Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Are You Human?

Semester Thesis

David Jan Lukas Werder

dwerder@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Tejaswi Nadahalli, Darya Melnyk
Prof. Dr. Roger Wattenhofer

July 13, 2020

Acknowledgements

Tejaswi Nadahalli and Darya Melnyk gave valuable guidance through the project. I would like to thank Tejaswi for providing a code for the graph analysis which generates a good & bad graph and connects them over some edges. The Distributed Computing Group of Prof. Dr. Roger Wattenhofer provided me with three Android Smartphones to test my App. Thanks to Andreas Werder for the orthography support.

Abstract

One approach to give a proof that a real person is behind an online user is an analysis of the interactions of the user. This project follows this approach by implementing an Android app which exchanges tokens between phones over Bluetooth. The tokens are uploaded to an Are You Human Server. This server constructs a graph which represents people interactions. This graph is analyzed to conclude whether a online user is a real person or not. Some use cases for this project are voting, reviews, surveys, log into a website, elections.

Contents

Chapter 1
Introduction

Are You Human? is an interesting question when it comes to the Internet because everyone will answer the question are you human with yes. There are many situations where it could be interesting to influence public opinions on the internet. Someone would like to get a higher rating for an apartment on Airbnb and a lower rating for all the other apartments, someone want's to get a lot of likes for one politician and a lot of dislikes for all the other ones. In many of such cases someone could have an interest in producing lots of fake humans.

In the real world, it is trivial to answer this question since we can tell whether it's a human is in front of us or not. But if it comes to the internet, it is easy to trick us, and we have to find a way to still be able to answer the question Are you Human. Here comes the characteristic of humans into the game that we often carry our smart phone with us.

1.1 Goal

The goal of this semester project is an implemented mobile proof of personhood concept, in which the server is not able to link an online user to a person but is able to distinguish users from fake users. In this thesis the example of voting is used but, in fact, the proposed implementation is compatible with many other use cases. The project follows the approach to collect data with a mobile phone over a time and send this data to the server, together with the vote. The data is collected into a graph which is analyzed to give a proof whether the origin is a person or a bot. In addition, the server is searching for multiple votes by the same person. The suggested concept uses an exchange of tokens over Bluetooth between phones and evaluates the exchanged tokens, just like it is done in some COVID-19 contact-tracing apps.

1.2 Thesis overview

The short Chapter 2 takes a look at the existing literature. Chapter 3 presents the concept of the implementation. It is followed by Chapter 4 which deals with possible Attacks. Such a system is exposed to attackers which want to change votes, modify an election, create fake reviews... Different attacks are discussed and some defence mechanism are implemented. Chapter 5 presents the implementation. An important part of is extracted to a separate Chapter 6, Graph analysis.

Chapter 2

Literature

Proof of personhood has been first introduced in the context of blockchain-based cryptocurrencies. In [1], Borge et al. suggest that it should be done through a party at which people must be present in person. Whereas Borge et. al. do the proof of personhood by face to face contacts, the Are You Human concept analyzes device to device contacts.

2.1 Approach of COVID-19 contact-tracing apps

Due to the COVID-19 outbreak many mobile apps have popped up which find out whether the holder of the phone has been in contact with an infected person and must therefore go into quarantine. Some of these apps use an approach which has potential to help solving the task of this project. The approaches that these apps use are quite similar: Each phone generates periodically, for example every 15 minutes, a new random token OWN that has no personal connection and stores it with a timestamp. The near field communication Bluetooth is used to detect if a person is in a close distance. If that is the case, the two phones exchange their tokens over Bluetooth and locally store the received one as a SOMEONE token with a timestamp. If later one of the two persons gets tested positive with COVID-19, the OWN from the last days (e.g. two weeks) is uploaded as INFECTED tokens to a server. The app periodically downloads the INFECTED tokens and compares them with the SOMEONE tokens. If they match, there has been a contact between a positive tested person and a person that has therefore to go into quarantine [2].

This semester project was started in the early days of the COVID-19 apps and the source code was not considered for the Are You Human code. Now, the SwissCovid app [3] has been published in Switzerland and seems to be compatible with the Swiss law. The SwissCovid app is based on DP3T [4] which is similar to most Bluetooth based Covid Apps. The documentation and source code are online.

Google and apple cooperated to create libraries which help developers to program COVID-19 contact-tracing apps. The libraries implement nearly the whole contact-traching app and are based on DP3T. In a next step, the contact-tracing functionality should be included in IOS and Android. Unfortunately, it is not yet allowed to use it for other purposes [5].

In this work we propose a bluetooth based approach for the proof of personhood. To the best of our knowledge such a proof has not yet been considered in the literature.

CHAPTER 3

Concept

During a week, most humans come as close as a few meters to each other while wearing their smartphone. This fact is used by the following concept.

3.1 Overview

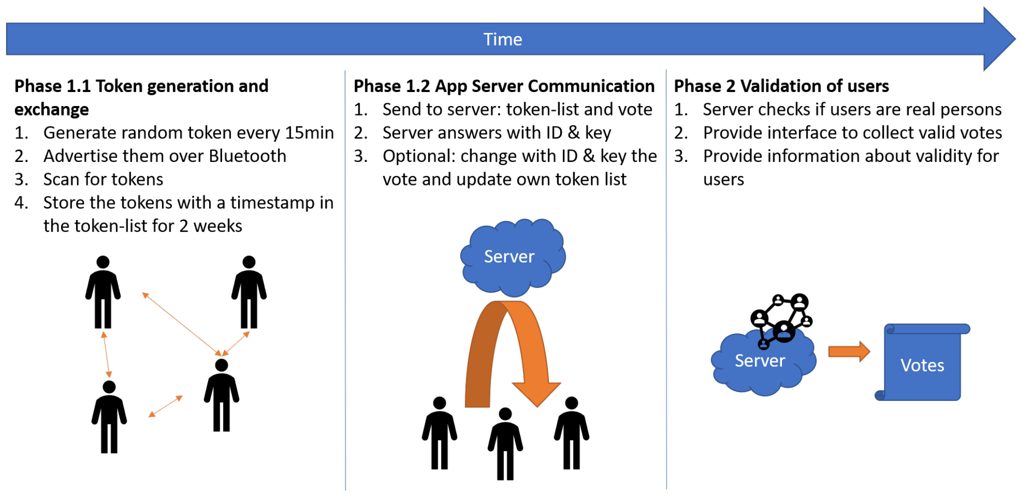

Time
Phase 1.1 Token generation and exchange
1. Generate random token every 15min
2. Advertise them over Bluetooth
3. Scan for tokens
4. Store the tokens with a timestamp in the token-list for 2 weeks
Phase 1.2 App Server Communication
1. Send to server: token-list and vote
2. Server answers with ID & key
3. Optional: change with ID & key the vote and update own token list
Server
Phase 2 Validation of users
1. Server checks if users are real persons
2. Provide interface to collect valid votes
3. Provide information about validity for users
Server
Votes

