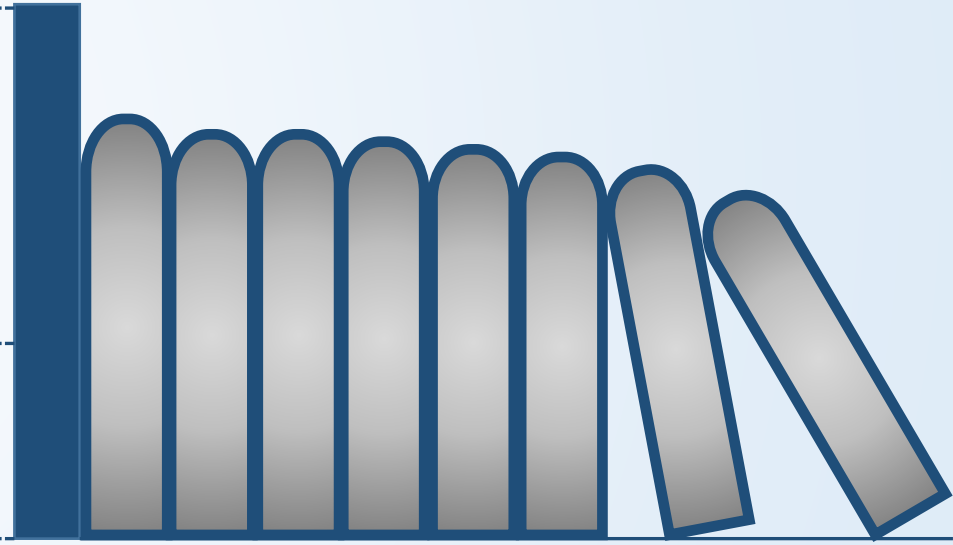
哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

