南開大學

数据安全 课程实验报告

对称可搜索加密方案

学院　网络空间安全学院

专业　信息安全

姓名　齐明杰

学号　2113997

2024 年 5 月 3 日

目　　录

图表

1 实验目的

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。

2 实验原理

2.1 对称可搜索加密（SSE）

对称可搜索加密是一种加密技术，允许用户在加密的数据上执行搜索操作而不需要先解密。这种技术在保证数据机密性的同时，也支持高效的关键词搜索功能，非常适合云计算环境，可以保护存储在外部服务器上的数据不被非法访问。

对称可搜索加密

对称可搜索加密通常包括以下四个基本组件：

- **加密过程**：用户使用密钥在本地对明文文件进行加密并将其上传至服务器；
- **陷门生成过程**：具备检索能力的用户使用密钥生成待查询关键词的陷门（也可以称为令牌），要求陷门不能泄露关键词的任何信息；
- **检索过程**：服务器以关键词陷门为输入，执行检索算法，返回所有包含该陷门对应关键词的密文文件，要求服务器除了能知道密文文件是否包含某个特定关键词外，无法获得更多信息；
- **解密过程**：用户使用密钥解密服务器返回的密文文件，获得查询结果。

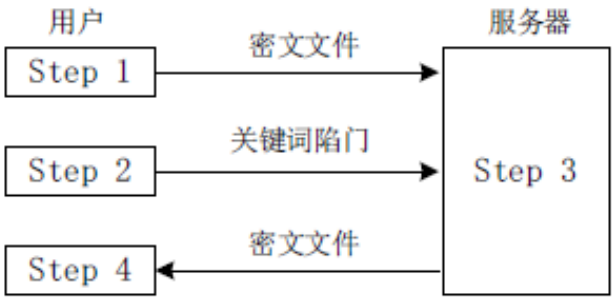

图 2.1.1: 可搜索加密过程

2.2 倒排索引

对称可搜索加密多数基于索引结构来提升检索的效率，比如倒排索引。

倒排索引是一种数据结构，用于存储一个关键词到一个或多个文档的映射。在 SSE 中，这意味着每个关键词都关联到包含该词的加密文档标识符列表。这种结构使得搜索效率得到极大提升，尤其是在文档集合很大时。倒排索引的使用在提升搜索效率的同时，必须确保不泄漏关键数据。因此，所有的索引项（包括关键词和文档标识符）都需要加

密处理。同时，对称可搜索加密方案设计应尽量减少关于搜索和数据访问模式的泄露，以避免敏感信息的泄露。

> **倒排索引**
>
> **倒排索引（Inverted index）**，也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。如图所示，关键词 w1 索引了一个列表，列表中存储了所有关联的文档的标识 ID。

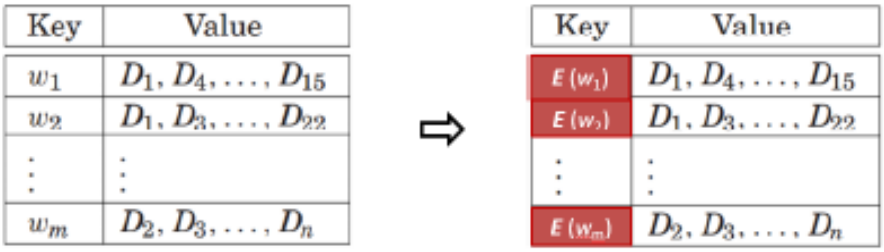

图 2.2.2: 倒排索引示例

3 实验过程

3.1 实验代码

本次我的代码将分为两个部分进行编写，以模块的形式封装加密类：

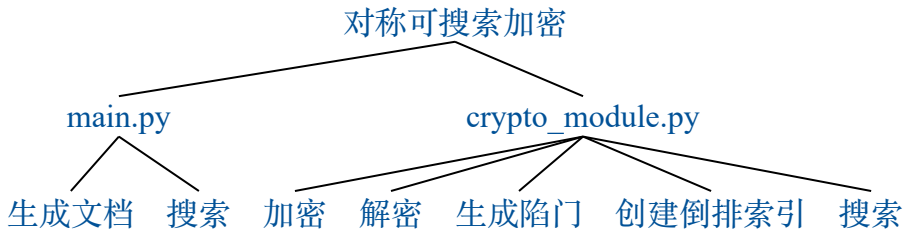

图 3.1.3: 代码结构

3.1.1 可搜索加密类

```python
from Crypto.Cipher import AES
from Crypto.Hash import SHA256, HMAC
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
import base64

class SearchableEncryption:
```

```
8       def __init__(self, key):
9           self.key = key
10      def encrypt(self, text):
11      def decrypt(self, encrypted_text):
12      def generate_trapdoor(self, keyword):
13      def create_index(self, documents):
14      def search(self, index, trapdoor):
```