Figure 3.1: Concept overview

3.2 Token generation and exchange

Each time two persons come in range of a Bluetooth connection, the phones exchange a random token. Every x minutes, the app generates a new token. This token will be advertised over Bluetooth until the next token is generated. In the meantime, the app is scanning for advertised tokens from other phones. The advertised and scanned tokens are saved with a timestamp. The tokens are deleted after some predefined time, for example two weeks. This token generation and exchange is always running in the background of the app in order that a user can in any time give a proof of personhood.

3.3 App Server Communication

A person that wants to vote presses a button that sends a vote or some other data to the Are You Human server, together with all advertised and scanned tokens. In response, the user gets an ID and a key for future communication with the server. The future communication enables the user to modify or add votes, to get updates to his validation status or to upload more tokens to proof of personhood. The vote is connected to a vote ID allowing the user to vote for different topics.

3.4 Validation of users

The server evaluates the tokens checking each user's personhood. It hosts an interface from where all the valid votes can be collected corresponding to a vote ID.

3.5 Timing

Tokens are stored on the phone and server for a time for example two weeks. After this time the tokens are deleted in order to keep anonymity. That a user can proof his personhood at any time the app is always running in background and exchanges tokens.

3.6 Vote and other use cases

The Are You Human app and server do not care about the content of the vote. It can be encrypted and decrypted by a third party, it can store multiple votes under one vote ID or it can store customizable date. Those data can be use as a token to get a proof of personhood, reviews, surveys, log into a website, elections...

Chapter 4

Attacks

Since the system is decentralized, it is prone to attacks by malicious users. In the following, we discuss the attacks that could generate wrong proofs of personhood.

4.1 Send fake or same tokens to vote

The attacker generates random advertisement tokens, scans tokens and sends them to the server. Alternatively, the attacker could send the same valid tokens multiple times. The goal is in both cases to vote multiple times. In order for the advertised fake token to be successful, a corresponding scan token must exist. Normal users only store advertised tokens as scanned tokens. To be a successful attacker you must send already advertised tokens. With a large number of random tokens the chance of a successful attack increases. This can be done trough multiple fake users. An attack with a big enough number of fake users leads to multiple entries with the same advertisement token in the same timeslot. In normal operation, the chance for this is small but not impossible because the tokens are generated decentralized on the phones.

To prevent this attack...

- ... long tokens are used. This implementation uses the longest tokens allowed by Bluetooth.
- ... double advertisements (same token at same time advertised) are counted and taken into account for the validation (implemented in Section 6.2.1).

It would not make sense to set the double scanned tokens to invalid, since it happens all the time that multiple phones scan the same token.

4.2 Set valid votes to false

The attacker tries to set a valid user to invalid by sending a large number of advertisement tokens from this user to the server. The server classifies the user as an attacker. Since the tokens are changed within a period larger than 15min an attacker must be close to a user for several hours to get enough tokens for an effect. Another way of getting enough tokens is to randomly send a large number of them to the server.

To address this issue...

- ...on request, the server sends a notification to the user informing if the votes are valid and the time of the last validation change. This information can detect an attack (implemented in Section 5.2).
- ... users which would be set to invalid because of double advertisement are compared to each other in order to find out the attacker (not implemented).

4.3 Take over the ID and key

If an attacker takes over the ID and the key, he or she can change the votes. This could happen on the phone, on the server or when the phone is voting over the internet.

To detect this attack...

- ...each time the user modifies the data on the server, the last modification times are compared (implemented in Section 5.2).
- ...the ID key pair is used to log in to a website to check the votes (which can be done by the app, not implemented).
- ...all votes and IDs are displayed on a website (not implemented). This method makes the system less anonymous.

4.4 Cluster of fake phones

If an attacker creates a bad graph, connects it to the good graph over some infected nodes, multiple fake votes can be produced. This is addressed by the graph evaluation analysis that will be discussed in Chapter 6.

4.5 Use fake app/device

An attacker can split the advertisement and scan recorded tokens in multiple slices. In that way he gets multiple users out of one user. To make it even worse, the tokens can be advertised only for a short time and multiple tokens can be advertised at the same time. Later, the tokens can be split again. If such a device is placed in a crowded place, it will lead to a lot of fake users.

To prevent this attack…

- …meetings are only counted if they last for a longer time or if there are multiple meetings (implemented in Section 6.2).
- …only persons are counted, which were met alone.

4.6 Fake App

Fake app, which does the same in the background and votes for you. This app could be in the Android store or installed via a virus (no prevention implemented).

Chapter 5

Implementation

5.1 Android app

5.1.1 Development environment

There are two options when developing an Android app. A native Android app is usually built in Android Studio, a cross-platform app in a one of many other development environments. The big advantage of a cross-platform app is that a part of the code has to be written only once. React Native is the cross-platform solution of Facebook. It is well distributed and uses the common language of JavaScript.

But React Native has the following disadvantages [6]:

- A long setup is needed, even Android Studio has to be installed on the way.
- React Native depends on different Companies/Communities such as Google, Windows, etc. Therefore, the R.N. community actually should do a fix for every change of operating system, libraries etc. which is more than the community can do. Therefore, the setups are complicated, only with old versions of software, compatibility is granted. React Native is not fully working with the newest Android development environment 29 which belongs to Android 10.
- Airbnb is an example of a company that must have had bad experiences with their cross-platform app. After two years, they changed back to two different platforms, one for Android, one for IOS [7].
- In React Native, no direct support of Bluetooth between two devices can be programmed, there is no such library [8]. So, a module in Java [9] must be written.

