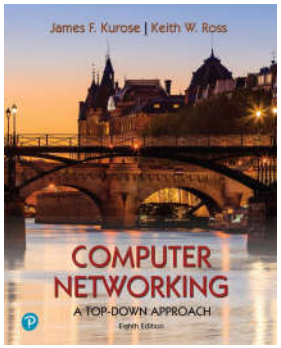
第 8 章
网络安全（中）

中国科学技术大学
自动化系 郑烇
改编自Jim kurose, Keith Ross



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

第八章 提纲

- 什么是网络安全?
- 加密原理
- 认证，报文完整性
- 安全电子邮件
- 使TCP连接安全：TLS
- 网络层安全性：IPSec
- 无线和移动网络的安全
- 实践中的网络安全：防火墙和IDS

Transport-layer security (TLS)

- 广泛部署的安全协议
  - 被几乎所有浏览器和Web服务器采用，如：https
  - 被QUIC采用（协作）：握手时的认证和秘钥集协商，之后的加密数据传输
  - https(Port 443)
- 能够提供
  - 机密性:采用对称加密
  - 完整性：通过加密的散列值
  - 认证：通过公开秘钥加密体系

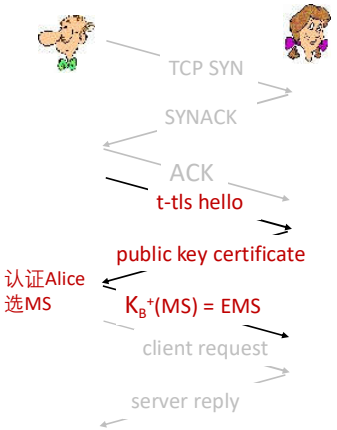
所有的技术我们都学过!

- 历史
  - 早期的研究和实现：安全网络编程，安全套接字接口
  - SSL3.0 RFC7568 (1996)-> TLS 1.0（1999）
  - TLS1.2 RFC 5246(2008)
  - TLS1.3 RFC 8846(2018)

Transport-layer security: 需求是什么?

- 渐进式地构建起一个玩具版的TLS协议： *t-tls*
- 安全通信的4个步骤:
  - *握手：* Alice和Bob使用各自证书，私钥来认证对方，交换或者创建共享密钥（对称密钥）
  - *秘钥导出：* Alice和Bob采用共享秘钥来生成一个密钥集
  - *数据传输：* 要传输的数据被分成若干记录的序列（数据块）
    - 不仅仅是一次性的事务性交互
  - *连接关闭：* 采用特殊的报文安全关闭连接

t-tls:初始化握手



t-tls 握手阶段:

- Bob 建立和Alice的TCP连接
- Bob 验证的Alice的身份
- Bob发送给Alice 主密钥key_A^+(MS),用来生成TLS会话的所有秘钥
- 可能存在的问题:
  - 在通信之前需要花费3 RTT (包括TCP连接建立)

t-tls:密钥生成

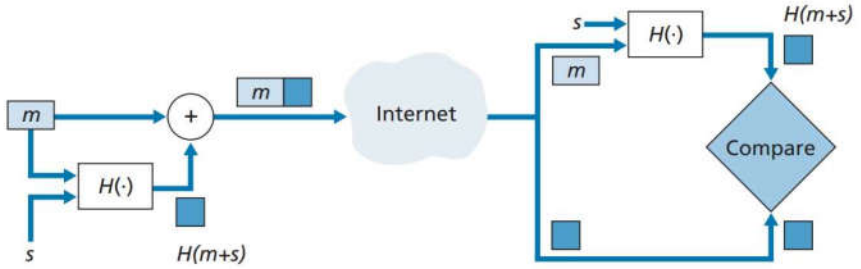
- 在多个加密操作中采用同一个密钥:不安全
  - 采用不同的密钥用于:报文认证(MAC)和加密
- 4 keys:
  - K_c = 客户端到服务器的加密密钥
  - M_c = 客户端到服务器的MAC密钥
  - K_s = 服务器到客户端的加密密钥
  - M_s = 服务器到客户端的MAC密钥
- 密钥由密钥key derivation function (KDF)导出函数导出
  - KDF创建密钥的输入:主密钥MS+(可能的)一些附加随机数据
    - 破除主密钥MS 和 4Keys的固定对应关系

