1. 可行性分析

最终目标是实现一个具有Web前端以及本地客户端的同步系统，借助Python、FUSE工具对本地文件操作，并应用HASH码实现数字签名，AES-CBC算法在本地加密，确保上传的文件不会被损坏或篡改。最后，在存储时，分块存储，将一个文件分割后存储到不同的服务器，一方面，保证文件损坏后的备份安全，另一方面，防止运营商对私人数据的泄露。

1. 理论依据
2. SHA-1码

在1993年，安全散列算法（SHA）由美国国家标准和技术协会(NIST)提出，并作为联邦信息处理标准（FIPS PUB 180）公布；1995年又发布了一个修订版FIPS PUB 180-1，通常称之为SHA-1。SHA-1是基于MD4算法的，并且它的设计在很大程度上是模仿MD4的。现在已成为公认的最安全的散列算法之一，并被广泛使用。

该算法的思想是接收一段明文，然后以一种不可逆的映射将它转换成一段（通常更小）密文，也可以简单的理解为取一串输入码（称为预映射或信息），并把它们转化为长度较短、位数固定的输出序列即散列值（也称为信息摘要或信息认证代码）的过程。

单向散列函数的安全性在于其产生散列值的操作过程具有较强的单向性。如果在输入序列中嵌入密码，那么任何人在不知道密码的情况下都不能产生正确的散列值，从而保证了其安全性。ＳＨＡ将输入流按照每块５１２位（６４个字节）进行分块，并产生２０个字节的被称为信息认证代码或信息摘要的输出。SHA-1是不可逆的、防冲突，并具有良好的雪崩效应。1bit数据的改变都会造成输出大大不同。

通过散列算法可实现数字签名实现，数字签名的原理是将要传送的明文通过一种函数运算（Hash）转换成报文摘要（不同的明文对应不同的报文摘要），报文摘要加密后与明文一起传送给接受方，接受方将接受的明文产生新的报文摘要与发送方的发来报文摘要解密比较，比较结果一致表示明文未被改动，如果不一致表示明文已被篡改。

SHA1产生相同消息摘要的可能性仅为1/(1\*10^48)，可以忽略这种可能性。

1. AES

在之前的相关工作中，已经提及AES加密算法的理论根据。此处略。

3. PyFilesystem

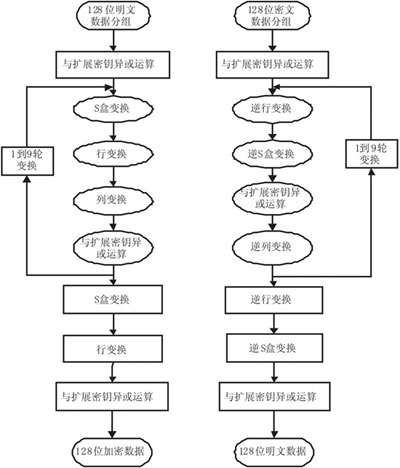
1. 技术依据
2. AES

在之前的相关工作中，已经提及AES加密算法，下面对其算法细节进行进一步阐述：

AES算法（即 Rijndael 算法）是一个对称分组密码算法。数据分组长度必须是 128 bits，使用的密钥长度为 128，192 或 256 bits。对于三种不同密钥长度的 AES 算法，分别称为“AES-128”、“AES-192”、“AES-256”。