On the other hand, Android Studio has the following advantages:

- It is better compatible. A Java code written for one app can be used in other apps, as well. Most development environments have a way to support Java code, even React Native.
- The documentation of Android Studio is better and the community larger than the ones for React Native.

Out of these reasons, a Java solution made in Android Studio is implemented despite the advantages of a cross-platform app.

5.1.2 Android and its difficulties

Android is widely used, there is a huge amount of apps. One would imagine that it is easy programming an Android app, especially if one has some Java experience. The reality is different, the drag and drop part for the app interface is not working as smoothly as one would expect. In the end, the layout of the app was written as code. A very large issue are the 30 versions of Android. Every time, it must be checked whether a function is supported for your target versions. A function that is working for older versions sometimes doesn't work for newer versions and vice versa. To overcome these difficulties, Google provides libraries which take into account the version differences. These libraries are most of the time up to date, but not always.

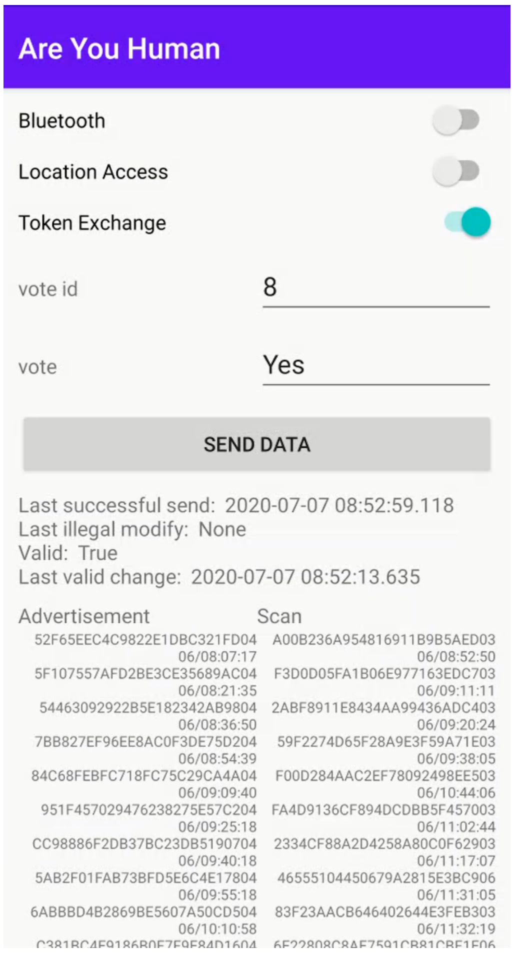
Android has a lot of permissions of which one must take care. These permissions are also version dependent. An app in the Play Store has to be actively supported, and changes due to Android upgrades have to be adopted [10]. This will most probably also be the case for the Are You Human app, since Bluetooth and background activities are often affected from changes.

5.1.3 Overview app

Figure 5.1 shows a screenshot of the implemented example interface of the Are You Human app. The code for the Are You Human part is accessible through the functions in the AreYouHuman class and strongly separated from the user interface.

On the top of the interface the switch to enable Bluetooth is displayed. If the switch is turned on, a popup will appear over which the necessary action can be done. After successfully enabling Bluetooth, the Switch disappears. The Location Access Switch works similarly but gives Location permission. The next Switch turns on and off the token exchange.

The SEND DATA button sends the vote ID, vote, tokens and ID-Key pair to the server. The next field displays info's for the user coming from the server. These info's update after each successful SEND DATA. They are saved locally.



Are You Human

Bluetooth: off
Location Access: off
Token Exchange: on

vote id: 8

vote: Yes

SEND DATA

```
Last successful send:  2020-07-07 08:52:59.118
Last illegal modify:  None
Valid:  True
Last valid change:  2020-07-07 08:52:13.635

Advertisement                       Scan
  52F65EEC4C9822E1DBC321FD04   A00B236A954816911B9B5AED03
                 06/08:07:17                  06/08:52:50
  5F107557AFD2BE3CE35689AC04   F3D0D05FA1B06E977163EDC703
                 06/08:21:35                  06/09:11:11
  54463092922B5E182342AB9804   2ABF8911E8434AA99436ADC403
                 06/08:36:50                  06/09:20:24
  7BB827EF96EE8AC0F3DE75D204   59F2274D65F28A9E3F59A71E03
                 06/08:54:39                  06/09:38:05
  84C68FEBFC718FC75C29CA4A04   F00D284AAC2EF78092498EE503
                 06/09:09:40                  06/10:44:06
  951F457029476238275E57C204   FA4D9136CF894DCDBB5F457003
                 06/09:25:18                  06/11:02:44
  CC98886F2DB37BC23DB5190704   2334CF88A2D4258A80C0F62903
                 06/09:40:18                  06/11:17:07
  5AB2F01FAB73BFD5E6C4E17804   46555104450679A2815E3BC906
                 06/09:55:18                  06/11:31:05
  6ABBBD4B2869BE5607A50CD504   83F23AACB646402644E3FEB303
                 06/10:10:58                  06/11:32:19
  C381BC4F9186B0F7F9F84D1604   6F22808C8AF7591CB81CBF1F06
```

Figure 5.1: Screenshot Are You Human app

On the bottom of the display are the advertised and scanned tokens with the timestamps. The tokens will not be visible in a release version.

5.1.4 Bluetooth

Bluetooth low energy BLE has some features, which are useful for this app. Apart from the fact that it does not use much energy, it is possible to advertise to and scan from multiple devices at the same time. The advertisement of BLE contains in the case of the Are You Human app:

- An UUID of 128bit which is used to verify that this advertisement is from a Are You Human app
- The advertisement token, which changes every x minutes

- A random BLE address which is automatically generated for each advertisement [11].

