UNIVERSITY OF WATERLOO
David R. Cheriton School of Computer Science

CS 458/658 **Computer Security and Privacy** **Spring 2014**
Urs Hengartner

ASSIGNMENT 3

Assignment due date: **Wednesday, July 30, 2014 @ 3:00 pm**

Total marks: 40

Written Response Questions TA: Hassan Khan
Office hours: Thursdays 09:30-10:30 am at DC3332, or by appointment.

Applied/Programming Questions TA: Jalaj Upadhyay
Office hours: Wednesdays 12:00-1:00 pm at DC3332, or by appointment.

Please use Piazza for all communication. Ask a private question if necessary.

Written Response Questions [20 marks]

Note: For written questions, please be sure to use complete, grammatically correct sentences where appropriate. You will be marked on the presentation and clarity of your answers as well as the content.

1. [8 marks total] **Anonymous communications systems.**
Answer the following questions. Refer to `https://www.torproject.org/docs/faq.html.en` if you encounter unfamiliar terminology or concepts, or if you want to know more about Tor.

   (a) [2 marks] Tor prevents a user from becoming linkable (in the long term) to a website based on her/his computer's source address. Give a way in which a website may link a user (in the long term) despite her/his usage of Tor.

   (b) [2 marks] Tor can be relatively slow. To mitigate this problem, Tor routers can self report their available bandwidth to the directory servers that provide a listing of Tor routers. A user can then choose routers with good bandwidth. Explain how an adversary can take advantage of this feature to compromise the user's privacy.

   (c) [2 marks] uWaterloo decides that its students spend too much time on Facebook and has its firewall drop all traffic to Facebook. Therefore, Alice, a CS 458 student and

Facebook addict, switches to Tor to access Facebook. As it turns out, Alice accesses only a very small set of websites from the university, all of them with Tor (e.g., Facebook, CBC, and digg), and the university knows about this set. How can the university still learn of each of Alice's visit to Facebook? Assume that Alice uses her laptop, which is not under the university's control, and that Alice always uses the same path of Tor routers. uWaterloo does not control the Tor routers or the websites accessed by Alice.

(d) [2 marks] uWaterloo notices that many students are switching to Tor and that they access many different websites, which makes the detection mechanism from the previous question less reliable. Explain how putting an advertisement on the Facebook website, where this advertisement is provided by a uWaterloo server, can let the university uniquely identify students who access Facebook. A user's browser retrieves the advertisement (via Tor) directly from the uWaterloo server, that is, Facebook is no longer involved. Assume that the link to the advertisement on the Facebook page is static and the same for all users. The advertisement is a static picture and its size (in bytes) is comparable to other advertisements on the Internet. Assume that a user's browser is configured not to send the Referer header to a server.

2. [6 marks total] **A Practical Attack.**
Suppose the CS458 course staff are testing a new "secure" mark submission system. Midterm and final exam marks are encrypted by a TA before being sent to the mark database. To convince you that the new system is secure, a TA demonstrates it using a fake student account:

```
$echo aaaaaaaa 100 100 > message

$openssl enc -aes-128-cbc -iv 00000000000000000000000000000000
  -pass password -in message -out ciphertext

$od -x ciphertext
0000000 6153 746c 6465 5f5f b935 2b69 2f53 2b0d
0000020 733f b288 72c1 26bf a340 2f3c f7b6 0959
0000040 8333 e650 277a 718b 540c 48a4 ee75 ecb7
0000060
```

This (ciphertext, IV) tuple is sent to the database to update marks. The encryption/decryption key is derived from the password. The TA never reveals the password during the demo. You can assume that it was read from a file or other private source and is not actually shown during the demo. The TA knows that an IV is not meant to be secret, and does not hide it.

The database receives the encrypted marks and it can recover them by decrypting:

```
$openssl enc -d -aes-128-cbc -iv 00000000000000000000000000000000
  -pass password -in ciphertext

aaaaaaaa 100 100
```

While answering student questions, the TA explains that each update is of the form [8 character userid][space][3 digit midterm mark][space][3 digit exam mark]. The database knows an update is valid if the userid is found in the database and the marks are between 000 and 100. The TA assumes that since AES is secure, no one but a TA with the password will be able to construct a valid mark submission.