t-tls:加密数据

- 回顾: TCP提供数据字节流传输的服务
- Q: t-tls可以对数据进行字节流流式加密?就像TCP Socket
  - A: 如是流式加密,完整性校验的MAC只能在最后,那么只有所有数据都被传输完毕才能够进行**完整性校验**!
  - 如:对于即时通信(持续时间很长),在终端显示这些字符之前,如何对 **所有字节** 进行完整性校验?
  - 方案:将字节流分割成一个个的记录
    - 每个记录C-S携带该记录的MAC,hash(采用M_C)
    - 接收端可以对每个到达的记录进行完整性校验
- t-tls记录采用对称秘钥K_c加密,交付给TCP传输: MAC=HASH(data+ M_C)

K_c(data | MAC)

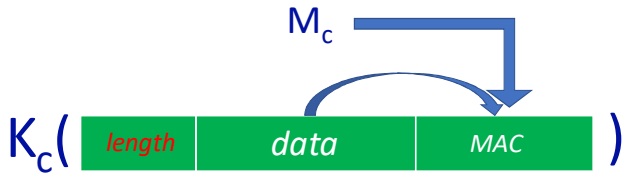
MAC:报文认证码



- 简单采用Hash作为MAC的问题:攻击方替换成:m',Hash(m')
- S(即:Mc)加入到hash的运算,H(m+Mc)
- 不需要加密,更不用采用代价高的私钥加密签署
  - m和H(m+s)反正要加密

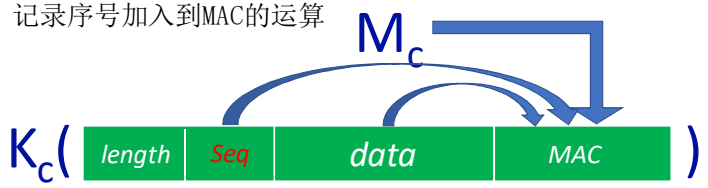
t-tls:加密数据

- Q：在记录中，接收端要区分数据与MAC
  - 采用可变长记录，用length指示数据长度
  - MAC=MAC(data，M_C)；Data和MAC加密传输
  - M_C加入MAC的运算，MAC不仅仅依赖于data，更安全
- t-tls记录采用对称秘钥加密，用秘钥K_c，交付给TCP传输：



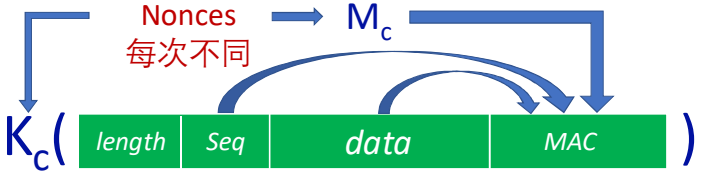
t-tls:加密数据（续1）

- 针对数据流的可能攻击?
  - *重排攻击：*中间攻击者可以捕获、重新排序攻击
  - Client->Server 2个TCP段
  - 中间人Trudy颠倒2个Tcp段的序号（TCP段没有加密，可以修改TCP头部序号）
  - Server TCP收到，交TLS，上交应用
  - 应用无法检查出颠倒的问题
- 方案:将记录的序号放到MAC中，MAC = MAC(M_c, sequence||data)
  - 发送者TLS使用记录序号0,1,2…（每发送一个+1）
  - 记录序号加入到MAC的运算



t-tls:加密数据（续2）

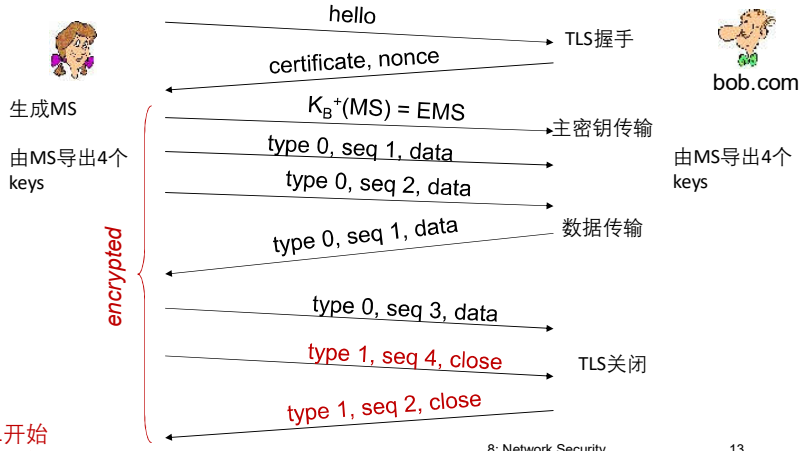
- 针对数据流的可能攻击?
  - *重放攻击：*攻击者重放所有的记录
- 方案：采用不重数nonce
  - 每次TLS握手，采用不同的nonces
  - 密钥4keys依赖于nonce，每次连接密钥不同
  - 重放攻击不起作用



