INFORMATION SECURITY

IPSec and SSL

Information Security

School of Data & Computer Science
Sun Yat-sen University

Lecture Notes:
courseware_is_sysu@163.com
Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

Outline

- **Chap 8. IPSec and SSL**
  - **IPSec**
    - Introduction
    - Some Basic Concepts about IPSec
    - ESP protocol
    - Key Management of IPSec
    - Gateway and Road Warrior Mode
  - **SSL/TLS**
    - Introduction
    - How TLS Works
    - Decryption of TLS Packet
  - **VPN**
    - Introduction to IPsec VPN
    - OpenVPN

Introduction

- IPsec, Internet Protocol Security, is officially specified by IETF.
  - RFC-2401 Security Architecture for the Internet Protocol
- IPsec is a protocol suite.
  - IPsec is a protocol suite for securing Internet Protocol communications by authenticating and encrypting *each IP packet* of a communication session.
  - IPsec also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session.
    - 每个 IP 包都得到认证、加密
    - 会话开始时建立双方的交互认证
    - 会话期间进行密钥协商

- IETF: the Internet Engineering Task Force

Introduction

- IPsec is an end-to-end security scheme.
  - IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite. It can be used in protecting data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*).
- IPsec protects any application traffic across an IP network.
  - Applications do not need to be specifically designed to use IPsec.
    - Oppositely, some other Internet security systems in widespread use operate in the upper layers of the TCP/IP model such as Secure Sockets Layer (SSL), Transport Layer Security (TLS) and Secure Shell (SSH). The use of TLS/SSL must be designed into an application to protect the application protocols.
- IPsec is a successor of the ISO standard Network Layer Security Protocol (NLSP).
  - NLSP was based on the SP3 protocol that was published by NIST, but designed by the Secure Data Network System project of NSA.

Introduction

- IP 协议的安全性
  - 传统的 IP 协议诞生于军用计划，设计之初并未考虑太多安全问题，因而存在很多安全隐患。
    - 比如数据明文传输，共用一个集线器的通信可以被互相监听，如果获得交换机权限，所有流经交换机的通信也可以被监听。攻击者即便没有交换机权限，也可以通过中间人攻击窃取用户的通信。
- IPsec 提供了网络层加密方案
  - 对 IP 协议进行安全加强的迫切需要催生了 IPsec。IPsec 在网络层将每个 IP 分组的内容先加密再传输，即使中途被截获，攻击者由于缺乏解密数据包所必要的密钥而无法获取其中内容。

Introduction

- IPsec 对数据进行加密的方式有两种：传输模式和隧道模式。
  - 传输模式只对 IP 协议报文的有效数据载荷部分 (payload) 进行加密，因此需要对原始 IP 报文进行拆装。
  - 隧道模式对整个 IP 协议报文进行加密，相当于把原始 IP 报文封装在一个安全的隧道进行传输，保持了原始 IP 报文的完整性。

What We Need to Protect Data

- Data Confidentiality 数据保密性
  - 用各种加密手法对数据进行加密，保证攻击者无法破解密密文。
- Data Integrity 数据完整性度量
  - 保证所收到的数据是完整的，在传输途中没有被恶意篡改。
- Origin Authentication 数据来源认证
  - 确认数据发送方和接收方的身份，防止攻击者的伪造。
    - 攻击者截获数据包后将发送地址改成自己的地址，诱骗接收方将回复包发送给攻击者。
- Prevent Replay-Attack 防止数据回放攻击
  - 攻击者截获交易命令的数据包后，将交易过程的所有数据包重复发送一次，就可能造成另一次重复的交易，从而使得被攻击者遭受损失。
  - 给每一个数据包打上一个唯一的标示 (比如时间戳或随机数序列)，及时鉴别重复数据包是一种可行的办法。

How IPsec Protects Data

- 数据保密性
  - 使用对称加密技术可以保证加解密操作的速度。
  - 由于对称密钥需要通过不安全的网络来传输，所以需要一个保护密钥的机制。公钥密码技术可以提供这样的保护机制以保证对称密钥的安全。
  - 采用公钥密码体制的公钥合法性需要加以保证，即需要确定给我们发送公钥证书的一方不是伪装的攻击者。因此需要寻求一个可以信任的第三方为通信双方进行身份验证，比如使用基于 PKIX 的 X.509 证书。
  - IPsec 的密钥传输不仅可以使用 X.509 证书，还可以使用预共享密钥 (PSK) 或 RSA 密钥。

How IPsec Protects Data

- 数据完整性度量
  - 对消息进行摘要并将摘要结果附在消息上传输 (比如 HMAC)，接收者收到数据后以同样的方式对数据进行摘要，再将结果与附在数据后面的摘要进行对比，获得数据的完整性。
  - 需要考虑摘要算法的抗碰撞性。
- 数据来源认证
  - 来源认证同样可以由消息摘要技术解决，此时的消息摘要中包含了发送方的地址信息。
- 防止数据回放攻击
  - 给数据加上时间戳和随机数，以确保能唯一区分每个新的消息。

Some Basic Concepts

- AH Protocol 认证头协议
  - RFC 2402 IP Authentication Header
  - AH (Authentication Headers) 协议能够在数据的传送过程中对数据进行完整性度量和来源认证，还可以防止回放攻击。
  - AH 协议在被保护的 IP 报文上添加一个称为认证报头的数据项，其中包含一个带密钥的对该 IP 报文计算的 Hash 值。对 IP 报文内容的任何更改将致使该值无效，从而提供了数据完整性保护。
  - AH 协议和 ESP 协议相比较具备更强的认证能力，它能保护通信免受篡改，但没有提供加密能力，因此不能防止被窃听，适合用于传输非机密数据。
  - AH 结构由 IP 协议号51标识，AH 的封装结构随着所采用的传输模式和隧道模式有所不同，需要分别讨论。
  - AH 可以单独使用，也可以与 ESP 协议结合使用。

Some Basic Concepts

- ESP Protocol 封装安全载荷协议
  - RFC 2406 IP Encapsulating Security Payload (ESP)
  - ESP (Encapsulating Security Payloads) 能够在数据的传输过程中对数据进行完整性度量和来源认证，可以选择加密，也可以选择防止回放保护。
  - ESP 服务依据建立的 SA (Security Association, 安全关联)，对可选项目有所限制：
    - 完整性检查和认证必须一起选择；
    - 仅当选择了完整性检查和认证时才能选择防止回放保护；
    - 防止回放保护只能由接收方选择。
  - ESP 的加密服务是可选的，但如果启用加密，则也就同时选择了完整性检查和认证。
    - 如果单独使用加密服务，入侵者可能发起密码分析攻击。

Some Basic Concepts

- ESP Protocol 封装安全载荷协议
  - ESP 提供了两种数据加密模式：传输模式和隧道模式
    - 传输模式下 ESP 不对整个原始 IP 报文加密，而只加密其中不包括 IP 头的有效载荷部分。但在端对端的隧道通信中，ESP 需要对整个原始 IP 报文加密 (隧道模式)。
  - ESP 结构由 IP 协议号50标识，ESP 的封装结构随着所采用的传输模式和隧道模式有所不同，需要分别讨论。
  - ESP 可以单独使用，也可以和 AH 结合使用。

Some Basic Concepts

- Tunnel Mode 隧道模式
  - 隧道模式下 IPsec 将要发送的原始 IP 报文作为数据内容，在这段“数据”前面加上 ESP 或 AH 协议头，再加上新的 IP 头，形成 IPsec 报文进行传输。
  - 原始 IP 报文的传输就像在一个安全的隧道中进行一样，传输过程中报文保持原有的完整结构，内容没有被修改。
- Transport Mode 传输模式
  - 传输模式下 IPsec 保护的仅仅是原始 IP 报文的数据内容部分 (即 IP 报文的有效载荷)，而不是整个原报文。在传输过程中原报文结构被拆装。
  - 在处理方法上，原始 IP 报文被拆解，在其有效载荷前面加上新的 ESP 或 AH 协议头，再装回原来的 IP 地址，形成新的 IPsec 报文。

How IPsec Organize All Things Together

- Need a structure to store keys and related things
  - SA (Security Association)
- Need a place to store SAs
  - SAD (Security Association Database)
- Need a structure to associate packets with SAs
  - SPI (Security Parameter Index)
- Need a place to store policies (or rules)
  - SPD (Security Policy Database)

How IPsec Organize All Things Together

- SA
  - Security Associations 安全关联。
  - SA 是 IPsec 的重要概念，可以理解为被 IPsec 保护的某个连接的唯一标示。SA 是单向的，即在一次安全的通信中，通信的两个方向 (发送和接收) 各需要创建一个 SA。
  - 一个 SA 所包含的内容是维护一次安全通信所需要的数据参数。通常，一个 SA 可以由目的地址、IPsec 所采用的协议 (AH 或 ESP) 和 SPI 来唯一确定。
  - 所有的 SA 都被存放在一个数据库中，称为 SAD。
  - SA 的建立和维护通过密钥交换协议 IKE 实现。

How IPsec Organize All Things Together

- SAD
  - Security Associations Database 安全关联数据库。每一个 SA 在 SAD 中都有一个对应的条目，保存 SA 的信息。
  - SAD 条目的内容：
    - 顺序号计数器 Sequence number counter for outbound communications
      - 在 AH 或 ESP 的头部，占32比特。SA 初次建立时置0，每发送一个数据包加1。
    - 顺序号溢出计数器 Sequence number overflow counter
      - 用来标志这个 SA 是否应该被弃用。如果顺序号已经溢出，当前的 SA 就应该被抛弃，否则会使得重放攻击成为可能。

How IPsec Organize All Things Together

- SAD
  - SAD 条目的内容：(续)
    - 防止回放窗口 Anti-replay Window
      - 占32比特。与 TCP 窗口的概念类似，引进窗口的原因是为了实现可靠的传输服务。
      - The destination maintains a 'sliding window' record of the sequence numbers of validated received packets; it rejects all packets with sequence numbers that are lower than the lowest in the window (i.e. too old) or that have already appeared in the window (i.e. duplicates/replays). Accepted packets, once validated, update the boundaries of the window (displacing the lowest sequence number out of the window if it was already full).

How IPsec Organize All Things Together

- SAD
  - SAD 条目的内容：(续)
    - SA 有效期 Lifetime of the SA
      - 通过字节计数 (byte count) 或时间帧 (time frame) 或两者的结合来记录一个 SA 的使用时间。若两者一起使用的话，以先到期限的那一个为准。当 SA 使用了一段时间后就应该被删除以确保安全。
    - AH 协议中所使用的算法以及密钥
      - 默认情况下，IPsec 至少要支持 HMAC-MD5 和 HMAC-SHA，算法需要密钥支持。

How IPsec Organize All Things Together

- SAD
  - SAD 条目的内容：(续)
    - ESP 协议用于认证以及完整性度量的算法以及密钥
    - ESP 协议用于加密数据的算法以及密钥
    - IPsec 运行的模式
      - 传输模式 (transport mode) 或者是隧道模式 (tunnel mode)。
    - PMTU (Path Maximum Transmission Unit)
      - 由 SA 的 ICMP 数据获得。MTU 值是传送数据包大小的最大上限，PMTU 是两个通信设备间的 MTU。

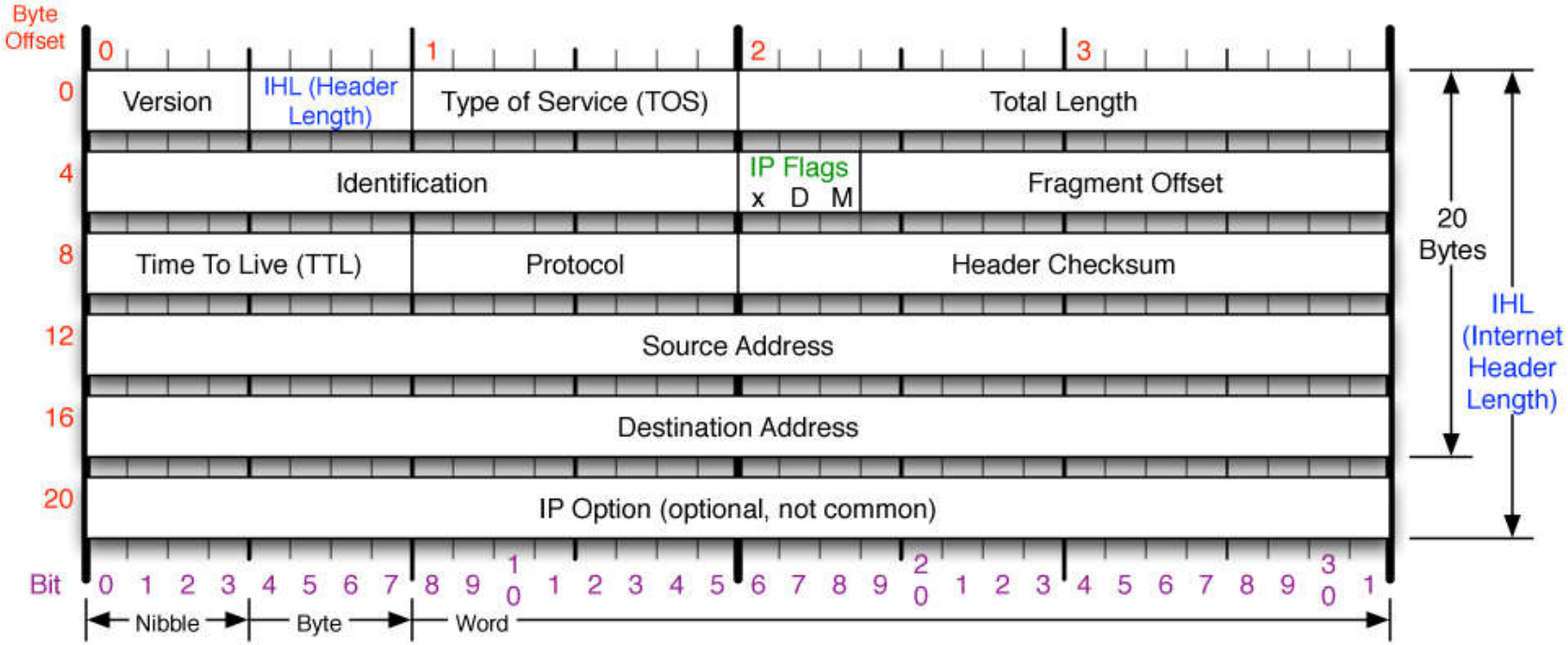
How IPsec Organize All Things Together

- SPI
  - Security Parameter Index 安全参数索引。
  - 用于将收到的 IPsec 数据包与其对应的 SA 进行关联。
- SPD
  - Security Policy Database 安全策略数据库。
  - IPsec 的策略就是规则。SPD 的策略告诉系统如何处理收到的数据包，例如将包处理成 IPsec 数据包从而进行保护，或者不保护直接转发，甚至直接丢弃。
- IKE
  - Internet Key Exchange 互联网密钥交换协议。
  - 默认情况下，IPsec 使用 IKE 自动管理密钥，也可以直接手动管理。