This question explores the security of TAs' approach to encryption. You might find Wikipedia's article on the CBC mode of operation useful for this question: `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher-block_chaining_.28CBC.29`

(a) [3 marks] Demonstrate that the above system is insecure. Specifically, show how you need to change the IV such that when sent with the ciphertext above to the database the update results in a mark update for your personal userid instead of user "aaaaaaaa" (1 mark). Give the technical reason why your attack works (2 marks). You do not need to compute the actual changed IV. You could assume that the message fits entirely within the first block.

(b) [3 marks] Which CIA principle(s) is/are violated (1 mark)? How would you fix it (2 marks)?

3. [6 marks total] **Database Security.**
SQL uses the GRANT operation to grant a user the right to execute an SQL operation on a table or a view, maybe only for a specific set of columns. For example, the operation

```
GRANT SELECT, UPDATE (Day, Flight)
  ON TABLE Diary
  TO Arthur, Zoe
```

grants Arthur and Zoe the right to select any field of table Diary and to update the fields Day and Flight in table Diary. It is also possible to grant access rights on a view (here, the keyword VIEW would be used instead of TABLE).

(a) [4 marks] Consider a table Accounts with records (CustomerName, AccountNumber, Balance, CreditRating). Define an access structure, e.g., through views, so that:

- Customer Alice can read (only) her own account;
- A clerk can read all fields other than CreditRating and update Balance for all accounts;
- A manager can create new records, read all fields, and update CreditRating for all accounts.

You can assume that access rights can be granted to groups of people (e.g., Manager).

(b) [2 marks] Explain how blind writes can happen in a view, that is, a user sees some data in her view of a table, modifies this data (assuming she has the required UPDATE privilege), and then the modified data disappears from the user's view.

Applied/Programming Questions [20 marks]

4. [8 marks total] **GnuPG.**
   A GnuPG public key for `jkupadhy@cs.uwaterloo.ca` is provided along with the assignment on the course website (jkupadhy.asc). Perform the following tasks. You can install GnuPG on your own computer, or use the version we have installed on the ugster machines.

   (a) [2 marks] Generate a GnuPG key pair for yourself. Use the RSA and RSA algorithm option, your real name, and an email address of your-userid@uwaterloo.ca. Export this key using ASCII armor into a file called **key.asc**. [Note: older versions of GnuPG might not have the RSA and RSA algorithm option, so check that the version you are using has this option. The ugster machines have a new enough version, but the student.cs machine does not.]

   (b) [2 marks] Use this key to sign (not local-sign) the jkupadhy@cs.uwaterloo.ca key. Its true fingerprint is: `86BB F73A 20E8 1304 8E18 6D83 DA04 A4D4 F2F7 C84E`. Export your signed version of the jkupadhy key into a file called **jkupadhy-signed.asc**; be sure to use ASCII armor. [Note: signing a key is not the same operation as signing a message.]

   (c) [2 marks] Create a message containing your userid and name. Sign it using the key you generated, and encrypt it to the jkupadhy key. You should do both the encryption and signature in a single operation. Make sure to use ASCII armor, and save the output in a file called **message.asc**.

   (d) [2 marks] Briefly explain the importance of fingerprints in GnuPG. In particular, explain how users should check fingerprints and what type of attacks are possible if users do not follow this procedure properly.

5. [12 marks total] **Database Record Re-identification**

   Not-So-Secure Hospital (NSSH) in Waterloo recently published a record of infectious diseases in Waterloo Region to help health researchers come up with better plans in the future. Understanding that it is important to preserve the privacy of these health-related records, they redacted the *Name*, *Telephone*, and the *date and month* in *Date of birth* from these patient records. A citizens' group has sued the hospital, claiming that the hospital did not do enough to anonymize the records. They have retained you as an expert witness. Your job is to demonstrate that, indeed, the released records do identify some individuals and the diseases they have.

   You have been provided three .csv (comma-separated value) files:

   - **Poll-Data.csv**: A (partial) public polling list from the recent provincial election. It contains citizen information with columns *Name, Telephone, Date of birth, Gender, Postal code*.
   - **Disease-Records.csv**: The hospital's infectious diseases records with columns *Name, Telephone, Date of birth (date and month), Date of birth (year), Gender, Postal code,*

Disease. Note that the columns *Name, Telephone*, and *Date of birth* (date and month) have been redacted. Note that, in Disease-Records.csv, the year is simply split off into a separate column in the redacted data-set.

