Total points: 75 (+15 bonus)

Assignment 3

Due: 11:59 PM, 1st Dec, 2017 (Fri)

Written assignment

1. Write down which of the four data privacy techniques (k-anonymity, differential privacy, secure multiparty computation, private information retrieval) is most suitable to resolve each of the following scenarios, and explain *why it is more suitable than the others*. (It is possible for the same technique to be best for two different parts.)

   (a) [4 points] Alice and Bob have recently been hired at a factory, and they asked the factory to compute the mean lifespan of all workers at the factory and tell them. The factory argues that this is private information, because it may reveal health details of their fellow workers.

   (b) [4 points] A social science researcher has a new theory about how people consume food in malls. The researcher wants to retrieve the company's big data to test this theory, but the company argues that the information is private. The company wants to know the researcher's theory to test it themselves, but the researcher doesn't want to reveal the theory because the researcher wants the academic credit for publishing the novel theory.

   (c) [4 points] You are about to start a new company, and you would like to purchase its web domain as the company's website. However, you are aware of the practice of cybersquatting; people may purchase the domain first if they know it is in demand, and sell it to you at an elevated price. You want to know if the domain is still available, but you are worried that attempting a DNS query for the domain will lead to some DNS servers purchasing it immediately for cybersquatting.

   (d) [4 points] You want to install a GPS-enabled application on your mobile device so that you can use it as a map for hiking. You need to regularly look at your surroundings to find the correct path, but you don't want to submit your location to the remote server with the map. (Assume that you trust the application will do what you want it to do.)

Solution: 1 point each for explaining right technique, and why the other techniques are wrong. For parts where the chief difficulty is computational or precision, half a mark is possible for claiming it is suitable when it is not or vice-versa if the explanation is correct. (This means at most 3 points are possible if the wrong technique is chosen as the right technique.)

(a) k-anonymity: no, too imprecise, and there is no control of how imprecise it is. Differential privacy: yes, add noise to lifespan of workers and return a result with only a small perturbation of noise. SMP: no, too computationally expensive for no reason. PIR: no, the data is private.

(b) k-anonymity: yes, can anonymize away sensitive and private information. Differential privacy: no, the company cannot know if the researcher's queries will be differentially private. SMP: no, the query is private. PIR: no, the data is private.

(c) k-anonymity: no, too imprecise. Differential privacy: no, the query is private. SMP: no, the query is private. PIR: yes, the query can be made in a privacy-preserving manner to stop the DNS server from knowing what you asked for. (This is assuming that the DNS server will cooperate in supporting such a query.)

(d) k-anonymity: no, too imprecise. Differential privacy: no, the query is private. SMP: no, the query is private. PIR: yes, same as above.

A hint about when PIR is appropriate is to ask yourself: Would it be appropriate to simply let the user download the entire database if bandwidth restrictions are ignored? If so, PIR could be appropriate to reduce bandwidth costs while still allowing the query to be private. In (c) and (d) it is appropriate for the user to know the entire database (the DNS server list, and the real world map, respectively).

2. Company A always backs up its data at 11:59 PM. Every Sunday, a full backup is performed; partial backups are performed on the other weekdays. Every Monday, Tuesday, and Wednesday, a differential backup is performed. Every Thursday, Friday, and Saturday, an incremental backup is performed.

   For simplicity, assume that the data consists of D files of the same size. On each day, a percentage p of the files is changed. Also assume that the files are small enough that it will not be subdivided into chunks for partial backups.

   (a) [3 points] Assume that no file is changed twice during a week. Calculate the amount of storage required to store each of the seven backups for the week, given in terms of the amount of files stored. Which partial backup is the largest?

   Solution: From Sunday to Saturday: $D, pD, 2pD, 3pD, pD, pD, pD$. Wednesday.

   (b) [4 points] Now, assume that a file can be changed twice. A file, if changed twice or more, will never return to its original state. Which of the above answers, if any, is expected to be different at the mean? Could the largest partial backup be different as well? Show your workings.

Solution: Only the sizes of Tuesday and Wednesday will change. For Tuesday, percentage of files changed zero times is expected to be $(1-p)^2$, so the backup will be of size $(1-(1-p)^2)D$. For Wednesday, percentage of files changed zero times is expected to be $(1-p)^3$, so the backup will be of size $(1-(1-p)^3)D$. The largest partial backup does not change.

(c) [3 points] When checking the integrity of the backups for a certain week, Company A finds that because one of the partial backups was corrupted, it is not possible to restore to 3 days of the week. Which partial backup has been corrupted? Explain why.

Solution: Thursday. If Thursday's backup is corrupted, then the company cannot restore to Thursday, Friday, and Saturday of that week.

(d) [2 points] Explain why backups can be encrypted versions of the original data, but backups cannot be hashes of the original data.

