



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

数据安全

频率隐藏 OPE 方案实现

穆禹宸 2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：刘哲理

2023 年 5 月 10 日

目录

一、 实验名称	1
二、 实验要求	1
三、 实验过程	1
(一) 实验原理	1
(二) 实验代码	2
1. CalPos 插入位置	2
2. Insert 展示插入位置和插入后的编码树	3
3. 主函数样例设计	3
四、 实验结果	4
五、 心得体会	9
六、 附录一：完整 client.py 代码	10
七、 附录二：完整输出结果	13

一、 实验名称

频率隐藏 OPE 方案实现

二、 实验要求

参照教材 6.3.3FH-OPE 实现完成频率隐藏 OPE 方案的复现，并尝试在 client.py 中修改，完成不断插入相同数值多次的测试，观察编码树分裂和编码更新等情况。

三、 实验过程

(一) 实验原理

2015 年, Kerschbaum 对现有的保留顺序编码方案进行了改进, 提出了频率隐藏的保留顺序编码方案, 进一步提高了安全性。该方案隐藏相同明文出现的频率, 在一定程度上提升了方案的安全性, 并抵御了一部分利用明文频率发起的攻击。该方案在客户端维护一个二叉排序树, 将明文插入到二叉排序树中。通过参数的设定来减少排序树的调整。但是排序树一旦发生调整, 将带来巨大的性能消耗。

2021 年 Li 等人提出了一个小客户端存储且无额外交互的频率隐藏 OPE 方案, 该方案同时为了降低了编码更新的频率, 提出了一个带有编码策略的 B+ 树。

为了在无额外交互下运行算法, 该方案设置了一个本地表作为客户端存储, 记录明文以及出现次数。每当新的明文 pt 被加密时, 在本地表的帮助下, 找出有多少现有的明文值小于 pt 和等于 pt 的。因此, 很容易确定相应明文在 B+ 树中的随机顺序。该方案改进了 B+ 树, 每个中间节点不再存储关键词, 而是存储节点后代中包含的密文的数量。从而完成在 1 次交互下将密文插入 B+ 树中。

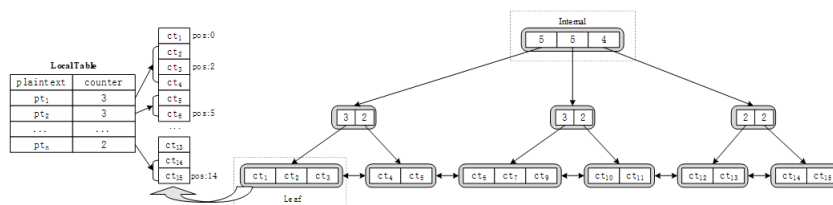


图 1: 无交互的 FH-OPE 方案

当树需要重新平衡时, 密文的路径编码即保序编码将被更新。如果更新涉及的密文很多, 会严重降低 OPE 方案的性能。对于服务端树, 为了减少密文重新编码的频率, 需要在调整树时不更新保序编码。该方案采用区域编码策略: 每个叶节点值区间为 $(a, b]$, 默认新节点编码策略: $((L+R))/2$, L 为左邻居的编码, R 为右邻居的编码。节点内编码更新策略: 区间 $(a, b]$ 、密文个数为 c , 更新后 $[1*(a+((b-a))/c), c*(a+(b-a)/c)]$ 。在这种情况下, 树的平衡调整不会引起编码的更新, 插入可能引发节点内数据密文编码更新, 但是其他节点不发生更新。

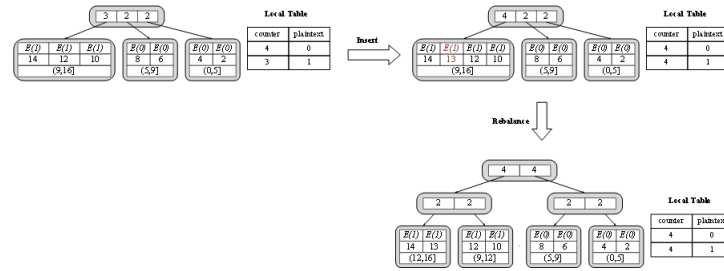


图 2: 整体示例

利用二进制明文域进行加密作为示例，有两种可能的明文值：0 和 1。构建一个 3 阶的编码树，并设置根的编码区间为 (0, 16]。一共有 8 个明文 0, 1, 0, 1, 0, 1, 0, 1，以下序列 0, 1, 0, 1, 0, 1, 0 已经插入到编码树中，其中 $E()$ 代表随机加密。接下来展示了客户端执行加密明文 1 的操作。客户端分配密文插入 B+ 树的随机顺序为 6 并将其与 $E(1)$ 一起提交给服务器，服务器将密文插入第三个叶子并为其分配一个编码 5。整体数据加密和编码树调节重新平衡后的情况如上图所示，从图中可以看到任何编码都没有更新。

