数据安全实验报告

Lab8 交互式发布 DP 方案评估
网络空间安全学院 信息安全专业
2112492 刘修铭 1036

1 实验要求

参考教材实验 5.1，对交互式发布方案进行 DP 方案设计，指定隐私预算为 0.1，支持查询次数为 20 次，对 DP 发布后的结果进行评估，说明隐私保护的效果。

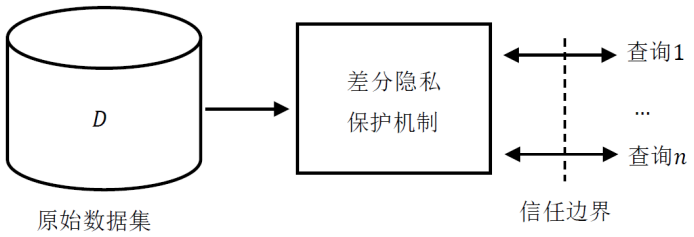
2 实验原理

差分攻击的核心即是通过寻找两个仅相差一条记录的数据集，对其分别做同样的查询，再比较返回结果的差异，从而获取两个集合所相差的记录的敏感信息。

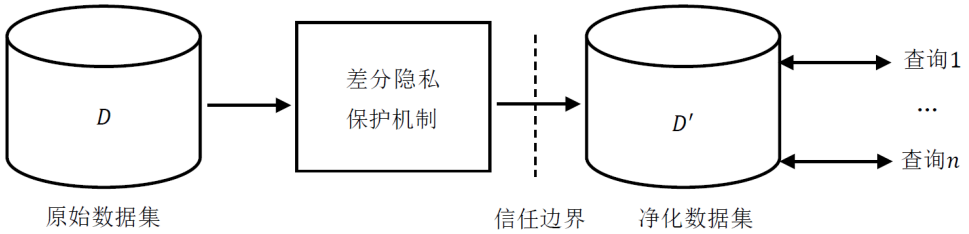
为了抵御差分攻击，最直观的想法是通过对查询结果加入一定的扰动（即反馈的查询结果并非是真实的结果），使得查询结果不再精确，攻击者在进行差分攻击时获得的查询结果无法用于区分只差一条记录的两个集合。使用差分隐私来量化一个随机化算法提供多强的隐私保护。当随机化算法满足差分隐私的定义时，我们称该算法为**差分隐私算法**。

差分隐私保证：如果数据分析者除了数据分析任务之外不能对数据集进行额外查询，就无法增加数据集中每条记录的隐私损失。也就是说，如果我们使用随机性算法保护了个人的隐私，那么数据分析者就不能仅通过背景知识以及算法的输出来增加隐私损失，无论是在正式定义中，还是在任何直观意义上。

根据对于多次查询的响应方法不同，差分隐私的发布模型分为交互式和非交互式两种。



（a）交互式数据发布方案



（b）非交互式数据发布方案

在交互式数据发布中，给定数据集 D 和查询集 $F = f_1, \ldots, f_m$，需通过一种数据发布机制，使其能够在满足差分隐私保护的条件下，逐个回答查询集 F 中的查询，直到耗尽全部隐私预算。发布机制的性能通常由精确度来衡量。交互式数据发布即是要在满足一定精确度的条件下，以给定的隐私保护预算回答尽可能多的查询。交互式设置的方法主要考虑事务数据库、直方图、流数据和图数据发布等。

在交互式数据发布中，给定数据集 D 和查询集 $F = f_1, \ldots, f_m$，需通过一种数据发布机制，使其能够在满足差分隐私保护的条件下，一次性回答 F 中的所有查询。数据管理者针对所有可能的查询，在满足差分隐私的条件下一次性发布所有查询的结果，或者发布一个原始数据集的净化版本，即带噪音的合成数据集，用户可对合成数据集自行进行所需的查询操作。非交互式数据发布方法主要集中在批查询、列联表发布、基于分组的发布方法以及净化数据集发布。相比于交互式数据发布场景每次查询都要消耗隐私预算，非交互式只需在发布合成数据集时消耗隐私，由于差分隐私的**后处理不变性**，对合成数据集的后续查询任务，不会进一步泄露原始数据集的隐私。