t-tls: 连接关闭

- 截断攻击:
  - 攻击者伪造TCP连接的报文段；伪造TCP连接关闭
  - 连接过早地被关闭，一方或者双方只接收到了比实际少的数据
- 方案：*TLS*记录类型，一个类型用于关闭
  - TLS类型0数据；类型1用于关闭
  - TLS会话记录type=1传输之后，才是TCP连接关闭
  - 本质上：上层TLS协议中包括下层TCP协议的一些控制信息
- MAC采用数据，类型type，和序号 来计算
- MAC = MAC(M_C, sequence||type||data)

t-tls: 过程

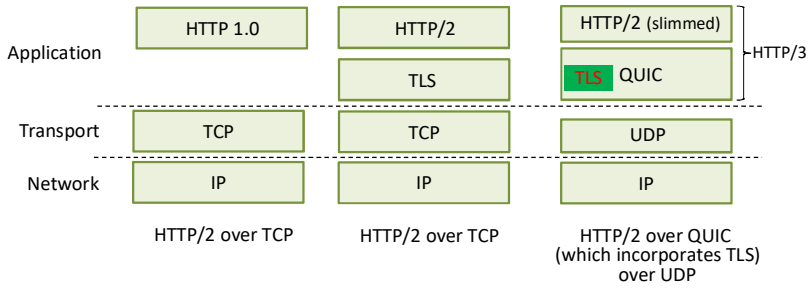


t-tls: 并不完整

- 字段多长?
- 那种加密算法?
- 希望提供协商机制
  - 允许客户端和服务器支持不同的加密套件
  - 在数据传输之前协商和选择特定的套件

Transport-layer security (TLS)

- TLS可为任何应用提供API
- HTTP视角的TLS



TLS借助QUIC交换报文
QUIC借助TLS协商套件和交换秘钥，认证，加密

TLS: 加密套件

- “加密套件”:秘钥生成算法，加密算法，MAC算法，数字签名算法
  - 并不捆绑具体算法
- TLS: 1.3 (2018): 提供比TLS 1.2（2008）支持更少的加密套件（防止降级攻击）
  - 只有5个选择，而不是37
  - Diffie—Hellman(DH)算法进行秘钥交换，而不是DH或RSA
  - 组合加密和认证算法（认证加密算法），而不是加密之后再认证
    - 4 based on AES
  - HMAC采用SHA(256或者284)加密散列函数
    - HMAC：用Hash做MAC（当然要加secret，M_C）

TLS 加密套件

密码套件的名称和其安全性属性的示例

密码套件名称	身份验证	密钥交换	密码	MAC	PRF
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	RSA	ECDHE	AES-128-GCM	–	SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDSA	ECDHE	AES-256-GCM	–	SHA384
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	RSA	DHE	3DES-EDE-CBC	SHA1	协议
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	RSA	AES-128-CBC	SHA1	协议
TLS_ECDHE_ECDSA_WITH_AES_128_CCM	ECDSA	ECDHE	AES-128-CCM	–	SHA256

