

视频大文件验证 实验报告

学号：SA20225172 姓名：郭俊勇

实验目的

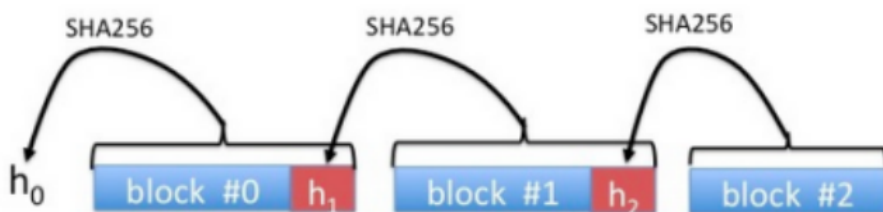
- 掌握数据完整性、哈希函数、MAC等概念
- 了解如何使用MACs来确保数据的完整性
- 通过编程，利用SHA256对视频大文件进行认证，学习如何使用哈希链来的实现大文件的分块认证。

编程语言

- Python

实验内容

- 假设某网站托管着一个任何人都可以下载的视频大文件F。下载文件的浏览器需要确保文件是真实的，然后才能向用户显示视频内容。一种可行的方法是让网站使用抗碰撞散列函数来散列F的内容，然后通过一些可信信道将得到的散列值 分发给用户（稍后我们将使用数字签名）。浏览器下载整个文件F，检查 $H(F)$ 是否等于可信的哈希值 h ；假如相等，浏览器便将视频显示给用户。
- 然而，这种方法意味着只有在下载好完整的视频之后才能开始播放视频内容。我们本次实验的目标是构建一个文件认证系统，使得浏览器在下载时可以对视频块进行身份验证和播放，而无需等待整个文件的下载。
- 网站不计算整个文件的散列值，而是将文件分成1KB块（1024字节）。它首先计算最后一个块的哈希值，并将值附加到倒数第二个块末尾。然后，它计算扩充后的倒数第二个块的哈希值，并将结果哈希值追加到第三个块的末尾。以此类推，直到处理完所有的块，如下图所示：

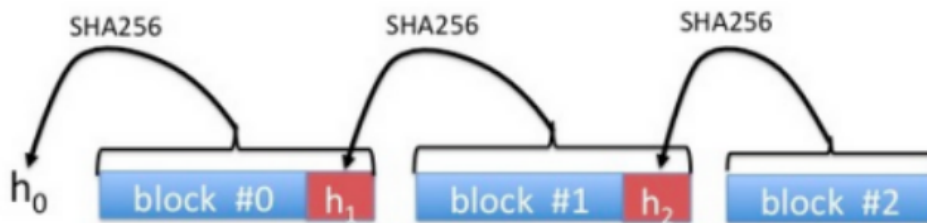


- 最终的哈希值（带有扩充哈希值的第一个块的哈希值）被通过可信信道分发给用户
- 现在，浏览器以每次一个块的方式下载文件F，其中每个块包含上图中的附加哈希值。当接收到第一个块后，浏览器检查是否等于；假如相等，浏览器就开始播放第一个视频块。当接受到第二个块后，浏览器检查是否等于；假如相等，浏览器就开始播放第二个视频块。此过程一直持续到最后一个块。这样，每个块都会在接收时进行认证和播放，无需等到整个文件下载完毕。
- 显然，如果哈希函数H时抗碰撞的，则攻击者无法在不被浏览器检测到的情况下修改任何视频块。事实上，由于，攻击者无法找到一对使得，因为这是违反哈希抗碰撞的。因此，在第一次哈希核验后，浏览器确信和都是可信的。相同的证明方法可以表明，浏览器确信和都是可信的，并以此类推剩余的所以块。