Hash 长度扩展攻击



实验目的

- 掌握 Hash 函数的计算原理和算法特点
- 理解 Hash 长度扩展攻击的工作原理
- 了解抵御 Hash 长度攻击的方法

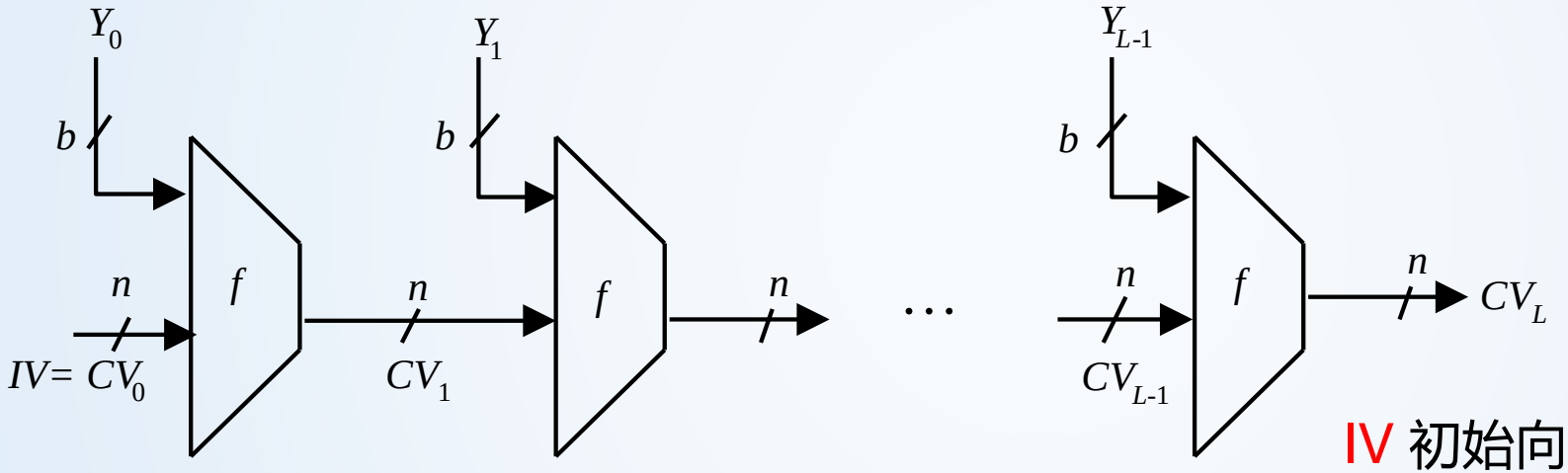
实验原理 ---基于哈希函数的MAC

将密钥和消息一起作为哈希函数的输入求哈希值，本次实验是针对秘密前缀Hash算法产生的MAC进行攻击。

- $T = \mathrm{MAC}_k(M) = h(k||M)$,称为秘密前缀(secret prefix) MAC
- 设消息M为一个分组序列(sequence of blocks)，即$M = (x_1, \cdots, x_n)$
- 则$T = \mathrm{MAC}_k(M) = h(k||x_1, \cdots, x_n)$
- 此种构造方式的问题在于，一致消息M的MAC值和k的长度，消息$M' = (x_1, \cdots, x_n, x_{n+1})$（其中$x_{n+1}$为任意的敌手设定的分组）的MAC值可以直接由$T$计算得出而不需要知道密钥$k$的值。

实验原理—哈希函数的特点

迭代型哈希函数的一般结构-MD结构，这种构造方式是每个消息块都会和一个输入向量做运算，第一个输入向量是初始化的，后面每个输入向量都是前面一个消息块的输出结果。



$$CV_0 = IV = n$$

$$CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \le i \le L$$

$$H(M) = CV_L$$

IV 初始向量

CV 链接值

Yi 第i个输入分组

f 压缩函数

n Hash值长度

b 输入分组长度

实验原理—SHA256算法过程

- **预处理**
  - 对消息填充
  - 初始化缓冲区，缓冲区为8个寄存器共256位
- **压缩过程，循环处理L个消息分组，每个分组处理分为如下两步**
  - 首先将消息Mi分为16个32位的字，并将其扩展为64个32位的字W_0 - W_{63}；
  - 然后将上一轮的输出H^{i-1}与产生行64轮次的循环，最后输出256位H^i为本次迭代的输出。
- **输出结果**
  - L个分组都被处理完后，最后一个H_{SHA256}的输出即为产生的消息摘要

实验原理

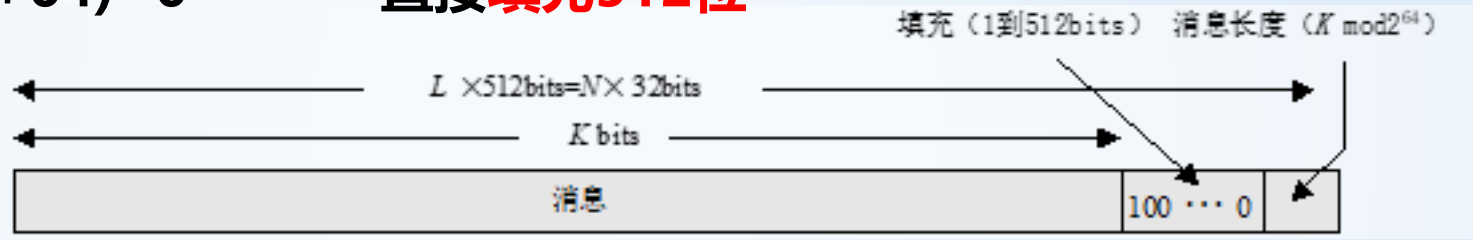
- **预处理—消息填充**

  因为SHA256是512bit一个分组，所以填充后消息长度必须为512的整数倍

  - 首先在末尾处附上64比特消息长度
  - 然后在消息原文后面填充，第一位为1，其余为0，至少需要填充1位，如果原始消息长度加上64bit消息长度刚好是512的整数倍，那么需要填充512位。
  - 填充后的消息必须恰好为512的整数倍

