Protected U

UCLA CS M117

Spring 2018

Chiao Lu (204848946)

SangJi Lyu (704850507)

Mu-Te Shen (904676710)

Shen Teng (104758168)

Motivation

Through continuous invention and renovation of technology, identifying yourself has become easier than ever before. Unfortunately, impersonating someone has become easier as well. As long as one has your physical identification card, they can pretend to be you. For example, the barcode on our UCLA ID card is simply plain text encoding of our ID number. To make the hacking process even simpler, one can take a picture of that barcode, and do a lot of things in the name of you, without being noticed since the card is not "lost." Students are already busy with school work, and the last thing they should worry about is the safety of their identity. Our goal is to propose a new way of identification with the security that algorithm can provide. Our service/ application will constantly create a string hashed from the ID number, timestamp, and secret salt every once in a while, and that's the only valid ID a student can use. Each time a student needs to pull out their ID, like in the dining hall or at libraries, they can either use the generated QR code, or use the NFC (simply tap) to connect with the receiver. With our app, one doesn't have to worry about lost ID, or malicious copying of the ID.

Technical Background

Near Field Communication (NFC)

In NFC peer to peer mode, devices are connected through physical contact, and allows exchange of information in adhoc style. This was done to achieve maximum security for our app. In our application, we worked closely with Android; thus, used a feature in Android (mobile) operating system called Android Beam. Android Beam authorize data exchange via NFC, and allows one to read NDEF messages from other NFC device. In other words, for every Peer to Peer exchange there are two active devices in which one functions as a target and other as an initiator. Thus, any NDEF message sent to a target, the target is able to alter the data that was sent, and send a message in response.

SHA-256 Cryptographic Hash Algorithm

SHA-256 Cryptographic Hash Algorithm. We used SHA-256 to generate a unique 32 byte signature for the student's ID. This is useful as hash is a one way function; thus, near impossible to be decrypted to the text it once was. Our app also adds a timestamp and a secret salt to the hashing, and the salt is known only to the server and the specific student. The timestamp adds the security so that the generated hash string cannot be reused by a theft.

NoSQL & Google Cloud Platform

Google Cloud Platform and NoSQL were both used for, mainly, data storage and retrieval of data. Google Cloud Platform was used for database such as students name, email, and student ID. Google Cloud Platform functioned as our server, since the server needed a public IP, and

could not be on our private computer (NAT).

-Implementation

Frontend

All the apps are written using Android Studio. Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

Because the raw NFC communication is too complicated, we chose Android Beam. Android Beam is a feature of the Android mobile operating system that allows data to be transferred via near field communication (NFC). In short, Android Beam is a wrapper API for Android NFC libraries.

One important technique we used is a method called `AsyncMethod`. Using this is important because calling `HttpRequest` will block. In other words, if we call `AsyncMethod` in the main thread, our app will block and crash.

The basic design flow is as follows. First, the user can type in `EditText` box. Then, the app will take the user input and pass it to the server using `HttpRequest`. The request is sent using `AsyncMethod` because we don't want to block the main thread. The server then sends an `HttpResponse`. The `HttpResponse` will be handled by our app and put inside `TextView` or `Toast`.

Backend

Server

We use Python Flask to develop our server. Python Flask is a microframework for Python based on Werkzeug and Jinja2. It is an easy framework for web design. For the purpose of this project, we only utilize the server part. Flask provides easily used URL routing mechanism, so that we can handle the request using Python.

Our server provide 3 functions for the application and 2 more function for testing. All of the data, to and from the server, are transferred in JSON string. We use HTTP GET request to contact server for its ease of use, and we'll switch to POST when our app is more mature.

The following 5 functions are available through our server:

Function: /createuser

This function takes the name and email the student is created, and return with a verification code that would be given to the student.

Input:

{“name”:”student”,
“email”:”email”}

Output:

{"result":"<true/false>",
"name":"name",
"vcode":"<6 digit niumber>"}

Function: /verifyuser

This function verify the student's identity after registrar has sent the student’s info to the server, and return with the student's id, primaryKey, and salt.

Input:

{“name”:”<student>”,
“vcode”:”<6digits code>”}

Output

{"result":"<true/false>",
"primaryKey":"<the key>",
"studentID":"<ID>",
"salt":"<salt>"}

Function: /verifyid

This function verified the hashed studentID with the database and return the result.

Input:

{“primaryKey”:”thekey”,
“IDHash”:”thehash”}

Output:

{"result":"<true/false>",
"name":"<studentname>",
"email":"<studentemail>"}

Function: /find ---for testing purpose

This function obtains the student’s info with name and email

Input:

{“name”:”student”,
“Email”:”email”}

Output:

{"result":"<true/false>",
"primaryKey":"<theKey>",

"studentID":"<id>",
"salt":"<salt>"}

Function: /retrieveInfo ---for testing purpose

This function obtains the student's info with the primary key.

Input:

{“primaryKey”:”thekey”}

Output:

{"result":"<true/false>",
"primaryKey":"<theKey>",
"name":"<studentname>",
"email":"<studentemail>",
"studentID":"<theID>",
"salt":"<theSalt>"}

The server is served on Google Cloud Platform with the address:

http://protected-u.appspot.com/<function>?data=<Input>

Database

Our NoSQL database includes 3 models with the following schema:

MaxID:

maxID: integer

PendingUser:

name: string
email: string
studentid: string
salt: string
vcode: string
timestamp: Datetime

User:

name: string
email: string
studentid: string
salt: string

Obstacles

Through great teamwork and communication, luckily, we did not come across many obstacles. Notable obstacles we overcame was when we were building the front hand and back hand. Front hand was written in Java and back hand was written in python; thus, it was difficult to produce hash results. After multiple different approaches to solve this issue, we fixed it by tweaking and adjusting the code. Another obstacle was that the Server needed a public IP therefore near impossible to put the server on our private computer (NAT). One solution to this was using Google Cloud Platform Server that allowed us to use the public IP and access the server from everywhere.

Future/Conclusion

Our group wish to make this application a reality and further improve on the application. We wish the make the application more presentable and user friendly. Such as changing the layout of the application, add more color, and simple setting options. Ever since the DUO application, two factor authentication process that was required for all UCLA students and faculty to have, everyone must have a active phone in order to online onto My UCLA or CCLE. Keeping this in mind, our app is another authentication using one's phone. Students and faculty often forget the importance of School ID. School ID allow us to borrow books, computers, chargers, etc. In addition to all of this it gives access to dormitories, restricted schools that only graduate students are able to access, and it functions as a debit card. If a student ID is lost so many things can go wrong, and our app ensures the safety of UCLA students and faculty from identity theft.

Responsibility Assignment

Server Development: Mu-Te Shen
Android App Development: Chiao Lu and Shen Teng
Testing: SangJi

Video demo: https://youtu.be/bVTEiDR5GSE