# SSL和TLS复习

2018.6.13 20：20开始23:00结束,MOD3 Secure Socket Layer (SSL) and TLS 复习。

## SSL是SECURE SOCKETS LAYER

- 是一个广泛部署的安全协议（被几乎所有的浏览器和网络服务器支持，https）
- 提供CIA，confidentiality，Integrity， Authentication

本意的目标：

- 电子商务的交易
- 加密
- 网络服务器验证
- 客户端验证
- 减少在新商品在贸易中的不便

可供所有TCP应用使用：secure socket interface

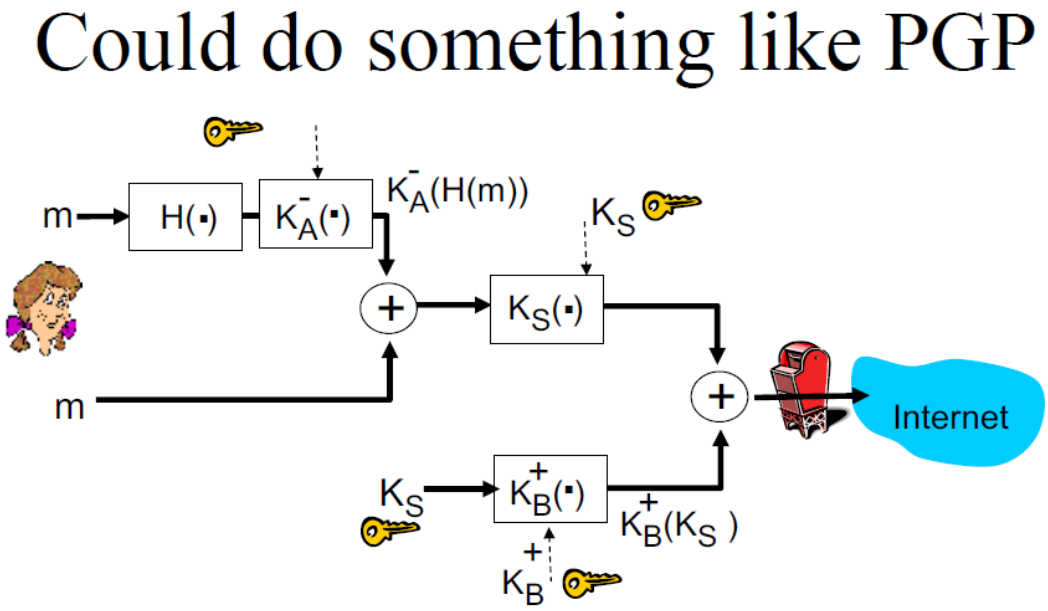
SSL在TCP/IP应用中的位置：

1. 网络层
   HTTP/SMTP/FTP
   SSL
   IPSEC/IP

2. 传输层
   HTTP/SMTP/FTP
   SSL OR TLS
   TCP
   IP

3. 应用层
   S/MIME PGP SET
   Kerberos SMTP HTTP
   UDP TCP
   IP

SSL/TLS可以做PGP一样的事情




- 但是需要发送字节流和交互数据
- 需要设置secret key
- 需要证书交换作为协议一部分： 握手阶段

## 案例分析：Toy SSL: a simple secure channel

握手： Alice和BOB使用他们的certificate和密钥来相互验证并且交换shared secret

key deriviation：alice和bob使用shared secret来生成set of keys

数据转移：被转移的数据被分成几个系列

data to be transferred is broken up into series of records

传输闭包connection closure：用来安全的关闭链接的特殊消息

握手：

ALICE --HELLO-> BOB
ALICE <- PUBLIC KEY CERTIFICATE - BOB 公钥证书
ALICE -KB+(MS) = EMS -> BOB 使用BOB的公钥来加密Master secret用来生成加密的master secret

生成key：

- 用多种key来作为MAC（消息验证编码）和加密
- four keys: 四种key， 数据和消息验证编码在客户端和服务器之间

```
- Kc = encryption key for data sent from client to server 加密key用来加密从客户端到
服务器的数据
- Mc = MAC key for data sent from client to server
- Ks = encryption key for data sent from server to client
- Ms = MAC key for data sent from server to client
```

- keys derived from key derivation function (KDF) 需要使用master secret并可能需要一些随机数据来产生key – takes master secret and (possibly) some additional random data and creates the keys

### data records数据记录：

因为需要check integrity所以数据必须要分段发送，不能直接发送加密的连续数据流（连续的数据流没地方放MAC)

- 每个记录都载有MAC
- 接收方可以单个处理
  对于每个record，需要区分数据的MAC,需要使用变长的记录 (LENGTH + DATA + MAC 这样的数据结构)

### sequence numbers序列号：

问题：攻击者可以捕获并重放记录
解决方案：把序列号放入MAC

- MAC = MAC(Mx, sequence||data)
- 需要注意的是这里并没有序列号的域

问题：攻击者还是可以重放攻击
解决方案：利用nounce

### control information控制消息：

问题：truncation attack 减少数据攻击
– attacker forges TCP connection close segment 攻击者重造TCP链接关闭区块
– one or both sides thinks there is less data than there actually is.双方或者单方认为数据量少了

解决方案：记录类型，用一种类型来表示关闭
– type 0 for data; type 1 for closure 0表示记录，1表示关闭
• MAC = MAC(Mx, sequence||type||data)
数据结构： LENGTH+TYPE+DATA+MAC

## SSL架构：

SSLHANDSHAKEPROTOCOL SSLCHANGECIPHERSPECPROTOCOL SSLALERTPROTOCOL HTTP
SSL RECORD PROTOCOL
TCP
IP

## 真实SSL握手：

目的:

- 服务器验证
- negotiation：加密算法
- 创建keys
- 客户端验证（optional）

1. 客户端发送一系列支持的算法和自己的nounce
2. 服务端选择算法，发送回：choice + certificate + server nonce
3. 客户端验证证书，提取服务器的公钥，生成PMS(PRE_MASTER_SECRET)，并通过客服务器的公钥加密并发送回服务器
4. 客户端和服务器通过PMS和nounces各自计算加密和MAC key
5. client sends a MAC of all the handshake messages客户端发送有所有握手信息的MAC
6. 服务端发送有所有握手信息的MAC

问题： 为什么最后两则消息交换MAC

- 客户端发送了一系列算法，有些强有些弱。MIMT攻击可以删掉强的算法。最后两条信息被加密则是用来防止MIMT.