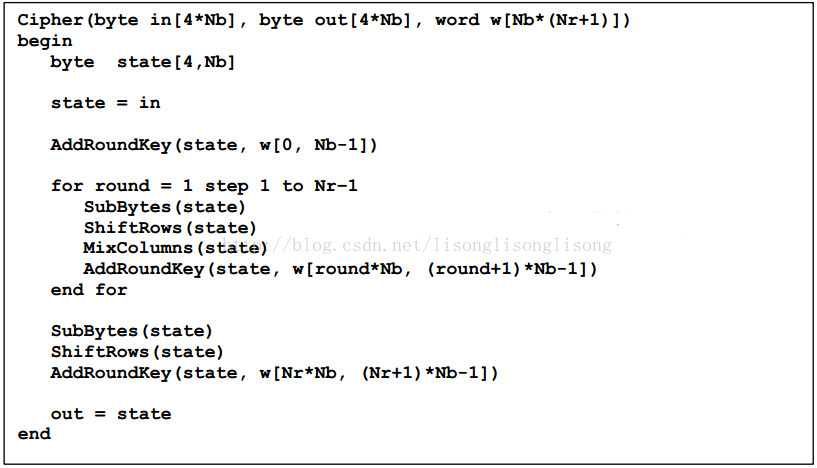
以AES-128为例，加密、解密流程图如下（事实上，迭代轮数与秘钥长度有关）：

IMG_256IMG_256



·加密

伪代码实现如图：



IMG_256从伪代码描述中可以看出，AES 加密时涉及到的子模块有SubBytes()、ShiftRows()、MixColumns()和AddRoundKey()。这些子模块介绍如下：

① S盒变换-SubBytes()

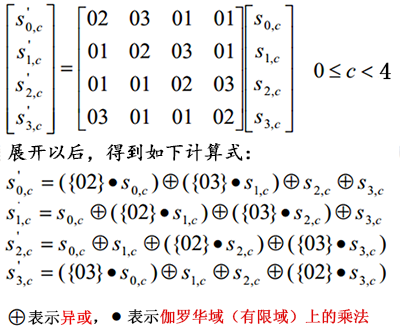
S盒是一个 16 行 16 列的表，表中每个元素都是一个字节。函数SubBytes()接受一个 4x4 的字节矩阵作为输入，对其中的每个字节，前四位组成十六进制数 x 作为行号，后四位组成的十六进制数 y 作为列号，查找S盒中对应的值替换原来位置上的字节。

② 行变换-ShiftRows()

行变换是将矩阵的每一行以字节为单位循环移位：第一行不变，第二行左移一位，第三行左移两位，第四行左移三位。右移的位数也与秘钥长度有关。

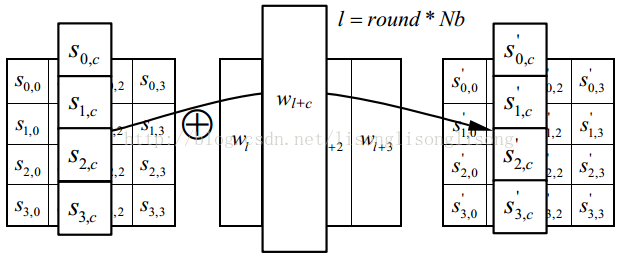
③ 列变换-MixColumns()

函数MixColumns()同样接受一个 4x4 的字节矩阵作为输入，并对矩阵进行逐列变换，变换方式如下：



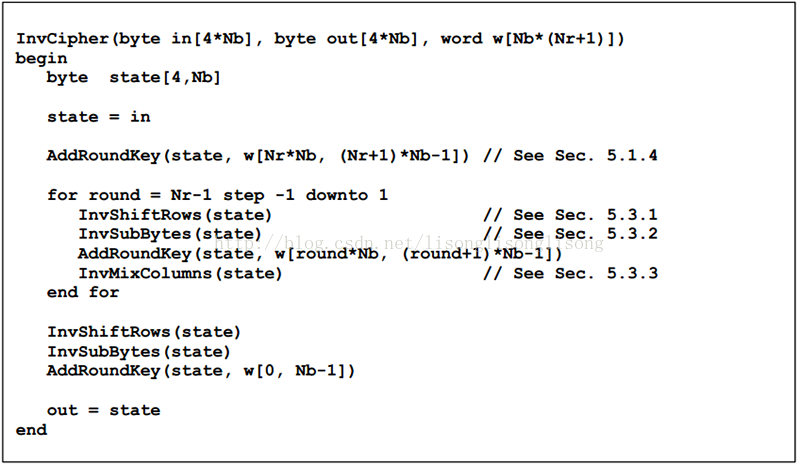
④ 与扩展密钥的异或-AddRoundKey()

扩展密钥只参与了这一步。根据当前加密的轮数，用w[]中的 4 个扩展密钥与矩阵的 4 个列进行按位异或。如下图：



·解密

根据 AES 解密的整体流程图（本文开头），伪代码如下：



从伪代码可以看出，我们需要分别实现 S 盒变换、行变换和列变换的逆变换InvShiftRows()、InvSubBytes()和InvMixColumns()。

① 逆行变换-InvShiftRows()