The maximum length of a token is 13 bytes, because the Bluetooth advertisement is limited in length. The fact that the BLE address is random and we advertise only the UUID and randomly generated tokens guarantees that there is no way to find out the user's identity based on the BLE advertisement. It is possible to locate a phone over Bluetooth scans, because some Bluetooth devices are associated with location. This is the reason why the app needs location permission by the rules of Android. Whenever the location permission is given and Bluetooth is turned on the app can scan and advertise tokens. The app scans for BLE advertisements; in case of a match with the corresponding UUID, it saves the token with a timestamp.

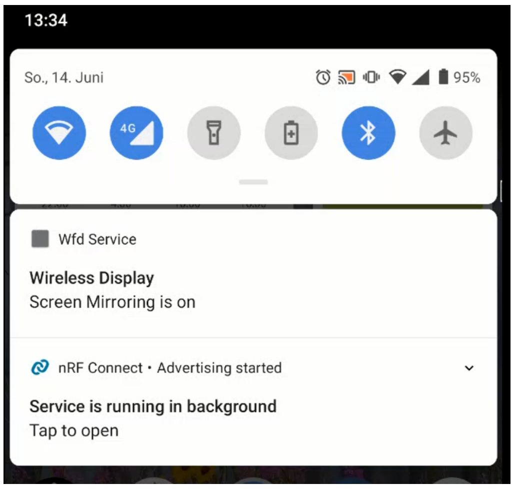
5.1.5 Save tokens and settings

The tokens are saved in a file, which is in the directory of the application. The files of the application are automatically decrypted in the newer versions of Android. To prevent the files from growing into infinity, the oldest tokens are deleted. If the file is still to big, some randomly chosen tokens are deleted. In that way a day with a lot of scanned tokens will not lead to a loss of all other tokens. Advertised tokens will not grow to infinity since tokens are valid and stored for two weeks. The settings for the app are also stored in a file like the one that is implemented for the tokens.

5.1.6 Work schedule

The token exchange is supposed to run all the time, even if the app is closed. To achieve this goal, there are two options. The service runs in foreground which leads to a notification which is displayed all the time. This would lead to a a bad user experience as it is shown in Figure 5.2.

The second option is to run a background service. The library Android Jetpack [12] from Google takes care of all the changes in regard of background services to different Android versions. The shortest supported schedule period is 15 minutes and starts automatically if the phone is rebooted. This library supports battery save mode like doze mode and other Android schedule rules, which lead to a longer period than 15 minutes. It can go up to a few hours, for example at night. As soon as the phone is used, it schedules it more frequently. For the Are You Human app it is not important that the advertising tokens are changed exactly every 15 minutes. It is welcome when the app does not drain down the battery. The implemented app works with the so called Work Manager and Worker classes from the library. The implemented preferred period is 15 minutes.



13:34

So., 14. Juni

95%

Wfd Service

Wireless Display

Screen Mirroring is on

nRF Connect • Advertising started

Service is running in background

Tap to open

Figure 5.2: Screenshot of notifications of foreground services as an example for a bad user experience

Each run of the scheduled task loads the settings from the file and checks if a new advertisement token is necessary. If so, it is generated and saved in the file. This makes sure that BLE is advertising and scanning. Then it saves the temporally saved scan tokens from the HashMap to the file. The HashMap has the advantage that the same token is not stored in the file multiple times.

5.1.7 Programmed settings

All the programmed settings are stored on top of the AreYouHuman class, in that way it is easy to change all the parameters and to keep track of these settings.

5.2 Communication between app and server

A person wanting to vote can enter a vote ID and a vote and press the send button. This executes an http post to the Are You Human server. The following parameters are included in the post:

- User ID
- Key
- Advertisement tokens
- Scan tokens
- Vote ID

- vote

The server checks if the ID key pair is valid. If not, it creates a new ID key pair. Then it saves the new tokens and saves the vote under the provided vote ID. The server does not immediately check if the user is valid or not since this would consume much time. The valid flag can still change if others upload their tokens. All the same, all users have a status of validity (default = true). Finally, the server responds with:

- User ID
- Key
- Last modification time
- Current modification time
- Validity status
- Last change of validity status

The phone stores the received ID key pair for future communication. The last modification times between the server and phone are compared. If they do not match a third party has modified the data on the server. If this is the case the phone saves the last modification time as last illegal modification time and displays it. In that way the user realizes if someone modified his data. The current modification time, valid flag and last valid change time are stored and displayed on the phone.

A user can update his votes on the server, check if the vote is valid and detect a modification by an attack Section 4.3.

5.3 Server

5.3.1 Flask and PostgreSQL

My supervisors suggested Nginx, Flask and PostgreSQL. The web server is written on Flask in Python using PostgreSQL as a database. Python is well distributed and favored by many programmers. Currently, many ETH students learn Python. Flask is the most used Python server. PostgreSQL is SQL based and SQL is the most distributed and known database language. PostgreSQL is open source based. Nginx was not needed for the test environment.

5.3.2 Debug website

The Are You Human server provides a debug website which displays all data from the database and gives opportunity to validate the votes, clear the validations or clear all data. In the attachment Figure B.1, a screenshot of the Website can be found.

5.3.3 Provide vote results

In order to get the vote results, an http post has to be made on “/votes” with a vote ID. The server responds with the votes of the valid users.

Chapter 6

Graph analysis

An important and dangerous attack is the creation of fake users, in order to manipulate the result of an online investigation, e.g. a vote. The goal of the graph analysis is a separation of real users and fake users. To decide if a user is a real person or not, the relations of this user are analyzed. The analysis is done by creating a graph from the collected data. Each user is represented by a node in this graph. The analysis is done in two steps. First the nodes are evaluated to detect simple attacks which only effect one user. In a second step the goal is to detect attacks which try to mimic multiple fake users. This is done by a graph evaluation.