(二) 实验代码

本次实验主要进行复现和修改代码，观察编码树的变化，**以下是我的修改和理解：**

1. CalPos 插入位置

下面的代码之中，变量 presum 表示本地表中所有小于要插入的明文的明文出现次数的总和。因为这是一个排序树，因此如果要插入的明文已经存在于本地表中，函数会将该明文的出现次数加 1，并在 $presum$ 到 $presum + local_table[plaintext] - 1$ 的范围内随机选择一个位置。最后，函数返回 the_pos 作为该明文在本地表中的位置。如果要插入的明文之前不存在于本地表中，函数会将该明文的出现次数设为 1，并直接返回 $presum$ 作为该明文在本地表中的位置。

```

1 def CalPos(plaintext):
2     # 插入 plaintext, 返回对应的 Pos
3     presum = sum([v for k, v in local_table.items() if k < plaintext])
4     print("本地表之中小于要插入明文的所有明文出现次数总和:", presum)
5     if plaintext in local_table:
6         local_table[plaintext] += 1
7         the_pos = random.randint(presum, presum + local_table[plaintext] - 1)
8         print("要插入的明文已经存在, 选择范围: [" + presum + ", " + presum +
9               ↪ local_table[plaintext] - 1, "]", end=' ')
10        print("随机选择的位置:", the_pos)
11        return the_pos
12    else:
13        local_table[plaintext] = 1
14        print('要插入的明文之前不存在, 直接选择位置:', presum)
15        return presum

```

我的修改：

因此，如果要插入的明文之前已经存在，那么我们把随机选择的范围和最终选择的结果打印出来，如果不存在，直接打印要插入的位置。

2. Insert 展示插入位置和插入后的编码树

这段代码的功能主要是调用 sql 之中的存储过程，实现插入明文。

```
1 def Insert(plaintext):
2     ciphertext = Random_Encrypt(plaintext)
3     # 连接数据库
4     conn = pymysql.connect(host='localhost', user='user',
5                             passwd='123456', database='test_db')
6     cur = conn.cursor()
7     the_result = CalPos(plaintext)
8     print(" 插入的明文:", plaintext, '位置:', the_result)
9     cur.execute(f"call pro_insert({the_result}, '{ciphertext}')" )
10    conn.commit()
11    print("-----此时编码树的结果-----")
12    cur.execute(
13        f"select ciphertext from example order by encoding")
14    results = cur.fetchall()
15    for result in results:
16        print(Random_Decrypt(result[0]), end=" ")
17    print("\n")
18    conn.close()
```

我的修改:

这部分主要有两点内容，一方面我展示了要插入的明文和插入的位置，另一方面，我撰写了如下 sql 语句：

```
1 select ciphertext from example order by encoding
```

然后通过 python 调用，打印了编码树的叶子节点的顺序。展示了编码树在节点插入之后的变化。

3. 主函数样例设计

我的修改:

在样例设计之中，我在实验指导书所给出的样例基础上，详细增加了 **apple** 和 **cherry** 的样例。这一点会在后面实验结果处详细说明。

同时，我也打印了本地表的情况和数据库之中的数据表项进行对比，来说明实验结果的正确性。

```

1 # 插入明文，同时设置了一部分重复的内容
2 print("-----")
3 print("-----下面展示实验过程-----")
4 print("-----")
5 # 加入了一大堆 apple 和 cherry，希望观察结果和树的变化
6 for ciphertext in ['apple', 'pear', 'banana', 'orange', 'cherry', 'apple',
↵ 'cherry', 'orange', 'apple', 'apple', 'apple', 'apple', 'cherry', 'cherry',
↵ 'apple', 'cherry']:
7     Insert(ciphertext)
8
9 # 假设我们搜索 b 和 p 之间的数据
10 print("-----")
11 print("-----假设我们搜索 b 和 p 之间的数据-----")
12 print("-----")
13 Search('b', 'p')
14
15 print("-----")
16 print("-----下面展示本地表内容-----")
17 print("-----")
18 print("local_table:", local_table)
19 total = sum([v for k, v in local_table.items()])
20 print(" 本地表共:", total, " 数据项")
21 # 连接数据库
22 conn = pymysql.connect(host='localhost', user='user',
23                         passwd='123456', database='test_db')
24 cur = conn.cursor()
25 cur.execute("select count(distinct encoding) from example")
26 results = cur.fetchall()
27 for result in results:
28     print(" 现在数据库之中共有:", result[0], " 个数据项")
29     if result[0] == total:
30         print(" 数据数量一致，实验成功！")

```

四、 实验结果

在测试样例之中，我的设计如下：

完整输出结果（明文）

```

1 ['apple ', 'pear ', 'banana ', 'orange ', 'cherry ', 'apple ', 'cherry ', 'orange ',
   'apple ', 'apple ', 'apple ', 'apple ', 'cherry ', 'cherry ', 'apple ', 'cherry
   ']

```

这是因为，多个 apple 会在排序树的叶节点的最前端进行插入，而多个 cherry 则会在排序树的叶节点的靠中间的位置插入，以此我们可以观察到排序树的多种变化。

```

myc@myc-virtual-machine:~/Desktop/lab6$ python3 client.py
-----下面展示实验过程-----
本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文之前不存在，直接选择位置： 0
插入的明文： apple 位置： 0
-----此时编码树的结果-----
apple
本地表之中小于要插入明文的所有明文出现次数总和： 1
要插入的明文之前不存在，直接选择位置： 1
插入的明文： pear 位置： 1
-----此时编码树的结果-----
apple pear
本地表之中小于要插入明文的所有明文出现次数总和： 1
要插入的明文之前不存在，直接选择位置： 1
插入的明文： banana 位置： 1
-----此时编码树的结果-----
apple banana pear
本地表之中小于要插入明文的所有明文出现次数总和： 2
要插入的明文之前不存在，直接选择位置： 2
插入的明文： orange 位置： 2
-----此时编码树的结果-----
apple banana orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 2
要插入的明文之前不存在，直接选择位置： 2
插入的明文： cherry 位置： 2
-----此时编码树的结果-----
apple banana cherry orange pear