https://blog.csdn.net/m0_37621078/article/details/106028622

SSL&TLS个版本支持的秘钥交换、协商和认证

Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
RSA	Yes	Yes	Yes	Yes	Yes	No	
DH-RSA	No	Yes	Yes	Yes	Yes	No	
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes	
ECDH-RSA	No	No	Yes	Yes	Yes	No	
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
DH-DSS	No	Yes	Yes	Yes	Yes	No	
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[48]	
ECDH-ECDSA	No	No	Yes	Yes	Yes	No	
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
PSK	No	No	Yes	Yes	Yes		Defined for TLS 1.2 in RFCs
PSK-RSA	No	No	Yes	Yes	Yes		
DHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes		
ECDHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes		
SRP	No	No	Yes	Yes	Yes		
SRP-DSS	No	No	Yes	Yes	Yes		
SRP-RSA	No	No	Yes	Yes	Yes		
Kerberos	No	No	Yes	Yes	Yes		
DH-ANON (insecure)	No	Yes	Yes	Yes	Yes		
ECDH-ANON (insecure)	No	No	Yes	Yes	Yes		

https://blog.csdn.net/m0_37621078/article/details/106028622

真实TLS：握手(1)

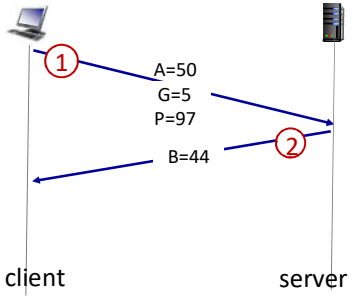
目的

1. 服务器认证（客户端认证服务器）
2. 协商：就加密算法和MAC算法协商一致
3. 建立秘钥
4. 客户端认证（可选项）

TLS 1.3:快速握手-Diffie-Hellman算法

客户端传到服务器
G（P的原根，如5）和P(素数,如97)

随机生成a(36)
A=G**a%P
=5^36%97=50

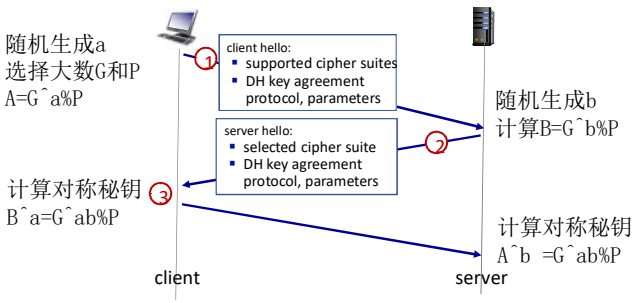


1)随机生成b=58
B=G**b%P
=5^58%97=44

计算对称秘钥
key=B^a%P
=44^36%97=75

2)key=A^b%P
=50^58%97=75

TLS 1.3 handshake: 1 RTT



G是P的原根

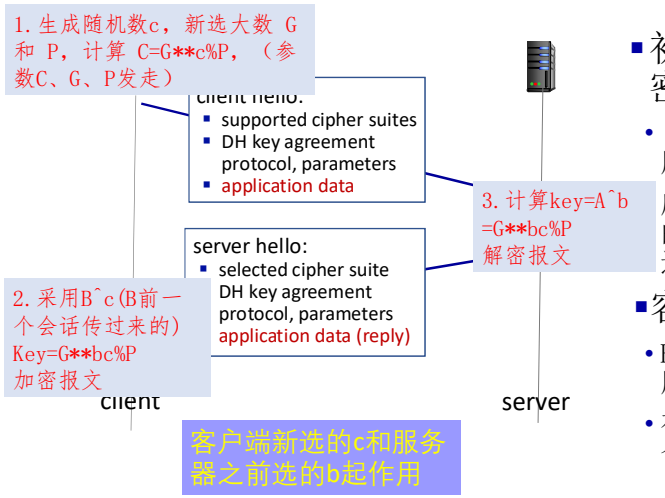
TLS1.3只需1RTT就可以传输数据，而不是2RTT

① 客户段TLS hello 报文:
- 猜测秘钥交换协议，参数大数G、P和A=G**a%P
- 支持加密套件支持列表

② 服务器TLS hello 报文选择
- 秘钥协商协议，参数 B=G**b%P
- 加密套件
- 签名了的证书(服务器端)

③ 客户端:
- 校验服务器的证书，认证对方身份
- 生成秘钥
- 可以发送应用报文了(e.g., HTTPS GET)