6.1 Create Graph for validation

The users are represented by nodes and contain a validity flag. The interactions between the users are represented by the edges of the graph. The edges have a direction, a weight and a mode. The modes are corresponding to the kind of connection between people. There are three different modes:

- Mode "adv" advertised valid: User A advertised to user B a valid token and user B scanned that token. This leads to a directed edge from A to B.
- Mode "double" double advertised invalid: Whenever two users advertise the same token in the same timeslot. The token is not valid and the event is represented as two edges in both directions.
- Mode "ind" indirectly met: User A advertises the same token to B and C. This is represented as two edges in both directions between B and C.

The graph uses weights, which represent the number of equal arrows in a direction. This is also the number of different tokens. An advertised token is valid for one hour. If the scan time is more than an hour later than the token advertisement, the scan is not counted.

To view the generated graphs a plot function is implemented which stores its image in graph.png. It can be selected which parts are visible.

6.2 Validation

X stands for the number of exchanged tokens. The tokens are changed every 15 minutes or later. If x=3 we consider four cases. Theoretically, a person must meet another person for at least 15min or at least twice to be counted. Two other cases are more likely for x=3: Either a person has been met three times or for approximately one hour.

The validation of the users is done in two steps, the first one being an evaluation of each node separately and the second one an analysis of the whole graph.

6.2.1 Node evaluation

The nodes are evaluated according to the following steps. For each node, the following parameters are extracted:

- count_double: Number of double advertisements, extracted of the “double” graph
- adv_to_ids: Advertised to IDs. A list of user IDs is created. To make it more robust only IDs are included that exchanged at least x tokens (e.g. x = 3).
- count_ind : The number of indirectly met IDs is extracted of the “ind” graph.
- count_scan_and_adv: The number of users with A advertised B and B advertised to A, if both edges have the minimum weight x.

The following formula decides if a node is valid or not (at least for the node evaluation). The numbers 0, 2, 0 can be adapted to the situation being analyzed.

$$\begin{aligned}&len(adv\_to\_ids) - count\_double > 0\\&and\ len(adv\_to\_ids) + count\_ind > 2\ and\ count\_scan\_and\_adv > 0\end{aligned} \tag{6.1}$$

- len(adv_to_ids) - count_double > 0
  A double advertisement represents in most cases an attack (that’s why the number is subtracted). A valid node must have at least one directed edge to another node with weight x (this yields a positive number).

- len(adv_to_ids) + count_ind > 2
  An indirect meeting is respected in order to allow the case were user A is not uploading the data but has met the users B and C at the same time. A user has to send data manually to the server in order to prove the personhood. If a user is inactive but the app is running in the background, other users can profit with this indirect method.
- count_scan_and_adv > 0
  Tokens must be exchanged in both directions.

6.2.2 Graph evaluation

The idea is to find bad parts of the graph that represent an attack. The attacker is generating a graph and connects it through some modified users to the rest of the graph. The goal is to vote multiple times with bot votes. A code which generates a good & bad graph and connects them over some edges helps to test the graph evaluation program.

In this part of the analysis, all valid nodes from the node evaluation are taken. An undirected, unweighted graph "graph_for_check" is built from the "adv" graph. For an edge to be included there must be two antiparallel edges with the minimum weight of x. The "graph_for_check" graph is checked if it has a minimum size, in the implemented code this size is 20 nodes. The assumptions for the check are:

- All people meet each other, but it could be indirect over other people
- People meet many other people in two weeks
- The bad parts of the graph are smaller than the good part
- It is hard to connect the good and bad graph, since some real users' phones have to be modified.

The Algorithm

1. Take the biggest connected subgraph and delete all other nodes.
2. Split the remaining graph into two subgraphs called good and bad with the Girvan Newman algorithm. The good graph is the one with more nodes.
3. Count all infected nodes in the good graph, those are all the nodes which had an edge to the bad graph before the graphs were split.
4. If the factor $\frac{number\ of\ bad\ nodes}{number\ of\ infected\ nodes}$ is bigger than a value, for example 1, start over at Step 1.

The Girvan Newman algorithm [13] removes one edge per step. We follow this algorithm until there are two subgraphs which are not connected anymore. The edge to remove is the one with the biggest betweenness centrality. Betweenness centrality is the number of paths going through an edge connecting pairs of nodes with the shortest tracks [14].

Step 4 decides whether the detected graph is a bad one or not. The assumption behind this decision is that it is hard to modify real users' phones. If this factor is equal to one the attacker has to modify for each illegal vote one phone, which would be a big effort.

6.2.3 Test graph evaluation

To test the graph evaluation, a test software is written. Test graphs with known good and bad parts are created and handed over to the graph evaluation function. The results are images of the evaluated graphs and a plot of the wrong true and wrong false evaluations.

6.2.4 Corruption factor

The corruption factor gives an indication of how strong the bad and the good graph are connected. This factor is equals the number of infected nodes while each infected node is connected to one or two bad nodes. One or two is randomly chosen. The test went from corruption factor zero to ten. For each factor, the mean value of five graphs is taken. The good graph has 40 nodes and the bad graph has ten nodes. This test Figure 6.1 shows how the corruption factor has an influence on the performance of the graph evaluation.

Only a few nodes are analyzed as wrong false, but these are the infected nodes. This is a good result. The first graph in Figure 6.2 shows an example of an infected node. If the corruption factor is too high, all bad nodes are highly connected to good nodes and can't be detected. This yields wrong true.

6.2.5 Factor bad nodes infected nodes

For this experiment, the same settings as for the corruption factor are taken except that this time the corruption factor is constantly three and the factor bad nodes / infected nodes is changed form 0.7 until 1.6. From the plot Figrue 6.3 and graphs Figure 6.4 it is visible that the evaluation is not working for very small values. In reality it is difficult for an attacker to realize more infected nodes than bad nodes. If the ratio equals one, which is still low, there are only infected nodes wrong validated, a result that is acceptable.