```

图 3: 开始实验

一开始，插入的明文并不重复，因此此时的编码树情况较为简单。

```

本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 1 ] 随机选择的位置： 1
插入的明文： apple 位置： 1
-----此时编码树的结果-----
apple apple banana cherry orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 3
要插入的明文已经存在,选择范围:[ 3 , 4 ] 随机选择的位置： 3
插入的明文： cherry 位置： 3
-----此时编码树的结果-----
apple apple banana cherry cherry orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 5
要插入的明文已经存在,选择范围:[ 5 , 6 ] 随机选择的位置： 5
插入的明文： orange 位置： 5
-----此时编码树的结果-----
apple apple banana cherry cherry orange orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 2 ] 随机选择的位置： 0
插入的明文： apple 位置： 0
-----此时编码树的结果-----
apple apple apple banana cherry cherry orange orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 3 ] 随机选择的位置： 3
插入的明文： apple 位置： 3
-----此时编码树的结果-----
apple apple apple apple banana cherry cherry orange orange pear
本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 4 ] 随机选择的位置： 1
插入的明文： apple 位置： 1
-----此时编码树的结果-----
apple apple apple apple apple banana cherry cherry orange orange pear

```

图 4: 出现相同明文

随着程序运行，**开始出现相同的明文**，此时可以观察到关于节点插入位置选择的变化。

```

本地表之中小于要插入明文的所有明文出现次数总和：0
要插入的明文已经存在,选择范围:[ 0 , 1 ] 随机选择的位置：1
插入的明文：apple 位置：1
-----此时编码树的结果-----
apple apple banana cherry orange pear

```

图 5: 相同的 apple 出现

可以从上图观察到 apple 的插入位置。此时 apple 可以插入的范围是 [0,1]，而随机选择的位置是 1，则进行插入。

```

本地表之中小于要插入明文的所有明文出现次数总和：7
要插入的明文已经存在,选择范围:[ 7 , 9 ] 随机选择的位置：9
插入的明文：cherry 位置：9
-----此时编码树的结果-----
apple apple apple apple apple banana cherry cherry cherry orange orange pear

本地表之中小于要插入明文的所有明文出现次数总和：7
要插入的明文已经存在,选择范围:[ 7 , 10 ] 随机选择的位置：7
插入的明文：cherry 位置：7
-----此时编码树的结果-----
apple apple apple apple apple apple banana cherry cherry cherry cherry orange orange pear

本地表之中小于要插入明文的所有明文出现次数总和：0
要插入的明文已经存在,选择范围:[ 0 , 6 ] 随机选择的位置：3
插入的明文：apple 位置：3
-----此时编码树的结果-----
apple apple apple apple apple apple apple banana cherry cherry cherry cherry orange orange pear

本地表之中小于要插入明文的所有明文出现次数总和：8
要插入的明文已经存在,选择范围:[ 8 , 12 ] 随机选择的位置：9
插入的明文：cherry 位置：9
-----此时编码树的结果-----
apple apple apple apple apple apple apple banana cherry cherry cherry cherry cherry orange orange pear

```

图 6: 全部插入结束

我们可以观察到排序是依据字典序进行的，每一步实验结果都和理论完全对应得上。从下图可以看出：

```

本地表之中小于要插入明文的所有明文出现次数总和：5
要插入的明文已经存在,选择范围:[ 5 , 6 ] 随机选择的位置：5
插入的明文：orange 位置：5
-----此时编码树的结果-----
apple apple banana cherry cherry orange orange pear

```

图 7: presum 结果

然后进行搜索和本地表的展示。可以看到，本地表的情况如下图所示，并且和数据库的数据项的数量也可以对应的上：


```

本地表之中小于要插入明文的所有明文出现次数总和：8
要插入的明文已经存在,选择范围:[ 8 , 12 ] 随机选择的位置：9
插入的明文：cherry 位置：9
-----此时编码树的结果-----
apple apple apple apple apple apple apple banana cherry cherry cherry cherry orange orange pear

-----假设我们搜索 b 和 p 之间的数据-----
ciphtertext: odBNHURLVNYX4FWQYtjn41j9FKuRa5N3NIND7TRDsdrsLZhwhSynv06IIdqxsN0q plaintext: banana
ciphtertext: WMhrFlaWozI8soQWNikoUD5VuSkgBBkLLACKbRI2L3lZtW/GqTrFd1KYUmumr+tv plaintext: orange
ciphtertext: 0zrXqEPooC6OZ4EUjt+9vRXT35LhwsOMBn+OuCUi9Lp1kDJ/zbkzIJC9UzlFEKaL plaintext: cherry
ciphtertext: 6dgsrvoHdMJ/8S68Z9qoHDUWh6pNhi5r2nApMtVwqUtqnBHe8PIrvfpAH9IIXjuD plaintext: cherry
ciphtertext: ZJAIMmrdIe6aXNNpXJ2BRDJ6LX92L6BfBFSFddQeYwy7q1mON+cYiAdjKzzT/IeA plaintext: orange
ciphtertext: nzJ9gv0+vetQLPykGUF+RQ3TGixubn5l9kYncuhsBGlP8+ytRqLjH0u19j69xjPq plaintext: cherry
ciphtertext: SVPPtB42mDRKTfrVuzpimQKSbQyNJK3IQRCLgLYWfjzQ/5299e4iUGS9K+/V5+3T plaintext: cherry
ciphtertext: eWQ/9lrb/GgfzQWvF7qIGFF5pJE95FqStdFeOKyjlBvUn0tR4WfFZmwdYZ5K24ax plaintext: cherry

-----下面展示本地表内容-----
local table: {'apple': 7, 'pear': 1, 'banana': 1, 'orange': 2, 'cherry': 5}
本地表共：16 数据项
现在数据库之中共有：16 个数据项
数据数量一致，实验成功！
myc@myc-virtual-machine:~/Desktop/lab6$

