CS4160 Blockchain Engineering

Report

Project H: Build your own Storagecoin

Dereck Bridie
David Gómez de Segura
Paul van der Knaap
Roelof Sol
Mitchell Olsthoorn

February 2, 2018

Chapter 1

Progress report

1.1 Initial project description

The title of the project was “Build your own storagecoin”. This concept is based on a state-of-the-art project named Filecoin [1], which aims to make people host files using their unused storage in exchange for ”filecoins”. This currency would then be traded on a number of exchanges (US Dollars, Bitcoin) and supported by multiple cryptocurrency wallets.

Tribler is a technology developed at the TU Delft. One of its upcoming features is that it can mine reputation by means of uploading and downloading content. Also, there is an Android application that implements the TU Delft style blockchain called TrustChain.The goal of the project was to change Tribler and the app to add these new features:

1. Use QR codes for offline one-way communication and transfers.
2. Transfer tokens from Pc-To-Android
3. Transfer tokens from Android-To-Android

Additional challenges included cooperating with other groups that concurrently worked on new features of the application. Working on a single application that included all the features the different groups was the ultimate goal. Therefore, this project also included maintaining proper communication with other groups to determine and detect possible conflicts and to align development plans to ensure that no unfixable conflicts would arise.

1.2 Project development

From the very first meeting, it is set clear that the main work will be developed around a native TU Delft android app called ”TrustChain Android” [2]. This app was built as part of the Blockchain Engineering course and implements

Trustchain. The project is also highly related to Tribler [3], an open source decentralised BitTorrent client which allows anonymous peer-to-peer connections.

During the discovery phase of the project the first step was to generate QR codes containing Tribler's TrustChain identity. The first steps towards finishing this sprint was generating QR codes in Python and experimenting how much data could fit in them, before scanning them did not work reliable anymore. Other time-consuming work included setting up the build environments for Tribler and the Android app and figuring out the structure of Tribler and understanding how the components were linked together, as the blockchain technology in Tribler is tightly coupled.

Before our first sprint, we familiarized ourselves with Tribler's GUI implementation and implemented a simple button that displays a QR code. Also, we created very simple Android application that could read a QR code and showed its contents.

This first sprint led to a lot of valuable insights into bottlenecks and next issues to solve during the project. The three main discoveries and their solutions were:

1. Character encoding is very important in QR codes. There are different QR code reader implementations that expect different encoding character sets. In our first sprint we directly put the private key binary data in the QR code but this led to a various of flaws with different phones and readers that led to random (incorrect) behavior. The foolproof solution that was eventually chosen is to encode all fields that have non-ascii characters as base64 so that there would be no conversion issues between different character sets.

2. The cryptographic library in Tribler was incompatible with the library used in the Android application. The Android application used the Bouncycastle [4] crypto library whereas the TrustChain in Tribler uses the state of the art Libsodium [5]. As Libsodium is future proof and the first pick for new projects, the decision was made to replace Bouncycastle with Libsodium. Fortunately there were bindings available for Java. The Java JNI bindings [6] made using Libsodium in and Android environment possible. However, it was not easy: the dependency to the existing Bouncycastle library was both tightly coupled and partly invisible: keys were quickly pulled out of their container into a byte array, which made it difficult to track the usage of the keys throughout the application.

3. Currently the identities from Tribler were transferred to the phone. After a discussion with the blockchain department, it was decided that this situation was not preferable as it would not be in line with the intuition of the user about the persistence of identity. The agreed solution was to make use of a temporary transfer identity to ensure that both the original identities in Tribler and the identity in the Android application would not

be replaced. So the idea went from transferring the identity to empting funds.

Afterward, we agreed upon a common transaction format that would be used to transfer the tokens. This took many iterations to reduce the size of a QR code down to an acceptable size and to find the minimal data that must be transferred in order to reconstruct a block.

At this stage we were able to split up the work.

1. A funds screen was created that could show transactions in your chain. This shows the transactions that are on your chain. It was requested that the interface looked polished.