TLS 1.3 握手: 0 RTT



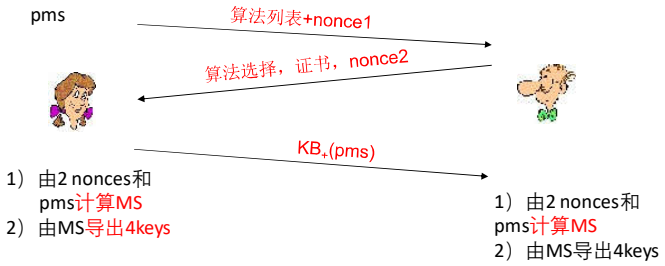
客户端新选的c和服务器之前选的b起作用

- 初始化hello报文包含加密应用数据!
  - “恢复resuming”之前客户端和服务器之间的连接
  - 应用数据采用之前的连接所使用的“resumption master secret”进行加密
- 容易受到重放攻击比较!
  - HTTP GET或者客户端请求不改变服务器的状态这种报文没问题的
  - 在QUIC协议中通过超时机制，几个小时过期

真实的TLS：握手(2)

1. 客户端发送过一个它自己支持的算法列表，同时发送的还有不重数nonce1
2. 服务器从列表中选择算法：发送回来它的算法选择+服务器证书+服务器不重数nonce2
3. 客户端验证证书，提取出服务器的公钥，生成pre_master_secret（采用服务器**公钥**加密的），发送到服务器
4. 客户端和服务器从pre_master_secret 和2个nonces独立地计算出**主密钥MS**
   1. 由MS切片得到2个数据传输密钥和2个MAC密钥
5. 客户端发送一个整个握手阶段报文的MAC
6. 服务器发送一个整个握手阶段报文的MAC

真实的TLS：握手(2)



密钥导出

- 客户端不重数、服务器不重数和pre-master secret 输入到为随机生成器
  - 产生出主密钥
- 主密钥pre-master secret和nonces 输入到另外一个随机数生成器：“key block”
- key 块被切分，再切分
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key

  如果采用：3DES或者AES，还生成
  - client initialization vector (IV)
  - server initialization vector (IV)

真实的TLS：握手(3)

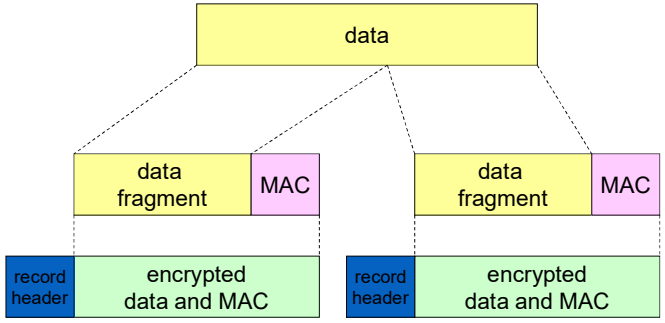
最后两步：保护握手阶段防止篡改

- 客户端通常提供一系列算法：有的很强，有的弱
- 中间人可能会从列表中删除比较强的加密算法（算法列表是明文传输的）
- 防止这种事情的最后2步骤（步骤5,6）
  - 客户端将握手阶段的所有报文的MAC发送给服务器，以便于服务器检查有无篡改
  - 同样，服务器将所有报文的MAC发送给客户端，以便于客户端检查有无篡改
  - 最后2个报文都是加密的

真实的TLS：握手(3)

- 为什么2个随机的不重数（客户端和服务器）？
  - 防止重放攻击
- 假设Trudy嗅探到Alice和Bob的所有报文
- 第二天，Trudy和Bob建立TCP连接，发送同样的记录序列（重放攻击：伪造Alice对Bob进行重放）
  - 如果秘钥只依赖于Alice一侧的nonce，2次会话生成的密钥都一样
  - Bob (Amazon) 可能会认为Alice提交了关于一个物品的2个订单
  - 解决：Bob在每个连接上发送不同的随机不重数，这导致2次的加密密钥不一样
  - Trudy的重放攻击报文无法通过完整性校验