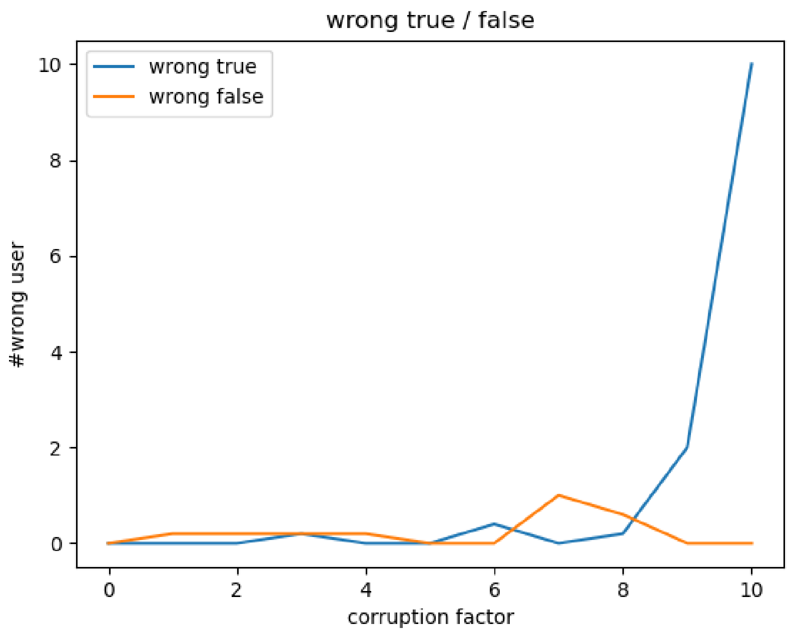


Figure 6.1: Corruption factor plot min value for five graphs

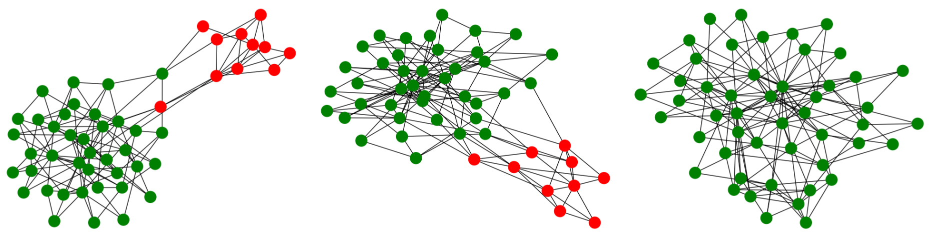


Figure 6.2: Corruption factor = 3, 8, 10 The corruption factor gives an indication of how strong the bad (red) and the good(green) graph are connected.

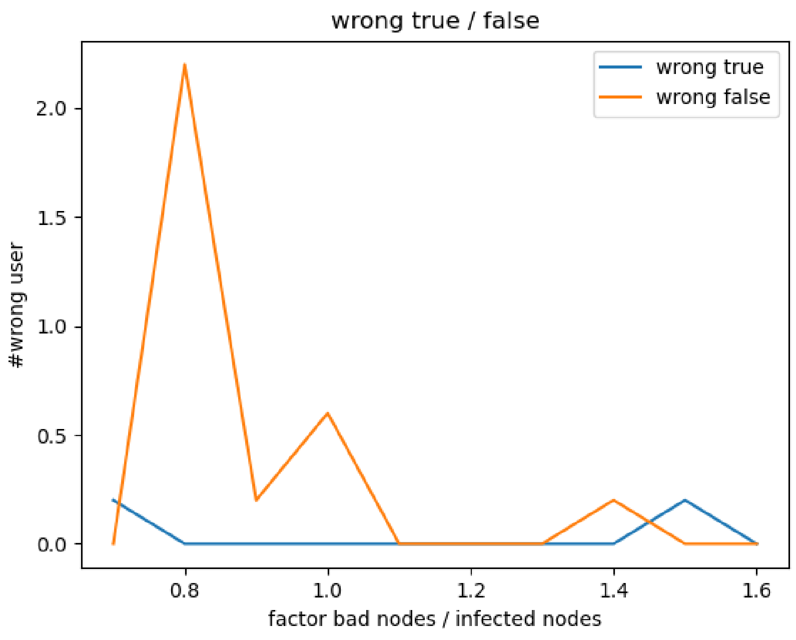


Figure 6.3: Ratio Bad/infected plot, mean value for five graphs is used

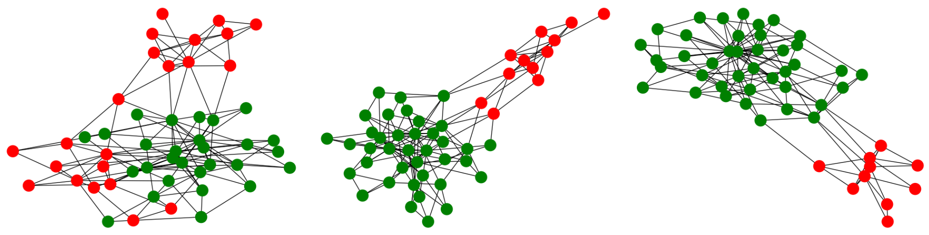


Figure 6.4: three graphs with different bad/infected ratio = 0.7, 1, 1.6

Chapter 7

Future work

7.1 Necessary improvements

The following problems and ideas for improvement were not in the focus of this project and are left out to be covered in future time.

- Phones in different time zones could lead to problems. To solve the problem, all times could be changed to time zone 0 (Unix time in ms is used, starting 1 January 1970 in Greenwich)
- If the Android app is fully closed, the phone stops exchanging tokens for approximately 15min until the Work Manager starts the service again.
- This kind of advertising and scanning for tokens without connecting may not be possible on IOS, since apps under IOS have limited background Bluetooth access [15].
- The validation takes some time, and a good time for this validation has to be chosen. Possibilities are all the time (starts over whenever it is finished and a change occurred), every night, user-forced, after each http post form the phone and others. At the moment, it is done by pressing the button on the debug website or before sending the valid votes. The second implemented option leads to timeouts if too many users have to be evaluated.

7.2 Possible improvements

Android app

- The BLE settings for advertisements and scans are on low power mode. It has to be checked if this is enough to detect a short interaction between people.

- The background work is done with the Work Manager class. The 15min are often extended to a long time up to hours. Another background service could solve this problem, but it must be dealt with Android version differences.
- Instead of saving the advertised/scanned tokens and settings in a file use the SQLite based database, which is provided by Android on each phone. Android has also a special settings support.
- Build the Are You Human app in a way that multiple other apps can use that service to get a proof of personhood. At the moment, this is not possible because if the app is installed twice, it works unnecessarily parallel.