IP Header

```
typedef struct _iphdr          // IP 首部, 20 bytes
{
  unsigned char ver-hlen       //4位 IP 版本号 + 4位首部长度
  unsigned char TOS;           //8位服务类型, TOS = PPPDTRC0
  unsigned short pkt_len;      //16位总长度, max = 65535
  unsigned short id;           //16位分片标识符 (同一分片具有相同标识)
  unsigned short flgs-offset;  //3位分片标记 + 13位偏移值
  unsigned char TTL;           //8位 IP包生存时间, 每跳减1, TTL=0 时丢弃
  unsigned char proto;         //8位上层 (封装) 协议号: 1-ICMP, 2-IGMP,
                               // 6-TCP, 17-UDP, 88-IGRP, 89-OSPF
  unsigned short hchecksum;    //16位 IP 首部校验和, 每跳必算
  unsigned int src_IP;         //32位源 IP 地址
  unsigned int dst_IP;         //32位目的 IP 地址
} IP_HEADER;
```

IP Header (version 4)



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

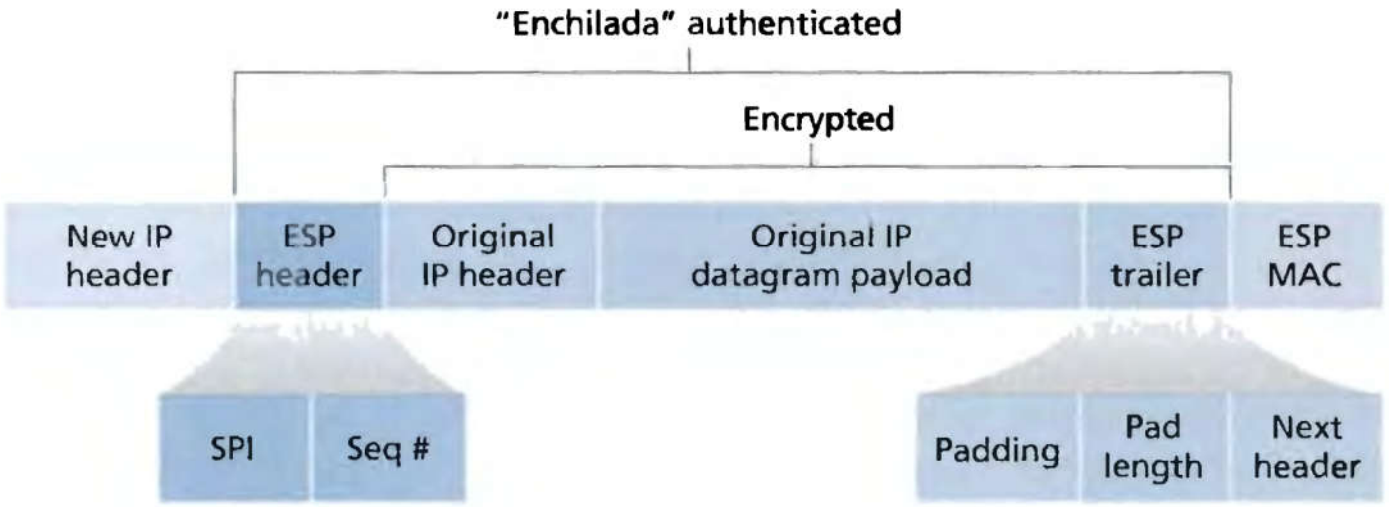
x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

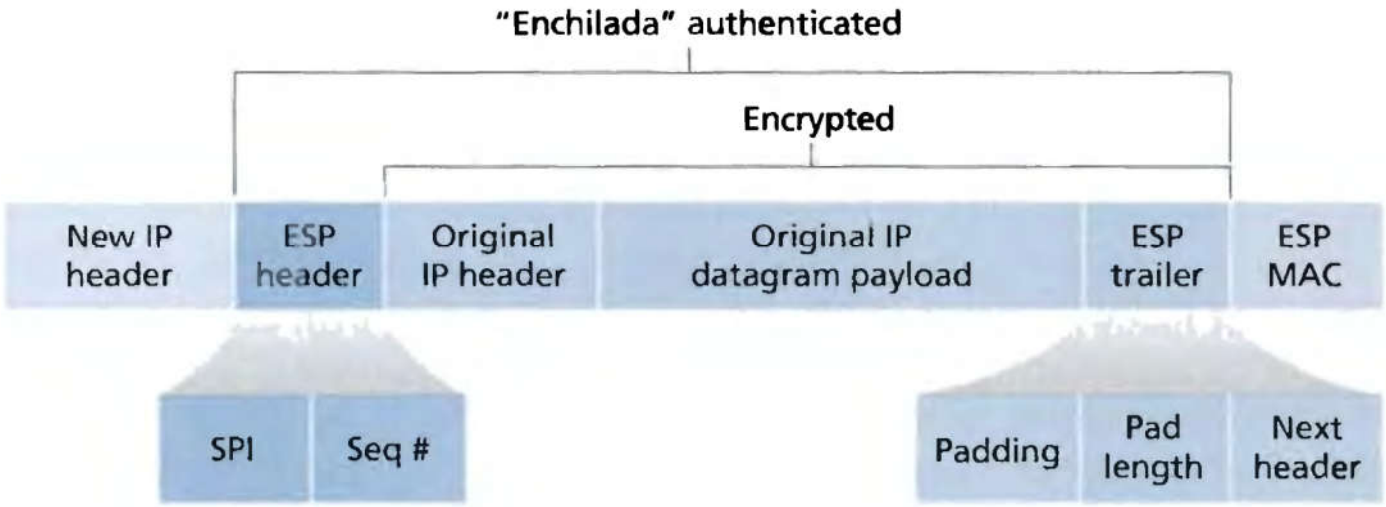
ESP Protocol in Tunnel Mode

- ESP Datagram in Tunnel Mode.



ESP Protocol in Tunnel Mode

- When a packet is going to be sent
  - (1) Append an ESP trailer
  - (2) Encryption
  - (3) Append an ESP header
  - (4) Append MAC
  - (5) Create a New IP Header

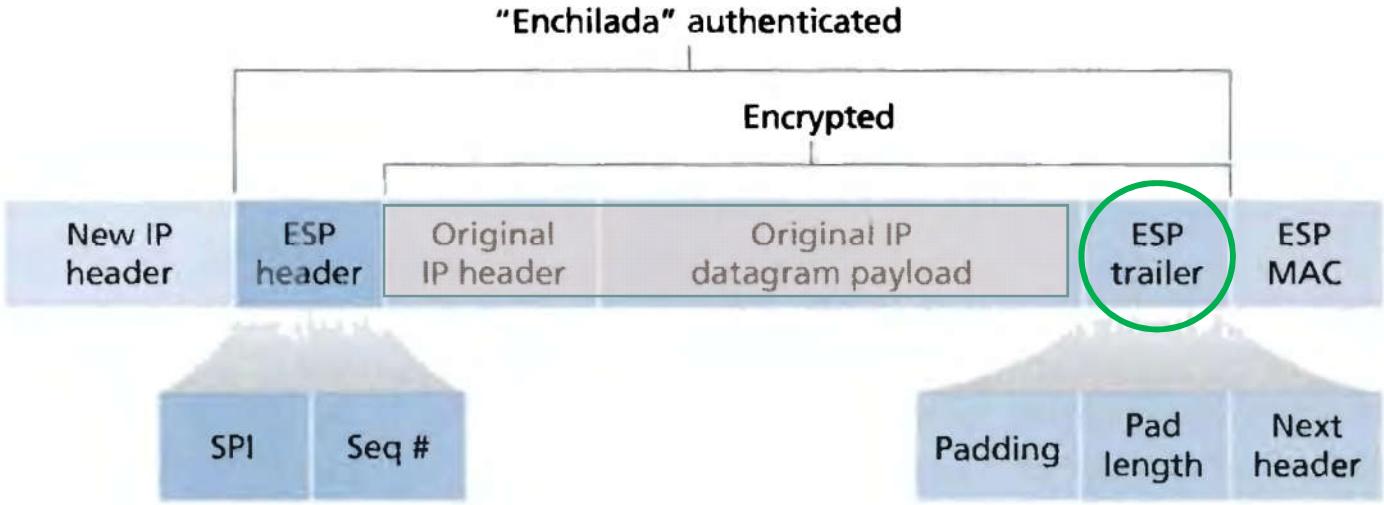


ESP Protocol in Tunnel Mode

- 装包过程

(1) 在原 IP 报文末尾添加 ESP trailer (尾部/挂载) 信息。

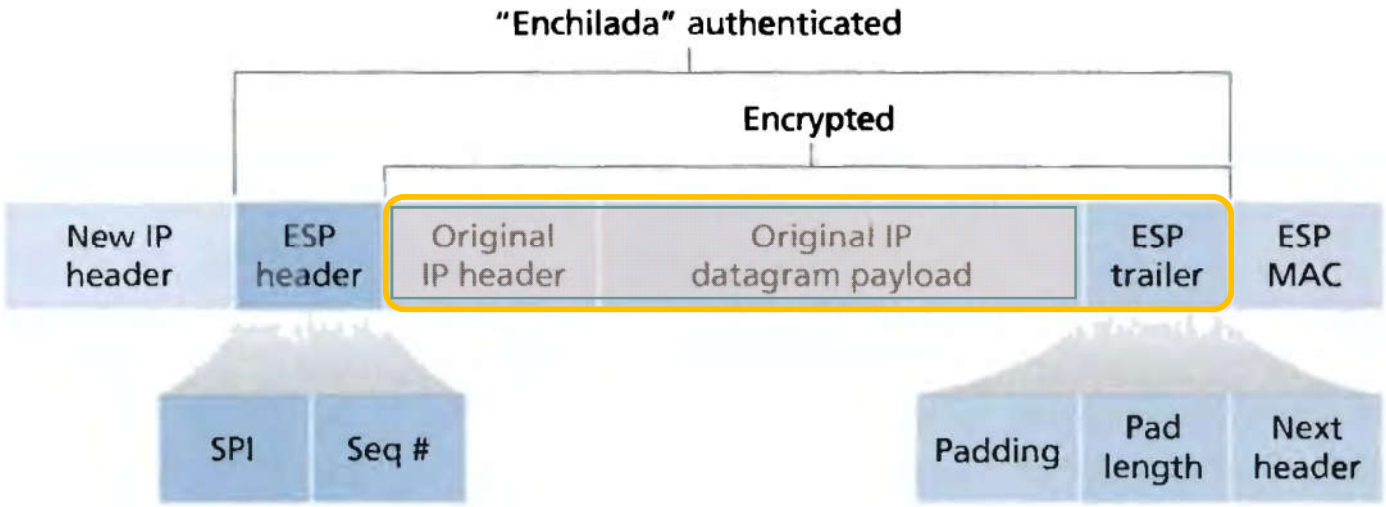
- ESP trailer 包含三部分。由于所选加密算法可能是块加密，当最后一块长度不足时就需要填充 (padding)，附上填充长度 (Pad length) 方便解包时顺利找出用来填充的那一段数据。Next header 用来标明被封装的原报文的协议类型，例如 4 (= IP)。