拉普拉斯机制通过向查询结果或原始数据加入服从拉普拉斯分布的噪声来实现差分隐私。

设有查询函数 $f: X^n \to R^d$，输入为一个数据集，输出为 d 维实数向量。对于任意相邻数据集 D 和 $D' \in X^n$，有 $GS_f = \max_{D,D'} ||f(D) - f(D')||_p$ 称为函数 f 的全局灵敏度。对于不同的机制，灵敏度的范数也不同，拉普拉斯机制使用 1 阶范数距离（即曼哈顿距离）。

给定数据集 $D' \in X^n$，随机算法 $M(D) = f(D) + (Y_1, \ldots, Y_K), Y_I\ Lap(\Delta f/\epsilon)$ 提供 ϵ - 差分隐私。注意，在交互式发布DP方案之中，每一次查询其实都会造成对应的隐私损失。因此，在数学上每次查询添加的噪声均服从 $Lap(0, K/\epsilon)$ 分布。如图展示了多次查询后被推测出的概率。

表 6-3 交互式发布的概率表

ε	噪声绝对值的数据分布				多次查询添加噪声的平均值落在危险区间内的概率			
	90%	95%	99%	99. 9%	100	1000	10000	10000
1	2. 29	2. 98	4. 50	6. 43	100. 00	100. 00	100. 00	100. 00
0. 1	23. 24	29. 99	45. 51	66. 56	25. 59	73. 75	99. 99	100. 00
0. 01	227. 97	296. 22	463. 48	677. 26	2. 72	9. 12	27. 85	73. 70

在隐私保护中，随机数种子的选择也是非常重要的，其分布满足如下条件：

1. $$u_1, u_2 \sim U(0,1)$$
2. $$x = \begin{cases} \beta ln(2u_1) + u_2 & u_1 \le 0.5 \\ u_2 - \beta ln(2(1-u_1)) & u_1 > 0.5 \end{cases}$$

3 实验过程（含主要源代码及实验结果分析）

3.1 实验环境配置

按照实验手册说明，进行实验的环境配置。

安装解释器软件。

```
lxm@lxmliu2002:~/datasecurity$ sudo apt-get install build-essential
[sudo] password for lxm:
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
The following packages were automatically installed and are no longer required:
  libjsoncpp1 librhash0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 54 not upgraded.
```

接着查看 gcc 版本，说明安装成功。

```
lxm@lxmliu2002:~/datasecurity$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/
OFFLOAD_TARGET_NAMES=nvptx-none:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversi
,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-ma
/usr/lib --without-included-gettext --enable-threads=
libstdcxx-abi=new --enable-gnu-unique-object --disabl
-enable-multiarch --disable-werror --with-arch-32=i68
build/gcc-9-9QDOt0/gcc-9-9.4.0/debian/tmp-nvptx/usr,h
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)
```

将给定的压缩包复制到给定文件夹中，并解压文件。

```
lxm@lxmliu2002:~/datasecurity$ tar -zxvf ./experiment1.tar.gz
./experiment1/
./experiment1/testraw.c
./experiment1/laplace.c
./experiment1/csvpackage.c
./experiment1/testhist.c
./experiment1/zoo_nb.csv
./experiment1/include/
./experiment1/include/csvpackage.h
./experiment1/include/laplace.h
./experiment1/zoo.csv
./experiment1/medicaldata.csv
./experiment1/Makefile
./experiment1/md_nb.csv
```

接着 make 进行编译。

```
lxm@lxmliu2002:~/datasecurity/experiment1$ make
gcc -I./include -c csvpackage.c
gcc -I./include -c laplace.c -lm
gcc -I./include -c testraw.c
gcc csvpackage.o laplace.o testraw.o -o testraw  -lm
gcc -I./include -c testhist.c
gcc csvpackage.o laplace.o testhist.o -o testhist -lm
```