TLS 记录格式

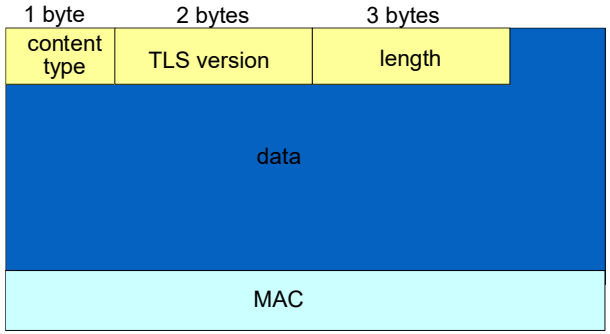


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

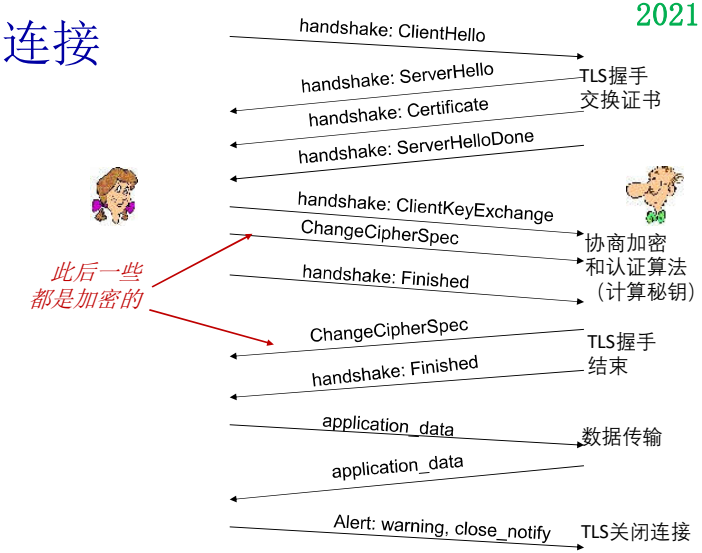
fragment: each TLS fragment 2^{14} bytes (~16 Kbytes)

TLS 记录格式



数据和MAC都是被加密的(对称算法)

真实TLS连接



第八章 提纲

- 什么是网络安全?
- 加密原理
- 认证，报文完整性
- 安全电子邮件
- 使TCP连接安全：TLS
- 网络层安全性：IPSec
- 无线和移动网络的安全
- 实践中的网络安全：防火墙和IDS

IPSec协议：位置

- *在2个网络实体（主机 or 路由器）之间提供安全性*
- IP之上、传输层协议之下
- 发送实体加密数据报载荷，载荷可以是：
  - TCP段、UDP数据报、ICMP报文、 OSPF报文等
- 所有从发送端实体到其他接收方的数据都可以被隐藏：
  - web pages, e-mail, P2P文件传输, TCP SYN 分组 …
- “地毯覆盖”
  - IP的载荷（如TCP段）是经过加密，认证，完整性检查的

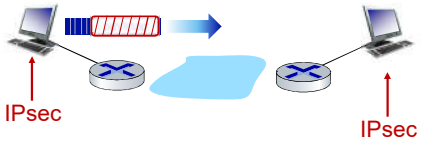
IP Sec：特性和组成

- 安全性：机密性，完整性，可认证性，重放攻击保护
- 2个IPSec子协议

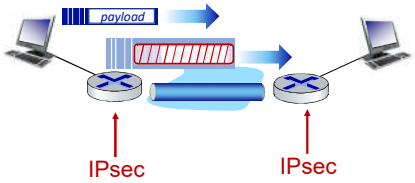
Authentication Header (AH) 协议 [RFC 4302]
提供源端的可认证性和数据完整性，但是不提供机密性

Encapsulation Security Protocol (ESP) [RFC 4303]
提供源端可认证性，数据完整性和机密性
比AH用的广泛

IP Sec的2个模式



传输模式:
- 主机之间
- 只有数据报载荷被加密和认证



隧道模式:
- 路由器之间
- 整个数据报都被加密和认证
- 加密的数据报被封装在一个新的数据报中，新的IP头部
- 2个路由器之间就像有个隧道一样，将2个网络连接起来

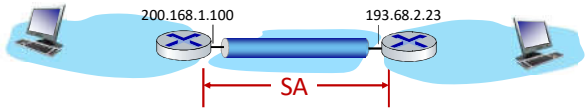
IP Sec：模式和协议的组合

Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

最常用，最重要

安全关联Security associations (SAs)