ESP Protocol in Tunnel Mode

- 装包过程

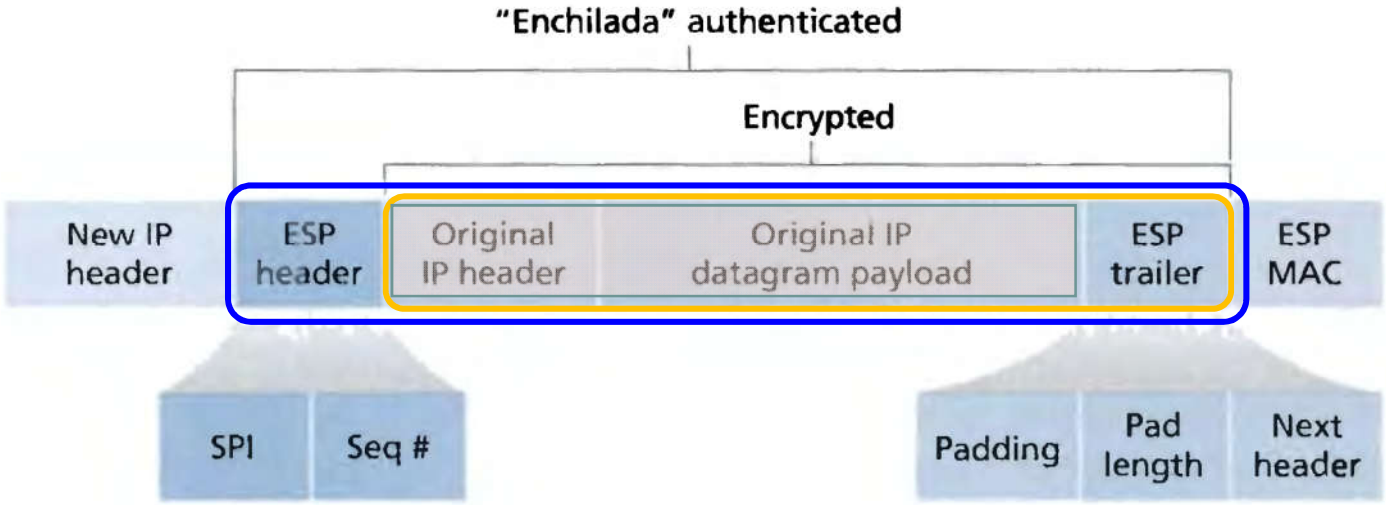
(2) 将原 IP 报文以及第1步得到的 ESP trailer 作为一个整体进行加密封装。具体的加密算法与密钥由 SA 给出。



ESP Protocol in Tunnel Mode

- 装包过程

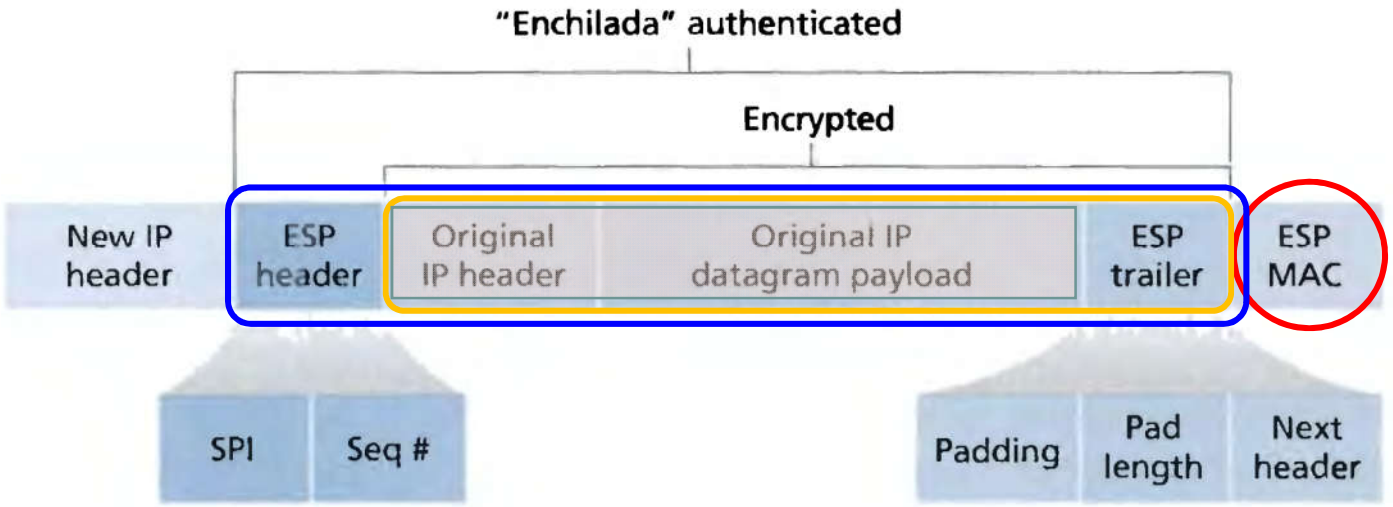
(3) 为第2步得到的加密数据添加 ESP header。ESP header由 SPI 和 Seq# 两部分组成。加密数据与 ESP header 合称为 “Enchilada”，构成认证部分。注意到被封装的原报文的协议类型受到保护，由加密的 ESP trailer 的 Next header 声明，而不出现在未加密的 ESP header 中。



ESP Protocol in Tunnel Mode

- 装包过程

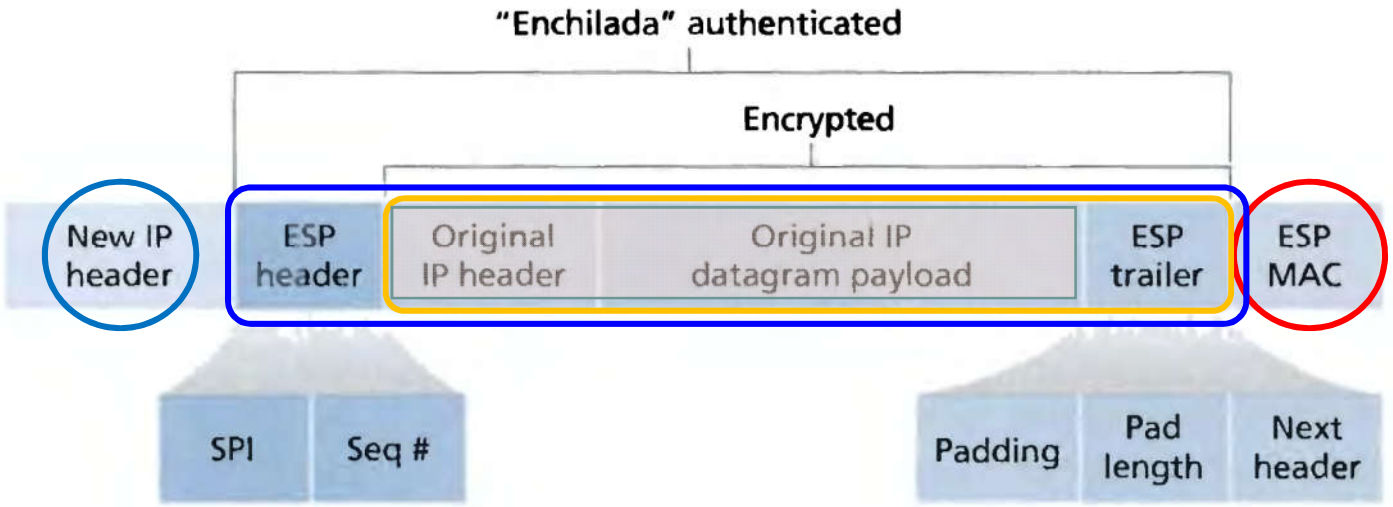
(4) 附加完整性度量结果 (ICV，Integrity check value)。对第3步得到的 “enchilada” 部分做认证，得到一个32位整数倍的完整性度量值 (消息认证码 MAC)，并附在 ESP 报文的尾部。完整性度量算法包括需要的认证密钥由 SA 给出。



ESP Protocol in Tunnel Mode

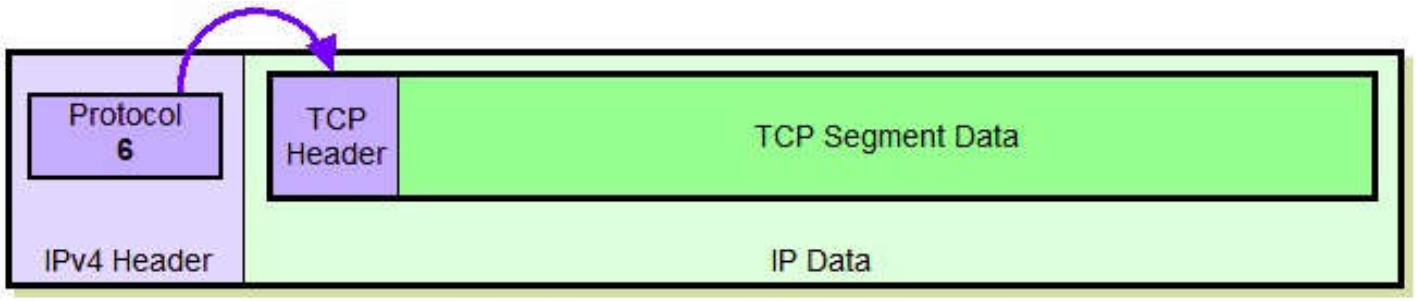
- 装包过程

(5) 加上新的 IP header 构成 IPsec 报文。新构造的 New IP header 附在 ESP 报文的前面组成一个新的 IP 报文。注意 New IP header 的 IP 地址由路由器和安全网关解释，可以和原报文 (由主机创建的 IP 地址) 不同。协议类型为50，说明它封装的是一个 ESP 报文。