2. The Android export functionality was created, exporting your current private key into a QR code

However, when we were about to merge these changes into the main app, it was discovered that the keys were not able to be used for signing and verifying blocks. This was due to a misunderstanding in how Tribler saved its keys: it used the DualSecret (key contains two cryptographic seeds, one for encryption and another for signing) specification to export its keys, instead of a private and public keypair as we had thought. Therefore, we had to recreate Libsodium's DualSecret implementation as this was not exposed in the bindings.

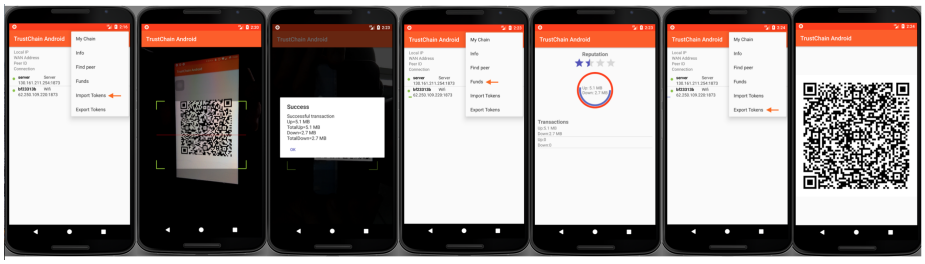
Another discussion on implementation was about how much tokens you could export, as theoretically it would be possible to have a negative balance by downloading more than uploading. For clarity and to prevent confusion, it was decided that the maximum amount of transferable tokens would be max(0, total uploaded - total downloaded). In this way, you could not get a negative balance and make no exports before uploading more than downloading.

Finally, in the last steps of the process, we ensured that the export in Tribler was made "idiot proof". Exporting the tokens first requires to go through a warning pop-up. The export tab also contains an explanation on what the export functionality does and what the risks are. —— To merge the code we had to confirm to Triblers code style , and its policy on user interface naming and design.

We worked together with the Overlay group to get each others final and major changes merged into the master of the Android Application. No major merge conflicts or issues were found to be blocking. The last steps included improving the documentation, polishing the in-code comments and ensure that the naming conventions from Tribler are maintained in the Android codebase.

1.3 Final implementation

The final implementation consists of two parts: the Tribler modifications and the Android TrustChain application. The total chain of events that shows all the implemented features is as follows:



- Export QR Code with funds in Tribler
- In the app go to “Import Tokens” in the menu
- Scan the QR code generated by Tribler: your tokens are now transferred!
- On the funds page you can see your transaction history as well as your current token balance.
- You can transfer your tokens to another android device by pressing “Export Tokens” in the menu. A QR code is generated which can be scanned again by another phone.

The full implementation details can be reviewed in the technical documentation. As a summary, exported tokens are stored in a temporary transfer identity, of which the key and the transaction are stored in the QR code. While importing the QR code the temporary transfer identity and the transaction blocks are reconstructed using the data in the QR code. The keys are then used to sign a transaction to transfer the funds to the receiving party.

Chapter 2

Technical Documentation

2.1 Crypto

The app maintains cryptographic compatibility with Tribler, using Libsodium. To achieve this, Java JNI bindings has been used to allow the Trustchain Android app to use native libsodium method invocations.

Libsodium has been chosen due to the fact that it is used by Tribler, and because it provides all of the core operations needed to build high-level cryptographic tools. It is a portable, cross-compilable, installable, packageable fork of NaCL with a compatible extended API. Libsodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x64), iOS, Android, as well as Javascript and Web Assembly.

Libsodium supports the notion of Dual Secrets, an object that supports both encryption and signing. The key formats used in the app match the keys used in Triber, using `xsalsa20poly1305` for identity management and `ed25519` for signing.

2.1.1 (De)serialization

During transmission and storage, keys are serialized and deserialized in the following manner:

- Public key pair: `"LibNaCLPk:"` + public key bytes + verify key bytes
- Private key pair: `"LibNaCLSk:"` + encryption seed + signing seed

The signing key is then generated using the signing seed. The private key can be generated from the encryption seed.

2.2 Tokens

We are compatible with the current Tribler transaction content. A transaction holds the values up , down , total_up , and total_down. When receiving the

first half of a block , the receiver flips up and down , and sets the total_up and total_down for itselfs (adjusted with the transaction content).