Solution: Hashing cannot be inverted, so you cannot recover the original data. Encryption can be inverted.

3. Assume the following numbers describe the Bitcoin ecosystem:

I. The size of each block is at most 1 MB.

II. Each transaction is at least 166 bytes in size. A block is allowed to contain nothing but transactions.

III. In total, all miners produce 10^{19} hashes per second. All miners cooperate to ensure that no two miners will attempt the same hash.

IV. It costs \$10,000 to buy a mining device to produce 10^{17} hashes per second, and \$1 to run it per second for every miner.

V. The reward for a successful block hash is 12 bitcoins. [1]

VI. Assume there are no transaction fees.

(a) [3 points] What is the minimum price of a bitcoin such that miners avoid making a running loss?

(b) [2 points] What is the maximum number of transactions per second?

(c) [3 points] Now, suppose that the price of a bitcoin stays the same as your answer in (a), but the reward for a successful block hash has been halved to 6 bitcoins. What is the mean transaction fee (in \$) required for each transaction so that miners continue to avoid making a running loss? (Show your workings clearly so that an error from (a) or (b) will not affect your marks for this part.)

[1] In reality, it is currently 12.5 bitcoins.

(d) [4 points] Suppose the price of a bitcoin stays at \$10,000 and the reward for a successful block hash is still 12 bitcoins. What is the minimum number of Bitcoin mining devices you would need to buy and run to earn at least \$1,000,000 profit in a year? (The cost of purchase and the cost of running are both counted as loss.)

Solution:

- 1 block per 600 seconds; number of bitcoins per second is $12/600 = 0.02$. Total running cost is $10^{19}/10^{17} * \$1 = \$100$ per second. So $\$100/0.02 = \$5000$ is the minimum price of Bitcoin to cover the running costs.
- If 1 MB = 1,000,000B: 1 MB / 166 bytes $\approx$ 6024 transactions per block, divided by 600 seconds per block $\approx$ 10.04 transactions per second. If 1 MB = 1,048,576B: 1 MB / 166 bytes $\approx$ 6316 transactions per block, divided by 600 seconds per block $\approx$ 10.53 transactions per second.
- Number of bitcoins per second is $6/600 = 0.01$, so 0.01 bitcoin per second is lost. Monetary loss needed to be covered by transaction fees = $\$5000 * 0.01 = \$50$ per second. Therefore the mean transaction fee needs to be $50/10.04 = \$4.98$ or $50/10.53 = \$4.75$.
- 1 device is enough. Total share of the bitcoin revenue is $10^{17}/(10^{17} + 10^{19}) = 1/101$, which is \$200/101 per second. Profit per second is therefore \$99/101. There are about $365.24 \cdot 86400 = 31556736$ seconds in a year. Therefore a single machine is enough to earn $\$31556736 * 99/101 - \$10000 \approx \$30931850$ in a year, exceeding the question's requirement.

  It is OK to claim that there are 365 days, 366 days, or 365.25 days in a year.

  -1 for forgetting the slight decrease in profit from us buying a machine ourselves.

  (The intent was to ask for the required number of machines for \$1,000,000 profit in one day, not year.)

Programming assignment

k-anonymity [35 points]

We learned in class that k-anonymity is used to prevent linkage between *quasi-identifiers (QIDs)* and *sensitive attributes*. k-anonymity has some limitations; in particular, it cannot account for background knowledge, especially from complementary data release. In this question, we will examine another issue of k-anonymity known as *minimality*. We want to minimize how much the anonymization procedure *changes* the data set, but it is sometimes computationally hard to ensure minimal change.

(a) [20 points] The government wishes to study a disease called the Phage. They have asked hospitals to release a data set to investigate a possible link between Age and Phage infection, while protecting data privacy. For this question, there will be one QID: **Age**, which is an integer. There is one sensitive attribute, whether or not the person has contracted a sensitive disease called the **Phage**, indicated by 0 or 1.

You are asked to write a program to test if a input file satisfies k-anonymity. Specifically, your program should output the maximum k such that the input file satisfies k-anonymity. Your program's first input argument should be the name of the input file, and it will be called with:

```
./ktest <inputfile>
```

It should **return** the maximum k such that the input file satisfies k-anonymity. (Its standard output will be ignored, so use a return function rather than a print function to output k.)

(b) [15 points] Now, you are asked to anonymize the input data set. To anonymize the data set, you are allowed to change any Age identifier (but not the Phage attribute). You must change the Age to another integer (e.g. change 28 to 25); you **cannot** change any data to a string (e.g. change 28 to 2X). Two people with the same Age are considered to be in the same anonymity set (even if they have a different Phage attribute).