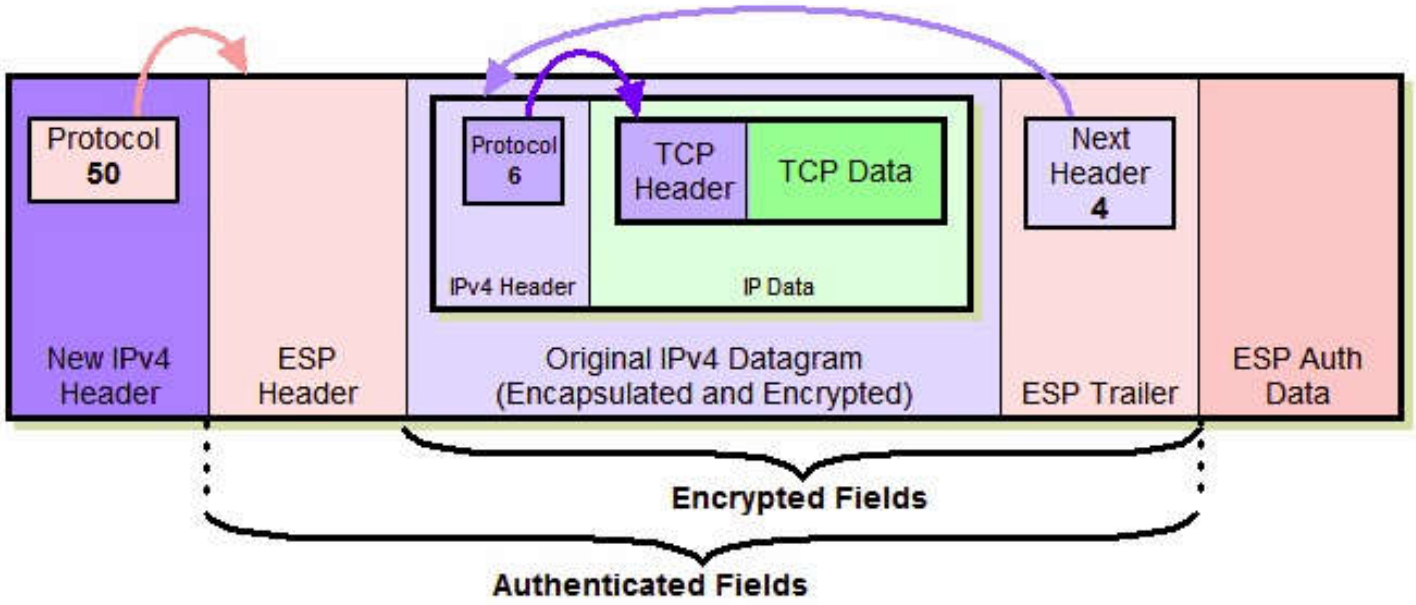


ESP Protocol in Tunnel Mode

- 装包过程



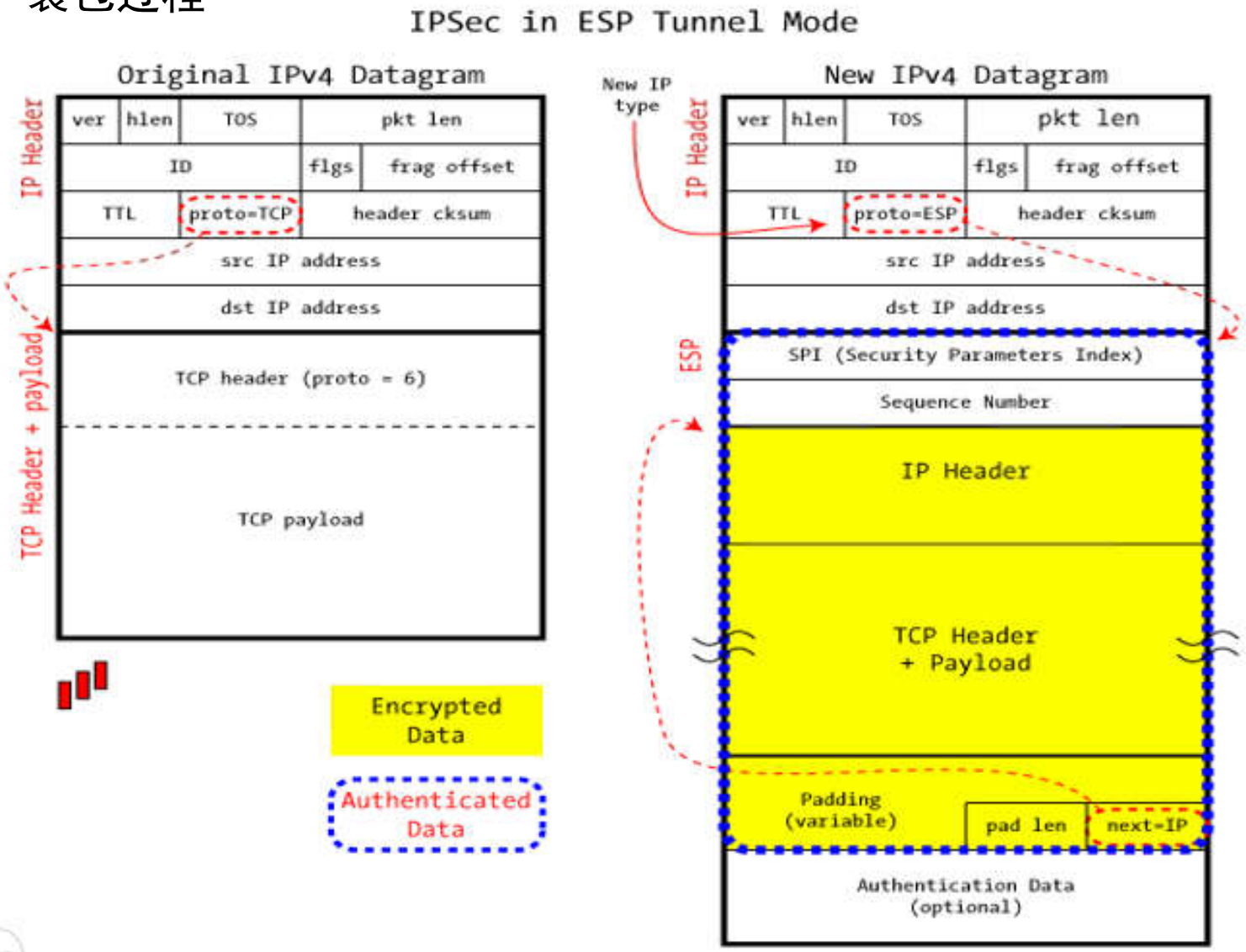
Original IPv4 Datagram Format



IPv4 ESP Datagram Format - IPSec Tunnel Mode

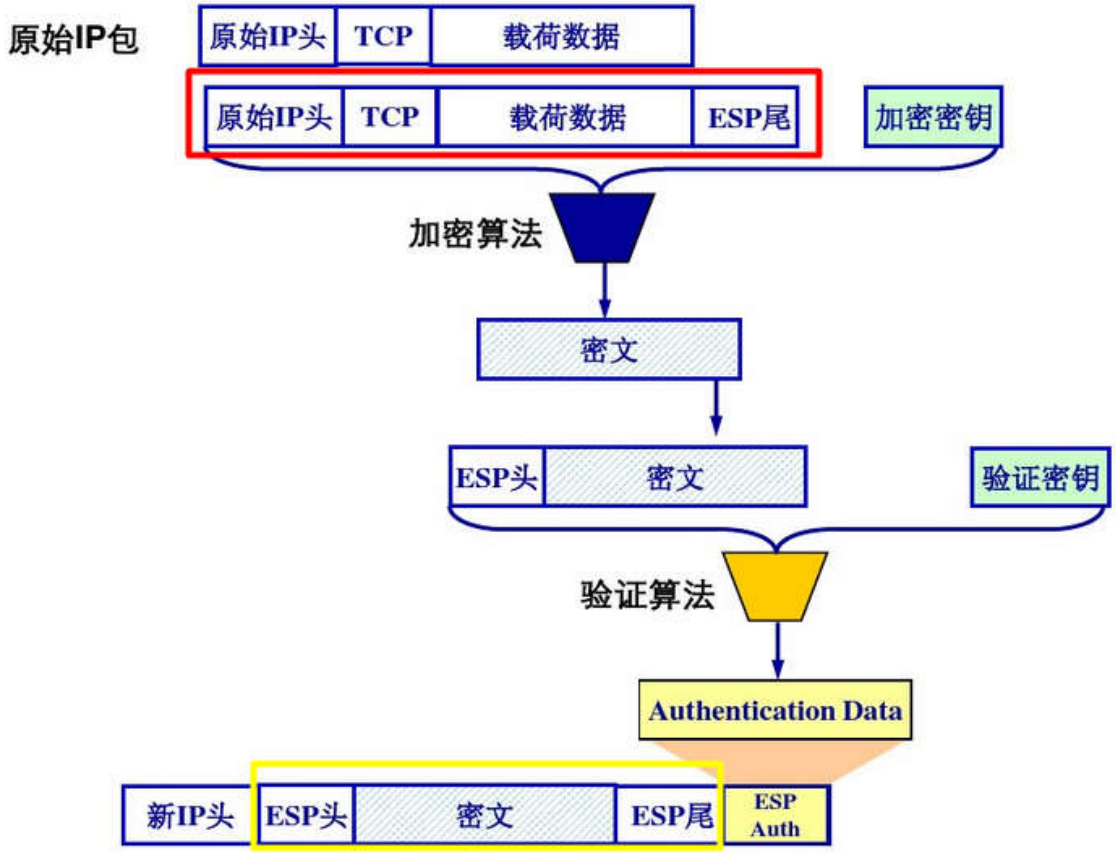
ESP Protocol in Tunnel Mode

- 装包过程



ESP Protocol in Tunnel Mode

- 隧道模式下的认证和传输区域
  - 红色区域是加密区域，黄色区域是验证区域。



ESP Protocol in Tunnel Mode

- When a packet is received
  - (1) Use SPI to determine SA
  - (2) Calculate MAC
  - (3) Check sequence number
  - (4) Decryption
  - (5) Remove padding
  - (6) Forward the original datagram

ESP Protocol in Tunnel Mode

- 拆包过程如下：

(1) 接收方收到 IP 报文后，发现协议类型是50，表明这是一个 ESP 包。首先查看 ESP header，通过 SPI 检索数据报文所对应的 SA，读取对应的模式 (tunnel/transport mode) 以及安全规范。

(2) 计算 “enchilada” 部分的摘要，与附在末尾的 ICV 做对比，验证数据完整性。

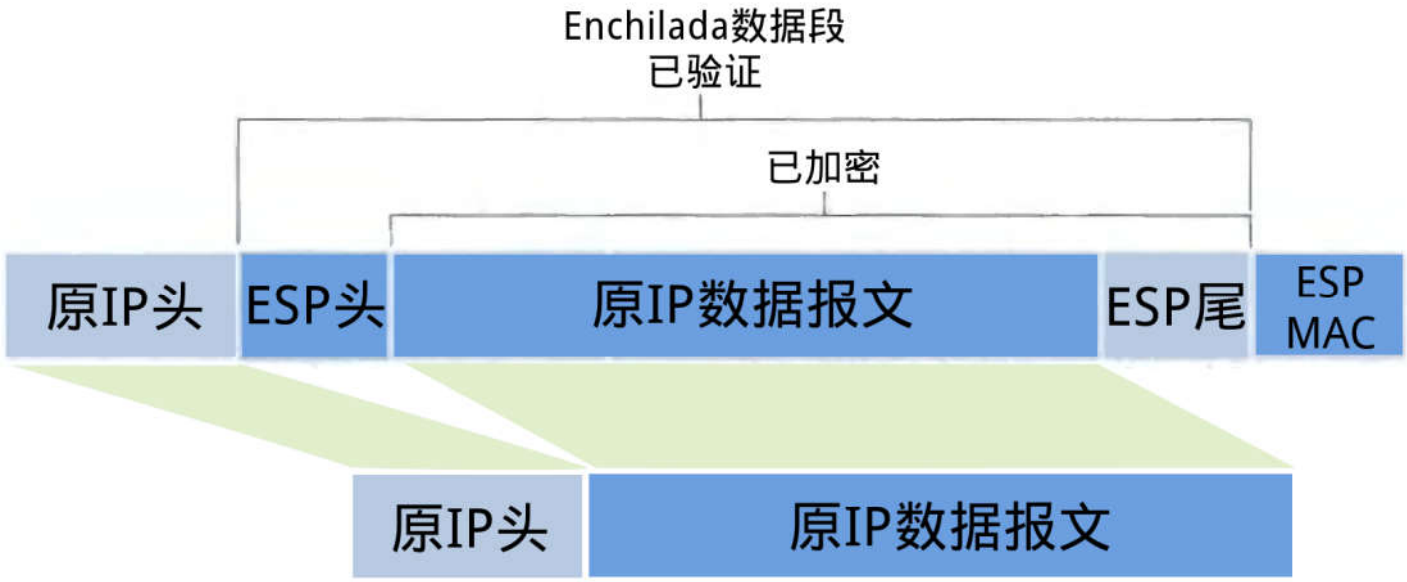
(3) 检查 Seq# 里的顺序号，保证数据是“新鲜”的。

(4) 根据 SA 所提供的加密算法和密钥，解密被加密过的数据，得到原 IP 报文与 ESP trailer。

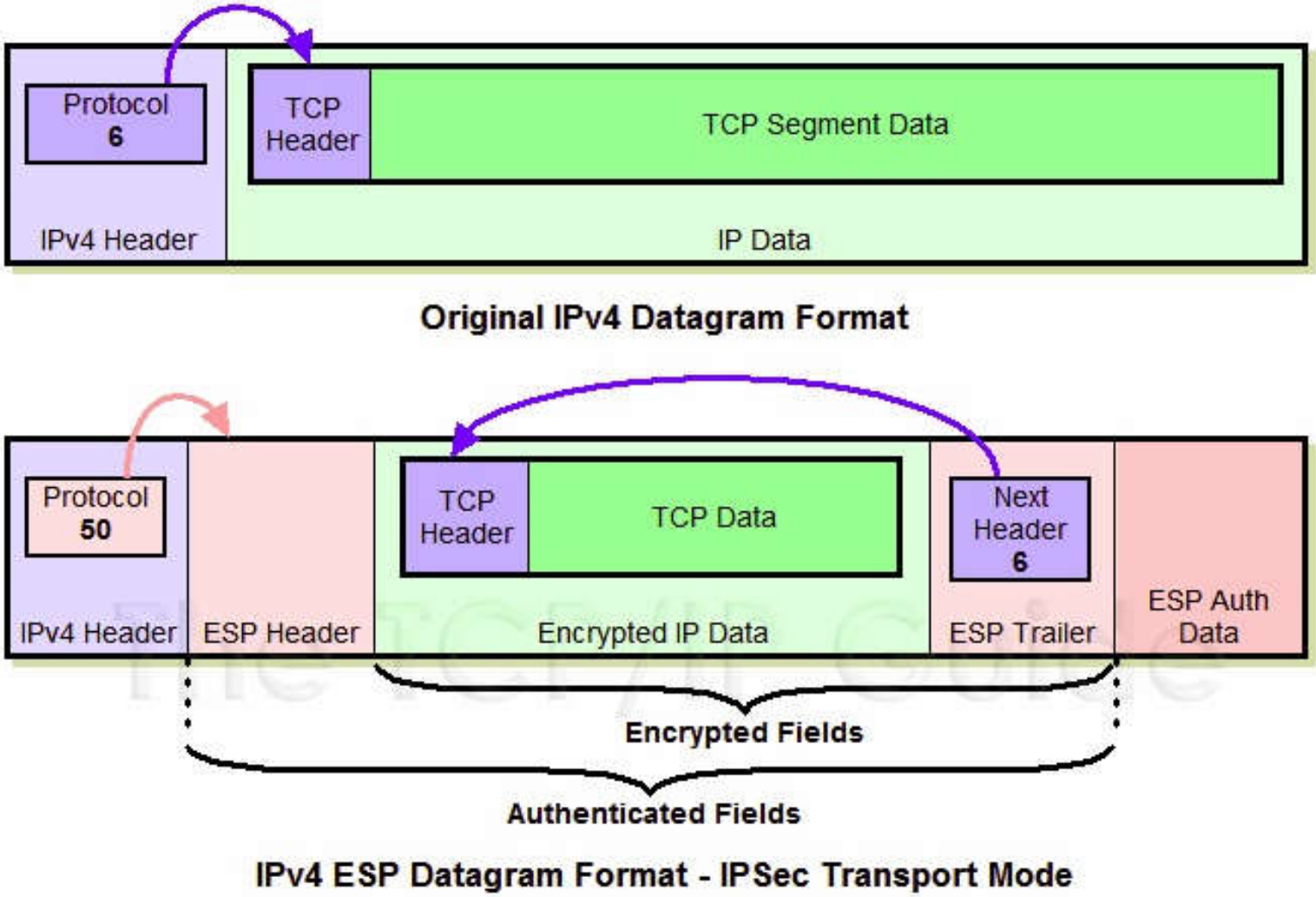
(5) 根据 ESP trailer 的填充长度信息，找出填充字段的长度，删去后得到原来的 IP 报文。