```

图 8: 搜索和本地表的展示

仔细观察搜索结果，这里是要求：“搜索字母 b 和 p 之间的所有内容”，那么根据我们最后展示的完整的编码树叶节点排序，可以得到从 banana 开始的一直到 pear 之前的所有单词，都是我们要搜索的结果，可以看到结果对应的上：

```

banana cherry cherry cherry cherry cherry orange orange pear

qG1Cw2zRWVJ/eNGXw1STAvIAF65uNhNKuciUd plaintext: banana
ncqh90+Dv+SlthSKAgHk779qeifTm+1dIH/Xi plaintext: orange
jjoymareZaIQjCuK+WSnE5X/whEN0e4aV1uXb plaintext: cherry
koIqnfr85u1Y29kcaTbZXdy0qDyL+fB6aEWb/ plaintext: cherry
BDSSMXgB2HCb+rRQM6secB9xvwg0y5eI1ug6H plaintext: orange
LSL0k9aQWW7KBKppSuH02jb28gBerIQffoMBz plaintext: cherry
cgIwJKT4FxcnnwNzFCc8tbTOARrSuY09SR6DA plaintext: cherry
JLKLbZLqUWarsIGcyLAN0hDajGiZrjIjY4gDL plaintext: cherry

```

图 9: 搜索结果

然后，我换了一种方式，改为通过编码观察编码树。

```

-----下面展示实验过程-----
本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文之前不存在，直接选择位置： 0
插入的明文： apple 位置： 0
-----此时编码树的结果-----
2305843009213693952

本地表之中小于要插入明文的所有明文出现次数总和： 1
要插入的明文之前不存在，直接选择位置： 1
插入的明文： pear 位置： 1
-----此时编码树的结果-----
2305843009213693952 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 1
要插入的明文之前不存在，直接选择位置： 1
插入的明文： banana 位置： 1
-----此时编码树的结果-----
2305843009213693952 2882303761517117440 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 2
要插入的明文之前不存在，直接选择位置： 2
插入的明文： orange 位置： 2
-----此时编码树的结果-----
2305843009213693952 2882303761517117440 3170534137668829184 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 2
要插入的明文之前不存在，直接选择位置： 2
插入的明文： cherry 位置： 2
-----此时编码树的结果-----
2305843009213693952 2882303761517117440 3026418949592973312 3170534137668829184 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在，选择范围：[ 0 , 1 ] 随机选择的位置： 0
插入的明文： apple 位置： 0
-----此时编码树的结果-----
1152921504606846976 2305843009213693952 2882303761517117440 3026418949592973312 3170534137668829184 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 3
要插入的明文已经存在，选择范围：[ 3 , 4 ] 随机选择的位置： 3
插入的明文： cherry 位置： 3
-----此时编码树的结果-----
1152921504606846976 2305843009213693952 2882303761517117440 295436135555045376 3026418949592973312 3170534137668829184 3458764513820540928

```

图 10: 通过编码观察实验结果

仔细观察可以发现，一方面树的平衡调整不会引起编码的更新，插入可能引发节点内数据密文编码更新，但是其他节点不发生更新。

```

本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 4 ] 随机选择的位置： 3
插入的明文： apple 位置： 3
-----此时编码树的结果-----
1152921504606846976 1729382256910270464 2017612633061982208 2161727821137838080
2305843009213693952 2882303761517117440 2954361355555045376 3026418949592973312
3098476543630901248 3170534137668829184 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 3 , 5 ] 随机选择的位置： 1
插入的明文： apple 位置： 1
-----此时编码树的结果-----
1152921504606846976 1441151880758558720 1729382256910270464 2017612633061982208
2161727821137838080 2305843009213693952 2882303761517117440 2954361355555045376
3026418949592973312 3098476543630901248 3170534137668829184 3458764513820540928

```

图 11: 对比

同时，由于编码唯一，我们也可以直观地观察到相同明文在插入前后的编码树的变化，并且也能证明试验成功。

```

本地表之中小于要插入明文的所有明文出现次数总和： 7
要插入的明文已经存在,选择范围:[ 7 , 10 ] 随机选择的位置： 8
插入的明文： cherry 位置： 8
-----此时编码树的结果-----
1152921504606846976 1441151880758558720 1729382256910270464 2017612633061982208
2161727821137838080 2305843009213693952 2882303761517117440 2918332558536081408
2936346957045563392 2954361355555045376 3026418949592973312 3098476543630901248
3170534137668829184 3458764513820540928

本地表之中小于要插入明文的所有明文出现次数总和： 0
要插入的明文已经存在,选择范围:[ 0 , 6 ] 随机选择的位置： 5
插入的明文： apple 位置： 5
-----此时编码树的结果-----
1152921504606846976 1441151880758558720 1729382256910270464 2017612633061982208
2161727821137838080 2233785415175766016 2305843009213693952 2882303761517117440
2918332558536081408 2936346957045563392 2954361355555045376 3026418949592973312
3098476543630901248 3170534137668829184 3458764513820540928

```

图 12: 观察插入位置

可以看到，结果已经完全地展示清楚了。**本次实验取得圆满成功！**

五、 心得体会

在本次实验中，我通过复现一个简单的频率隐藏 OPE 方案，深入理解了这个方案的原理和实现细节。

在实验中，我首先实现了一个哈希表，用于存储明文和它们的哈希编码。哈希表的实现基于 Python 的字典，它将明文作为键，将哈希编码作为值。我使用了 Python 的内置哈希函数来计算哈希编码，并将哈希编码映射到一个随机的位置上，从而实现了对其加密和排序。

