Assignment 0x07

Frank Kaiser – 1742945, Jan Martin – 1796943

January 27, 2018

Contents

1 k-Anonymity of a Dataset

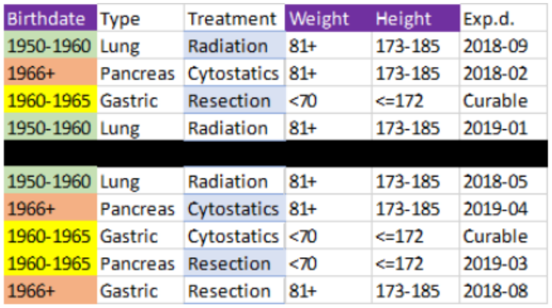
1.1 Assign the columns

1. **Identifiers**: Name, Telephone
   - (a) Name: Names in general are kind of unique, especially in combination with other quasi-identifiers.
   - (b) Telephone: A telephone number can usually be mapped to a single household or a person.
2. Quasi-identifiers: Birth date, Weight, Height
   - (a) Birth date: Birth dates tend to be pretty unique, too and therefore can be used to identify a person.
   - (b) Weight: While with weight alone it is most often not possible to identify a person, it may very well help to do so if other information is available. Also the weight of a person could change since the measurement. So this could fit in sensitive information, too.
   - (c) Height: For height the argumentation is pretty similar to weight, however, height usually does not change as quickly as weight can. Making it better to identify individuals.
3. Sensitive data: Type, Treatment, Expected death
   - (a) Type: The type of cancer is the reason for this datatable and cannot be changed
   - (b) Treatment: Same as Type
   - (c) Expected death: This is also relevant information to the type and treatment and should not be changed.

1.2 Anonymize the dataset (k=3)

- We removed all Identifies, in this case name and phone number.
- We grouped all Quasi-Identifiers that could reasonably be used to identify a person; This includes abstracting the birthdate to year ranges and grouping the weight and height into ranges, as well.
- We removed one outlier that could not be anonymous: The person in question was unusually young, unusually small, and had a vastly different weight to most other entries in the list.

The resulting table can be seen in figure 1.



Birthdate	Type	Treatment	Weight	Height	Exp.d.
1950-1960	Lung	Radiation	81+	173-185	2018-09
1966+	Pancreas	Cytostatics	81+	173-185	2018-02
1960-1965	Gastric	Resection	<70	<=172	Curable
1950-1960	Lung	Radiation	81+	173-185	2019-01
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
1950-1960	Lung	Radiation	81+	173-185	2018-05
1966+	Pancreas	Cytostatics	81+	173-185	2019-04
1960-1965	Gastric	Cytostatics	<70	<=172	Curable
1960-1965	Pancreas	Resection	<70	<=172	2019-03
1966+	Gastric	Resection	81+	173-185	2018-08

Figure 1: Anonymized Table

1.3 Explain the homogeneity attack against k-anonymized datasets. Can this be applied to your anonymized dataset? Explain your answer.

Homogeneity Attack: If all members of a group of k records have the same sensitive data in a column, an attacker can find out that value despite anonymization (its always the same, thus predictable). This is hard to avoid completely - our 1950-1960 age group is extremely similar in nearly all respects; thus, an attacker will find out they have lung cancer and are receiving radiation treatment.

2 Intersection Attacks on Mix Cascades

2.1 Part 1

2.1.1 How many requests are contained in the left-log and right-log, respectively?

In left-log there are 1191923 and in right-log 1192045 requests.

2.1.2 How many different users, identified by their IP address, exist, how many webservers?

There are 199 users and 29469 webservers.