(6) 最后根据得到的原 IP 报文的目的地址进行转发。

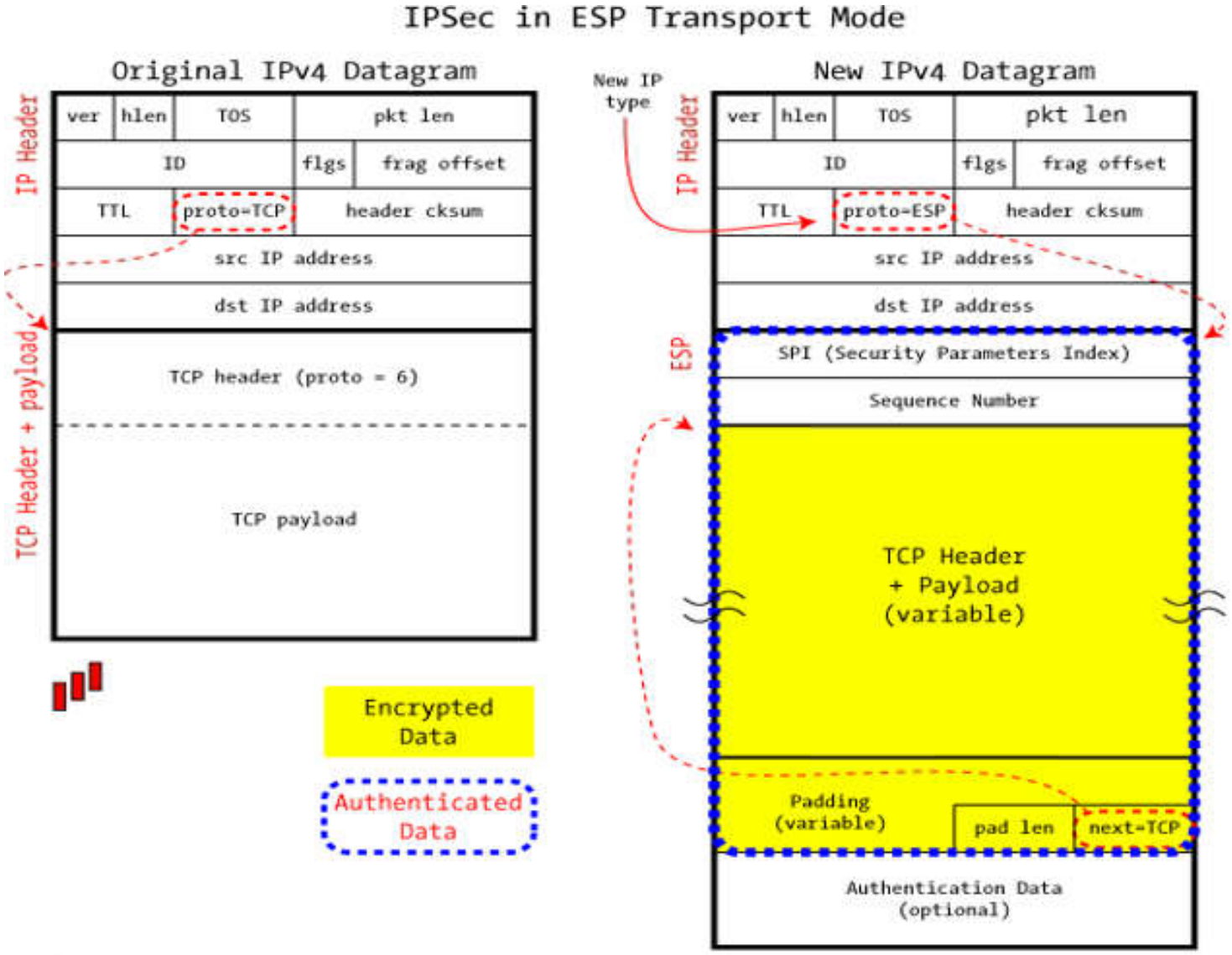
ESP Protocol in Transport Mode



ESP Protocol in Transport Mode

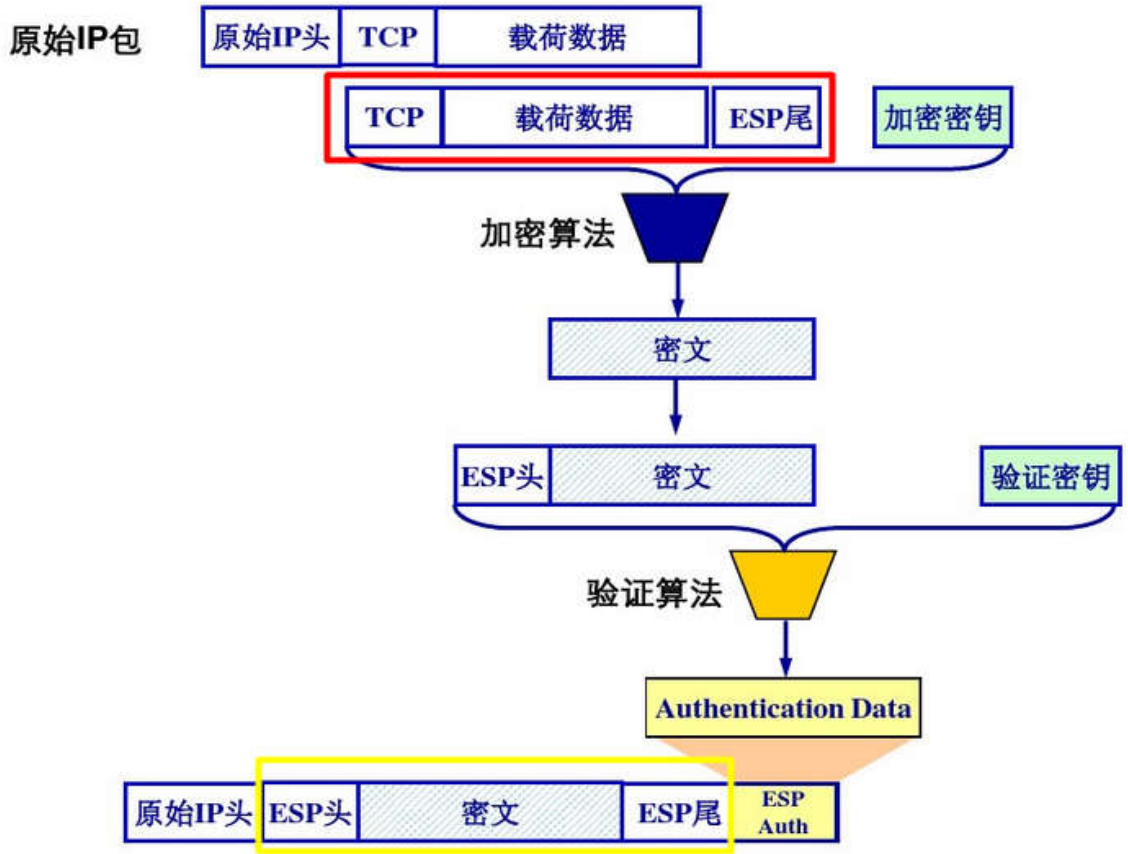


ESP Protocol in Transport Mode

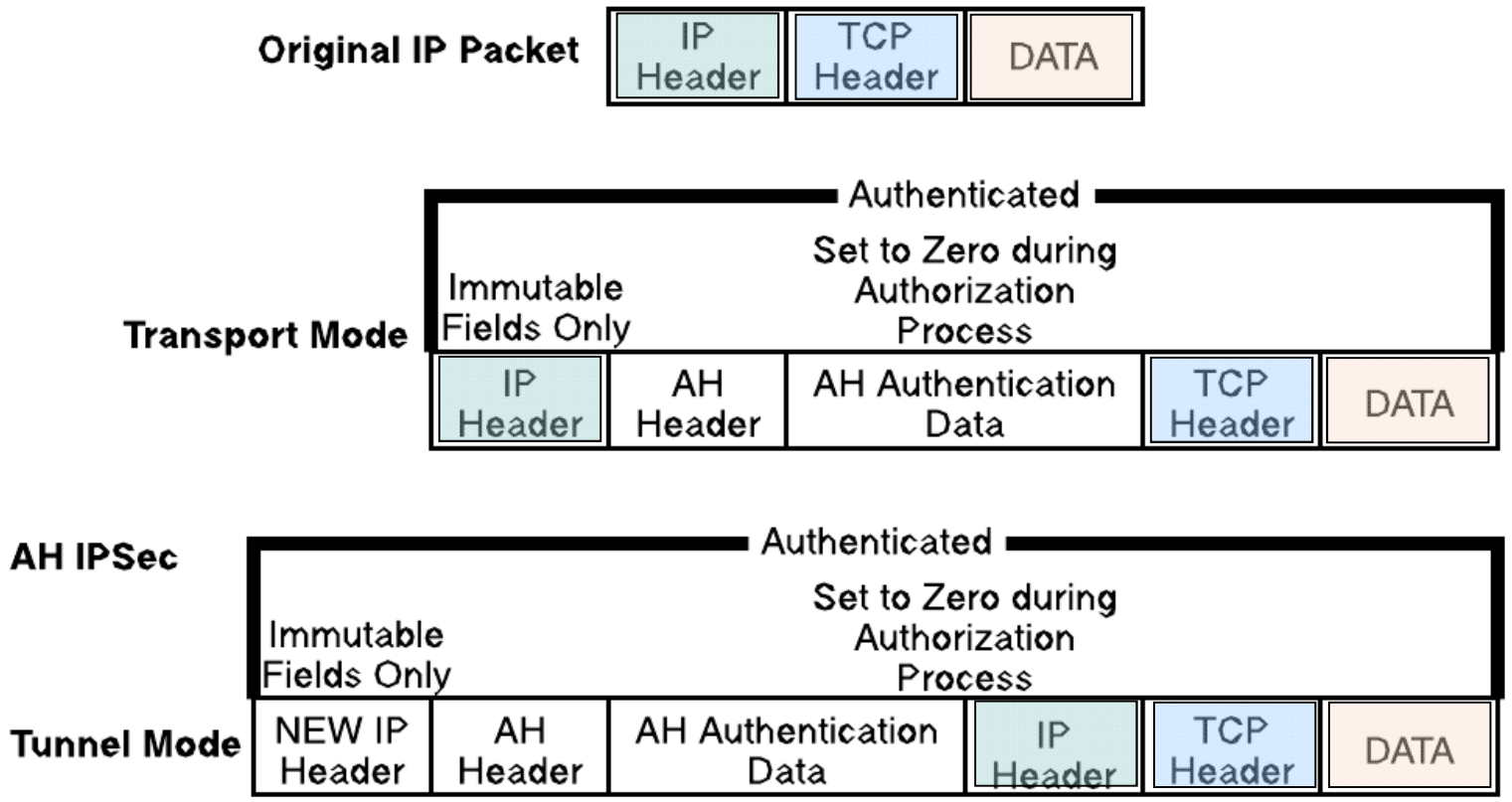


ESP Protocol in Transport Mode

- 传输模式下的认证和传输区域
  - 红色区域便是加密区，黄色区域是验证区。



A Brief Mention of AH Protocol



A Brief Mention of AH Protocol

- AH 协议能够对数据进行完整性度量和来源认证，但不提供任何的加密服务，所以它只适用于数据不需要保密的情况。
- AH 协议头结构：

0 7	8 15	16 31
Next Header	Payload Len	Reserved
Security Parameters Index (SPI)		
Sequence Number Field		
Authentication Data (Variable)		

A Brief Mention of AH Protocol

- Next Header 占8位，存放了连接在认证头后面的有效负载 (payload) 的类型。
- Payload Len 指定 AH 的长度，具体的计算方法是以32位为单位来表示 AH 的长度，再减去2。例如一段占64位的认证数据加上固定的96位协议头，一共是160位，那么，最终算出的 Payload Len 应是 [(96+64)/32]-2，即 3。
- Reserved 占16位，为预留区域，方便以后扩充。目前应该全为0。这部分在计算认证数据时也会参与计算。
- SPI 占32位，用于将 AH 数据报文与相应的 SA 做映射。
- Sequence Number Field 占32位，存放了一个递增的计数值，用于抵抗重放攻击。该字段是强制要求使用的，无论是否启用了反重放攻击的功能。当 SA 建立时置为0。
- Authentication Data，认证数据，即报文的完整性度量值 (Integrity Check Value)。长度是可变的，但必须是32位的倍数，不足时需要填充。

