信息安全基础

（二）网络安全
——传输层安全协议

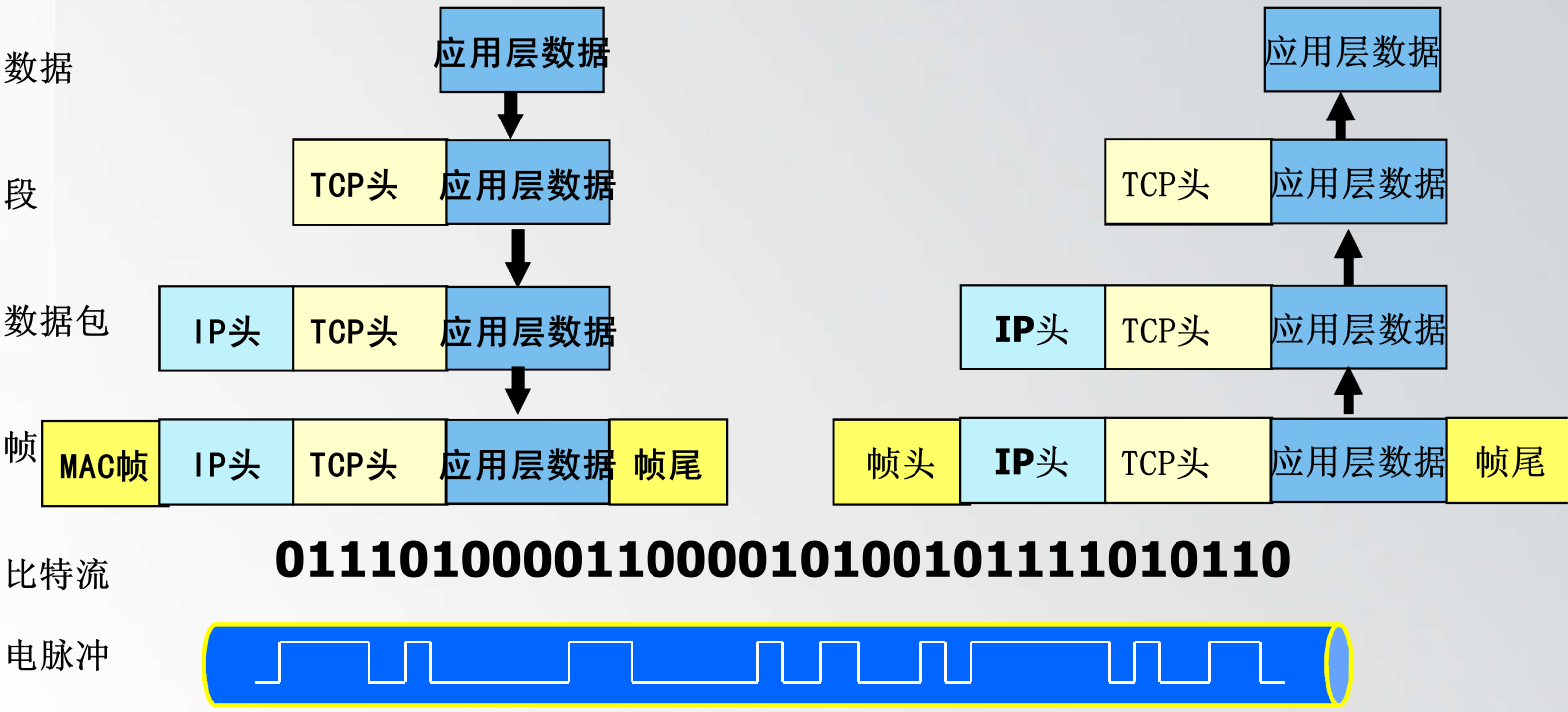
安全协议

1 SSL/TLS——传输层

2 IPSEC——网络层

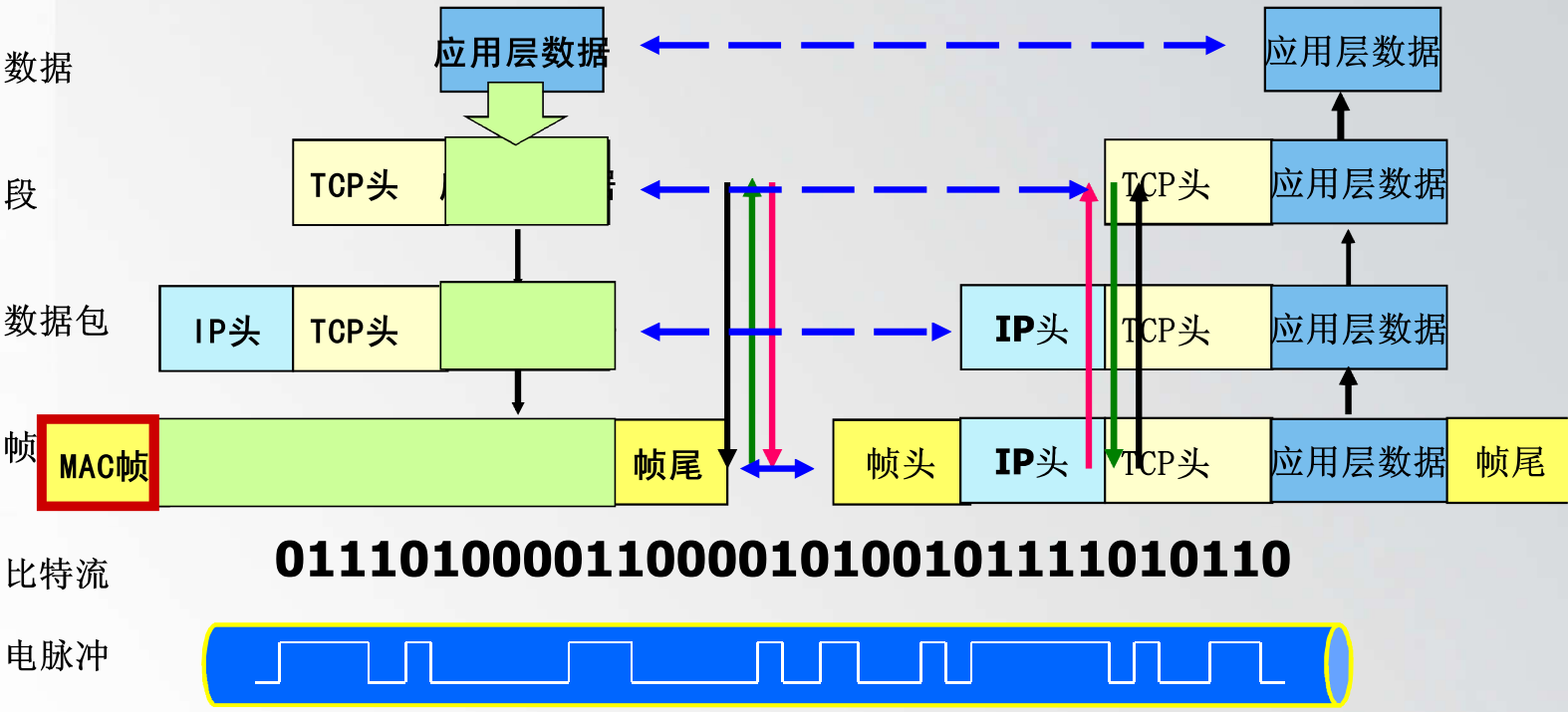
3 应用层安全（WEB、远程登录、邮件）

回顾：
网络通信实质上就是在两端进行数据封装/拆封



回顾：

每层协议并不一定是立即封装数据的，前期有一些控制处理



TCP/IP协议栈

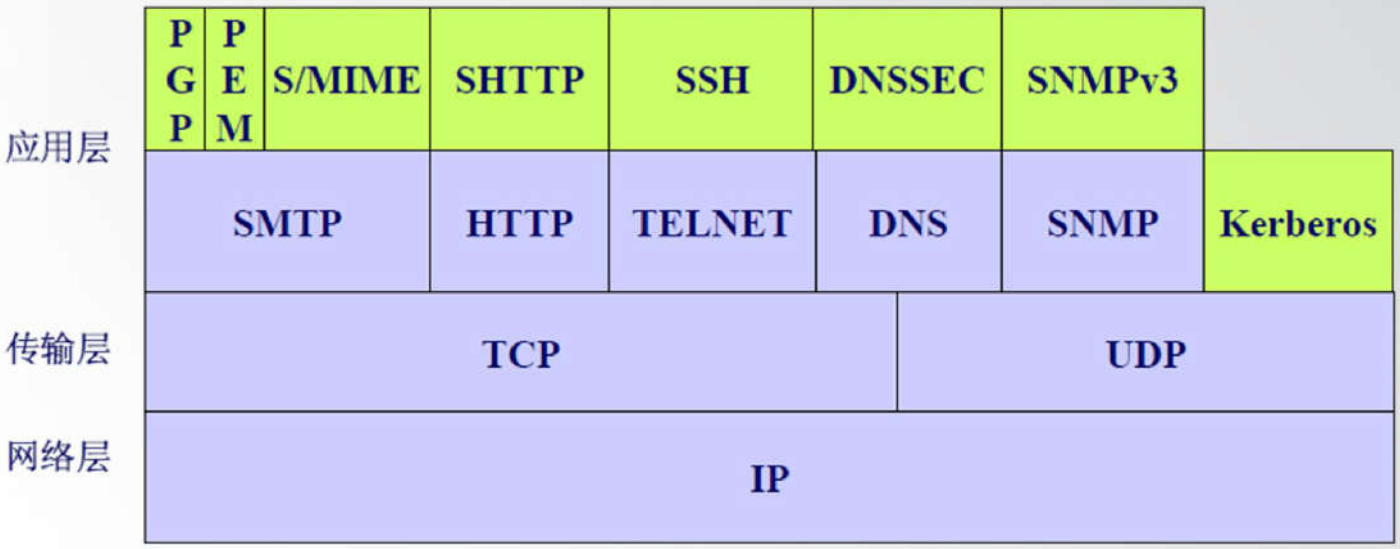
<table>
<tr><td>应用层</td><td>SMTP</td><td>HTTP</td><td>TELNET</td><td>DNS</td><td>SNMP</td></tr>
<tr><td>传输层</td><td colspan="3">TCP</td><td colspan="2">UDP</td></tr>
<tr><td rowspan="3">网络层</td><td colspan="2">ICMP</td><td colspan="3">IGMP</td></tr>
<tr><td colspan="5">IP</td></tr>
<tr><td colspan="2">ARP</td><td colspan="3">RARP</td></tr>
<tr><td colspan="6">网络接口层</td></tr>
</table>

Internet安全性途径

- 应用层——**S/MIME, PGP, PEM, SET, Kerberos, SHTTP, SSH**
- 传输层—— **SSL / TLS**
- 网络层——**IP** 安全性**(IPSec)**

