

数据安全 -- 对称可搜索加密

学号：2013921

姓名：周延霖

专业：信息安全

一、实验名称

对称可搜索加密

二、实验要求

根据正向索引或者倒排索引机制，提供一种可搜索加密方案的模拟实现，应能分别完成加密、陷门生成、检索和解密四个过程

三、实验思路

为了实现可搜索加密方案的模拟实现，在本次实验中我选择倒排索引机制，各个过程具体如下所示：

1、加密

首先，需要选择一个安全的加密算法（如AES）和一个安全的哈希函数（如SHA-256），对于每个文档，使用加密算法将其加密，并计算其哈希值，然后，对每个文档的关键词进行哈希，并将哈希值作为关键词的标识符，对于每个关键词，将其标识符和对应的文档哈希值存储到倒排索引中

假设我们要对明文m进行加密，加密过程如下：

1. 随机生成一个密钥k，可以是一个随机数或者一个字符串
2. 对明文m进行加密，可以使用对称加密算法如AES或DES，也可以使用非对称加密算法如RSA，加密后得到密文c
3. 将密钥k和密文c一起返回给用户

2、陷门生成

对于每个搜索请求，需要使用哈希函数将其转换为一个固定长度的哈希值，然后，使用加密算法将哈希值加密，并将结果作为陷门返回

假设我们要生成一个陷门d，使得只有满足特定条件的明文才能被检索出来，陷门生成过程如下：

1. 首先确定一个特定的条件，如只能检索出长度为10的明文
2. 针对这个条件，生成一个哈希函数H，将满足条件的明文映射到一个固定长度的哈希值
3. 将哈希值作为陷门d返回给用户

3、检索

对于每个搜索请求，需要使用哈希函数将其转换为一个固定长度的哈希值，然后，使用加密算法将哈希值加密，并将结果作为陷门发送到服务器，服务器使用陷门进行搜索，并返回匹配的文档哈希值列表，客户端将文档哈希值解密，并使用哈希函数计算每个文档的哈希值，客户端比较文档哈希值与服务器返回的哈希值，如果匹配，则将该文档添加到结果列表中

假设我们要检索出满足特定条件的明文，检索过程如下：

1. 用户输入一个陷门d，表示要检索出满足特定条件的明文
2. 使用哈希函数H对陷门d进行计算，得到哈希值h
3. 在加密索引表中查找哈希值为h的记录，得到对应的密文c和密钥k
4. 使用密钥k对密文c进行解密，得到明文m
5. 判断明文m是否满足特定条件，如果满足，则返回明文m，否则返回空

4、解密

对于每个结果，客户端使用加密算法将文档哈希值解密，得到原始的文档哈希值，然后，客户端使用哈希函数计算原始的文档哈希值，并比较它与服务器返回的哈希值是否匹配，如果匹配，则客户端使用解密算法将文档解密，并将其显示给用户