实验原理分析

MAC：消息验证码

根据原理图：

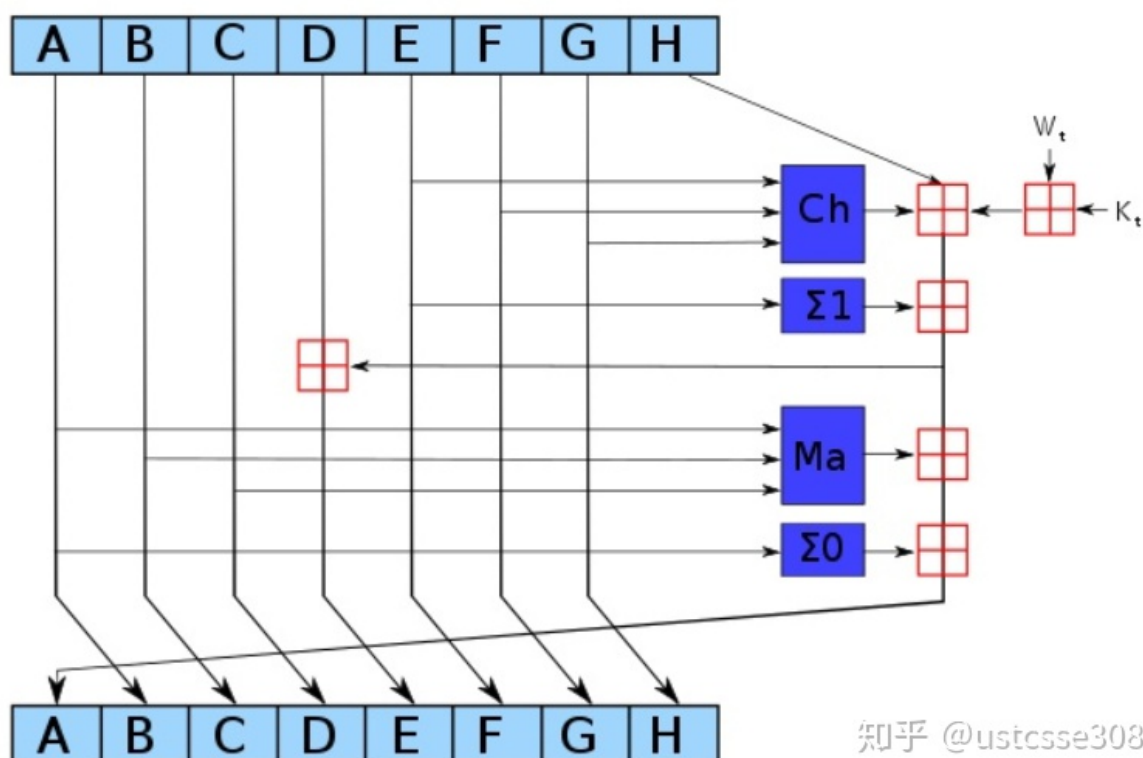


首先我们将视频文件的2进制流进行分组（1024）然后从后往前计算。第一步计算最后一个块的哈希值，扩展block的内容=block内容+后一个块整体的哈希值。逐步向前迭代，直到计算出第一个扩展块SHA256后的数值h0。

正向验证过程：

检测block0+h1是否和h0相等，然后以此往后面检测，知道最后一块。

SHA256原理：



知乎 @ustcsse308

$$\begin{aligned} \text{Ch}(E, F, G) &= (E \wedge F) \oplus (\neg E \wedge G) \\ \text{Ma}(A, B, C) &= (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \\ \Sigma_0(A) &= (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22) \\ \Sigma_1(E) &= (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25) \end{aligned}$$

ABCDEFGH 为前8个素数取平方根，前32位小数。

Wt是输入

Kt是自然数中前64个质数取立方根，前32位小数。

Python代码实现（源码）

```
from Crypto.Hash import SHA256
from binascii import b2a_hex

arr = []
```

```

blocks = []

def getFile(file, sizeNum):
    # 打开文件，以二进制数据读取
    f = open(file, 'rb')
    data = f.read()
    f.close()
    # 分块，1KB大小
    bytes = len(data)
    blocks = []
    for i in range(0, bytes, sizeNum):
        blocks.append(data[i:i + sizeNum])

    return blocks

def getHash(file):
    sizeNum = 1024
    res_hash = b''
    # 获取块内容
    blocks = getFile(file, sizeNum)

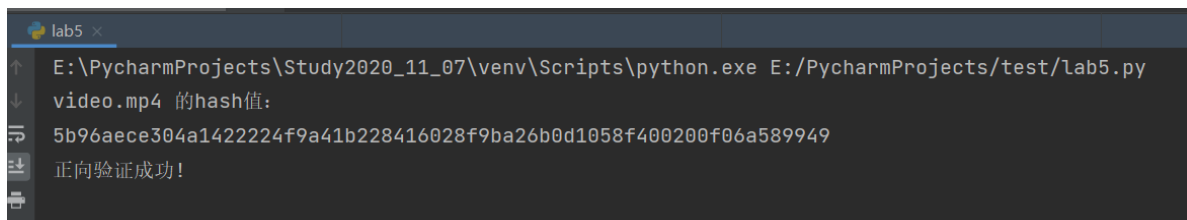
    for i in range(len(blocks) - 1, -1, -1):
        # 块内容=块内容+后一块的hash值
        blocks[i] = blocks[i] + res_hash
        # 得到新hash
        hash = SHA256.new(blocks[i])
        # 更新hash值
        res_hash = hash.digest()
        arr.append(res_hash)
    return b2a_hex(res_hash)

# 正向验证
def isValidate(file, arr):
    sizeNum = 1024
    # 获取块内容
    blocks = getFile(file, sizeNum)
    for i in range(0, len(blocks) - 1, 1):
        hash = SHA256.new(blocks[i] + arr[len(arr) - i - 2])
        if hash.digest() != arr[len(arr) - i - 1]:
            return 0
    return 1

if __name__ == '__main__':
    test_file1 = "E:\\研一\\第二学期\\现代密码学\\lab\\lab5\\实验五\\实验五\\test.mp4"
    test_file2 = "E:\\研一\\第二学期\\现代密码学\\lab\\lab5\\实验五\\实验五\\video.mp4"
    # print("test.mp4 的hash值: ")
    # print(getHash(test_file1).decode("utf-8"))
    # print("-----")
    print("video.mp4 的hash值: ")
    print(getHash(test_file2).decode("utf-8"))
    if isValidate(test_file2, arr) == 1:
        print("正向验证成功!")

```

运行结果



```
lab5 x
E:\PycharmProjects\Study2020_11_07\venv\Scripts\python.exe E:/PycharmProjects/test/lab5.py
video.mp4 的hash值:
5b96aece304a142224f9a41b228416028f9ba26b0d1058f400200f06a589949
正向验证成功!
```