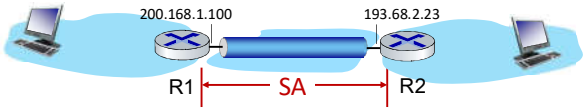
- 在发送数据之前，要在发送实体和接收实体之间建立“安全关联(SA)”
  - SA复杂：一个方向一个安全关联
  - 单向安全数据通信的关系
- 发送端、接收端实体维护SA的状态信息
  - 维护通信的状态
  - 回顾:TCP端结点也维护了状态信息
  - IP是无连接的；IP-sec是面向连接的（不是有连接的）！
- VPN一共有多少个SA，一个总部，一个分支和n个旅行中的销售?
  - 2+2n



安全关联（SAs）

R1为SA存储：

- 32-bit identifier: *Security Parameter Index (SPI)*
- 源SA接口 (200.168.1.100)
- 目标 SA 接口 (193.68.2.23)
- 采用的加密类型(e.g., 3DES with CBC)
- 加密的秘钥
- 完整性校验类型(e.g., HMAC with MD5)
- 认证的key



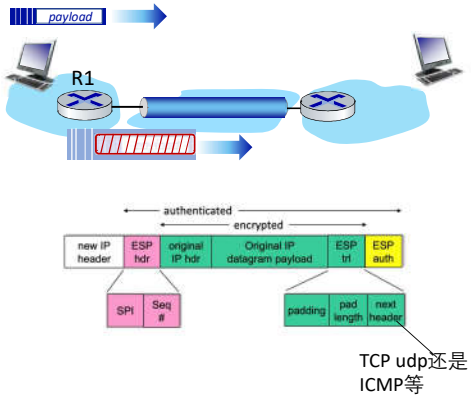
Security Association Database (SAD)

- 端结点（路由器或主机）将SA的状态维护在*security association database (SAD)*，可以使用它们在处理过程中访问到所用的信息
- 公司有n个销售，在R1的SAD中有2 + 2n个SA
- 当发送IPsec数据报时，R1访问SAD来决定如何处理该数据报
- 当IPSec数据报到达R2时，R2检查IPsec数据报的SPI，用该值检索SAD，对应地处理该数据报
  - 解密，完整性检查和可认证性检查

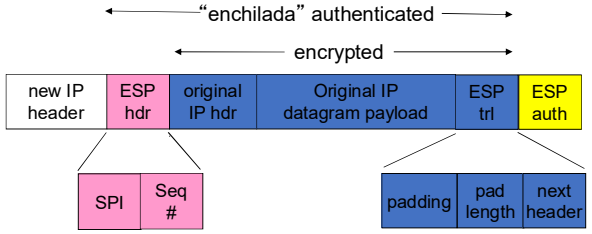
SAD: "how" to do it

ESP 隧道模式：R1动作

ESP、隧道模式

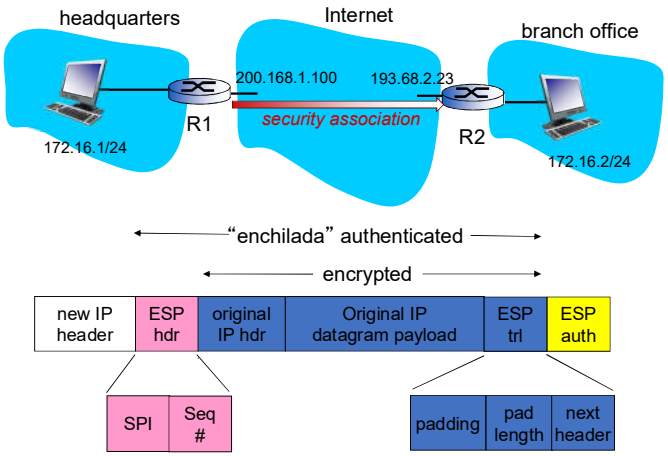


R1的动作:IP数据报封装在IPSec中



- ESP 尾部：块加密需要填充成块的整数倍
- ESP 头部：
  - SPI，让接收实体知道如何处理
  - 序号，防止重放攻击
- ESP中的MAC认证字段，采用共享秘钥创建

R1的动作:IP数据报封装在IPSec中



1. 在原始数据报（包括原始数据报的头部字段）的尾部附加“ESP尾部”字段
2. 采用SA指定的算法和key加密以上结果
3. 将被加密部分的前面，插入ESP头部，创建“enchilada”
4. 对全部的“enchilada” 创建鉴别MAC，采用SA中指定的算法和key，在“enchilada”后面附上MAC，形成载荷
5. 创建新的IP头部，包括典型IPv4的头部字段，插在载荷的前面
   - 协议字段：50，标识 ESP来处理
   - 源IP R1，目标IP R2