本次实验并不困难，但是我通过这个实验学习到了 **UDF** 的使用，并且通过编写 sql 语句，复习了我在数据库的课程上所学到的知识。同时，我也仔细阅读了实验指导书上的 C++ 源代码，研究了这篇论文的实验细节。

当然，本次实验也遇到一些困难，最重要的一点是大量底层的实现都通过 C++ 完成封装，使用 python 有时候无法观察到底层的细节，这给我带来了一些困难，我只能尽力通过 sql 语句展示对数据操作的前后变化，来观察数据库内部的变化。还有就是当程序运行一次之后，最好（甚至必须）要重置数据库的表项，不然在多次实验后，数据库内部已有大量数据表项，会对实验结果的分析造成困难。

最后，我尝试了复现论文之中关于其他内容的统计结果，但遗憾的是单纯改变 python 代码不能取得很好的效果，好像直接修改底层 c++ 代码是更好的方案。这方面可以日后继续探索。

六、 附录一：完整 client.py 代码

实验完整代码如下所示：

完整 client.py 代码

```
1 import pymysql
2 import random
3 from Crypto.Cipher import AES
4 from Crypto.Random import get_random_bytes
5 from Crypto.Util.Padding import pad, unpad
6 import base64
7
8 local_table = {}
9 key = get_random_bytes(16)
10 base_iv = get_random_bytes(16)
11
12 def AES_ENC(plaintext, iv):
13     # AES加密
14     aes = AES.new(key, AES.MODE_CBC, iv=iv)
15     padded_data = pad(plaintext, AES.block_size, style='pkcs7')
16     ciphertext = aes.encrypt(padded_data)
17     return ciphertext
18
19 def AES_DEC(ciphertext, iv):
20     # AES解密
21     aes = AES.new(key, AES.MODE_CBC, iv=iv)
22     padded_data = aes.decrypt(ciphertext)
23     plaintext = unpad(padded_data, AES.block_size, style='pkcs7')
24     return plaintext
25
26 def Random_Encrypt(plaintext):
27     # 随机生成iv来保证加密结果的随机性
28     iv = get_random_bytes(16)
29     ciphertext = AES_ENC(iv + AES_ENC(plaintext.encode('utf-8'), iv), base_iv)
30     ciphertext = base64.b64encode(ciphertext)
31     return ciphertext.decode('utf-8')
32
33 def Random_Decrypt(ciphertext):
34     plaintext = AES_DEC(base64.b64decode(ciphertext.encode('utf-8')), base_iv)
35     plaintext = AES_DEC(plaintext[16:], plaintext[:16])
36     return plaintext.decode('utf-8')
37
38 def CalPos(plaintext):
39     # 插入plaintext, 返回对应的Pos
40     presum = sum([v for k, v in local_table.items() if k < plaintext])
41     print("本地表之中小于要插入明文的所有明文出现次数总和：", presum)
```

```

42     if plaintext in local_table:
43         local_table[plaintext] += 1
44         the_pos=random.randint(presum, presum + local_table[plaintext] - 1)
45         print("要插入的明文已经存在,选择范围:[" ,presum,",",presum +
46             local_table[plaintext] - 1,"]",end=' ')
47         print("随机选择的位置:",the_pos)
48         return the_pos
49     else:
50         local_table[plaintext] = 1
51         print('要插入的明文之前不存在,直接选择位置:',presum)
52         return presum
53
54 def GetLeftPos(plaintext):
55     return sum([v for k, v in local_table.items() if k < plaintext])
56
57 def GetRightPos(plaintext):
58     return sum([v for k, v in local_table.items() if k <= plaintext])
59
60 def Insert(plaintext):
61     ciphertext = Random_Encrypt(plaintext)
62     # 连接数据库
63     conn = pymysql.connect(host='localhost', user='user',
64                             passwd='123456', database='test_db')
65     cur = conn.cursor()
66     the_result = CalPos(plaintext)
67     print("插入的明文:",plaintext,'位置:',the_result)
68     cur.execute(f"call pro_insert({the_result},{ciphertext})")
69     conn.commit()
70     print("-----此时编码树的结果-----")
71     cur.execute(
72         f"select ciphertext from example order by encoding")
73     results = cur.fetchall()
74     for result in results:
75         print(Random_Decrypt(result[0]),end=" ")
76     print("\n")
77     conn.close()
78
79 def Search(left, right):
80     # 搜索[left, right]中的信息
81     left_pos = GetLeftPos(left)
82     right_pos = GetRightPos(right)
83     # 连接数据库
84     conn = pymysql.connect(host='localhost', user='user',
85                             passwd='123456', database='test_db')
86     cur = conn.cursor()
87     cur.execute(
88         f"select ciphertext from example where encoding >= FHSearch({left_pos}

```

```

        }) and encoding < FHSearch({right_pos}))
89     rest = cur.fetchall()
90     for x in rest:
91         print(f"ciphtertext: {x[0]} plaintext: {Random_Decrypt(x[0])}")
92
93 if __name__ == '__main__':
94     # 插入明文, 同时设置了一部分重复的内容
95     print("-----")
96     print("-----下面展示实验过程-----")
97     print("-----")
98     # 加入了一大堆 apple 和 cherry, 希望观察结果和树的变化
99     for ciphertext in ['apple', 'pear', 'banana', 'orange', 'cherry', 'apple',
100                        , 'cherry', 'orange', 'apple', 'apple', 'apple', 'apple', 'cherry', '
101                        cherry', 'apple', 'cherry']:
102         Insert(ciphertext)
103
104     # 假设我们搜索b和p之间的数据
105     print("-----")
106     print("-----假设我们搜索 b 和 p 之间的数据-----")
107     print("-----")
108     Search('b', 'p')
109
110     print("-----")
111     print("-----下面展示本地表内容-----")
112     print("-----")
113     print("local_table:", local_table)
114     total=sum([v for k, v in local_table.items()])
115     print("本地表共:", total, "数据项")
116     # 连接数据库
117     conn = pymysql.connect(host='localhost', user='user',
118                            passwd='123456', database='test_db')
119     cur = conn.cursor()
120     cur.execute("select count(distinct encoding) from example")
121     results = cur.fetchall()
122     for result in results:
123         print("现在数据库之中共有:", result[0], "个数据项")
124         if result[0]==total:
125             print("数据数量一致, 实验成功!")

