BankChain - Final Report

Chainable Technologies

H.G. van de Kuilen
rvandekuilen, 4226151

J.C. Kuijpers
jckuijpers, 4209915

M.R. Kok
mrkok, 4437659

I. Dijcks
idijcks, 4371151

June 2017

Contents

1 Introduction

One of the largest issues that online interaction currently faces concerns trust. It is nigh impossible to verify who someone is and that no man-in-the-middle-attack is taking place. One possible solution to increase trust in other users is creating a web of trust. This web of trust allows you to build a trust score. This trust score depends on if other users have announced their trust in you. The problem is that you need secure methods to announce your trust in another user, making sure that user is who he says that he is, so you don't announce your trust in someone who is spoofing that user. One solution is a face to face exchange, however that is not always practical.

Our solution uses bank accounts. Controlling a bank account gives an inherent form of trust, since it is not trivial to obtain one. One needs to identify himself and link his name to the account. This makes it unfeasible to open hundreds of bank accounts automatically or illicitly. These premises allow us to increase the trust score of someone if he can proof he owns a bank account with his name attached to it.

Our application allows a user to send a challenge in the description of a bank transfer. If the other responds to the challenge, signed with his signing key, we can verify that his key-pair belongs to the same person that has control of the bank account. This allows you to increase the trust in that user. The bank account also has a legal name bound to it, which can be used for reference or other trust methods.

2 Overview

The product as described in the introduction is in fact a proof of concept: can an app be created that enhances a web of trust using bank transfers with IBAN addresses? The goal was to create such an app and see what problems occur and if it is a viable concept.

The requirements of the proof of concept: find a trusted way to verify a public key and IBAN combination by usig an API from a bank. This goal has been achieved using the Bunq bank. An optional task was to integrate our IBAN verification with other project groups. The second group makes a blockchain, and the third group created a key exchange mechanism using bluetooth transfers.

2.1 Use cases

The proof of concept we created can be used to enhance the trust in a person in a web of trust. Our product does not create a web of trust or initiate basic trust: that is done by other project groups. Our project just increases the trust when the supplied IBAN can be verified for the public key in the blockchain. (Our product has a basic storage that mimics a very simple blockchain). The product is excellent to verify that a trusted person owns (or has access to) given

bank account. This way trusted contact info can be collected. A system can automatically make sure that the right person will receive the money.

2.2 Method

The product is an open source Android app that implements all required functionality, and a basic user interface to demo the backend work.

The app is able to send a create a challenge and send it to a peers bank account. It is also able to read from its own bank account. From the received transactions, it can either send a response of a challenge, or act on the received response: that is, adding more trust to the peer in the blockchain. At every step, the signatures of challenges, responses, and the bunq API are verified. It is also possible to do manual challenges: you do the bank-part yourself by copy-pasting values from and to your own bank app. This allows easier testing what happens and allows usage of other banks without implementation in the project. We implemented a simple storage to mimic a blockchain so we can store that the public key has a verified IBAN.

2.3 API for other apps

Near the end of the project, a fork has been created to make the functional bits of the project into a library, to be used by the Nervous Fish Game Studio group. They created the key exchanged using bluetooth. The library, named BankVer, can be easily implemented by the Nervous Fish group to enhance the trust they create with their exchange. Sadly, the blockchain group was not ready to integrate as well, otherwise a full web of trust could have been created.

The API exists of two classes and an interface, easily implementable and usable with not more than a handful of methods, all documented. It allows synchronizing all challenges and responses, updating the blockchain when needed, and sending new challenges, either manual or automatic ones.

3 Reflection

Our process was met with hardship very early in the duration of the course.

3.1 Team composition

Early on in the process, communication between one team member and the rest of the team was not good. Messages were not responded to, the phone was not picked up. After a short while (about two weeks) the member started showing up all together. A couple of weeks later the member did not seem to be participating in the project anymore. Sadly, we had assigned a deliverable entirely to that person in week two and we did not verify his work. This resulted in an insufficient mark for the Product Vision. We should have worked on it together and verified each other instead of relying on a single person. Since then

we started reviewing each others work and not assuming what the person said is actually done is true.