Communication between app and server

- For the communication the json format could be used (I did not know this format until I was finished with the communication code).
- The votes are stored as hex strings, which is not efficient.

Server

- The server is vulnerable against many attacks from the internet. No encryption is used for the communications, with a http post all data can be deleted.
- Split the server code into different classes.

Graph analysis

- The Girvan Newman algorithm targets the few edges which connect the good and the bad parts of the graph. This works good on the provided test graphs, but an attacker could create multiple connections from the few modified phones to the bad graph. So, an algorithm, which tries to identify the infected nodes may be more effective in the real world.
- I have not found a database which simulates the Bluetooth traces of natural interactions. I hope that there will be more realistic datasets that one could use in the future. With the upcoming of the Corona apps the chances are good that the focus is going to turn to Bluetooth interactions.

Bibliography

[1] M. Borge, E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, "Proof-of-personhood: Redemocratizing permissionless cryptocurrencies," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2017, pp. 23–26.

[2] R. Canetti, A. Trachtenberg, and M. Varia, "Anonymous collocation discovery: Harnessing privacy to tame the coronavirus," 2020.

[3] Ubique, "Swisscovid: Dp3t android app for switzerland," 2020, [accessed 6-July-2020]. [Online]. Available: https://github.com/DP-3T/dp3t-app-android-ch

[4] Troncoso et al., "Dp3t - decentralized privacy-preserving proximity tracing," 2020, [accessed 7-July-2020]. [Online]. Available: https://github.com/DP-3T/documents

[5] Wikipedia contributors, "Exposure notification — Wikipedia, the free encyclopedia," 2020, [accessed 9-July-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Exposure_Notification&oldid=966007569

[6] R. Patel, "The other side of react native — limitations and opportunities of react native," 2019, [accessed 6-July-2020]. [Online]. Available: https://medium.com/@ronak8036/limitations-of-react-native-704094f6e299

[7] G. Peal, "Sunsetting react native," 2018, [accessed 6-July-2020]. [Online]. Available: https://medium.com/airbnb-engineering/sunsetting-react-native-1868ba28e30a

[8] Polidea, "React native ble plx," 2020, [accessed 6-July-2020]. [Online]. Available: https://github.com/Polidea/react-native-ble-plx

[9] Facebook Inc., "Native modules," 2020, [accessed 6-July-2020]. [Online]. Available: https://reactnative.dev/docs/native-modules-android

[10] Google Developers, "Migrating your apps to android 10," 2020, [accessed 6-July-2020]. [Online]. Available: https://developer.android.com/about/versions/10/migration

[11] Bluetooth SIG, Inc. , "Bluetooth technology protecting your privacy," 2020, [accessed 6-July-2020]. [Online]. Available: https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/

[12] Google Developers, “Android jetpack,” 2020, [accessed 6-July-2020]. [Online]. Available: https://developer.android.com/jetpack

[13] NetworkX Developers, “networkx algorithms community centrality girvan newman,” 2019, [accessed 6-July-2020]. [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html

[14] NetworkX Developers, “edge betweenness centrality,” 2014, [accessed 6-July-2020]. [Online]. Available: https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.algorithms.centrality.edge_betweenness_centrality.html

[15] Apple Inc., “Core bluetooth background processing for ios apps,” 2013, [accessed 11-July-2020]. [Online]. Available: https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothBackgroundProcessingForIOSApps/PerformingTasksWhileYourAppIsInTheBackground.html

Appendix A

Code

The code is saved at

https://gitlab.ethz.ch/disco-students/fs20/dwerder-are-you-human

A.1 Android app

The modified code from the Android app is (MainActivity.java holds set up instructions)

- The Are You Human classes
  in Android Studio/app/src/main/java/com/example/areyouhuman:
  - **AreYouHuman.java** This is the main class of Are You Human part and holds all defines
  - AreYouHumanWorker.java
  - Bluetooth.java
  - FileModifier.java
  - Helper.java
  - ServerCommunication.java
  - Settings.java
- The app interaction (Example of using the AreYouHuman class)
  - **MainActivity.java** Interface class between Are You Human functions and user input
    in Android Studio/app/src/main/java/com/example/areyouhuman
  - **activity_main.xml** Holds the layout of the app
    in Android Studio/app/src/main/res/layout
  - **AndroidManifest.xml** Holds the permissions
    in Android Studio/app/src/main

  - **build.gradle** Holds the settings and imported libraries in Android Studio

A.2 Server

Files to run the server

- app.py holds python part of the server and is the main file
- templates/home.html is the visible debug website
- templates/valid.html shows all valid IDs
- static/.. holds the logo and included copied files for the website