```

七、 附录二：完整输出结果

完整输出结果（明文）

```
1  _____
2  _____下面展示实验过程_____
3  _____
4  本地表之中小于要插入明文的所有明文出现次数总和： 0
5  要插入的明文之前不存在，直接选择位置：0
6  插入的明文：apple 位置：0
7  _____此时编码树的结果_____
8  apple
9
10 本地表之中小于要插入明文的所有明文出现次数总和： 1
11 要插入的明文之前不存在，直接选择位置：1
12 插入的明文：pear 位置：1
13  _____此时编码树的结果_____
14  apple pear
15
16 本地表之中小于要插入明文的所有明文出现次数总和： 1
17 要插入的明文之前不存在，直接选择位置：1
18 插入的明文：banana 位置：1
19  _____此时编码树的结果_____
20  apple banana pear
21
22 本地表之中小于要插入明文的所有明文出现次数总和： 2
23 要插入的明文之前不存在，直接选择位置：2
24 插入的明文：orange 位置：2
25  _____此时编码树的结果_____
26  apple banana orange pear
27
28 本地表之中小于要插入明文的所有明文出现次数总和： 2
29 要插入的明文之前不存在，直接选择位置：2
30 插入的明文：cherry 位置：2
31  _____此时编码树的结果_____
32  apple banana cherry orange pear
33
34 本地表之中小于要插入明文的所有明文出现次数总和： 0
35 要插入的明文已经存在,选择范围:[ 0 , 1 ] 随机选择的位置：1
36 插入的明文：apple 位置：1
37  _____此时编码树的结果_____
38  apple apple banana cherry orange pear
39
40 本地表之中小于要插入明文的所有明文出现次数总和： 3
41 要插入的明文已经存在,选择范围:[ 3 , 4 ] 随机选择的位置：3
42 插入的明文：cherry 位置：3
43  _____此时编码树的结果_____
44  apple apple banana cherry cherry orange pear
```



```
45
46 本地表之中小于要插入明文的所有明文出现次数总和： 5
47 要插入的明文已经存在,选择范围:[ 5 , 6 ] 随机选择的位置： 5
48 插入的明文： orange 位置： 5
49 -----此时编码树的结果-----
50 apple apple banana cherry cherry orange orange pear
51
52 本地表之中小于要插入明文的所有明文出现次数总和： 0
53 要插入的明文已经存在,选择范围:[ 0 , 2 ] 随机选择的位置： 0
54 插入的明文： apple 位置： 0
55 -----此时编码树的结果-----
56 apple apple apple banana cherry cherry orange orange pear
57
58 本地表之中小于要插入明文的所有明文出现次数总和： 0
59 要插入的明文已经存在,选择范围:[ 0 , 3 ] 随机选择的位置： 3
60 插入的明文： apple 位置： 3
61 -----此时编码树的结果-----
62 apple apple apple apple banana cherry cherry orange orange pear
63
64 本地表之中小于要插入明文的所有明文出现次数总和： 0
65 要插入的明文已经存在,选择范围:[ 0 , 4 ] 随机选择的位置： 1
66 插入的明文： apple 位置： 1
67 -----此时编码树的结果-----
68 apple apple apple apple apple banana cherry cherry orange orange pear
69
70 本地表之中小于要插入明文的所有明文出现次数总和： 0
71 要插入的明文已经存在,选择范围:[ 0 , 5 ] 随机选择的位置： 1
72 插入的明文： apple 位置： 1
73 -----此时编码树的结果-----
74 apple apple apple apple apple apple banana cherry cherry orange orange pear
75
76 本地表之中小于要插入明文的所有明文出现次数总和： 7
77 要插入的明文已经存在,选择范围:[ 7 , 9 ] 随机选择的位置： 9
78 插入的明文： cherry 位置： 9
79 -----此时编码树的结果-----
80 apple apple apple apple apple apple banana cherry cherry cherry orange orange
    pear
81
82 本地表之中小于要插入明文的所有明文出现次数总和： 7
83 要插入的明文已经存在,选择范围:[ 7 , 10 ] 随机选择的位置： 7
84 插入的明文： cherry 位置： 7
85 -----此时编码树的结果-----
86 apple apple apple apple apple apple banana cherry cherry cherry cherry orange
    orange pear
87
88 本地表之中小于要插入明文的所有明文出现次数总和： 0
89 要插入的明文已经存在,选择范围:[ 0 , 6 ] 随机选择的位置： 3
90 插入的明文： apple 位置： 3
```



```

91  |—————此时编码树的结果—————
92  |apple apple apple apple apple apple apple apple banana cherry cherry cherry cherry
   |orange orange pear
93  |
94  |本地表之中小于要插入明文的所有明文出现次数总和： 8
95  |要插入的明文已经存在,选择范围:[ 8 , 12 ] 随机选择的位置： 9
96  |插入的明文： cherry 位置： 9
97  |—————此时编码树的结果—————
98  |apple apple apple apple apple apple apple apple banana cherry cherry cherry cherry
   |cherry orange orange pear
99  |
100 |—————
101 |—————假设我们搜索 b 和 p 之间的数据—————
102 |—————
103 |ciphtertext: odBNHURLYNYX4FWQYtjn41j9FKuRa5N3NIND7TRDsdrsLZhwhSynvO6lJdqxsN0q
   |plaintext: banana
104 |ciphtertext: WMhrFlaWOzI8soQWNlkoUD5VuSkgBBkLLACKbRI2L3lZtW/GqTrFd1KYUmumr+tV
   |plaintext: orange
105 |ciphtertext: 0zrXqEPooC6OZ4EUjt+9vRXT35LhwsOMBn+OuCUi9Lp1kDJ/zbkzIJC9UzlFEKaL
   |plaintext: cherry
106 |ciphtertext: 6dgsrvoHdMJ/8S68Z9qoHDUWh6pNhi5r2nApMtVwqUtqnBHe8PIrvfpAH9IlxjuD
   |plaintext: cherry
107 |ciphtertext: ZJAIMmrdIe6aXNNpXJ2BRDJ6lX92L6BfBFSFddQeYwy7q1mON+cYiAdjkzzT/IeA
   |plaintext: orange
108 |ciphtertext: nzJ9gvO+vetQLPykGUF+RQ3TGixubn5l9kYncuhsBGlp8+ytRqLjH0u19j69xjPq
   |plaintext: cherry
109 |ciphtertext: SVPpTB42mDRKTfrVuzpimQKSbQyNJK3lQRCLgLYWfjzQ/5299e4iUGS9K+/V5+3T
   |plaintext: cherry
110 |ciphtertext: eWQ/9lrb/GgfzQWvF7qIGFF5pJE95FqStdFeOKyjlBvUn0tR4WfFZmwdYZ5K24ax
   |plaintext: cherry
111 |—————
112 |—————下面展示本地表内容—————
113 |—————
114 |local_table: {'apple ': 7, 'pear ': 1, 'banana ': 1, 'orange ': 2, 'cherry ': 5}
115 |本地表共： 16 数据项
116 |现在数据库之中共有： 16 个数据项
117 |数据数量一致，实验成功！