3.2 Android

Android is a well known platform and a lot of apps exist for it, but it was not easy to get everything up and running with modern APIs. The main cause is that Java 8 has only been available since Android API 25 (7.1, Noga). Due to Android fragmentation, just very few devices have 7.1 or heigher (Google, Inc., 2017a): our development devices did not. So we used Jack to tranform Java 8 code into Java 7 code, and together with Java 8 library fillers we could use nicer asynchronous code and lambdas. Jack had a lot of issues and support by Google was discontinued (Google, Inc., 2017b), so we switched to Retrolambda, which worked faster and better. Build also became faster: from 15 minutes to 30 seconds.

In retrospect we should never have bothered and just go the more messy way using Async Tasks and Runners. We would have had less problems. We want it too nice and it cost us a lot of time. The switch to retrolambda was too late: we should have done it earlier in our process.

3.3 GUI tests

Android GUI tests are a pain to get working right and deterministic. It has not been easy to make it work properly on all simulators and hardware. If it works on one device, it does not necesserily work on other devices. The test, that is. The app works just fine.

In the end we put more time into writing and fixing the tests for non-existend problems. Whenever tests failed, it was not due to bad code but due to a bad test. It did not help to improve the product, it only cost a lot of time.

3.4 Working with external parties: Bunq

The Bunq public API, used by our app to do the transactions, is a very new API (april 2017) and is very unpolished. After a couple of weeks into the project we also discovered more undocumented limitations and some documenation that is simply wrong. For example, the documentation on the listing of transaction states that it is "a listing of all Payments" (bunq B.V., 2017). However, it is actually limited to the latest ten transactions, as we discovered when we began using that API.

The biggest limitation of the API is the coupled IP address. When registering the device with the API (a step in the creation of the session, see our architecture document), the current IP address is bound to the API key from Bunq. When the IP addresses of the device changes, the session becomes invalid. This does not happen when running on a server in a datacenter but is very common on a phone. When you switch between WiFi and 4G you change IP. We added code to invalidate keys when the IP address changes. On top of this, the API

key is bound to the first IP used. So when your IP changes, you can't start a new session with your new IP address: a new API key from the Bunq app is required. This is a serious showstopper and ended the possibility to use the Bunq API for the purposes of this app. To get around it, we created an HTTP proxy server that proxies all requests, so all requests only carry the IP of that proxy server. This however, is against the terms of services of the Bunq API and is thus not a viable solution for an actual app in production.

A last limitation we found is that throttling is required on payment creation API calls because only six payments per six seconds is allowed. A simple synchronization of all challenges makes payments very quickly. This limitation is not documented either.

It cost a lot of time to find the issues Bunq was reporting and to work around it.

In retrospect, we should have looked at other banking APIs as well, for example the Open Banking API specification (TESOBE Ltd., 2017). This is a specification not in open use currently but can get traction among banks. It would have given us a more abstract view of what banks require. Our current bank abstractions are solely based on how Bunq and their API works.

3.5 CI and Android apps

We have had a lot of issues with the Continuous Integration tooling in combination with Android. Build are non-deterministic, and require a lot of restarting. The biggest problem is the running of tests on the headerless Android simulator. Installing and starting the simulator takes very long (up to 5 minutes), or it does not connect correctly at all. We have not been able to get the GUI tests to work on the CI, so we disabled these on Travis but kept them on our local machines using some magic with pre-build commands and text replacement. This is messy and not what we wanted but after four weeks of getting this to work, we gave up and spent our time on the actual code.

Improvements to this problem could be made by using a system like Jenkins instead. A system that allows more flexibility and configurability than Travis CI. However, this takes a lot of time to set up as well. And the builds will probably still be slow. It is also harder to set up GitHub integration.