假设我们要解密一个密文c，解密过程如下：

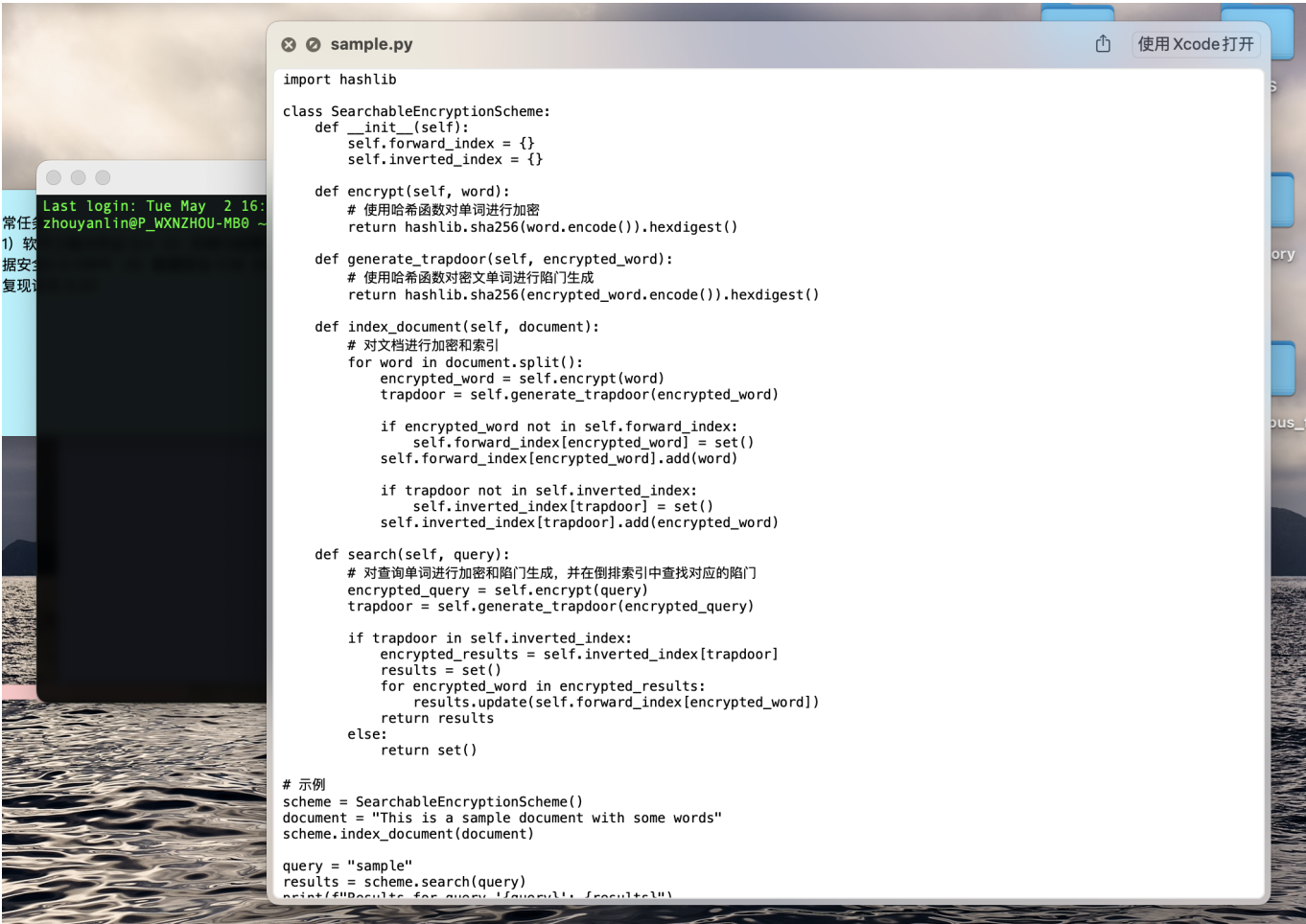
1. 用户输入密文c和密钥k
2. 使用密钥k对密文c进行解密，得到明文m
3. 返回明文m