至此，本次实验的环境配置完成。

3.2 实验文件分析

- testraw.c：提供了数据集读取，给定隐私预算的噪音生成和加噪后结果展示的功能。
- testhist.c：提供了数据集读取，给定隐私预算的噪音生成和加噪后统计值输出的功能。
- laplace.c：为实验代码生成拉普拉斯分布的随机数，用于加噪。
- csvpackage.c：实验代码的 csv 读取和预统计函数实现，针对样例代码提供读取为 Animals 结构体的功能。
- zoo.csv：该数据集描述了一个动物园喂食的场景，第一列中数据为动物名称，第二列中数据为动物每天消耗的胡萝卜数量。查询定义为“每日进食超过 55 根胡萝卜的动物数量”。
- zoo_nb.csv：相邻数据集（在 zoo.csv 的基础上去掉了“Dugong”这一项数据），来进行对比演示加入不同规模的噪音对统计结果的影响。我们利用这两个隐私预算分别基于原始数据集和相邻数据集生成加噪数据并进行前述查询，以此来对噪声的影响进行展示和比较。
- medicaldata.csv：其第一列数据为一特定年龄区间，第二列数据为该年龄区间患有某种疾病的人数。要发布的直方图即是以第一列数据为直方图的桶（Bucket）发布的统计数据。

- md_nb.csv：相邻数据集，是在 medicaldata.csv 的基础上将其中“30-40”区间的统计值 -1 而产生的，模拟原始医疗数据集内一名患者退出数据共享。

3.3 非交互式方案解读

3.3.1 Csvpackage.c

这个文件主要完成 csv 的文件的读取，对原始数据做一些变换，或将原始数据存储为其他结构体，不是本次实验重点，故跳过。

3.3.2 laplace.c

在这个文件中主要有两个函数。

1. uniform_data 函数：利用混合同余法产生（a,b）区间上均匀分布的随机数，其中 a 为给定区间的下限，b 为给定区间的上限，*seed 是伪随机数种子。混合同余法的公式为 $X_{n+1} = aX_N + c \mod m$。在此，设定 a = 2045，m = 1048576，c = 1。其中，2045 和 1048576 是同余法中使用的常数，能够产生高质量的伪随机数序列。具体来说，2045 是一个较大的质数，它可以确保生成的随机数序列具有较长的周期，即生成的随机数序列不会很快地重复。

```
/*
函数功能:    利用混合同余法产生（a,b）区间上均匀分布的随机数
输入参数说明:
a       给定区间的下限
b       给定区间的上限
seed    长整型指针变量， *seed 为伪随机数的种子
*/
double uniform_data(double a, double b, long int *seed)
{
    double t;
    *seed = 2045.0 * (*seed) + 1;
    *seed = *seed - (*seed / 1048576) * 1048576;
    t = (*seed) / 1048576.0;
    t = a + (b - a) * t;
    return t;
}
```

2. laplace_data 函数：主要用于求解 laplace 分布概率累积的反函数，并利用该反函数产生 laplace 分布的随机数。相关数学公式见**实验原理**部分。

```
/*
函数功能:    求解laplace分布概率累积的反函数，利用该反函数产生laplace分布的随机数
输入参数说明:
beta        拉普拉斯分布参数
seed        长整型指针变量， *seed 为伪随机数的种子
*/
double laplace_data(double beta, long int *seed)
{
    double u1, u2, x;
    u1 = uniform_data(0.0, 1.0, seed);
    u2 = uniform_data(0.0, 1.0, seed);
    if (u1 < 0.5)
    {
        x = beta * (log(2 * u1) + u2);
```

```
    }
    else
    {
        x = u2 - (beta * log(2 * (1 - u1)));
    }
    return x;
}
```