Unless Travis-CI improves Android support, or Android improves CI support (or either), running Android apps on Travis is not worth the time and hassle that it creates. In retrospect, we probably spent as much time trying to fix Travis with Android issues as we spent on building the app itself.

4 Functionalities

The aim of this Chapter is to give an overview of the developed functionalities implemented in our final product.

4.1 Sending challenges

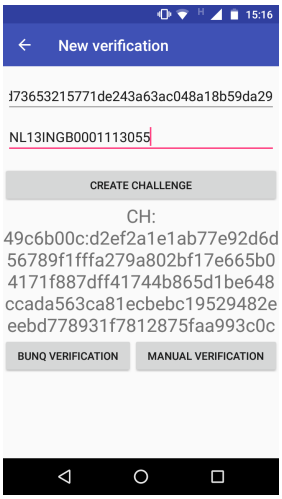
On the main screen of our application, there is a blue button that opens a new screen where the user can start a new verification. By entering the IBAN and public key of the verifee, a new challenge is created and presented to the user. The user can now choose to manually send the verification using an external application, or use the built-in Bunq verification. The display of the activity is shown in figure 1(a). When the Bunq verification is picked, the application uses the Bunq key to connect to the Bunq API, and send the message in the description field of a €0,01 transaction to the selected IBAN.

4.2 Verifying challenges

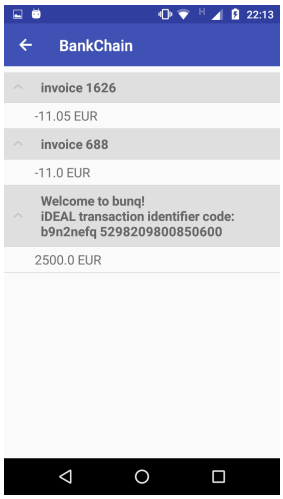
When the user has received a challenge from another user with an external bank application, the user can enter the challenge on the main screen to verify that it is a valid challenge. When the challenge is verified, a response is generated that can again be manually send with an external application, or automatically with a built-in Bunq transaction.

4.3 Viewing transactions

It is possible for the user to see the recent transactions that have been made with the Bunq bank. On the main screen there is a big button that takes the user to a new screen, where all the recent transactions are nicely presented in an expandable list. This screen can be seen in figure 1(b).



(a) Visual representation of NewVerification.java



(b) Visual representation of RecentTransactionsActivity.java

Figure 1

5 Human interaction design

5.1 Introduction

Our main application is an API so there is no available GUI to test. We did however also build a version that has a GUI to demo features without being dependant on other teams. this is the version we decided to test. ideally everything should be done automatically but for most functions a manual version is available. We wanted to know how usable these manual features are to an average user with no prior knowledge of the application. So our research question is: What part of completing a manual verification is the most difficult to perform according to the user?

5.2 Method

Because no prior knowledge can be expected of the user we decided to create a scenario with instructions that the participant had to follow to navigate the application. This scenario, as shown in Appendix B contains 3 sections. The first is to configure the application, the second to create a challenge, and the third to validate the response received from the challenge. This does have one downside because we were not only testing your application but also the clarity of our instructions. To evaluate this result we used 3 methods. The first was to time every step and see how long it took the user. The second was a questionnaire, as shown in Appendix A. These questions start off vague to try

to get there opinion first before leading into a specific direction. we tried to evaluate both the application as well as the instructions in order to get a clear picture where the main problems were. The final way we looked at the results was by transcribing the events and looking for patterns.

The analysis of our data depends on the specific question some are easily quantifiable (e.g. rate usability 1-10 or easiest step) other are more difficult. So for question like "What would you improve?" we decided to look for repeating answers. We also did the same looking at how our participants used the application and trying to find common choke points.