本次实验我将会完成上述的各个函数，最终这个模块将能够实现对称加解密，生成倒排索引，生成陷门，搜索的操作。

3.1.1.1 加密文本

```python
def encrypt(self, text):
    """ 加密文本 """
    iv = get_random_bytes(16)
    cipher = AES.new(self.key, AES.MODE_CBC, iv)
    encrypted_text = cipher.encrypt(pad(text.encode(), AES.block_size))
    return base64.b64encode(iv + encrypted_text).decode()
```

解析

此方法使用 AES 加密算法的 CBC 模式对文本进行加密。首先生成一个随机的初始化向量（IV），保证即使同一数据多次加密也将产生不同的加密结果。文本首先被转化为字节，然后填充到适合 AES 块大小的长度。加密后的数据前面附加 IV，然后整个结果使用 base64 编码，以便可以安全地以文本形式存储或传输。

3.1.1.2 解密文本

```python
def decrypt(self, encrypted_text):
    """ 解密文本 """
    data = base64.b64decode(encrypted_text)
    iv = data[:16]
    encrypted_text = data[16:]
    cipher = AES.new(self.key, AES.MODE_CBC, iv)
    original_text = unpad(cipher.decrypt(encrypted_text), AES.block_size)
    return original_text.decode()
```

解析

解密过程是加密过程的逆过程。首先从 base64 编码的字符串中解码出原始的二进制数据，提取出用于加密的 IV，并获取实际的加密文本。然后使用同一密钥和提取的 IV，通过 AES 的 CBC 模式解密文本。最后，去除填充并将字节转换回字符串形式。

3.1.1.3 生成陷门

```python
def generate_trapdoor(self, keyword):
    """ 生成陷门（加密关键词），依赖密钥的参与 """
    hmac = HMAC.new(self.key, digestmod=SHA256)
    hmac.update(keyword.encode())
    return base64.b64encode(hmac.digest()).decode()
```

解析

陷门生成使用了 HMAC 和 SHA-256 散列函数。这种方式通过结合密钥和关键词生成一个安全的陷门，这个陷门可以用于后续的搜索操作，而不暴露原始关键词。这种方法不仅保护了关键词的安全，也确保了只有知道密钥的实体才能生成有效的搜索陷门。

> **HMAC（Hash-based Message Authentication Code，基于哈希的消息认证码）**
>
> HMAC 的工作原理是通过一个加密哈希函数（如 SHA-256）和一个秘密密钥对数据（在本案例中是关键词）进行处理，生成一个固定长度的哈希值。这个哈希值作为陷门，用于后续的索引和搜索操作。由于每次生成的 HMAC 值不仅依赖于关键词，还依赖于密钥，因此即使相同的关键词在不同的系统或不同的密钥下生成的陷门也会不同。这种依赖密钥的特性增强了系统的安全性，因为只有知道正确密钥的实体才能生成或验证有效的 HMAC 值。
>
> 使用 HMAC 的一个重要好处是它的安全性不完全依赖于哈希函数的抗碰撞性，这意味着即使哈希函数本身存在潜在的安全问题，只要密钥保持安全，使用 HMAC 的系统仍然能提供较高的安全保障。因此，在对称可搜索加密系统中引入 HMAC，可以有效地防止未授权用户利用关键词泄漏信息或篡改索引。

3.1.1.4 创建倒排索引

```python
def create_index(self, documents):
    """ 创建倒排索引 """
    index = {}
    for doc_id, content in documents.items():
        words = set(content.split())
        for word in words:
            encrypted_word = self.generate_trapdoor(word)
            if encrypted_word in index:
                index[encrypted_word].add(doc_id)
            else:
                index[encrypted_word] = {doc_id}
```

```
12      return index
```

解析

此方法创建一个倒排索引，用于存储每个加密关键词与包含该关键词的文档 ID 之间的关系。这样，搜索操作可以快速定位包含特定关键词的文档。倒排索引是通过加密每个词并将其存储在字典中实现的，其中字典的键是加密后的关键词，值是文档 ID 集合。

3.1.1.5 根据陷门搜索

```python
def search(self, index, trapdoor):
    """ 根据陷门进行搜索 """
    return index.get(trapdoor, set())
```

解析

搜索方法使用陷门来查询倒排索引。给定一个陷门，此方法返回包含对应加密关键词的所有文档的 ID 集合。如果没有文档包含该关键词，返回一个空集合。这允许系统在不解密文档内容的情况下进行高效的关键词搜索。