IPsec 序号

- 对于一个新的SA，发送端初始化该序号为0
- 每一次在SA上发送数据报
  - 发送方增加序号的值
  - 将该序号值放入seq #字段中
- 目标:
  - 阻止嗅探和重放攻击
  - 如果接收到重复的，经过认证的IP数据报可以中断服务
    - 攻击者只改序号是不行的，经过认证的，对应的MAC字段也需要修改
- 方法：
  - 目标端检查重复
  - 不维护所有接收到分组的状态，而是采用窗口范围内的检查

Security Policy Database (SPD)

- 策略：给定一个数据报，发送实体知道它是否该使用IPsec，还是普通IP数据报
- 需要知道采用哪个SA关联的数据
  - 可能会使用：源和目标的IP地址，协议号
- 在SPD中的信息指示：对于到来的数据报到底该做什么
- SAD中的信息指示：如何做
  - 按照具体参数对进出的数据报进行IPSec的处理

IPSec服务安全性分析

- 假设Trudy在R1和R2之间，她不知道keys.
  - Trudy能够看到数据报的原始内容吗？源IP、目标IP地址，传输层协议，应用端口号能够探知吗？
    - 原始数据报加密，IP地址，端口号信息不可探知
  - 可以更改其中的某些bits，而且不被检测到吗？
    - 序号在IPSec分组中，有完整性检查机制
  - 伪装成R1，采用R1的IP地址呢？
    - 也通不过完整性检查，没办法形成正确的MAC
  - 重放一个数据报呢？
    - 有序号，通不过完整性检查

IKE: Internet Key Exchange

- *前面的例子：*在IPsec端节点上手工设置IPsec SA:

  Example SA

  SPI: 12345
  Source IP: 200.168.1.100
  Dest IP: 193.68.2.23
  Protocol: ESP
  Encryption algorithm: 3DES-cbc
  HMAC algorithm: MD5
  Encryption key: 0x7aeaca…
  HMAC key:0xc0291f…

- 手工设置这些key对于100s端节点的VPN来说是不切实际
- *IPsec IKE (Internet Key Exchange)：自动建立SA*
  - 交换证书（认证），协商：鉴别和加密算法
  - 交换用于生成Key的信息，生成keys
  - 建立SA

IKE：阶段1

- 条件：如果双方没有建立起VPN的管理连接IKE SA
- 目的：建立管理连接IKE SA(双向，不同于SA)
  - 安全参数协商（加密算法，MAC算法）
  - 密钥生成（HF算法），双方共享IKE SA的密钥
  - 相互认证对方（对称方式，RSA非对称方式）
- 模式：主模式和侵略模式（贪婪模式）
  - 主模式提供身份保护，更加弹性化
  - 贪婪模式采用更少的报文，更快

IKE：阶段2

- 目的：在阶段1建立的IKE SA基础上，安全地建立2个数据连接SA(2个单向)
- 结果：
  - 协商加密算法，鉴别算法
  - 建立加密密钥，鉴别密钥
  - 周期性的对数据连接进行密钥更新

IKE: PSK and PKI

- 相互认证对方身份（证明你是谁）
  - pre-shared secret (PSK)，对称加密体系
  - pubic/private keys and certificates (PKI)，公开密钥加密体系
- PSK：双方都从一个secret开始
  - 运行IKE来认证对方，而且建立起 IPsec SAs（每个方向一个），包括加密和认证的keys
- PKI：双方开始于public/private key 对和证书
  - 运行IKE来相互认证对方，获得 IPsec SAs（每个方向一个）
  - 和SSL握手类似

IPsec 总结

- IKE 交换报文用于：协商算法，生成和交换keys，生成SPI，自动建立SA

- AH或者ESP协议
  - AH提供完整性，源端的可认证性
  - ESP（相对于AH来说）提供附加的机密性

- IPsec 对可以是2个网络实体间的安全通信关系
  - 2个路由器/防火墙
  - 1个路由器/防火墙，另外一个是端系统
  - 2个端系统之间