IKE – Key Management of IPsec

- IKE (Internet Key Exchange)
  - V1: RFC2409, 1998. / V2: RFC 5996, 2010. / RFC 7296, 2014
  - ISAKMP (RFC2408, 1998)
    - Internet Security and Key Management Protocol
      - Used for establishing Security Associations (SA) and cryptographic keys in an Internet environment. It only provides a framework for authentication and key exchange and is designed to be key exchange independent.
  - Oakley (*Hilarie K. Orman*, RFC2412, 1998)
    - The Oakley Key Determination Protocol
      - a key-agreement protocol that allows authenticated parties to exchange keying material across an insecure connection using the *Diffie–Hellman* key exchange algorithm.
  - SKEME (*Hugo Krawczik*, 1996)
    - Secure Key Exchange MEchanism for Internet

IKE – Key Management of Ipsec

- IPSec 的通信双方需要事先协商好将要采用的安全策略，包括使用的加密算法、密钥、密钥的生存期等，亦即创建 SA。AH 和 ESP 都需要使用 SA，而 IKE 的主要功能就是 SA 的建立和维护。
- IPsec 使用 IKE (Internet Key Exchange) 协议来进行自动的密钥管理。IKE 实际上是一个混合协议，它包含协议
  - ISAKMP
  - Oakley
  - SKEME
- ISAKMP 协议是 IKE 协议的主要组成部分，它负责指定密钥的协商过程。ISAKMP 协议只是一个框架，并未规定具体使用的加密算法。
- Oakley 和 SKEME 协议可以理解为加密算法的具体规定。

IKE – Key Management of Ipsec

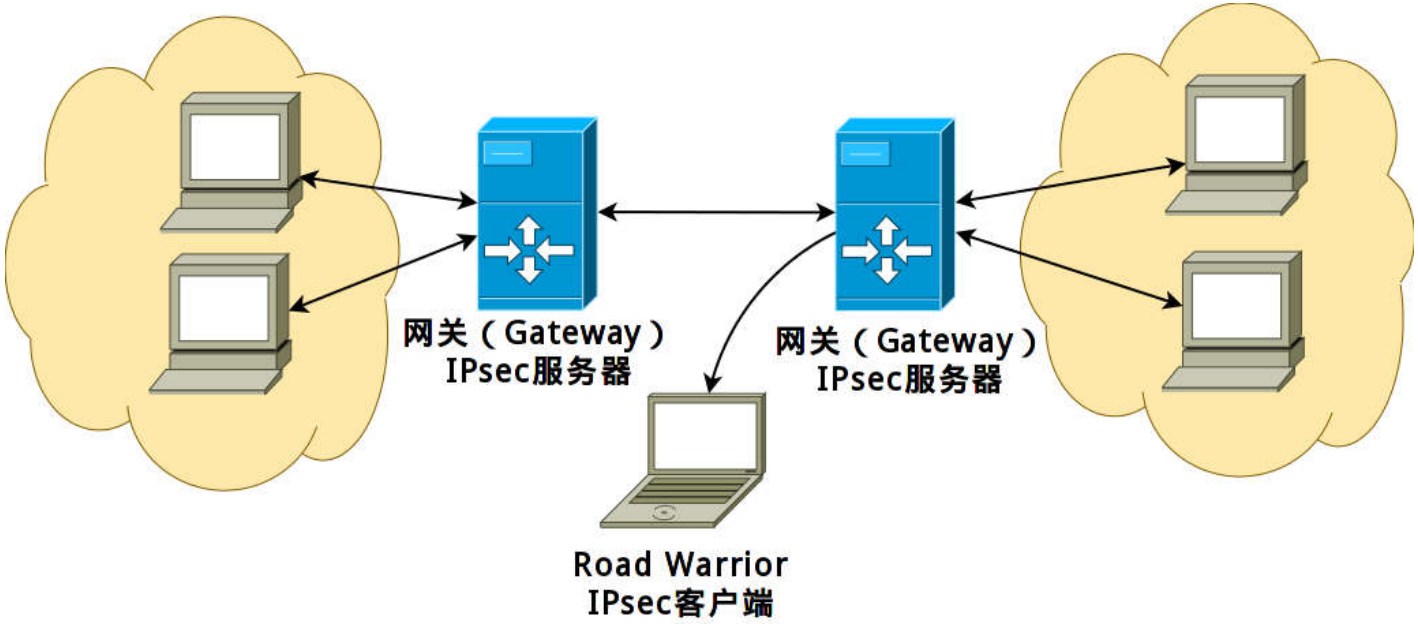
- 使用 IKE 的 IPsec 的密钥协商分为两个阶段：
  - 阶段一：建立 IKE-SA
    - 双方使用 *Diffie-Hellman* 算法创建两个方向的 IKE-SA，这里的 IKE-SA 与前面讲的 SA 有所不同。事实上通信时只有一个 IKE-SA 被建立，由通信的两个方向共享。期间还会生成阶段二协商所需用到的密钥。
  - 阶段一又可细分为主模式和激进模式。
    - 主模式 (Main Mode)；激进模式 (Aggressive Mode)。
    - 主模式需要进行多次数据交换，在交换密钥后对双方的身份消息进行加密，防止中间人攻击。虽然速度慢，但更安全。激进模式需要较少的数据交换，速度更快，但安全性有所降低。

IKE – Key Management of Ipsec

- 使用 IKE 的 IPsec 的密钥协商分为两个阶段：
  - 阶段二：协商 IPsec SA
    - 阶段二又称为快速模式 (quick mode)。阶段二的协商在阶段一所建的安全信道中进行，例如选用 ESP 还是 AH，加密用的密钥等等。 从这一阶段起后面的数据都是经过加密的。

Gateway and Road Warrior Models

- IPsec 通常应用于两种情况，一是两个私有网络通过因特网 的对接，另一种情况是外出办公的内部人员通过因特网连入私有网络，两者都需要保护好通信数据。如图所示：

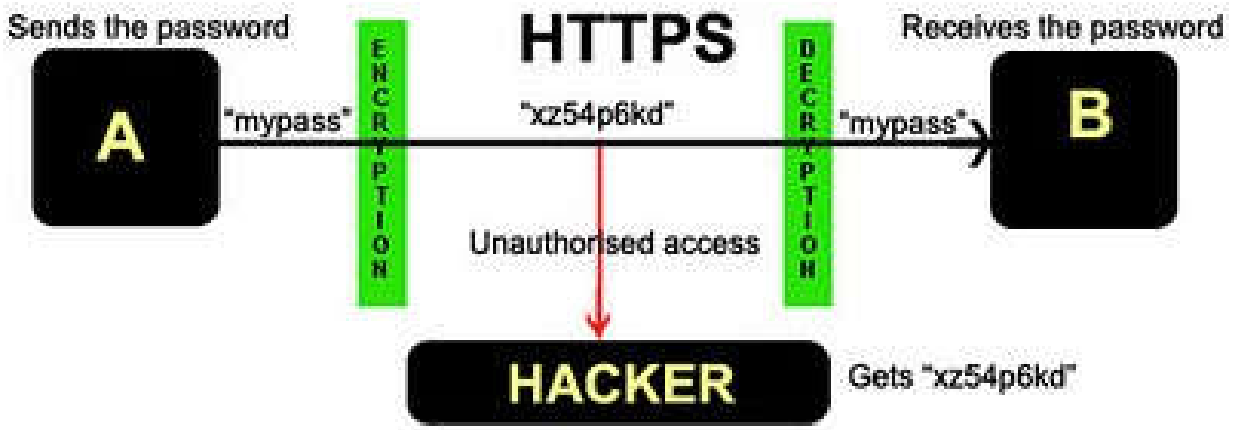


Gateway and Road Warrior Models

- Gateway 模式是两个网关之间的通信，只需要在两边的网关上做好 IPsec 的设置。
- Road Warrior 模式一头是网关，另一头是单个的客户端 (例如一台笔记本)。
  - 无论是 Gateway 模式还是 Road Warrior 模式，至少有一边是网关。想要通过 Internet 向网关后面某个内部主机发送信息时，需要先将经过 IPsec 保护的数据包发给网关，由网关将这个报文解包转发到真正的目的地。
- 私有网络内的主机通常被认为是可信的，因此在一个私有网络内部，没有必要为每一台设备配置 IPsec，而只需要在私有网络对外的出口，即网关处配置。

Security in the Transport Layer

- Every thing will be transported in plain text!
  - *Example.* http
- How to keep data in secret
  - Encrypt my data with *Symmetric Key* before sending it
  - Protect *Master Key* with my public key (PKI)
  - (Receiver) verify my *Public Key* with *Certificate* (by CA)
  - Prevent my data from tempering with *HMAC* (Hash-based Message Authentication Code)



What is SSL/TLS

- SSL/TLS 是工作在传输层的协议，目的是防止数据被窃听和篡改。
  - SSL (Secure Sockets Layer) 最早由 NetScape 发明，其2.0版本被认为有安全缺陷， 3.0版本发布于1996年并得到广泛应用。后来 IEEE 对 SSL 进行标准化，成为 TLS (Transport Layer Security, 传输层安全协议)。
  - 最先发布的 TLS1.0 即是 SSL3.1。实际上 TLS1.0 并没有对 SSL3.0 做太大的改动，详细说明参见 RFC2246。
  - https 是 SSL/TLS 的一个应用。
    - https 使用443端口，区别于原来 http 的80端口
  - 本课件的讨论基于 TLS 1.2 (RFC5246, 2008)，即 SSL3.3。更高级别的版本被禁止从美国出口。
  - TLS 1.3 于2018年8月发布，详细说明参见 RFC8446。

What is SSL/TLS

- A Simplified Picture of SSL
  - (1) Client sends out connection request.
  - (2) Server accepts the request and send back Server's public key PU(s).
  - (3) Client builds an *E*(PU(s), *K*) by encrypting a secret key *K* with PU(s) and send the *E*(PU(s), *K*) to the server.
  - (4) Server decrypts the *E*(PU(s), *K*) with its private key PR(s) and gets the secret key *K.*
  - (5) Client and serve start communicating with the secret key *K.*

What is SSL/TLS

- 一个简化的 SSL/TLS 模型：

  (1) 客户端向服务器发出连接请求。

  (2) 服务器接收请求，向客户端发送服务器公钥 PU(s)。

  (3) 客户将随机生成的密钥 *K* 用服务器的公钥加密，并将加密结果 *E*(PU(s), *K*) 发给服务器。

  (4) 服务器用自己的私钥 PR(s) 解密 *E*(PU(s), *K*) 得到 *K*。

  (5) 通信的双方都已经得到了密钥 *K*，接下来的通信使用刚才协商的密钥 *K* 进行加密。

What is SSL/TLS

- 一个简化的 SSL/TLS 模型：
  - 从上面的过程可以看出，SSL/TLS 其实与 IPsec 类似，两者处理的都是加密通信前密钥交换的问题，交换好密钥后再使用协商密钥来保护接下来的通信。
  - 在进行真正的 SSL/TLS 通信时，情况要复杂一些，例如，服务器不会简单的向客户发送一个公钥，而是发送一份数字证书。SSL/TLS 不仅仅要保证数据的保密性，还要保证传输数据的完整性，所以会用到 HMAC。
  - 考虑到 SSL 与 TLS 的密切关系，我们主要以 RFC5246 为基础，讨论 TLS 的实现。

Some Basic Concepts

- Session
- Connection
- Write State
- Read State
- Cipher Suites
- Pre-master Secret
- Record Layer Protocol
- Handshake Protocol
- ChangeCipherSpec Protocol
- Alert Protocol
- Application Data Protocol

Some Basic Concepts

- Session 会话
  - 每一次 SSL 握手初始化都会创建一个 SSL 会话，一次握手恢复不会创建新的会话，而是继续沿用之前已经建立的会话。通常进行 SSL 通信的双方只会创建一个 SSL 会话。
- Connection 连接
  - 每一个连接归属于某一个会话，而一个会话可以包含多个连接。通信的双方进行新的数据交换时只需要新建一个连接，而不需要重新建立会话，这样的设计可以减少握手开销。
- Write State 写模式
  - 在一次 SSL 会话中，通信双方都会分别纪录自己的写模式的信息。对于 *Alice* 与 *Bob* 的通信来说，*Alice* 的写模式描述了由 *Alice* 到 *Bob* 的通信方向，即*Alice* 所发送的消息所遵循的与安全通信相关的约定，例如加密算法和密钥。

Some Basic Concepts

- Read State 读模式
  - 同样，在一次 SSL 会话中，通信双方都会维护自己的读模式的信息。*Alice* 的读模式描述了由 *Bob* 到 *Alice* 的通信方向，即 *Alice* 所接收的消息所遵循的约定。
  - 读模式和写模式描述的是不同的通信方向，但读写模式是对应的。对于 *Alice* 与 *Bob* 的通信，*Alice* 的写模式就是 *Bob* 的读模式，两者纪录的信息 (密钥，算法) 完全一样。同样的，*Alice* 的读模式就是 *Bob* 的写模式。
  - *Alice* 的写模式和 *Bob* 的写模式可以不同，这意味着通信的两个方向可以采用不同的密钥和加密、散列算法。