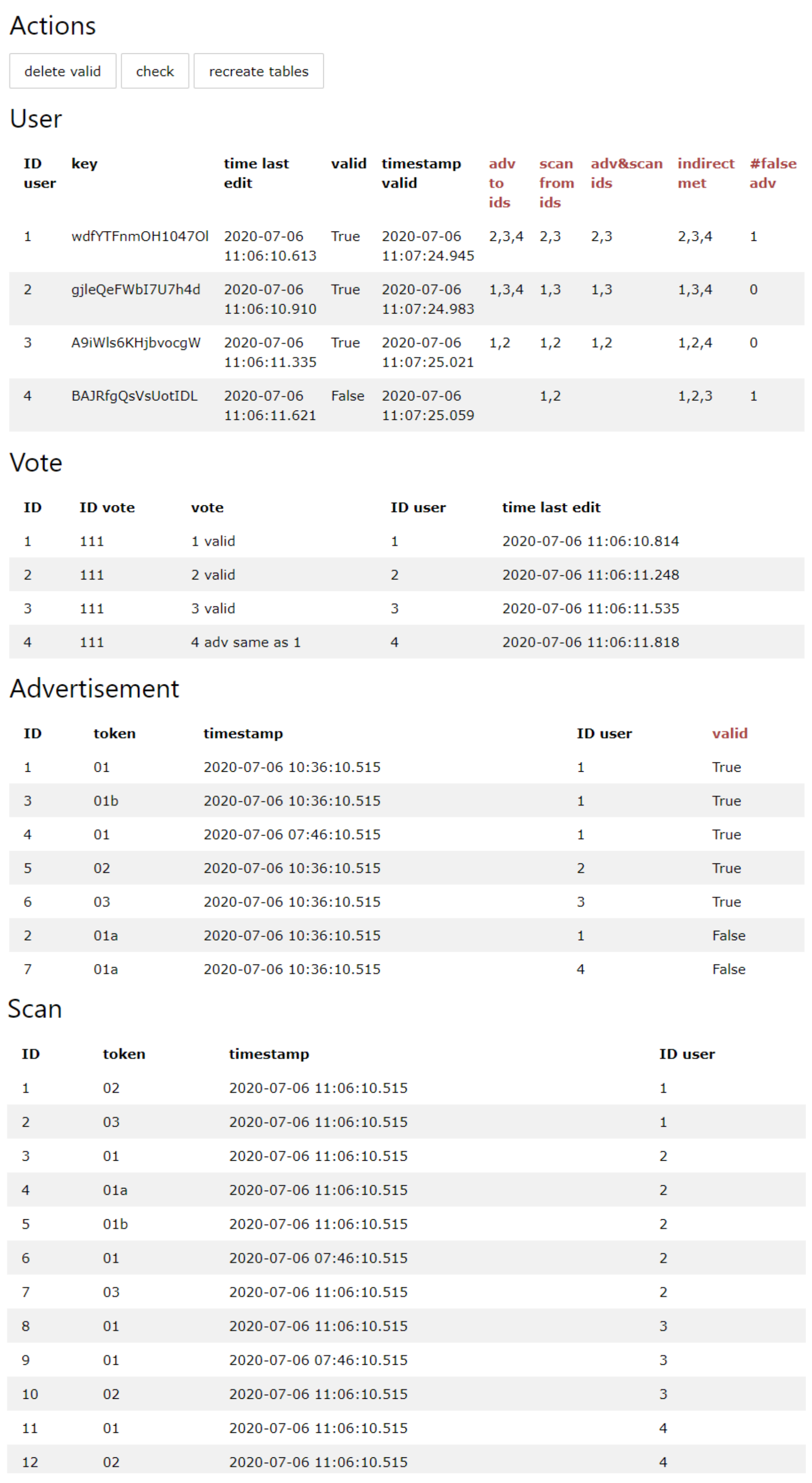
Files which help but not used for the server

- generate.py Generates graphs
- clients.py Sends test users to the server, for example from the generated graphs
- getVotes.py Example of a third-party server which collects the valid votes
- readme.txt Holds the instruction to set up Visual Studio Code and PostgreSQL on Windows 10
- commands_raspi.sh Instructions to set up a Raspberry Pi with Flask and PostgreSQL (In the end I used only VS code on windows 10)

Appendix B

Example for Node evaluation

This is a generated example where we have three good nodes and one bad node. On the debug website we see the tokens and the evaluation. The graphs are created from the code.



Actions

User

ID user	key	time last edit	valid	timestamp valid	adv to ids	scan from ids	adv&scan ids	indirect met	#false adv
1	wdfYTFnmOH1047Ol	2020-07-06 11:06:10.613	True	2020-07-06 11:07:24.945	2,3,4	2,3	2,3	2,3,4	1
2	gjleQeFWbI7U7h4d	2020-07-06 11:06:10.910	True	2020-07-06 11:07:24.983	1,3,4	1,3	1,3	1,3,4	0
3	A9iWls6KHjbvocgW	2020-07-06 11:06:11.335	True	2020-07-06 11:07:25.021	1,2	1,2	1,2	1,2,4	0
4	BAJRfgQsVsUotIDL	2020-07-06 11:06:11.621	False	2020-07-06 11:07:25.059		1,2		1,2,3	1

Vote

ID	ID vote	vote	ID user	time last edit
1	111	1 valid	1	2020-07-06 11:06:10.814
2	111	2 valid	2	2020-07-06 11:06:11.248
3	111	3 valid	3	2020-07-06 11:06:11.535
4	111	4 adv same as 1	4	2020-07-06 11:06:11.818

Advertisement

ID	token	timestamp	ID user	valid
1	01	2020-07-06 10:36:10.515	1	True
3	01b	2020-07-06 10:36:10.515	1	True
4	01	2020-07-06 07:46:10.515	1	True
5	02	2020-07-06 10:36:10.515	2	True
6	03	2020-07-06 10:36:10.515	3	True
2	01a	2020-07-06 10:36:10.515	1	False
7	01a	2020-07-06 10:36:10.515	4	False

Scan

ID	token	timestamp	ID user
1	02	2020-07-06 11:06:10.515	1
2	03	2020-07-06 11:06:10.515	1
3	01	2020-07-06 11:06:10.515	2
4	01a	2020-07-06 11:06:10.515	2
5	01b	2020-07-06 11:06:10.515	2
6	01	2020-07-06 07:46:10.515	2
7	03	2020-07-06 11:06:10.515	2
8	01	2020-07-06 11:06:10.515	3
9	01	2020-07-06 07:46:10.515	3
10	02	2020-07-06 11:06:10.515	3
11	01	2020-07-06 11:06:10.515	4
12	02	2020-07-06 11:06:10.515	4

Figure B.1: Website

The red indicated columns are only created for this website. On a release version delete the columns in the database.

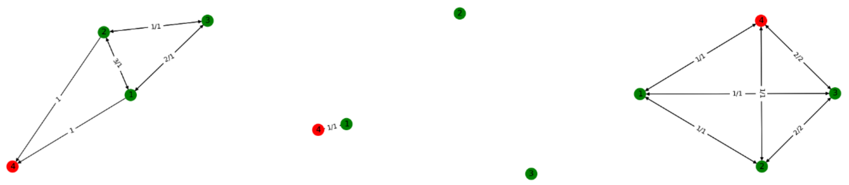


Figure B.2: Graphs: adv, double, ind