```

完整输出结果（编码）

```

1  |—————
2  |—————下面展示实验过程—————
3  |—————
4  |本地表之中小于要插入明文的所有明文出现次数总和： 0
5  |要插入的明文之前不存在，直接选择位置： 0
6  |插入的明文： apple 位置： 0
7  |—————此时编码树的结果—————
8  |2305843009213693952
9  |

```

```
10 本地表之中小于要插入明文的所有明文出现次数总和： 1
11 要插入的明文之前不存在，直接选择位置：1
12 插入的明文： pear 位置：1
13 -----此时编码树的结果-----
14 2305843009213693952 3458764513820540928
15
16 本地表之中小于要插入明文的所有明文出现次数总和： 1
17 要插入的明文之前不存在，直接选择位置：1
18 插入的明文： banana 位置：1
19 -----此时编码树的结果-----
20 2305843009213693952 2882303761517117440 3458764513820540928
21
22 本地表之中小于要插入明文的所有明文出现次数总和： 2
23 要插入的明文之前不存在，直接选择位置：2
24 插入的明文： orange 位置：2
25 -----此时编码树的结果-----
26 2305843009213693952 2882303761517117440 3170534137668829184
    3458764513820540928
27
28 本地表之中小于要插入明文的所有明文出现次数总和： 2
29 要插入的明文之前不存在，直接选择位置：2
30 插入的明文： cherry 位置：2
31 -----此时编码树的结果-----
32 2305843009213693952 2882303761517117440 3026418949592973312
    3170534137668829184 3458764513820540928
33
34 本地表之中小于要插入明文的所有明文出现次数总和： 0
35 要插入的明文已经存在,选择范围:[ 0 , 1 ] 随机选择的位置：0
36 插入的明文： apple 位置：0
37 -----此时编码树的结果-----
38 1152921504606846976 2305843009213693952 2882303761517117440
    3026418949592973312 3170534137668829184 3458764513820540928
39
40 本地表之中小于要插入明文的所有明文出现次数总和： 3
41 要插入的明文已经存在,选择范围:[ 3 , 4 ] 随机选择的位置：3
42 插入的明文： cherry 位置：3
43 -----此时编码树的结果-----
44 1152921504606846976 2305843009213693952 2882303761517117440
    295436135555045376 3026418949592973312 3170534137668829184
    3458764513820540928
45
46 本地表之中小于要插入明文的所有明文出现次数总和： 5
47 要插入的明文已经存在,选择范围:[ 5 , 6 ] 随机选择的位置：5
48 插入的明文： orange 位置：5
49 -----此时编码树的结果-----
50 1152921504606846976 2305843009213693952 2882303761517117440
    295436135555045376 3026418949592973312 3098476543630901248
    3170534137668829184 3458764513820540928
```

```
51
52 本地表之中小于要插入明文的所有明文出现次数总和： 0
53 要插入的明文已经存在,选择范围:[ 0 , 2 ] 随机选择的位置： 1
54 插入的明文： apple 位置： 1
55 -----此时编码树的结果-----
56 1152921504606846976 1729382256910270464 2305843009213693952
    2882303761517117440 2954361355555045376 3026418949592973312
    3098476543630901248 3170534137668829184 3458764513820540928
57
58 本地表之中小于要插入明文的所有明文出现次数总和： 0
59 要插入的明文已经存在,选择范围:[ 0 , 3 ] 随机选择的位置： 2
60 插入的明文： apple 位置： 2
61 -----此时编码树的结果-----
62 1152921504606846976 1729382256910270464 2017612633061982208
    2305843009213693952 2882303761517117440 2954361355555045376
    3026418949592973312 3098476543630901248 3170534137668829184
    3458764513820540928
63
64 本地表之中小于要插入明文的所有明文出现次数总和： 0
65 要插入的明文已经存在,选择范围:[ 0 , 4 ] 随机选择的位置： 3
66 插入的明文： apple 位置： 3
67 -----此时编码树的结果-----
68 1152921504606846976 1729382256910270464 2017612633061982208
    2161727821137838080 2305843009213693952 2882303761517117440
    2954361355555045376 3026418949592973312 3098476543630901248
    3170534137668829184 3458764513820540928
69
70 本地表之中小于要插入明文的所有明文出现次数总和： 0
71 要插入的明文已经存在,选择范围:[ 0 , 5 ] 随机选择的位置： 1
72 插入的明文： apple 位置： 1
73 -----此时编码树的结果-----
74 1152921504606846976 1441151880758558720 1729382256910270464
    2017612633061982208 2161727821137838080 2305843009213693952