Some Basic Concepts

- Cipher Suites 安全套件
  - 安全套件是通信过程中使用的各种加密算法与散列算法的集合，用于表示客户端所支持的加密与散列函数。安全套件中的每一个条目表示一种组合 (SSL 选项)，每种组合用两个字节表示，定义了会话的某个方向所使用的密钥交换算法、对称加密算法、密钥长度以及散列算法。
  - 例如 0x00, 0x3D 代表的就是

    TLS_RSA_WITH_AES_256_CBC_SHA256
- Pre-Master Secret 预主密钥
  - 当通信双方协商选用 RSA 加密算法后，客户端会先生成一个预主密钥 PMS，用于计算后续的密钥。

Some Basic Concepts

- Record Layer Protocol 记录层协议
  - 记录层协议是 TLS 中所有协议的基础，所有的 TLS 通信都是通过这个协议来传输的，但它只是一个“外套”，其协议头包含了很少的信息，而数据段则负载了 TLS 的各种子协议。
  - 负载部分可以是用于通信过程控制的三个子协议之一，即握手协议、更改秘钥规格协议或者告警协议。
  - 当记录层协议的类型字段为23 (应用数据的标识) 时，负载部分是已加密的数据，这时记录层协议就是一个应用数据协议。

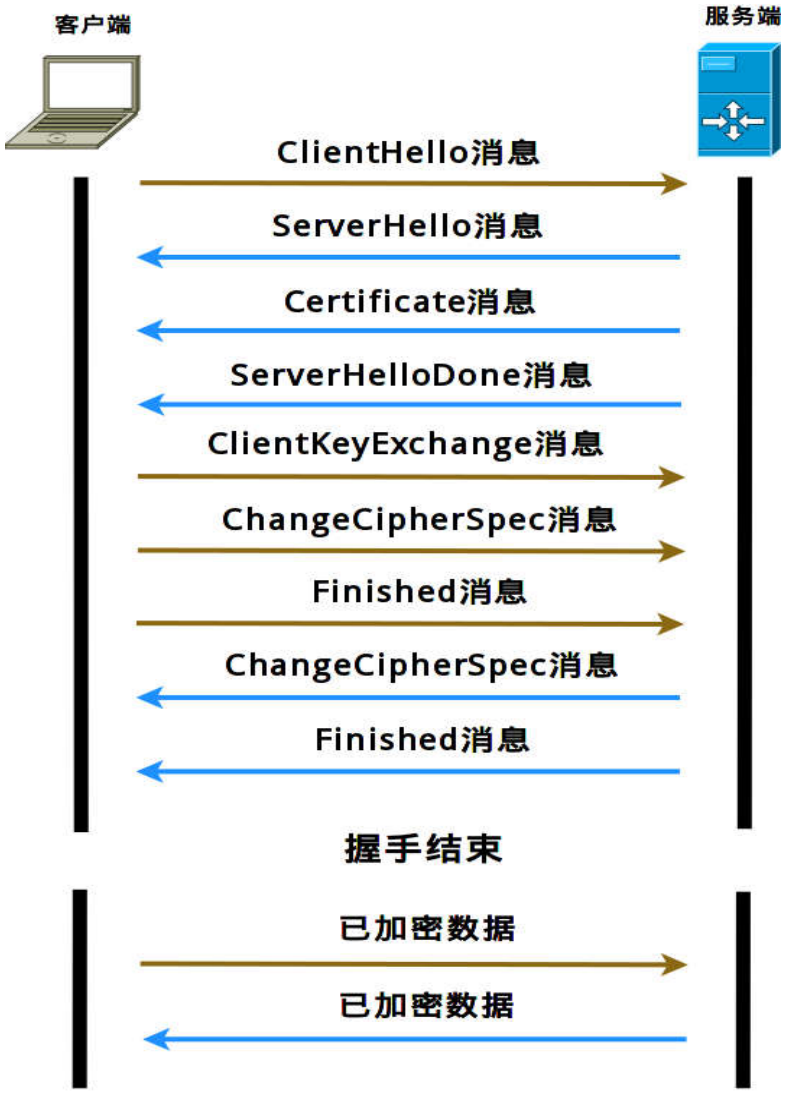
Some Basic Concepts

- TLS 的三个子协议
  - Handshake Protocol 握手协议
    - 握手协议主要用于正式通信前的密钥协商。
  - ChangeCipherSpec Protocol 更改秘钥规格协议
    - 更改秘钥规格协议是一个简单协议，它的负载部分只含有一个字节，其值取0或1，用于激活通信双方的读写模式。
  - Alert Protocol 告警协议
    - 告警协议用于传输通信过程中的错误信息。

Some Basic Concepts

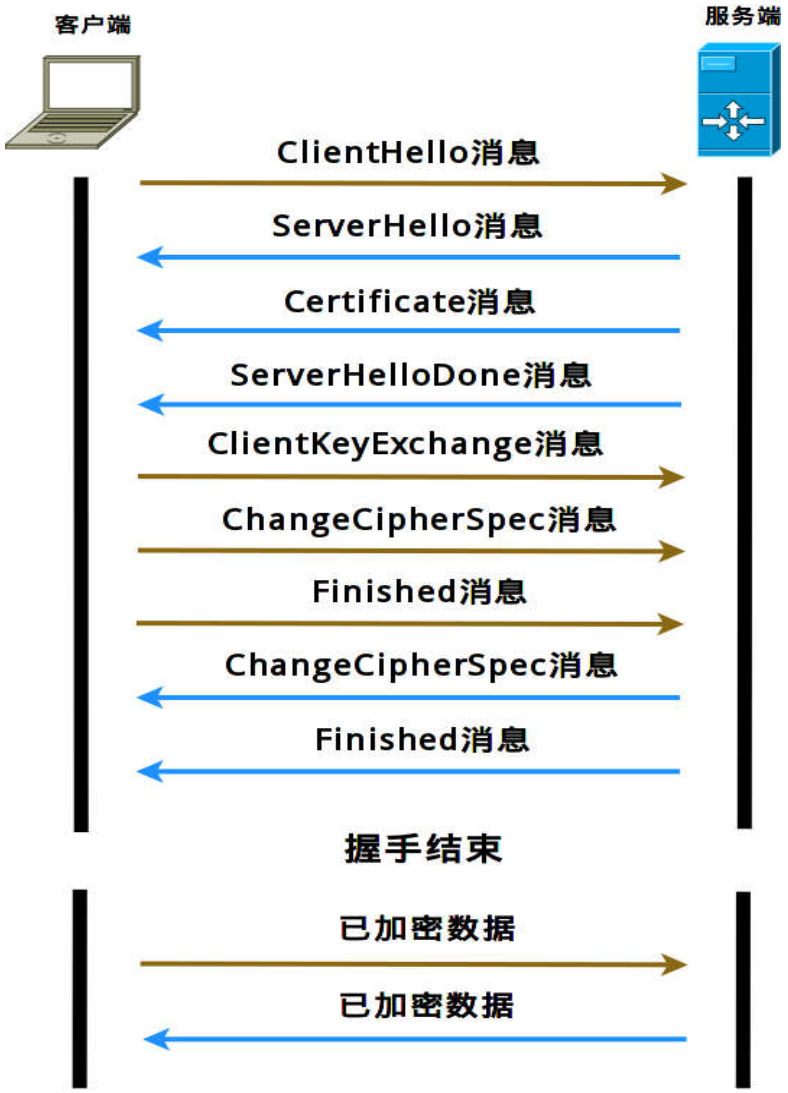
- Application Data Protocol 应用数据协议
  - 所有的受保护数据都将通过应用数据协议来传输，应用数据协议传输的是真正需要通信的数据。
  - 应用数据协议严格来讲不是一个单独的协议，而是记录层协议的一个类型。当记录层协议的类型字段填为23 (应用数据的标识)，而且负载部分填上已加密的数据后，它就是一个应用数据协议。
  - 应用数据协议在具体实现上与记录层协议同级，但在逻辑上又与其他3个子协议同级。因此逻辑上应用数据协议也可以理解为记录层协议的一个子协议。

- **TLS Handshake**
  - ClientHello
  - ServerHello
  - Certificate
  - ServerHelloDone
  - ClientKeyExchange
  - ChangeCipherSpec
  - Finished
  - ChangeCipherSpec
  - Finished



TLS Handshake

- 从简化的 SSL/TLS 模型可以看出，建立会话的很大部分耗费用在了密钥协商这一部分，这一过程被称为 SSL/TLS 握手。一旦握手成功，意味着通信双方都已经安全的得到了共享密钥。



TLS Handshake

- 客户端：ClientHello
  - 客户端向服务端器发出连接请求。ClientHello 消息里面还包含了客户端生成的随机数 (后面计算主密钥要用到)、会话 ID (Session ID) 以及客户端所支持的 SSL 选项。初次握手时，会话 ID 为0，表示客户端要求建立新的会话。
- 服务端：ServerHello
  - 服务端回应请求，从客户端提供的 SSL 选项中选取一个并告知客户端。
  - 由于 ClientHello 消息的会话 ID 为0，所以服务端会返回一个新的会话 ID，用于标识本次会话。同时服务端也会生成一个自己的随机数并发送给客户端。

TLS Handshake

- 服务端：Certificate
  - 在发送 ServerHello 消息后服务端会发送自己的公钥信息。当选用 RSA 算法时，服务端会发送自己的证书。
- 服务端：ServerHelloDone
  - 服务端通知客户端它已经完成密钥交换的准备工作，本身不含任何特殊的信息。注意到直到这一步，所有的信息都是没有加密的。客户端收到这个消息后可以开始与服务端交换密钥。
- 客户端：ClientKeyExchange
  - 由于选用的是 RSA 算法，客户端在这一步会随机生成一个预主密钥 PMS，用服务端的公钥加密并将加密结果发给服务端。至此，SSL 的初期协商结束。

TLS Handshake

- 客户端：ChangeCipherSpec
  - 客户端为之后的会话激活自己的写模式选项，并通知服务端激活其读模式的选项，包括密钥、加密算法、HMAC 算法等。
  - 这些选项在 Hello 消息和 ClientkeyExchange 消息已经协商过，协商结果在这之前处于待定 (Pending) 状态，只有在这个消息发出去后所有协商的选项才会被真正的激活。
  - 注意到 ChangeCipherSpec 信息由 ChangeCipherSpec 协议定义，它本身不属于握手协议的一部分，但是在握手过程中要用到。

TLS Handshake

- 客户端：Finished
  - 客户端告知服务端握手成功。
  - Finished 消息通常紧接着 ChangeCipherSpec 消息被发送，同时它也是第一个使用协商好的密钥和加密算法加密的数据。该消息还包含对以下3个内容做的摘要：
    - 密钥信息
    - 在此之前所有握手信息的内容
      - 攻击者插入或修改信息将导致的客户端和服务端两边的 hash 值不同
    - 一个指明发送方是服务端还是客户端的值
  - 这个摘要值可以保证握手过程的完整性，即中途没有被攻击者篡改。

TLS Handshake

- 服务端：ChangeCipherSpec
  - 与客户端发送的 ChangeCipherSpec 类似，服务端为之后的会话激活自己写模式的选项，也就是客户端读模式的选项。这里的服务端写模式选项可以与客户端的写模式，也就是服务端的读模式选项不同，即通信的两个方向可以选用不同的选项。
- 服务端：Finished
  - 服务端告知客户端握手成功。附带的摘要信息如前所述。

- 至此，SSL 握手结束，接下来的信息都可以使用握手协商得到的共享密钥来进行保护。

Key Generation

- Client randomly generates a PMS
  - PMS (Pre-Master Secret) 预主密钥
- Four keys are generated according to PMS
  - Encryption key for client to server
  - Encryption key for server to client
  - MAC key for client to server
  - MAC key for server to client

Key Generation

- TLS 需要的密钥
  - 通信双方密钥协商成功后，可以利用密钥来保护真正的通信。
  - 通信的一方会有一个写模式和读模式，而每一种模式会采用对应的对称密钥和 HMAC 密钥，所以一共需要 2x2=4 个密钥。
    - 注意到不是 2x2x2=8 个密钥，因为其中一方的写模式就是另一方的读模式，所以只需要计算两个模式。
  - 问题：前面握手的时候只协商了一个预主密钥 PMS。
- 伪随机数函数 PRF
  - TLS 利用前面协商得到的 PMS 来计算出所需要的4个密钥。这其中起关键作用的就是伪随机数函数 PRF。下面是 RFC5246 中给出的其中一种 PRF 定义。

Key Generation

- RFC5246 中给出的一种 PRF 定义：

```
if (i == 0)
    A(i) = seed ;
Else
    A(i) = HMAC_hash(secret, A(i-1))① ;
P_hash(secret, seed) =  HMAC_hash(secret, A(1)+seed) +②
                        HMAC_hash(secret, A(2)+seed) +
                        HMAC_hash(secret, A(3)+seed) +
                        ... ③ ;
PRF(secret, label, seed) = P_hash(secret, label + seed) ;
```

① 定义中用到的 HMAC_hash 函数在 RFC5246 中使用 SHA-256。

② “+” 号表示字符串的连接。

③ PRF 生成的随机数由多个散列值叠代连接而成，散列值的个数 (也就是叠代的次数) 由所需要的数据块长度决定。

Key Generation

- 主密钥的计算：

```
key_block0 = PRF(PMS①, “master key” ②, 客户端随机数 + 服务端随机数)
MS = key_block0[0..47]③
```

① Key_block0 是以 PMS 为种子应用 PRF 生成的一个数据段。

② 这里的 “master key” 就是字符串 “master key” 的 ASCII 码串。

③ 取 key_block0 的前48个字节得到主密钥 MS。

Key Generation

- 密钥块的计算：

  key_block = PRF(MS①, “key expansion” ②, 客户端随机数 +服务端随机数)

  ① Key_block 是以 MS 为种子应用 PRF 生成的一个数据段。

  ② 这里的 “key expansion” 就是字符串 “key expansion” 的 ASCII 串。

客户端MAC写密钥 client_write_MAC	服务端MAC写密钥 server_write_MAC	客户端写密钥 client_write_KEY	服务端写密钥 server_write_KEY	客户端写IV client_write_IV	服务端写IV server_write_IV	剩余部分... remaining...