- **举例说明，计算需要填充的长度**

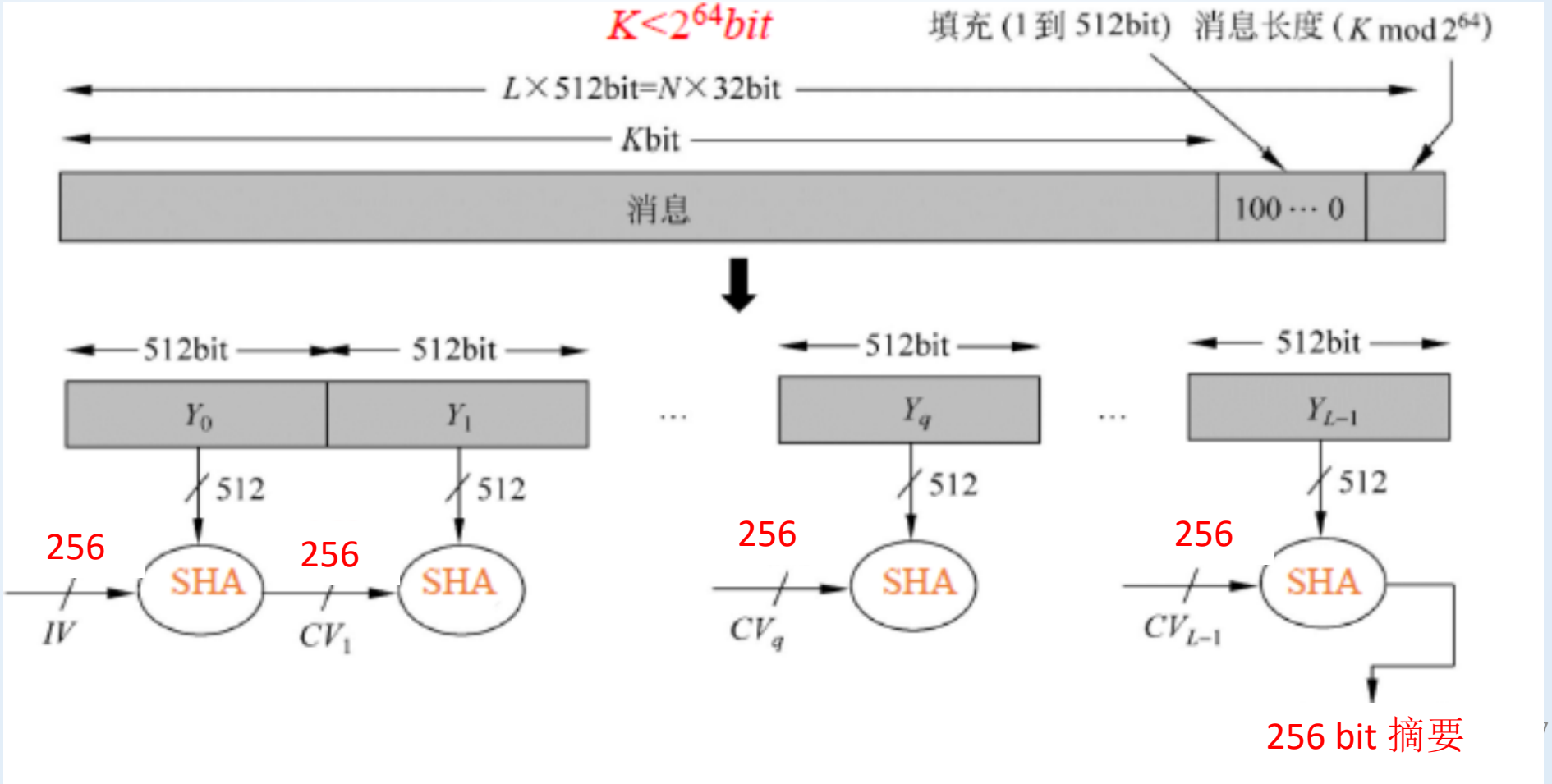
  - 700位 -> 1024-(700+64)=260 填充260位
  - 960位 -> 1024-(960+64)=0 直接填充512位