    2882303761517117440 2954361355555045376 3026418949592973312
    3098476543630901248 3170534137668829184 3458764513820540928
75
76 本地表之中小于要插入明文的所有明文出现次数总和： 7
77 要插入的明文已经存在,选择范围:[ 7 , 9 ] 随机选择的位置： 7
78 插入的明文： cherry 位置： 7
79 -----此时编码树的结果-----
80 1152921504606846976 1441151880758558720 1729382256910270464
    2017612633061982208 2161727821137838080 2305843009213693952
    2882303761517117440 2918332558536081408 2954361355555045376
    3026418949592973312 3098476543630901248 3170534137668829184
    3458764513820540928
81
82 本地表之中小于要插入明文的所有明文出现次数总和： 7
83 要插入的明文已经存在,选择范围:[ 7 , 10 ] 随机选择的位置： 8
```

```

84 插入的明文: cherry 位置: 8
85 此时编码树的结果
86 1152921504606846976 1441151880758558720 1729382256910270464
    2017612633061982208 2161727821137838080 2305843009213693952
    2882303761517117440 2918332558536081408 2936346957045563392
    295436135555045376 3026418949592973312 3098476543630901248
    3170534137668829184 3458764513820540928
87
88 本地表之中小于要插入明文的所有明文出现次数总和: 0
89 要插入的明文已经存在,选择范围:[ 0 , 6 ] 随机选择的位置: 5
90 插入的明文: apple 位置: 5
91 此时编码树的结果
92 1152921504606846976 1441151880758558720 1729382256910270464
    2017612633061982208 2161727821137838080 2233785415175766016
    2305843009213693952 2882303761517117440 2918332558536081408
    2936346957045563392 295436135555045376 3026418949592973312
    3098476543630901248 3170534137668829184 3458764513820540928
93
94 本地表之中小于要插入明文的所有明文出现次数总和: 8
95 要插入的明文已经存在,选择范围:[ 8 , 12 ] 随机选择的位置: 12
96 插入的明文: cherry 位置: 12
97 此时编码树的结果
98 1152921504606846976 1441151880758558720 1729382256910270464
    2017612633061982208 2161727821137838080 2233785415175766016
    2305843009213693952 2882303761517117440 2918332558536081408
    2936346957045563392 295436135555045376 3026418949592973312
    3062447746611937280 3098476543630901248 3170534137668829184
    3458764513820540928
99
100
101 假设我们搜索 b 和 p 之间的数据
102
103 ciphertext: 5sf0dkiCs/WMSNKAYtJB4E1mNtMn6nH2dGmUEthB+iw6Q5Vrz0lwUxd6Q/bDBBG6
    plaintext: banana
104 ciphertext: Do3GfKyAPbm7hYyHLwLAviq/I3VD8gGODGbAfCgn2lWK6jiWr0OEOPufqh3CF0CD
    plaintext: orange
105 ciphertext: tSKOcZyf5P2BHOFJThaqZgAj1nVGhi7N74IxlOIA9TPvcT7Hol3oK6RlXH6BjFw
    plaintext: cherry
106 ciphertext: q9SqW/kkPtziVsjtFTnHv2ZZpRJInOzju+Bx1jSFngneCrjLmla4LsnMgSmDfHR6
    plaintext: cherry
107 ciphertext: AF4dZEjDaZol2cbMzSisD3yhUTBFJwep9sw2kGu3iBh0ufZgrVrooMy3Ac+08LtV
    plaintext: orange
108 ciphertext: JKzl0H2juUlsyx0pc0WcqJjA9sBuf61NaUJi6IC9sVjJrvQ9V5uxOuxhKI0LN4S
    plaintext: cherry
109 ciphertext: hYgdfcDOuxeMqphH0C/R0nZ3MPnZ64xsIk7+o1Ov64FchQnjNE991OBSzDN3pGAs
    plaintext: cherry
110 ciphertext: m7Bo6Q6fvu3H8ZfLHfOk+kQubPMcD37knLxx5Lxjcb4AWGqyLIGn9hlhmgfvA2Q5
    plaintext: cherry

```

```
111 |  
112 |——下面展示本地表内容——|  
113 |  
114 |local_table: {'apple ': 7, 'pear ': 1, 'banana ': 1, 'orange ': 2, 'cherry ': 5}  
115 |本地表共: 16 数据项  
116 |现在数据库之中共有: 16 个数据项  
117 |数据数量一致, 实验成功!
```

NIKU