3.1.2 主函数

使用 SearchableEncryption 类来创建和搜索加密的文档索引。主要步骤包括生成文档、加密、创建索引、搜索关键词，并显示搜索结果。

3.1.2.1 生成文档

```python
fake = Faker()
documents = {}
for doc_id in range(1, NUM_DOCUMENTS + 1):
    document = ' '.join(fake.sentence() for _ in
    range(SENTENCES_PER_DOCUMENT))
    documents[doc_id] = document
```

解析

首先，使用 Faker 库生成一定数量的假文档。每个文档由指定数量的句子组成，这些句子是随机生成的，模拟真实的文档内容。Faker 是一个非常有用的库，用于在测试和模拟数据中生成各种真实感数据。

3.1.2.2 加密文档并创建索引

```python
encrypted_docs = {doc_id: se.encrypt(text) for doc_id, text in documents.items()}
index = se.create_index(documents)
```

解析

使用 SearchableEncryption 类的实例对每个文档进行加密，并创建一个倒排索引。加密过程涉及到 AES 算法，确保文档内容的安全性。索引是根据文档中的每个词创建的，每个词通过生成陷门（即加密的关键词）来标识，这些陷门和文档 ID 相关联存储在索引中。

3.1.2.3 搜索

```python
results = {}
for keyword in keywords:
    trapdoor = se.generate_trapdoor(keyword)
    found_docs = se.search(index, trapdoor)
    results[keyword] = found_docs
```

解析

搜索过程中，首先为每个关键词生成陷门。然后使用这些陷门在创建好的索引中查找，以确定哪些文档包含这些关键词。

3.1.2.4 解密

```python
for keyword, found_docs in results.items():
    print(f"Keyword '{keyword}': Found Documents IDs {found_docs}")
    for doc_id in found_docs:
        print(f"  Document {doc_id}: {se.decrypt(encrypted_docs[doc_id])}")
```

解析

最后，打印每个搜索关键词及其对应的找到的文档 ID，并解密显示这些文档的内容。这一步展示了系统的实用性，即用户能够在保持文档加密的同时检索关键词。这一步调用的是 SearchableEncryption 类的解密方法，将加密的文档内容解密为原始文本。

3.2 运行结果

运行 main.py，选定若干个**关键词**进行搜索，本次我选择如下几个词：

quick town enemy

然后运行代码：

```
Generate Docs: 1000
Keyword 'quick': Found Documents IDs set()
Keyword 'town': Found Documents IDs {897, 66, 964, 871, 840, 268, 175, 400, 502, 57, 62}
  Document 897: Different energy according. Realize product wall build star unit. Guy and at oil drop town science simple.
  Document 66: Third strong under surface so early agree. History draw clearly reality visit town let. Specific degree age general behavi
or onto.
  Document 964: Tough town professional would camera future side see. Just store test begin Mr if quickly. Hospital three even reflect pi
cture floor.
  Document 871: Want in find boy town per. Ask party decision high also son. Say security authority detail old audience.
  Document 840: Road study available citizen window since deep. Raise improve city town goal talk professional. Cultural husband after ge
neral authority.
  Document 268: Attack chance outside energy. Student husband inside good town keep. Less lawyer here without.
  Document 175: Record or town democratic bit small. Make growth physical spring this body most. Attack speech sign well art.
  Document 400: Across call line major avoid condition statement southern. By and water nor test town president. Around wife company some
times official play degree clear.
  Document 502: Would evidence stage change gun discover see. Window blue military measure. Guy town never party including way.
  Document 57: Candidate challenge shake economic need. Consider onto business brother peace some. Nor town money fine.
  Document 62: Care about town defense change summer pick receive. Together guy door partner. What ok floor travel mind.
Keyword 'enemy': Found Documents IDs set()
```

图 3.2.5: 运行结果

代码生成了 1000 个随机句子，对于 quick 并没有搜到结果，对于 town 搜到若干个结果，并列出了相应的句子，对于 enemy 也没有搜到结果，这样就完成了**基于倒排索引的可搜索加密**。

4 实验心得

在完成这个对称可搜索加密方案的实验中，我深刻体会到了隐私保护技 术的重要性和实用价值。通过使用陷门来保护查询的关键词，我们能够有效地保护 云服务中的信息隐私，防止云隐私的泄漏。同时倒排索引技术使得查询变得十分高效，只需一次索引即可找到相应信息。

对称加密算法的选择，我本次使用了 AES，使我对数据科学和隐私保护领域有了更深的理解和认识。这次实验不仅增强了我的技术能力，也提升了我在处理真实世界数据问题时的敏感性和 责任感。