2.1.3 What are the five most visited websites?

We wrote a python script (web_site_count.py) to count the websites. This is the result:

Webserver	Number of visits
static.cache.l.google.com	37780
www.google-analytics.com	28315
www.jetztspielen.de	22410
tbn0.google.com	21406
www.vtunnel.com	19507

This is the script:

```
import re
import operator

regex = re.compile(r'http://(.+)"')
web_servers = {}
with open('right.txt', 'r') as file:
    lines = file.readlines()
    for line in lines:
        match = re.search(regex, line)
        if match:
            web_site = match.group(1)
            if web_site in web_servers:
                web_servers[web_site] += 1
            else:
                web_servers[web_site] = 1

sorted_servers = sorted(web_servers.items(), key=operator.itemgetter(1),
                        reverse=True)
print(len(sorted_servers))
print(sorted_servers[:5])
```

2.2 Part 2: Can you find out his IP address if you have no further information about the user? If not, list all candidate IP addresses.

It is not possible to find a unique user, but 30 candidates. The 31th most used ip address in this timeframe is used only 7 times.

To find it out we used another script (user_site_count.py)

This was the output:

```
(’10.1.2.54’, 49), (’10.1.2.80’, 48), (’10.1.2.73’, 46), (’10.1.2.77’, 46), (’10.1.2.62’,
46), (’10.1.2.72’, 46), (’10.1.2.76’, 46), (’10.1.2.59’, 45), (’10.1.2.74’, 44), (’10.1.2.60’,
44), (’10.1.2.66’, 44), (’10.1.2.55’, 43), (’10.1.2.64’, 43), (’10.1.2.70’, 43), (’10.1.2.65’,
43), (’10.1.2.52’, 43), (’10.1.2.68’, 43), (’10.1.2.61’, 42), (’10.1.2.67’, 42), (’10.1.2.51’,
41), (’10.1.2.58’, 41), (’10.1.2.69’, 40), (’10.1.2.79’, 39), (’10.1.2.71’, 39), (’10.1.2.75’,
39), (’10.1.2.63’, 39), (’10.1.2.53’, 38), (’10.1.2.56’, 36), (’10.1.2.78’, 35), (’10.1.2.57’,
34) (’10.1.2.50’, 7)
```

And this the script:

```python
import re
import operator

def time_check(time):
    """makes sure that the time is in the desired timerange:
        20:14:30 - 20:15:30"""

    hr, m, s = time.split(":")
    hr, m, s = int(hr), int(m), int(s)

    if m == 14 and s >= 30:
        return True
    elif m == 15 and s <= 30:
        return True
    else:
        return False

regex = re.compile(r'(.+)\s.+(20:1[4-5]:\d+)')
ips = {}
with open('left.txt', 'r') as file:
    lines = file.readlines()
    for line in lines:
        match = re.search(regex, line)
        if match:
            ip = match.group(1)
            time = match.group(2)
            timeValid = time_check(time)
            if timeValid:
                if ip in ips:
                    ips[ip] += 1
                else:
                    ips[ip] = 1

sorted_ips = sorted(ips.items(), key=operator.itemgetter(1), reverse=True)
print(len(sorted_ips))
print(sorted_ips[:31])
```

2.3 Part 3: Can you uniquely identify Julias IP address if you assume that the delay of the whole mix cascade is only up to five seconds?

We checked all requests to tv-movie.de in right.txt (by Hand :P) and listed requesting IPs from 5 seconds earlier until the time of the request in the left.txt. After only 3 days, the IP 10.1.2.32 is the only one in all eligible timeslots. This means that this has to be Julia's IP address.

T2 154430 - 154638
Zeit: 22:00:11 - 22:00:16 (154629 - 154632) (**32**, 61, 64, 75)
T3 324846 - 325085
Zeit: 22:00:18 - 22:00:23 (325079 - 325082) (10, **32**, 40, 64)
T4 494813+
Zeit: 22:00:47 - 22:00:52 (495077-495079) (10, **32**, 40)
665241+
Zeit: 21:59:47 - 21:59:52 (665447 - 665462) (57, 59, 101, 186, 53, 64, 69, 72, 37, 51, 56, **32**, 51, 70, 75, 65)