Tips：消息填充是必须的，在任何情况下都需要消息填充

实验原理—SHA256算法描述

- 算法输入：小于2^{64}bit长的任意消息，分为512bit长的分组，每次迭代处理512bit的消息分组
- 算法输出：256bit长的消息摘要
- 算法的框图如下所示：



实验原理 ---基于前缀哈希函数的MAC的攻击

Alice **敌手** **Bob**

$M = (x_1, \cdots, x_n)$

截获(intercept)

M, T ←

$M' = (x_1, \cdots, x_n, x_{n+1})$

$T' = h(T||x_{n+1})$

M', T' ←

- 计算$h(k||M') = h(k||x_1, \cdots, x_n, x_{n+1})$
- 验证$T = h(k||M')$通过

- 该攻击可能成功是因为哈希函数的迭代结构。注意哈希函数中的压缩函数的输入为当前的分组和处理完前面分组得出的结果。
- 因此，添加一个新分组x_{n+1}之后，新的消息哈希值可以由前面计算得到的哈希值T作为压缩函数的输入得到。

实验原理 ---基于前缀哈希函数的MAC的攻击

http://www.seedlab-hashlen.com/?myname=JohnDoe&uid=1001&lstcmd=1
&mac=dc8788905dbcbceffcdd5578887717c12691b3cf1dac6b2f2bcfabc14a6a7f11

```
def verify_mac(key, my_name, uid, cmd, download, mac):
    download_message = '' if not download else '&download=' + download
    message = ''
    if my_name:
        message = 'myname={}&'.format(my_name)
    message += 'uid={}&lstcmd='.format(uid) + cmd + download_message
    payload = key + ':' + message
    app.logger.debug('payload is [{}]'.format(payload))
    real_mac = hashlib.sha256(payload.encode('utf-8', 'surrogateescape')).hexdigest()
    app.logger.debug('real mac is [{}]'.format(real_mac))
    if mac == real_mac:
        return True
    return False
```

Key + message

生成摘要与传递来的mac比对

实验原理 ---基于前缀哈希函数的MAC的攻击

```
http://www.seedlab-hashlen.com/?myname=SEEDManual&uid=1001&lstcmd=1
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%50
&download=secret.txt
&mac=1dd23663f5cdfd371347601393e503066502f5497d17bb8a85b4e7d7a87a3450
```

1、已知key的长度为6

2、已知，正常访问时的mac，如下红色部分

http://www.seedlab-hashlen.com/?myname=SEEDManual&uid=1001&lstcmd=1
&mac=3912fe08949c7c09bd2825b01a2e8e9c151d84be0106e8584e9006b88a22555f

3、 填充消息

%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%50和一个和附加的消息&download=secret.txt

4、根据步骤2和3，可以根据已知的mac计算出填充和附加消息后新的mac

实验内容

本次实验来自https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Hash_Length_Ext/

共需要完成如下4个小任务。通过这4个任务完成对一个服务器网站的 Hash 扩展攻击，并对 Hash 扩展攻击进行防御验证。

- 任务1：发送请求来列出所有文件
- 任务2：创建Padding
- 任务3：长度扩展攻击
- 任务4：使用 HMAC 来抵御长度扩展攻击

实验要求

- 提交内容

① 实验结果截图

- 截止时间

一周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。

- 登录网址：： http://grader.tery.top:8000/#/login
- 推荐浏览器： Chrome
- 初始用户名、密码均为学号，登录后请修改

附录

Hashpump是hash长度扩展攻击的工具，下载方式如下，本次实验我们没有使用，在长度扩展攻击中，该工具是常用工具。

```
git clone https://github.com/bwall/HashPump
sudo apt-get install g++ libssl-dev
cd HashPump
make
sudo make install
```

关于Hash 长度扩展攻击的参考链接：
https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks

请同学们开始实验！