3.3.3 testraw.c

本文件中主要有两个函数。

1. csv_analysis 函数：对传入的csv文件进行处理，提取其中数据并生成拉普拉斯分布的噪音进行加噪。

```
/*
函数功能：    对传入的csv文件进行处理，提取其中数据并生成拉普拉斯分布的噪音进行加噪
输入参数说明：
path        csv文件的存储位置
beta        拉普拉斯分布参数
seed        长整型指针变量， *seed 为伪随机数的种子
*/
void csv_analysis(char *path, double beta, long int seed)
{
    FILE *original_file = fopen(path, "r+"); // 读取指定路径的数据集
    struct Animals *original_data = NULL;
    original_data = csv_parser(original_file);
    int sum = 0, i = 0;
    double x = 0;
    while (original_data[i].name) // 循环为原始数据集内各条数据生成拉普拉斯噪音并加噪
    {
        x = laplace_data(beta, &seed); // 产生拉普拉斯随机数
        printf("Added noise:%f\t%s\t%f\n", x, original_data[i].name,
original_data[i].carrots + x); // 此处分别列出了每条具体添加的噪音和加噪的结果。当投入较少预算时，可能会出现负数
        if (original_data[i].carrots + x >= 55)
        {
            sum++;
        }
        i++;
    }
    printf("Animals which carrots cost > 55 (Under DP): %d\n", sum); // 输出加噪后的数据集中，每日食用胡萝卜大于55的动物个数
}
```

2. main 函数：设置隐私预算并调用 csv_analysis 函数。在 main 函数中，使用不同的隐私预算来对原始数据集和相邻数据集进行处理，并输出加噪后的结果。在此代码中，分别设置两类隐私预算，分别代表极小、极大。while 循环表示查询次数。

```
/*
参数表：
seed        长整型指针变量， *seed为伪随机数的种子
sen         数据集的敏感度
x           用于储存拉普拉斯分布噪音的临时变量
```

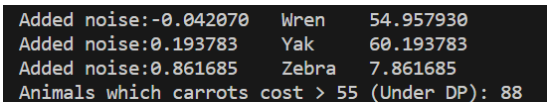
```
beta        隐私预算，在输入后根据公式转换为拉普拉斯分布参数
*/
int main()
{
    long int seed;
    int sen = 1; // 对于一个单属性的数据集，其敏感度为1
    double eps[] = {10, 0.1}; // 指定两类隐私预算，分别代表极小，极大
    srand((unsigned)time(NULL)); // 生成基于时间的随机种子（srand方法）
    int i = 0;
    while (i < 2)
    {
        printf("Under privacy budget %f, sanitized original data with animal name and laplace noise:\n", eps[i]);
        double beta = sen / eps[i]; // 拉普拉斯机制下，实际公式的算子beta为敏感度/预算
        seed = rand() % 10000 + 10000; // 随机种子产生
        csv_analysis("./zoo.csv", beta, seed); // 先调用原始数据集
        printf("==================Using neighbour dataset==================\n");
        seed = rand() % 10000 + 10000; // 随机种子更新
        csv_analysis("./zoo_nb.csv", beta, seed); // 再调用相邻数据集
        printf("============================================================\n");
        i++;
    }
    return 0;
}
```

运行该文件，查看输出结果。

当隐私预算极大（为 10）时，可以看到噪声在正负一之间，在该预算下，加噪前和加噪后的响应一致。原始情况响应为 90，加噪后响应为 89，相差很小，数据可用性更好。