问题： 为什么用两个随机的NOUNCES

- 假设C嗅探了A和B之间所有的包，C再向B建立TCP连接，发送所有记录中提取的序列号
- B（AMAZON）以为A分两次买了两个同样的东西
  解决方案：B为每次连接发送不同的随机nounce，C就在Integrity check中失败。

## SSL RECORD PROTOCOL：

---

data

---

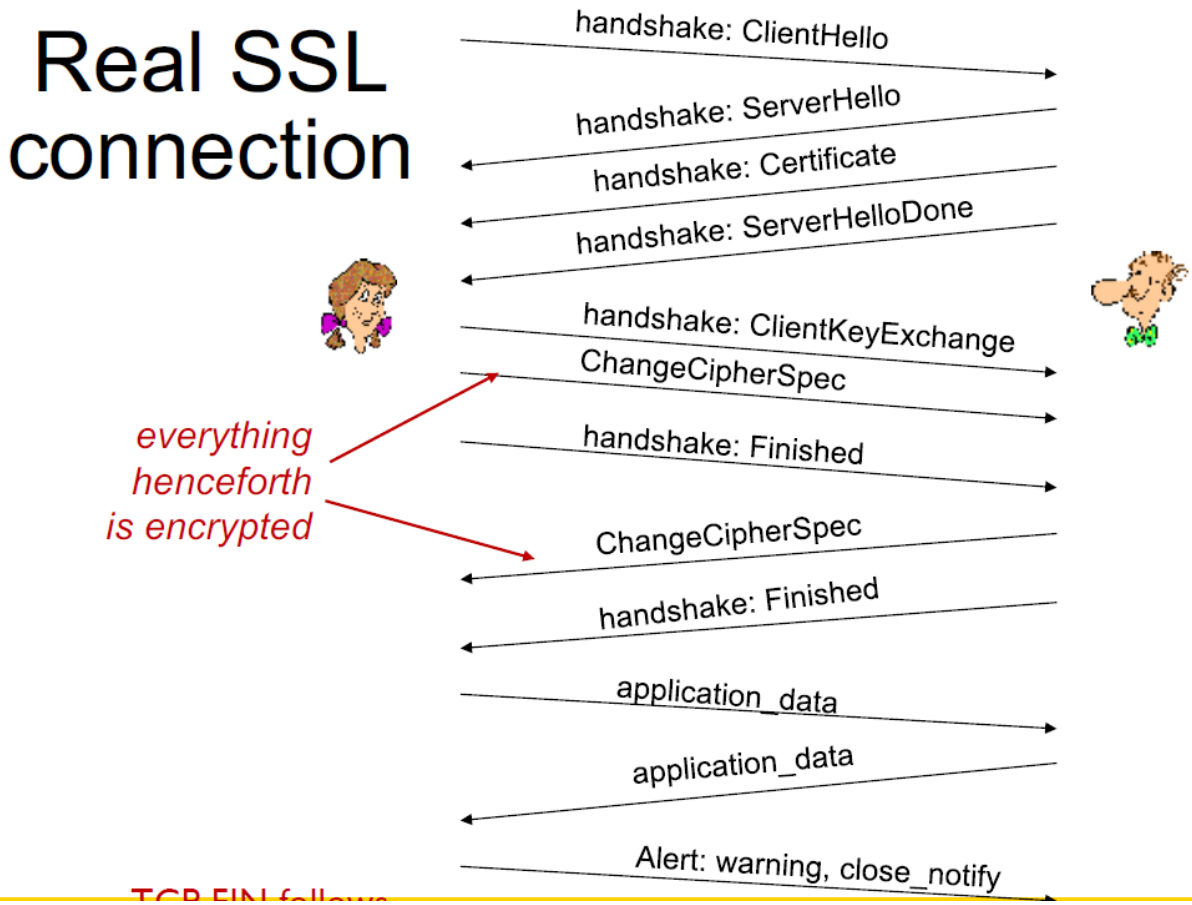
data fragment(each SSL fragment 214 bytes (~16 Kbytes)) + MAC(includes sequence number, MAC key Mx)

---

record header(content type; version; length) + encrypted data and MAC

---

SSL RECORD format SSL记录格式：
content type(1 byte)+SSL VERSION(2 bytes)+length(3 bytes)+ data + MAC
数据和mac都经过对称加密




Key derivation:

- 使用用户nounce，服务器nounce和pms一并输入生成器生成master secret
- ms和新的nounces输入另一个随机生成器（key block）
- key block sliced and diced:

```
- client MAC key
- server MAC key
- client encryption key
- server encryption key
- client initialization vector (IV)
- server initialization vector (IV)
```

## Transport Layer Security数据传输层安全

- The same record format as the SSL record format.数据格式和SSL数据格式相同

```
• Defined in RFC 2246.
• Similar to SSLv3.
• Differences in the:
- version number
- message authentication code
- pseudorandom function
- alert codes
- cipher suites
```

```
- client certificate types
- certificate_verify and finished message
- cryptographic computations
- padding
```