5.3 Results

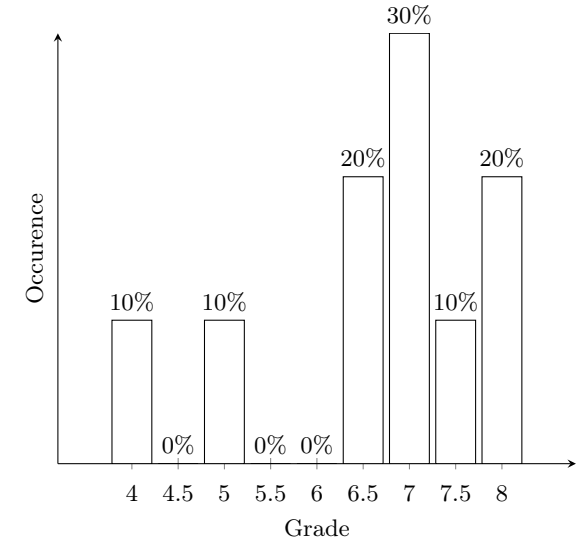
5.3.1 Improvements to the application

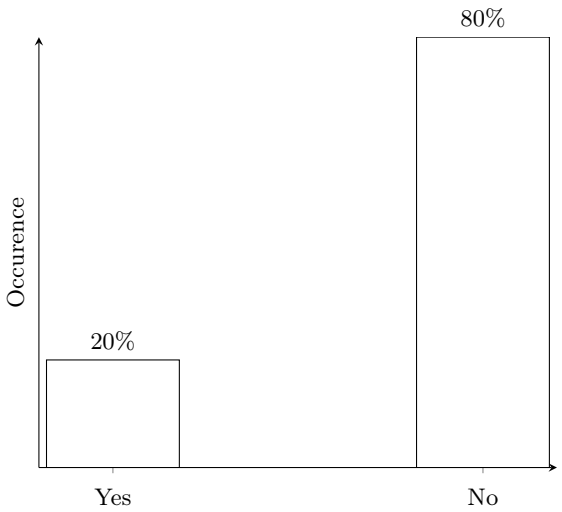
The response that we got the most when asked what the tester would improve to our application, was 'change the settings menu' and 'make verifying the response easier'. In the application we used to test, the settings menu is hidden in an overflow menu on the top-right of the screen. This is standard android practice, and we noticed that most testers who had trouble with this did not have an android phone. We also received feedback that hiding a setting in an overflow menu does not make sense when it is the only option in the menu.
Verifying the response was too difficult because there are also input boxes on the main screen that are used for verifying a challenge. Most testers got confused and did not know which input box to use.

5.3.2 Improvements to the instruction sheet

The feedback we received most was that some of the wording on the instruction sheet did not match with the application. In the instruction sheet (which was also in Dutch), we used the word 'valideer'. Which corresponded to 'verify' in our application. Some testers were confused by this, and took a long time to find the right buttons.

user	Settings	Challenge	Verification
1	1:32	2:43	2:01
2	0:35	1:43	3:00
3	1:32	3:43	1:50
4	0:34	1:56	1:08
5	0:43	2:11	1:17
6	0:18	1:02	2:38
7	0:40	1:30	2:23
8	0:30	2:10	1:13
9	0:51	1:29	1:41
10	2:20	2:00	2:40
average	00:51	02:03	1:59





Has the tester correctly explained what they just did.

5.4 Conclusion

When looking at the results it is quite clear that the most important step to remove is the constant copy pasting of information. This part takes too much time and is quite error prone. We completely agree with this and are trying to make our application as automated as possible.

However there are other interesting results. we noticed for example that it took IPhone users a long time to find the settings menu while android users were used to its location. Another interesting result was that people tried to

Step	Occurence
Open the BankChain application	2x
Go to settings and enter your private key	4x
Press the blue button and create a new challenge	4x
Verify that the response is correct	2x

Table 1: The steps that our testers found the easiest.

Step	Occurence
Go to settings and enter your private key	3x
Enter the public key and iban and create a manual challenge	2x
Copy the response and verify it	2x

Table 2: The steps that our testers found the most difficult.

copy the challenge instead of first pressing the button manual challenge.