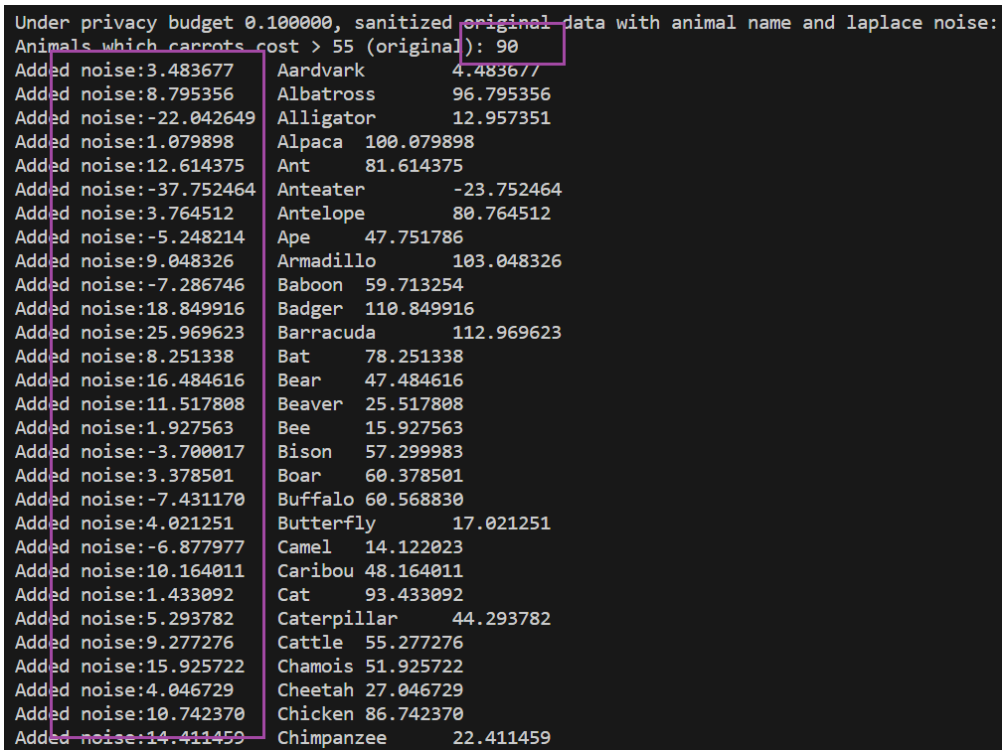
```
lxm@lxmliu2002:~/datasecurity/experiment1$ ./testraw
Under privacy budget 10.000000, sanitized original data with animal name and laplace noise:
Animals which carrots cost > 55 (original): 90
Added noise:0.401386    Aardvark        1.401386
Added noise:-0.016149   Albatross       87.983851
Added noise:0.009721    Alligator       35.009721
Added noise:0.760822    Alpaca  99.760822
Added noise:0.048870    Ant     69.048870
Added noise:0.009574    Anteater        14.009574
Added noise:0.137995    Antelope        77.137995
Added noise:0.614964    Ape     53.614964
Added noise:0.104366    Armadillo       94.104366
Added noise:-0.015100   Baboon  66.984900
Added noise:-0.010895   Badger  91.989105
Added noise:-0.329488   Barracuda       86.670512
Added noise:0.540771    Bat     70.540771
Added noise:0.226063    Bear    31.226063
Added noise:-0.148243   Beaver  13.851757
Added noise:0.019149    Bee     14.019149
Added noise:-0.039511   Bison   60.960489
Added noise:-0.018367   Boar    56.981633
Added noise:0.508988    Buffalo 68.508988
Added noise:0.692845    Butterfly       13.692845
Added noise:0.953360    Camel   21.953360
Added noise:0.005226    Caribou 38.005226
Added noise:0.070334    Cat     92.070334
```

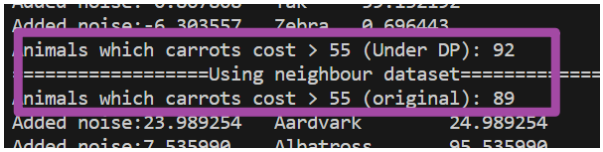
```
Added noise:0.103548    Yak     60.103548
Added noise:0.038042    Zebra   7.038042
Animals which carrots cost > 55 (Under DP): 90
==================Using neighbour dataset==================
Animals which carrots cost > 55 (original): 89
Added noise:0.023333    Aardvark        1.023333
```