2.3 Wallet

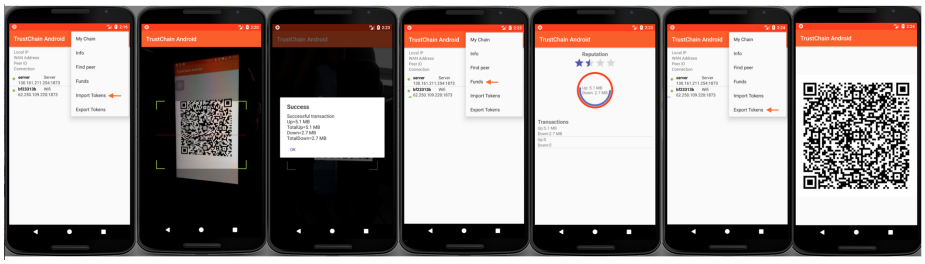
The wallet management works with a new Tribler concept, called the reputation of each user. This reputation is based on the amount of data uploaded and downloaded, and the simple subtraction of these two quantities provides a number representing it. This number gives a positive value when the user is uploading more content than downloading, therefore contributing positively to the overall system. It will be referred in terms of tokens, that account for the reputation of the user, and can be transferred.

2.3.1 Import of tokens from PC

This feature is done by generating a throw-away identity in the PC, to which a chosen amount of Tribler tokens are transferred. Then, this identity is exported to the phone by printing a QR code with the necessary information in the screen, and scanning it with the TrustChain Android app. To this mean, an option called "Import tokens" is implemented in the app's menu. Once both identities are settled in the phone, the final transfer is performed and only the phone identity outlasts. Once this process has been completed, the phone has successfully received the tokens from Tribler and they can be checked in the "Funds" menu option. However it must be clarified that, as it is an offline process, there are no guarantees for preventing the double-spending problem.

2.3.2 Import/Export of tokens between phones

The functioning is exactly the same as the import of tokens PC-phone, only that now it is performed between two phones using the Trustchain Android app. In order to go through with it, the sender uses the menu option "Export tokens", which displays in the screen a QR code with all the necessary data to export the total amount of tokens. Then, the receiver should make use of the "Import tokens" option to scan it and complete the transaction. As it is an offline process, same security concerns as before apply here.



2.3.3 QR code

The QR code has to be able to transfer the throw-away identity, while having a total size small enough to make it readable. To that effect, only the essential information is included in it:

- Private key of the throw-away identity, which includes the private key special crypto format.
- Transaction object, with the up and down quantities.
- Block hash and sequence number belonging to the half block of the transaction between the sender identity and the throw away one.

This information is encoded in a JSON string before being put into the QR code.

Once the QR code has been read, the receiver uses this information to reconstruct the transaction and throw-away identity. The QR code used is the version 13, which has a capacity between 1440-3424 data bits depending on the ECC level.

2.3.4 Example

An example of the data in the QR code is the following:

```
{
    "private_key": "TGliTmFDTFNLOvHyazzyYvb00cdAIb+
        xmDUzflFOsnzYTm3vbFcRV0FfuxWh827LrDLxYljG5+ga\n/
        m0SUkDYcDiHRnuf5BQ1HAI=\n",
    "transaction": {"down": 0, "up": 11114175918},
    "block": {"block_hash": "7Jh0+S93fbtoqWwKQlYmsPMjC8eU7Bzo91NaKy/0
        d0w=\n", "sequence_number": 1}
}
```

Which will result in the following QR code:

Bibliography

[1] "Filecoin: A descentralized storage network," Protocol Labs, Tech. Rep., August 2017.

[2] (2018, January). [Online]. Available: http://trustchain-android.readthedocs.io/en/latest/

[3] (2018, January). [Online]. Available: https://github.com/tribler/tribler

[4] (2018, January). [Online]. Available: https://www.bouncycastle.org

[5] (2018, January). [Online]. Available: https://download.libsodium.org/doc/

[6] (2018, January). [Online]. Available: https://github.com/joshjdevl/libsodium-jni

[7] (2018, January). [Online]. Available: https://en.wikipedia.org/wiki/PDF417