In order to preserve the usefulness of the data set, you are asked to change the identifiers minimally. The definition of minimal change is as follows. Suppose there are n entries, and the list of Age was $a_1, a_2, \ldots, a_n$ Suppose after your anonymization changes, they become $a'_1, a'_2, \ldots, a'_n$ respectively. The **change** is defined as:

$$\sum_{i=1}^{n}(|a'_i - a_i|)$$

In other words, the change is computed by summing all the differences between the Ages of the original data set and the anonymized data set.

Write a program, `kanon`, which implements 4-anonymity on a data set with **minimal change**. The name of the data set file will be the first argument, so that `kanon` would be called as follows to anonymize `inputfile`:

```
./kanon <inputfile>
```

The data set is in CSV format, where each line is an entity's Age and Phage, separated by commas (a sample has been provided). `kanon` should overwrite the original file with the anonymized data set. **Do not sort your data; keep the original ordering in the input file.**

The following fact may be useful for you: suppose a minimal change k-anonymity solution to the set $a_1, a_2, \ldots, a_n$ (sorted ascendingly) is to divide it into m anonymity sets: $A_1, A_2, \ldots, A_{m-1}, A_m$. If $A_m = \{a_f, a_{f+1}, \ldots, a_n\}$, then $A_1, A_2, \ldots, A_{m-1}$ is a minimal change k-anonymity solution to the set $a_1, a_2, \ldots, a_{f-1}$. You can apply this fact in dynamic programming to arrive at an answer.

Your program should be fast; it should not take more than a few seconds on a input file with 500 rows.

(c) [15 points (bonus)] It turns out that the previous study did not reveal any notable relationship between Age and Phage. The government has asked hospitals to release another data set, with more QIDs: Age, Height, Weight, Width, Shoe Size, and Number of Children (all integers). The last column will be the sensitive attribute, Phage, which is 0 or 1, and should not be changed. Similarly, you are asked to achieve minimal change.

Note that the dynamic programming strategy above no longer works.

Write a program, `kanon2`, which implements 4-anonymity on a data set with approximately minimal change. The name of the data set file will be the first argument, so that `kanon2` would be called as follows to anonymize `inputfile`:

```
./kanon2 <inputfile>
```

`kanon2` should overwrite the original file with the anonymized data set. **Do not sort your data; keep the original ordering in the input file.**

Note that the problem of minimal change is in general a computationally hard (NP-hard) problem. In other words, achieving minimal change efficiently in general is impossible. You should simply try your best to minimize change as much as possible while still achieving 4-anonymity. (See below for testing for what grades you will get.)

Submission instructions

All submissions should be done through the CASS system. For this assignment, there is **no** Milestone deadline. Submit the following programs:

- `a3.pdf`, containing all your written answers.
- The programming assignment, detailed below:

For the programming assignment, submit your code; do not submit any compiled files.

C++: Submit ktest.cpp, kanon.cpp and kanon2.cpp. I will compile them and call `./ktest <inputfile>`.

Py: Submit ktest.py, kanon.py and kanon2.py. I will call `python ktest.py <inputfile>`.

Java: Submit ktest.java, kanon.java and kanon2.java I will call `javac ktest.java` and then `java ktest <inputfile>`.

If there is a Makefile in your folder, the Makefile will override all of the above. This implies if you are not writing in C++, Python, or Java, you must include a Makefile.

Keep in mind that plagiarism is a serious academic offense; you may discuss the assignment, but write your assignment alone and do not show anyone your answers and code.

The submission system will be closed exactly 48 hours after the due date of the assignment. Submissions after then will not be accepted unless you have requested an extension before the due date of the assignment. You will receive no marks if there is no submission within 48 hours after the due date.

Testing for Progamming Assignment

For part (b), there will be 10 data sets, ranging from simple, small data sets to large ones with a maximum of 500 elements. I will run your code for at most 10 seconds on each data set. You will score 2 points per test for a correct answer and no points for an incorrect answer.

For part (c), I will run your program on 10 randomly generated data sets. Each data set has 500 entities with reasonable random values for all attributes. After at most 10 seconds, I will terminate the program, and evaluate the change you achieved by reading the input file again. The bonus mark you get will depend on how well you did, compared to other students:

Percentile	Score
95th	15
90th	14
80th	13
70th	12
50th	10
25th	9
10th	8
0th	7

Any incorrect answer (failing to achieve k-anonymity) scores 0 points. Your final bonus mark is the average mark achieved for all 10 randomly generated data sets. Students who do not submit will not be counted in the percentile.

Furthermore, I will also include my own algorithm in the competition. My algorithm is relatively simple. If you beat my algorithm on a data set, you will automatically score a 15 on that data set no matter how well you did compared to other students.

Since the programs will be terminated at the time limit, your code should overwrite the original file frequently, perhaps whenever it finds a better solution.