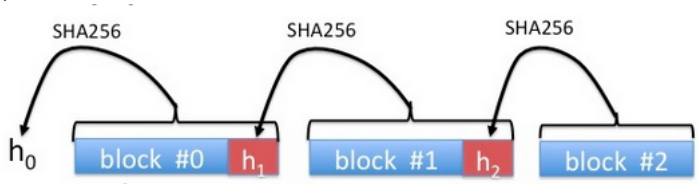
实验五　视频大文件验证

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$h = H(F)$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h_0（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$(B_0 \parallel h_1)$后，浏览器检查$H(B_0 \parallel h_1)$是否等于h_0；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$(B_1 \parallel h_2)$后，浏览器检查$H(B_1 \parallel h_2)$是否等于h_1；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$h_0 = H(B_0 \parallel h_1)$，攻击者无法找到一对$(B_0', h_1') \neq (B_0, h_1)$使得$h_0 = H(B_0' \parallel h_1')$，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B_0和h_1都是可信的。相同的证明方法可以表明，浏览器确信B_1和h_2都是可信的，并以此类推剩余的所以块。

实验任务

编写代码来计算文件*video.mp4*（见附件）的哈希值h_0，并完成对文件块的验证过程。

内容提示

- 使用**SHA256**作为哈希函数。对于SHA256的实现，使用现有的加密库，如**PyCrypto**[1]（Python），**Crypto++**[2]（C++）或任何其他语言和库。
- 将哈希值附加到每个块时，以二进制数据形式进行，即32个未编码的字节（256 位）。
- 如果文件大小不是1KB的倍数，则最后一个块将短于1KB，但所有其他块确保是1KB。
- 可以使用文件*test.mp4*（见附件）来检查自己的代码。该文件的h_0的十六进制编码为：**03c08f4ee0b576fe319338139c045c89c3e8e9409633bea29442e21425006ea8**

实验要求

- 请在线提交源码和实验报告。
- 实验报告需包括实验结果（需包含实验截图）、重要代码段解释以及本次实验总结。

[1]https://www.dlitz.net/software/pycrypto/
[2]http://www.cryptopp.com/