四、实验过程

- 操作系统:macOS Monterey
- 编译环境:python

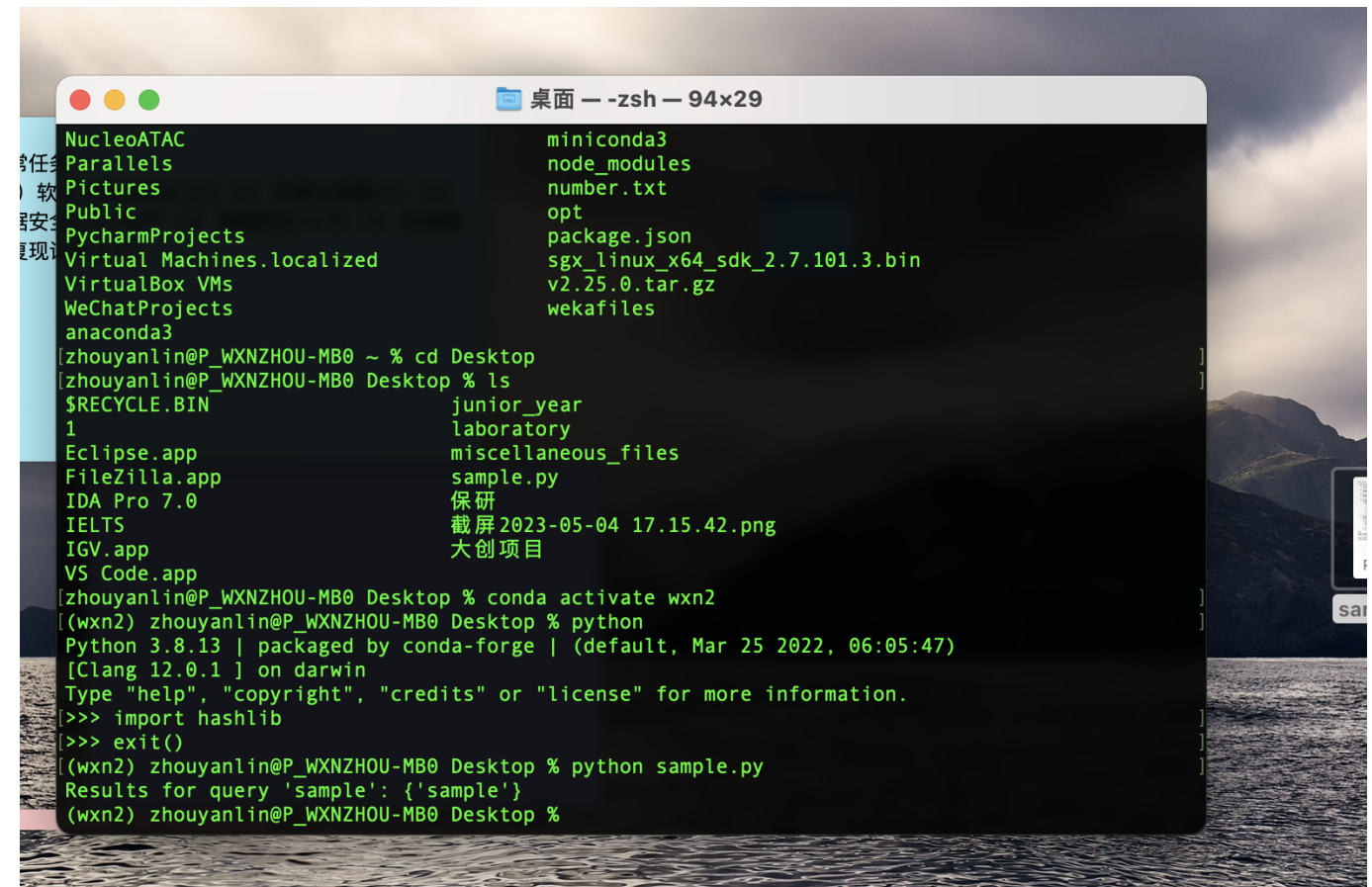
1、创建文件

在桌面上创建sample.py文件如下所示，并打开终端准备接下来的运行程序：



2、运行代码

运行代码如下所示：



The screenshot shows a terminal window titled '桌面 — zsh — 94x29'. It displays two directory listings. The first listing shows files like NucleoATAC, Parallels, Pictures, Public, PycharmProjects, Virtual Machines.localized, VirtualBox VMs, WeChatProjects, anaconda3, miniconda3, node_modules, number.txt, opt, package.json, sgx_linux_x64_sdk_2.7.101.3.bin, v2.25.0.tar.gz, and wekafiles. The second listing shows \$RECYCLE.BIN, 1, Eclipse.app, FileZilla.app, IDA Pro 7.0, IELTS, IGV.app, VS Code.app, junior_year, laboratory, miscellaneous_files, sample.py, 保研, 截屏 2023-05-04 17.15.42.png, and 大创项目. Below the listings, the user runs 'conda activate wxn2', 'python', and 'python sample.py'. The output shows 'Python 3.8.13 | packaged by conda-forge | (default, Mar 25 2022, 06:05:47)' and the results of the 'sample.py' script: {'sample': {'sample': 'sample'}}

可以看出代码运行结果与预期相符

五、心得体会

在本次实验中，首先学习到了对称可搜索加密的相关概念，并探究其和非对称可搜索加密的区别

还了解到其正向索引和倒排索引的具体概念并最后将倒排索引在实验中复现出来

最后通过本次实验对所学到的理论知识进行相应的应用，期待自己未来更好的发展，心想事成、万事胜意、未来可期

六、附录——完整代码

```
import hashlib

class SearchableEncryptionScheme:
    def __init__(self):
        self.forward_index = {}
        self.inverted_index = {}

    def encrypt(self, word):
        # 使用哈希函数对单词进行加密
        return hashlib.sha256(word.encode()).hexdigest()

    def generate_trapdoor(self, encrypted_word):
        # 使用哈希函数对密文单词进行陷门生成
```

```
        return hashlib.sha256(encrypted_word.encode()).hexdigest()

def index_document(self, document):
    # 对文档进行加密和索引
    for word in document.split():
        encrypted_word = self.encrypt(word)
        trapdoor = self.generate_trapdoor(encrypted_word)

        if encrypted_word not in self.forward_index:
            self.forward_index[encrypted_word] = set()
            self.forward_index[encrypted_word].add(word)

        if trapdoor not in self.inverted_index:
            self.inverted_index[trapdoor] = set()
            self.inverted_index[trapdoor].add(encrypted_word)

def search(self, query):
    # 对查询单词进行加密和陷门生成，并在倒排索引中查找对应的陷门
    encrypted_query = self.encrypt(query)
    trapdoor = self.generate_trapdoor(encrypted_query)

    if trapdoor in self.inverted_index:
        encrypted_results = self.inverted_index[trapdoor]
        results = set()
        for encrypted_word in encrypted_results:
            results.update(self.forward_index[encrypted_word])
        return results
    else:
        return set()

# 示例
scheme = SearchableEncryptionScheme()
document = "This is a sample document with some words"
scheme.index_document(document)

query = "sample"
results = scheme.search(query)
print(f"Results for query '{query}': {results}")
```