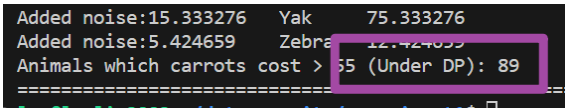


但是，观察标记后对相邻数据集的处理情况，我们可以发现，加噪后数据集对该查询的响应为 88，仍与数据集的变化基本一致，不能有效抵御对该查询的差分攻击。

当隐私预算极小（为 0.1）时，可以看到产生的拉普拉斯噪音更大，对查询结果产生影响。







观察相邻数据集，原始查询是 89，相对于 90 少 1（因为移除了 Dugeng 项），但是加噪后的查询结果为 89，与原始数据集相同，与数据集变化方向不同，不再能反映出“Dugong”项移除的影响。这就意味着，投入较少的隐私预算时，虽然数据的可用性降低了，但是能够更好地抵御差分攻击的影响。

3.3.4 testhist.c

此文件与 testraw.c 基本相同，唯一不同的地方为噪声的处理。本文件中，加噪声的对象不再是数据本身，而是对数据进行分桶统计后的计数值进行加噪。

```
while (original_data[i].bucket) // 循环为原始数据集内各桶数据生成拉普拉斯噪音并加噪
{
    x = laplace_data(beta, &seed); // 产生拉普拉斯随机数
    printf("Added noise:%f\t%s\t%f\n", x, original_data[i].bucket, original_data[i].count + x);
// 此处分别列出了每条具体添加的噪音和加噪的结果。当投入较少预算时，可能会出现负数
    i++;
}
```

运行代码，得到如下结果：

当隐私预算极大（为 10）时，由于加入噪音量级较小，相邻数据集的变化仍能被体现。即加噪后的数据和原始数据并没有太大的差别。

分区	原始	加噪	加噪 + 相邻
20 - 30	405	405.577487	405.700195
30 - 40	436	436.064704	434.832390
40 - 50	421	421.026895	420.856021
50 - 60	457	456.974492	457.923189
60 - 70	463	463.976042	463.699772

当隐私预算极小（为 0.1）时，由于噪音规模的提高，在相邻数据集的变化影响下，查询结果不减反增。即，虽然数据可用性变差，但能保护实际数据的变化不被攻击者获取，可抵御差分攻击。

分区	原始	加噪	加噪 + 相邻
20 - 30	405	386.329299	390.637195
30 - 40	436	432.750870	433.050928
40 - 50	421	454.224493	426.334131
50 - 60	457	457.998244	463.287818
60 - 70	463	464.643677	165.718378

3.4 交互式方案

3.4.1 方案设计

new_testraw.c

交互式方案较非交互式方案而言，是对查询返回的结果添加噪音。

重复攻击是针对差分隐私的攻击方式。因为拉普拉斯机制添加噪声的特点是无偏估计，多次查询后均值为 0，如果攻击者向数据库请求重复执行同一个查询语句 ，将结果求和平均，就有极大的概率获得真实结果。

要考虑保护多次查询的话，需要为每次查询进行预算分配：假定隐私预算为 ε，允许的查询次数为 k，则每次查询分配的预算为 ε/k，这样才能达到 ε - 差分隐私的目标。因此，对于统计查询而言，如果在查询结果上进行反馈，则需要定义所能支持的次数，进而，按照上述方式对每次查询进行预算的分配。换句话说，这种添加噪音的方式，会使每次查询都消耗一定的隐私预算，直到隐私预算都被消耗干净，就再也不能起到保护的作用。