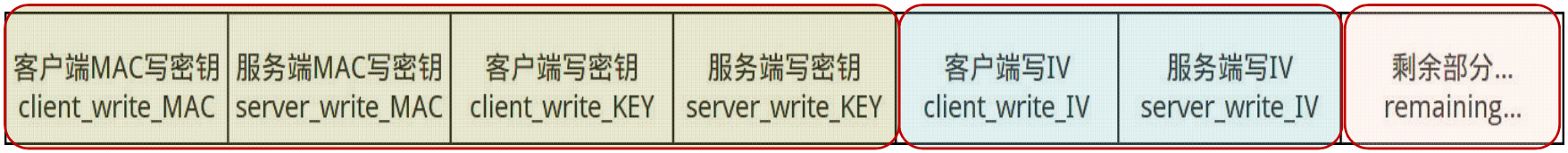
Key Generation

- 密钥块的计算：

  key_block = PRF(MS①, “key expansion” ②, 客户端随机数 +服务端随机数)

  ① Key_block 是以 MS 为种子应用 PRF 生成的一个数据段。

  ② 这里的 “key expansion” 就是字符串 “key expansion” 的 ASCII 串。



- 从 key_block 中顺序划分出4个密钥和2个初始向量 IV (选用块加密算法时需要用到 IV)，剩余部分被舍弃。
- 现在已经得到了所需的4个密钥，可以用以保护传输的数据。
- TLS 使用应用数据协议来进行加密数据的传输。

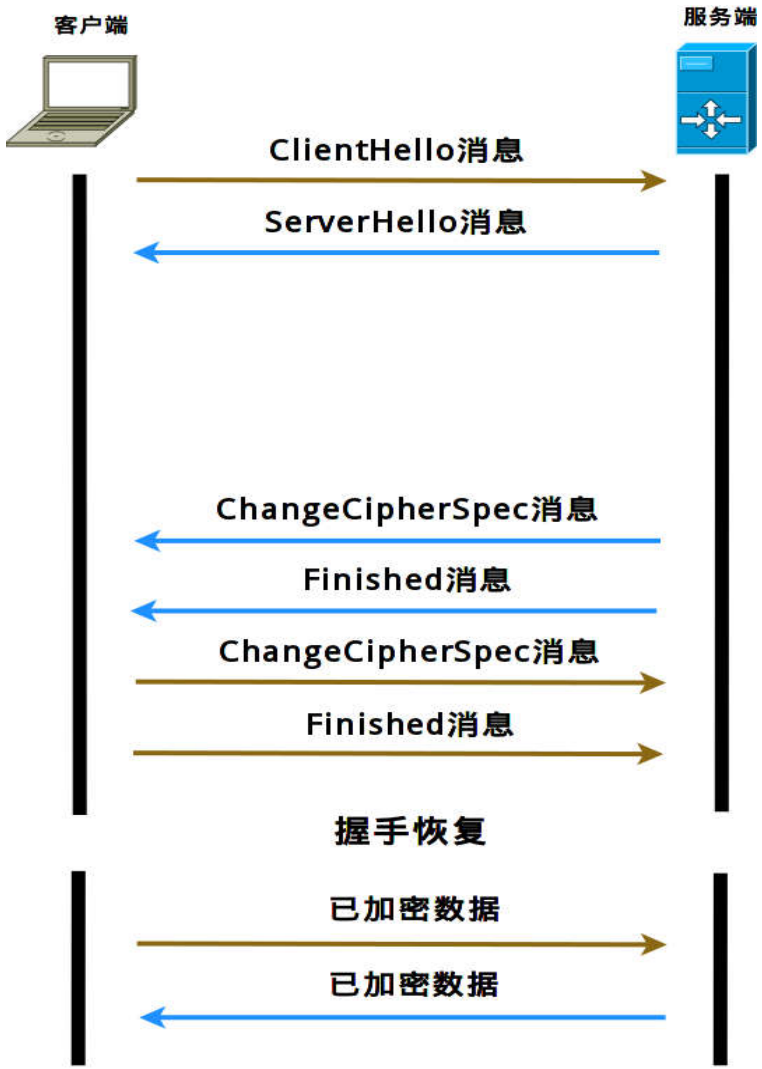
Application Data Protocol

- 类型为23的 TLS 记录层协议就是应用数据协议，结构如下：

0			**5**
23 (Content Type)	主版本号 Major version	子版本号 Minor version	长度 Length
数据内容 (已加密)			
MAC (已加密)			

Resume of TLS Handshake

- ClientHello
- ServerHello
- ChangeCipherSpec
- Finished
- ChangeCipherSpec
- Finished



Resume of TLS Handshake

- TLS 握手过程比较繁复，所以 TLS 提供了会话恢复机制以避免每次进行信息交换都要重复握手。
- 客户端：ClientHello
  - 客户端发送会话请求，与首次握手类似，不同的是客户端会在会话 ID 字段填上首次握手建立的会话 ID。
- 服务端：ServerHello
  - 服务端回复请求，确认会话 ID 的有效性。
- 服务端：ChangeCipherSpec
  - 服务端发送 ChangeCipherSpec 消息来重新激活自己的写模式选项，并通知客户端重新激活它的读模式。
- 服务端：Finished
  - 服务端告知客户端握手成功。

Resume of TLS Handshake

- 客户端：ChangeCipherSpec
  - 客户端发送 ChangeCipherSpec 消息来重新激活自己的写模式选项，并通知服务端重新激活它的读模式。
- 客户端：Finished
  - 客户端告知服务端握手成功。
- 上述过程可见，恢复握手时省略了密钥协商部分，直接使用以前协商的密钥和算法，这样可以减少密钥计算的消耗。

Decryption of TLS Packet

- Reading …

VPN

- 虚拟私有网络
- IPsec VPN
  - IPsec VPN 利用 IPsec 架构保护私有网络的通信，从而在逻辑上将私有网络与 Internet 隔离。
  - 基于 GNU/Linux 提供 IPsec VPN 服务的软件
    - FreeS/WAN
    - Openswan

OpenVPN

- an open source software application under the GNU General Public License (GPL) by *James Yonan*.
- implementing virtual private network (VPN) techniques for creating secure point-to-point or site-to-site connections in routed or bridged configurations and remote access facilities.
- using a custom security protocol that utilizes SSL/TLS for key exchange
- capable of traversing Network Address Translators (NATs) and firewalls
- OpenVPN allows peers to authenticate each other using a pre-shared secret key, certificates, or username/password. When used in a multi-client-server configuration, it allows the server to release an authentication certificate for every client, using signature and Certificate authority. It uses the OpenSSL encryption library extensively, as well as the SSLv3/TLSv1 protocol, and contains many security and control features.

OpenVPN

- OpenVPN 工作在传输层，借助的是 SSL 协议。
- 在使用 OpenVPN 服务时，需要建立一张虚拟网卡。
- OpenVPN 程序作为一个中间者，负责协调虚拟网卡和物理网卡之间的数据转换，所有接收到的数据都会先经过物理网卡，而所有发送的数据最终由物理网卡出去。

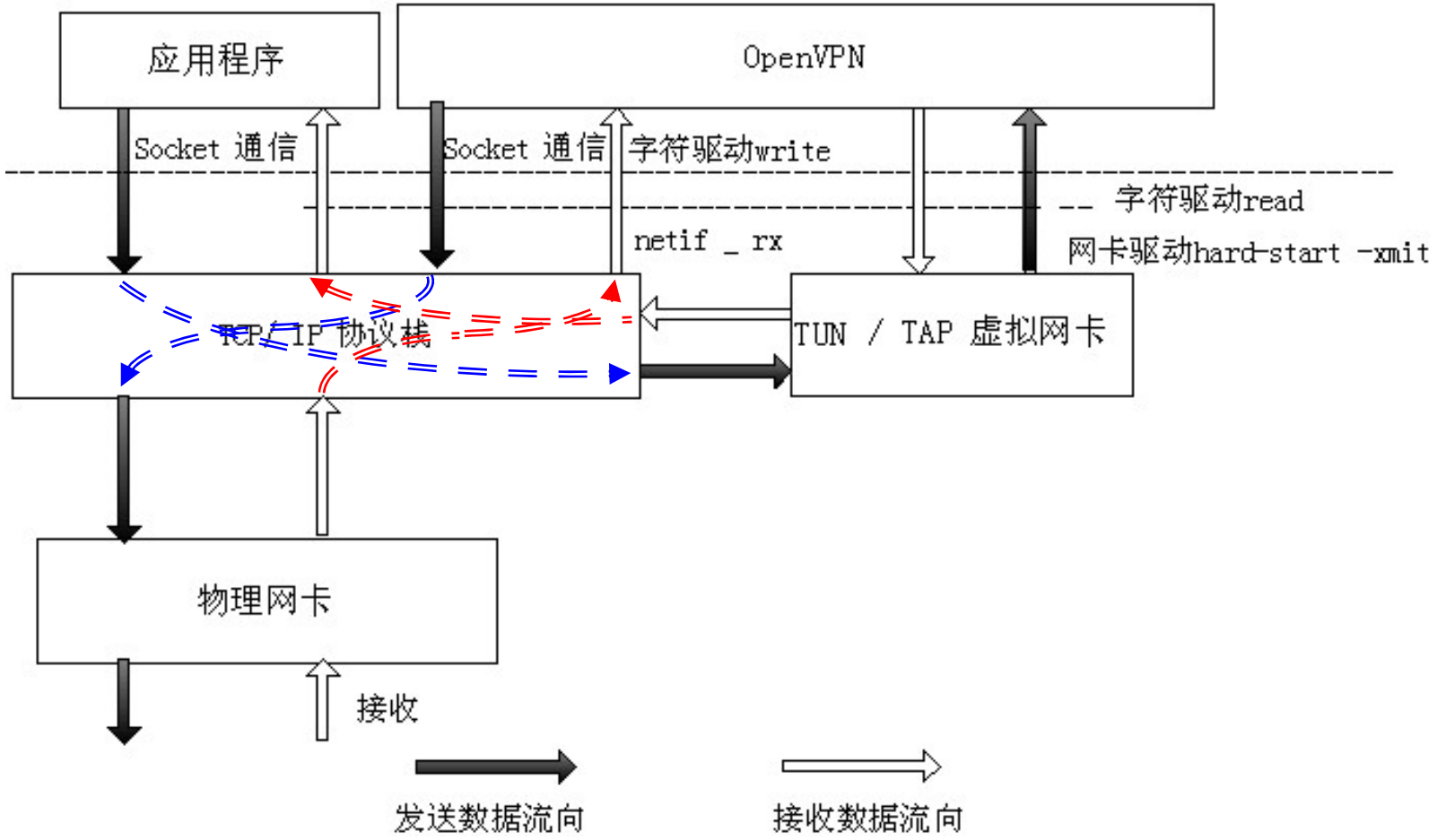
OpenVPN

- 发送数据流程：
  - 应用程序将数据发送到 TCP/IP 协议栈；
  - TCP/IP 协议栈将数据发送到虚拟网卡；
  - OpenVPN 检查到达虚拟网卡的数据，将数据取出作加密保护；
  - OpenVPN 将保护好的数据发送至 TCP/IP 协议栈；
  - TCP/IP 协议栈将数据发送给物理网卡；
  - 物理网卡收到要发送的数据并真正将数据发出。

OpenVPN

- 接收数据流程：
  - 物理网卡接收到传来的数据，交付 TCP/IP 协议栈；
  - OpenVPN 通过 Socket 通信检查接收到的数据，如果数据被加密过，则进行处理，否则抛弃；
  - OpenVPN 将解密后的明文报文发送至虚拟网卡；
  - 虚拟网卡将明文报文发送至 TCP/IP 协议栈；
  - 应用程序通过 Socket 从 TCP/IP 协议栈中读取信息。

OpenVPN


应用程序
OpenVPN
Socket 通信
Socket 通信
字符驱动write
字符驱动read
netif _ rx
网卡驱动hard-start -xmit
TCP/IP 协议栈
TUN / TAP 虚拟网卡
物理网卡
接收
发送数据流向
接收数据流向