Even though it is against standard android practices, we will change the settings menu to always show in the toolbar. It did not make sense to have the only option hidden behind a 'more options' menu. Because we noticed many testers saying the many input-boxes on the main screen being too confusing, we decided to group each of them under their own activity. This will make the main screen much less cluttered, and should lead to a cleaner user experience. Since we will perform a single HCI study during this project, we will not process the feedback we got on our instruction sheet.

There are a couple of limitations to our study the first being our participants. Our participants where all TU Delft students and were relatively tech savvy the results would probably be completely different if we asked men and women in their sixties. Another limitation was the scenario itself it contained a few vague instructions that at times confused the participants. Finally an interesting result of doing a face to face experiments is that the participants are far to nice both in trying to get trough the scenario and in answering the questionnaire. Participants generally don't like to give insufficient marks.

6 Outlook

We did build a functional application but there is still a lot to improve depending on how much time there is and what direction we would like to go. The main problem is that verifying manually is to cumbersome and complicated to be interesting for ordinary users. Therefore we should aim for complete automation which means we have to be able to create and read transactions from their bank accounts. These functionalists are currently not publicly available so if we were to pursue this we would have to either work together with someone that did reverse engineer this. or reverse engineer these apps ourselves. The problem is in both cases that these implementations will probably be shutdown as soon as the usage starts to increase as it is not wanted by the banks itself. Even Bunq

bank, that has a publicly available api has to redone since their api does not contain enough functionality for practical use. The main issues being their main issues being linking the api key to specific ip addresses and only returning the last 10 transactions.

Another way to improve the software is by allowing new public keys to be added using the challenge response system. At the moment you can only strengthen existing connections with public keys you already have access to it might be possible to use a different encoding (e.g. base 64) to increase information density so bot the challenge and a public key fit within 140 characters.

These are all ways to improve the bank part of this validation implementation. Another way might be to increase the possible ways to increase your trust score e.g. by attaching your key to your twitter account or email address. The possibilities are endless as long as there is some form off communication that can only be accessed trough the given account. One problem does arise when you start to attribute trust because then you have to evaluate how much a specific connection type adds to the overall trust score.

References

bunq B.V. (2017, May 2). *Payment.* Retrieved June 22, 2017, from `https://doc.bunq.com/api/1/call/payment/method/list`

Google, Inc. (2017a, June 21). *Dashboard.* Retrieved June 22, 2017, from `https://developer.android.com/about/dashboards/index.html`

Google, Inc. (2017b, June 22). *Jack.* Retrieved June 22, 2017, from `https://source.android.com/source/jack`

TESOBE Ltd. (2017, June 22). *Open bank project.* Retrieved June 22, 2017, from `https://openbankproject.com/`

A Human Computer Interaction study questionnaire template

We asked our test subjects to answer the following questions (in Dutch):
Proefpersoon X <date> (<time>)

- Heb je een android telefoon?
- Weet je wat een public en private key is?
- Wat was je eerste reactie?
- Kan je zelf uitleggen wat je zojuist hebt gedaan / bereikt?
- Moeilijkste stap?
- Makkelijkste stap?
- Wat zou je verbeteren aan de applicatie?
- Wat zou je verbeteren aan het stappenplan?
- Gebruiksvriendelijkheid?
- Zou jij dit een nuttige toevoeging vinden aan je online betalingen?

Opmerkingen

B Human Computer Interaction study instruction sheet

Kijk in de notitie app voor de private key.

1. Open de bankchain app.
2. Ga naar settings en vul de gegeven private key in.
3. Ga terug naar het hoofdscherm.

De Iban en public key voor een challenge staan in de notitite app.

4. Druk op de blauwe knop en kies voor nieuwe challenge.
5. Vul de gegeven key en Iban in en maak de challenge selecteer hiervoor de handmatige challenge.
6. Kopieer de challenge naar de notitie app.

De response staat in de notitie.

7. Open in het hoofdmenu de validatie optie.
8. Kopieer de response en valideer deze.
9. Controleer of de response correct is.