基于此，对 `testraw.c` 文件进行修改。

1. csv_analysis 函数。修改函数的返回值类型，修改函数逻辑，将函数返回值设定为查询结果加上随机生成的噪声。

```c
/*
函数功能：    对传入的csv文件进行处理，提取其中数据并生成拉普拉斯分布的噪音进行加噪
输入参数说明：
path        csv文件的存储位置
beta        拉普拉斯分布参数
seed        长整型指针变量， *seed 为伪随机数的种子
*/
double csv_analysis(char *path, double beta, long int seed)
{
    FILE *original_file = fopen(path, "r+"); // 读取指定路径的数据集
    struct Animals *original_data = NULL;
    original_data = csv_parser(original_file);
    int sum = 0, i = 0;
    double x = 0;
    while (original_data[i].name) // 循环为原始数据集内各条数据生成拉普拉斯噪音并加噪
    {
        x = laplace_data(beta, &seed); // 产生拉普拉斯随机数
        // printf("Added noise:%f\t%s\t%f\n", x, original_data[i].name, original_data[i].carrots + x); // 此处分别列出了每条具体添加的噪音和加噪的结果。当投入较少预算时，可能会出现负数
        if (original_data[i].carrots + x >= 55)
        {
            sum++;
        }
        i++;
    }
    // printf("Animals which carrots cost > 55 (Under DP): %d\n", sum); // 输出加噪后的数据集中，每日食用胡萝卜大于55的动物个数
    return sum + x;
}
```

2. main 函数。在这个函数中，首先添加了查询次数的设定，需要用户手动输入想要查询的次数。除此之外，函数内部还显式地指定了可允许的最大查询次数。最后，对查询平均值进行统计求取平均。

```c
/*
参数表：
seed        长整型指针变量， *seed为伪随机数的种子
sen         数据集的敏感度
x           用于储存拉普拉斯分布噪音的临时变量
beta        隐私预算，在输入后根据公式转换为拉普拉斯分布参数
*/
int main()
{
    long int seed;
    int sen = 1;                    // 对于一个单属性的数据集，其敏感度为1
    double eps[] = {10, 0.1};       // 指定两类隐私预算，分别代表极小，极大
    srand((unsigned)time(NULL)); // 生成基于时间的随机种子（srand方法）
    int i = 0;
    int search_times = 20;
    printf("Please input search times:");
    scanf("%ld", &search_times);
    for (int j = 0; j < 1; j++)
```

```
    {
        printf("Under privacy budget %f, sanitized original data with fake animal name and laplace noise:\n", eps[j]);
        printf("Every single search privacy budget %f\n", eps[j] / 20);
        eps[j] = sen / (eps[j] / 20); // 拉普拉斯机制下，实际公式的算子 beta 为敏感度/预算
        double avg_old = 0;
        for (int i = 0; i < search_times; i++)
        {
            seed = rand() % 10000 + 10000;                      // 随机种子产生
            avg_old += csv_analysis("./zoo.csv", eps[j], seed); // 先调用原始数据集
        }
        printf("Avg Old Search Result: \t%f\n", avg_old / search_times);
        printf("==================Using neighbour dataset==================\n");
        double avg_new = 0;
        for (int i = 0; i < search_times; i++)
        {
            seed = rand() % 10000 + 10000;                          // 随机种子更新
            avg_new += csv_analysis("./zoo_nb.csv", eps[j], seed); // 再调用相邻数据集
        }
        printf("Avg New Search Result: \t%f\n", avg_new / search_times);
    }
    return 0;
}
```

3.4.2 效果评估

1. 隐私预算为 10，查询次数为 20

当隐私预算极大（为 10）时，在该预算下，加噪前和加噪后的响应一致。原始情况响应为 90，加噪后响应为 89，相差很小，数据可用性更好，但不能抵御差分攻击。

```
lxm@lxmliu2002:~/datasecurity/experiment1$ ./new_testraw
Please input search times:20
Under privacy budget 10.000000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.500000
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 90
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 88
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 89
```

但是，观察标记后对相邻数据集的处理情况，我们可以发现，加噪后数据集对该查询的响应为 88，仍与数据集的变化基本一致，能抵御对该查询的差分攻击，但效果有限。

```
==================Using neighbour dataset==================
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 86
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 91
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 88
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 89
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 85
```

