to the previous block.
- Cryptographic hashing function
  A cryptographic hash function is a hashing protocol, but it has some more properties, which ensure more security. For example, it is a one-way protocol; it is not feasible to find the input of the data using the result of the hashing function.
- Git
  Git is a way to control the versions of your program. You could update, delete or edit your version of the program using this.
- GitHub
  GitHub is a website that allows storage and parallel development of code for programming teams.
- Google
  Google is a large IT company which mainly focuses on developing software and hardware.
- Hash
  A Hash is a protocol to map any information to a particular arbitrary size.
- IBAN
  IBAN is an abbreviation for International Bank Account Number.
- Integration test
  An integration test tests whether multiple components can cooperate as expected.
- Java
  Java is a object-oriented programming language for developing software.
- Middleware
  Middleware is the layer that enables a subsystem to communicate with another subsystem. One of these subsystems does not have to be in the application itself.

- Native application
  A native application is an application, which is built with the SDK of the device and therefore is operating system dependent.
- Open-source
  Open-source means that the source code of a program is available to use to the public.
- Public Key
  A public key is one of the two keys used in cryptography to encode information.
- Regression test
  A regression test tests whether new bugs have been introduced due to updated code.
- Server
  A server is a remote computer, which could be used to persist data or handle any other requests.
- SHA-256
  SHA-256 is a cryptographic hashing function of the second SHA family.
- SQL
  SQL stands for Structured Query Language and is used to make mutations to a database.
- SQLite
  SQLite is a database engine which works with a SQL database.
- Test Driven Development
  Test Driven Development is a way of developing software, of which you write the tests first and build the program around it.
- Unit test
  A unit test tests a single component to verify its behaviour.
- Web application
  A web application is an application in the form of a website.
- Web-of-trust
  Web-of-trust is a concept in the cryptography to create the trustworthiness between a user and its public key.