Homework 5

COM402 — Information Security and Privacy 2019

- Submission instructions are on Moodle. Submissions sent after the deadline **WILL NOT** be graded.
- In the event that you find vulnerabilities, you are welcome to disclose them to us (can even have a bonus !)
- Do not forget to submit your source files to Moodle along with your tokens.

(30p) Exercise 1: Listen Carefully

Steganography is the art of hiding data within data. Data can be in the form of files (images, video, audio etc.) or messages. The goal of this exercise is to help you explore a simple steganographic technique and understand how pieces of information can be concealed in files without attracting undue attention.

You are given an audio clip in the WAV file format. You have to find the token in this file. The token is of the form: COM402{**<token>**}. When you submit the token, only submit the portion within the curly brackets, i.e., remove the COM402{} part.

You can find your file in the folder at:
https://drive.google.com/drive/folders/190X0pJ3KxRwKloWf1MThOIMbV3WLGUJ2?usp=sharing
Your file name will be of the form <your_email_address>.wav.
To solve this exercise, you need to look at commonly used steganographic techniques and figure out which technique has been used to hide the token.

Note: While the exercise can be completed without any packages, the wave Python module can be useful when dealing with WAV files. Link: https://docs.python.org/3/library/wave.html **.**
Another useful tool in Steganography analysis is ExifTool - it can give you basic information about different file types.

(30p) Exercise 2: Just In Time

In this exercise, you're being asked to guess credentials on the website.

To login, you must send a POST request with a JSON body looking like:

```
{ "email": <youremail>,  "token": <yourguessedtoken> }
```

to

`http://com402.epfl.ch/hw5/ex2`

Don't try to brute force the token. There are much faster ways to guess the correct token.

For example, the developer here used a modified function to compare strings which express some very specific timing behavior for each valid character in the submitted token…

The response code is `500` when the token is invalid and `200` when the token is valid. Look at the body of the response, you can get some useful information too. The "real" token that you can submit to moodle will be in the response body.

This exercise will require some patience and trial-and-error, as time in networks is never 100% accurate. In order to be precise, you should calibrate your measurements first, before trying to do any guessing on the token.

Exercise 3: Not so secure "AI"

A company named SecureHealth™ provides an API to query what they call a "Health AI" in a private way. You send as input a feature vector representing different health-related features of a patient, and get back a score of how likely is some given diagnosis.

According to their fancy website, the company uses some complicated "novel medical AI method". Your institution has paid a lot of money to get access to *two* of their AI models, which are used for different health diagnoses.

SecureHealth™ relies on homomorphic encryption so as not to see the users' inputs. Homomorphic encryption is a malleable form of encryption which allows the computation of one or more operations on ciphertexts without decrypting them. In particular, they use Paillier's Encryption Scheme (see lectures).

Before encryption, the feature vectors that are provided to SecureHealth™ API should contain 12 binary values, encoding different health-related features of a patient. Given an encryption of this feature vector, the API returns an encrypted integer score. You should be able to decrypt this score and use it for diagnosis.

Here are the details of their API.

SecureHealth™ API

`POST http://com402.epfl.ch/hw5/ex3/`**`securehealth/prediction_service`**
Parameters:

- `email`

- `pk: the Paillier public key modulus part (n), an integer.`
- `encrypted_input: Paillier-encrypted vector, a list of integers.`
- `model: ID of the SecureHealth™ model to use, one of {1, 2}`

Parameters should be encoded as JSON.

If all inputs are good, returns 200 and the JSON response:

- `encrypted_prediction: Paillier-encrypted output, an integer`

Part a. (15p)

You want to use model **1** to get a score of a health diagnosis of a patient. You should use the following API to obtain a feature vector containing the patient's private feature vector:

`POST com402.epfl.ch/hw5/ex3/`**`get_input`**
Parameters:

- `email`

Returns the patient's feature vector as JSON response:

- `x: the patient's feature vector, a list of integers`

Now, you should homomorphically encrypt this vector using a Paillier cryptosystem. You can use common encryption libraries or code it yourself. If using the library, please note that the public key here is simply the integer *n* (considering *g=n+1*). Once the vector is encrypted, submit it via a POST to **`securehealth/prediction_service`** with the public key associated with the encryption. It will return an encrypted prediction that you should be able to decrypt with the secret key.

Once you have the *decrypted* prediction, use the following API endpoint to obtain your token.

`POST http://com402.epfl.ch/hw5/ex3/`**`get_token_1`**
Parameters:

- `email`
- `prediction: Decrypted prediction from the model, an integer`

If all good, returns a JSON response:

- `token: your beautiful token`

Part b. (25p)

You found an interview of a former SecureHealth™ chief data scientist. Apparently, their "AI" actually uses a method for medical classification called SLIM, which has great properties, e.g. its decisions are easy to interpret.

You decide to read up on SLIM a bit, and find out that a SLIM model is a linear model. Thus, the outputs from the API, after decryption, are of this form: `y = w*x + b`, where `x` is your input feature vector, `w` is their model's weights, `b` is the model's bias term, and * denotes the dot product. The weights and bias are constrained to be non-negative integers.

Although once a SLIM model is trained, it looks like a simple linear model, the training itself is computationally expensive and requires medical data. You are wondering if you can steal their precious model (see lectures).

Can you figure out the parameters of the model **2**? Use the following API to get your token.

`POST com402.epfl.ch/hw5/ex3/`**`get_token_2`**
Parameters:

- `email`
- `weights: Your guessed w vector, a list of integers`
- `bias: Your guessed bias term b, an integer`

If all is good, returns a JSON response:

- `token: your beautiful token`