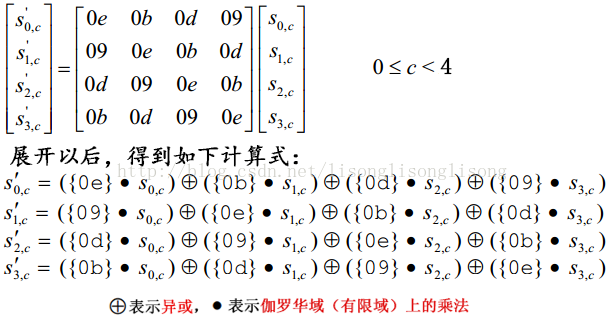
加密时ShiftRows()是对矩阵的每一行进行循环左移，所以InvShiftRows()是对矩阵每一行进行循环右移。

② 逆 S 盒变换-InvSubBytes()

与 S 盒变换一样，也是查表，查表的方式也一样，只不过查的是另外一个置换表（S-Box的逆表）。

③ 逆列变换-InvMixColumns()

与列变换的方式一样，只不过计算公式的系数矩阵发生了变化。如下图：



由此，可以完整的实现AES算法。

在密码分组链接模式（Cipher Block Chaining (CBC)）下，先将明文切分成若干小段，然后每一小段与初始块或者上一段的密文段进行异或运算后，再与密钥进行加密。

1. PyFilesystem

PyFilesystem是文件系统的抽象层。实际上，任何包含文件和目录的东西（硬盘，压缩文件，FTP服务器等等）都可以封装成一个共同的接口。使用这个模块，可以不需要知道文件确切的物理位置。FS对象提供的抽象意味着可以编写与文件物理位置无关的代码。例如，编写了一个在目录中搜索重复文件的函数，则它将在硬盘驱动器上的目录，或zip文件，FTP服务器上，Amazon S3等上进行更改.

以下是可以使用Pyfilesystem访问的一些文件系统：

DavFS访问WebDAV服务器上的文件和目录

FTPFS访问FTP服务器上的文件和目录

MemoryFS访问文件和存储在内存中的目录（非永久但非常快）

MountFS创建一个从其他文件系统构建的虚拟目录结构

MultiFS是将文件系统列表组合到一个的虚拟文件系统，并在打开文件时按顺序进行检查

OSFS是本地文件系统

存储在Secure FTP服务器上的SFTPFS访问文件和路由

S3FS访问存储在Amazon S3存储上的文件和目录

TahoeLAFS访问存储在Tahoe分布式文件系统上的文件和目录

ZipFS访问文件和包含在zip文件中的目录

使用PyFilesystem要比操作底层的接口更加简单。只要选择的文件系统（或任何类似于文件系统的数据存储）中存在FS对象，就可以使用相同的API。这意味着可以推迟将数据存储在以后的决定。如果决定在云中存储配置，则可能是单行更改，而不是主要重构。

PyFiles系统也可用于单元测试; 通过交换内存中文件系统和操作系统文件系统，可以编写测试，而无需管理（或模拟）文件IO。代码可以在Linux，MacOS和Windows上工作。

3.已有的云运营商提供API接口

Google Drive提供其API接口，对其用户提供一系列语言中的库以供调用。其中包括Google APIs Client Library for Python。使用 .json文件存储用户ID以及用户密码以及其他OAuth 2.0参数。

API又分为：

1. 简单API访问

不获取用户私人数据，调用的程序只需要证明其为Google API项目即可。

1. 授权API访问

获取用户私人数据。因此，在调用该API之前必须获得用户授权。且只能进行允许范围内的一些操作，并在用户许可下进行。用户许可后，会提供给应用访问码以及更新码。访问码可以授权API调用，但具有时限。访问码一旦过期，可以用更新码来获取新的访问码。

获得相应权限后，通过创建对象并发起请求，即可对云上文件进行操作。

1. 创新点

通常的分布式存储加密都是在服务端进行，不适用于私人用户。而采取在客户端的加密以及签名，实现方式简洁，且易于用户管理。（啊我编不下去了）