2. 隐私预算为 10，查询次数为 50

查询次数耗尽的情况下，可以看到两个数据集上的均值十分接近，效果差。

```
^[[Alxm@lxmliu2002:~/datasecurity/experiment1$ ./new_testraw
Please input search times:50
Under privacy budget 10.000000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.500000
Avg Old Search Result:  90.406173
==================Using neighbour dataset==================
Avg New Search Result:  89.229485
```

3. 隐私预算为 0.1，查询次数为 20

可以看到，在隐私预算为 0.1 时，原始查询偏差较大，在相邻数据集上也相较于原始查询表现出了较大偏差，难以探寻数据集变化的踪迹。隐私保护效果较好。

```
Under privacy budget 0.100000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.005000
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 109
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 116
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 113
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 133
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 127
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 119
Animals which carrots cost > 55 (original): 90
Animals which carrots cost > 55 (Under DP): 128
Animals which carrots cost > 55 (original): 90
```

```
==================Using neighbour dataset==================
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 108
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 129
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 119
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 128
Animals which carrots cost > 55 (original): 89
Animals which carrots cost > 55 (Under DP): 122
```

4. 隐私预算为 0.1，查询次数为 50

当查询次数耗尽时，二者的均值开始接近相同。

```
Under privacy budget 0.100000, sanitized original data with fake animal name and laplace nois
Every single search privacy budget 0.005000
Avg Old Search Result:  129.090164
==================Using neighbour dataset==================
Avg New Search Result:  207.379822
```

5. 隐私预算为 0.1，查询次数为 100

```
lxm@lxmliu2002:~/datasecurity/experiment1$ ./new_testraw
Please input search times:100
Under privacy budget 10.000000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.500000
Avg Old Search Result:  90.282538
==================Using neighbour dataset==================
Avg New Search Result:  89.775190
Under privacy budget 0.100000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.005000
Avg Old Search Result:  130.637660
==================Using neighbour dataset==================
Avg New Search Result:  159.784195
```

6. 隐私预算为 0.1，查询次数为 1000

```
lxm@lxmliu2002:~/datasecurity/experiment1$ ./new_testraw
Please input search times:1000
Under privacy budget 10.000000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.500000
Avg Old Search Result:  90.179056
==================Using neighbour dataset==================
Avg New Search Result:  89.644218
Under privacy budget 0.100000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.005000
Avg Old Search Result:  156.520276
==================Using neighbour dataset==================
Avg New Search Result:  172.554425
```

7. 隐私预算为 0.1，查询次数为 10000

```
lxm@lxmliu2002:~/datasecurity/experiment1$ ./new_testraw
Please input search times:10000
Under privacy budget 10.000000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.500000
Avg Old Search Result:  90.407239
==================Using neighbour dataset==================
Avg New Search Result:  89.449023
Under privacy budget 0.100000, sanitized original data with fake animal name and laplace noise:
Every single search privacy budget 0.005000
Avg Old Search Result:  168.248618
==================Using neighbour dataset==================
Avg New Search Result:  161.965114
```

可以看到，随着查询次数的不断增加，平均值会落在前面提到的危险区间中，隐私面临泄露的风险。

3.4.3 方案改进

better_new_testraw.c

修改显式指定的查询次数，将其设定为按照输入的查询次数进行计算。

```
1 eps[j] = sen / (eps[j] / search_times); // 拉普拉斯机制下，实际公式的算子 beta 为敏感度/预算
```

运行此方案，得到如下统计数据。

search_times	old_avg1	new_avg1	old_avg2	new_avg2
20	90.219531	89.303122	160.863114	179.218571
50	91.477675	92.796737	286.193377	383.090234
100	97.573308	93.315034	380.892396	265.118141
1000	139.115738	137.873842	1986.020166	2229.406086
10000	360.142331	363.358394	22121.405222	24465.252760

可以看到，其均值不在一个区间中，隐私信息不会泄露。

4 遇到的问题及解决方案

本次实验中，遇到的问题主要是对 DP 方案的不熟悉。通过询问同学们以及查阅相关资料后，问题得以解决。

5 参考

本次实验主要参考实验手册内容完成。