- **Queries.csv**: Uncertain records with columns *Name, Telephone, Year of birth, Gender, Postal code, Disease*. These queries are generated by an adversary who would like to know whether a given person has a given disease.

You have been assigned two tasks to aid in the case. The first task (part (a)) is to find all patients that can be re-identified with 100% certainty. Furthermore, to better understand the scope of the privacy leak, your second task (part (b)) is to provide a tool that can compute the probability of a patient having a given disease.

(a) Based on Poll-Data.csv and Disease-Records.csv, create a comma-separated value file **Reidentified-Data.csv** with the columns *Name, Telephone, Year of birth, Gender, Postal code, Disease*. It should contain as many confirmed re-identified records as you can find. In particular, you should find all cases where a single Poll-Data entry matches a single Disease-Records entry. There are other scenarios where re-identification may be possible (e.g., using a homogeneity attack) but you can ignore these for simplicity.

(b) Based on Disease-Records.csv, for each record in the file Queries.csv, compute the probability that the patient identified by the given attributes has the given disease. Create a file **Query-Probs.txt** with one number per line (rounded to two decimal places), corresponding to the probability of each diagnosis in the query file (in the same order, no carriage returns, one newline at the end of each line).

Write a program to help you with this. You can use any programming language you like, but it will need to compile (if applicable) and run on the **ugster** (*not UML*) environment. In addition to the data files described above, we have also provided you with a small test data set and the correct output files (“Small-*”). Furthermore, the following is the SHA1 hash of the correct Reidentified-Data.csv output file of the full data set (sorted with no additional whitespace between data items, no carriage returns in the file, and a single newline at the end of the file). There should be 49 items in the Reidentified-Data.csv file.

```
$ sort Reidentified-Data.csv | sha1sum
8c1a12d7300bee5980f4c2930532c895b7695849  -
```

Note that we *will* run your code on other data sets as well. The other data sets will be well formed; that is, they will not contain syntax errors.

You will need to **submit the following deliverables:**

- Your source code implementing the described functionality.
- Your output for part (a) in a file called “Reidentified-Data.csv”.
- Your output for part (b) in a file called “Query-Probs.txt”.

- A **Makefile**, with the following three targets:
  1. **build**: Typing "make build" should compile your code.
  2. **id**: Typing "make id" should run your re-identification app on the two files "Disease-Records.csv" and "Poll-Data.csv", which can be assumed to exist in the working directory. The output should be sent to `stdout`.
  3. **query**: Typing "make query" should run your query app on the three files "Disease-Records.csv", "Poll-Data.csv", and "Queries.csv" which can be assumed to exist in the working directory. The output should be sent to `stdout`.

Marking:

- 6 marks for each of (a) and (b):

  +1 mark if your program produces the correct output for the small data set

  +2 marks if your program produces the correct output for the provided full data set

  +3 marks if your program produces the correct output for our additional tests
- -2 marks if the Reidentified-Data.csv and Query-Probs.txt files you submit do not match the ones output by your program
- -2 marks if your Makefile does not work as specified

What to hand in

It is very important that you follow the rules outlined in the Assignments section of the LEARN course site for submitting your assignment. Otherwise we may not be able to mark your assignment and you may lose partial or all marks.

By the **a3 deadline**, you are required to hand in:

a3.pdf: A PDF file containing your written answers for all questions.

key.asc: Your GnuPG public key from question 4.

jkupadhy-signed.asc: Your signed version of jkupadhy's public key from question 4.

message.asc: Your encrypted and signed message from question 4.

Makefile: The Makefile used to compile your program from question 5. It must have targets "make build", "make id", and "make query", as explained in question 5. Submission of a Makefile is mandatory; otherwise we won't be able to mark your solution.

q5*: The source code to your program from question 5. All of your source code filenames should start with “q5”, but can be anything (with any extension) after that. This allows you to write your program in any language, provided it will compile on the ugster machines.

Reidentified-Data.csv, Query-Probs.txt: Your results for the given data set from question 5.