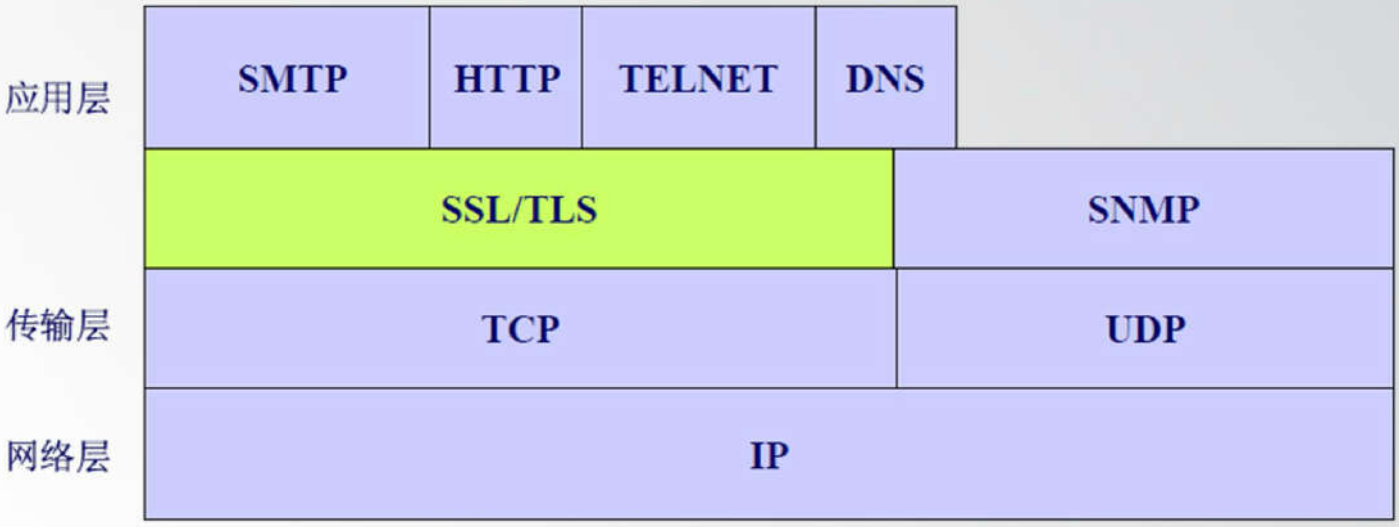
应用层安全

- 将安全服务嵌入在应用程序中（不透明），在相应的应用中提供机密性、完整性和不可抵赖性等安全服务，从而在应用层实现通信安全
  - 这一层协议较多，如**S/MIME**、**PGP**是用于安全电子邮件的一种标准。 还有安全交易协议**SET**（**Secure Electronic Transaction**）等等；



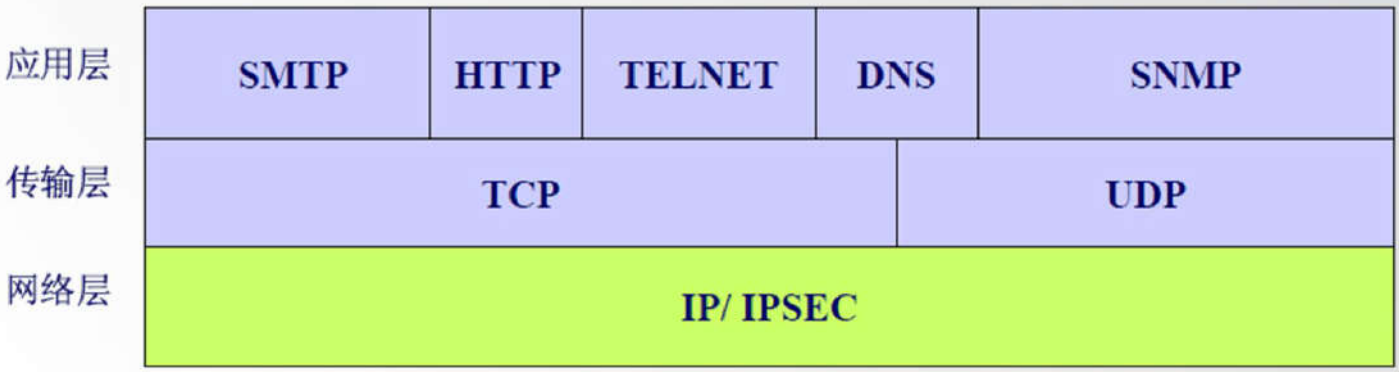
传输层安全

- 在传输层之上实现数据的安全传输是另一种安全解决方案。一般SSL都是可执行协议软件包的一部分，从而对应用是透明的，也可以嵌入到特殊软件包中（IE浏览器都配置了SSL）。



网络层安全

- **IPSec**可提供端到端的安全性机制，可在网络层上对数据包进行安全处理。
  - 可在路由器、防火墙、主机和通信链路上配置，实现端到端的安全、虚拟专用网络和安全隧道技术等。对终端用户和应用是透明的，提供了一种一般性的解决方案。



Web安全概述

- Web是一个运行于Internet和TCP/IP Intranet之上的基本的Client/Server应用。Web安全性涉及前面讨论的所有计算机与网络的安全性内容。
  - Web具有双向性，Web Server易于遭受来自Internet的攻击，
  - 实现Web浏览、配置管理、内容发布等功能的软件异常复杂，其中隐藏许多潜在的安全隐患。
  - 按web威胁发生的地域可分为三类：web服务器、浏览器、web流量。前两者可归为计算机系统安全，流量安全可归为网络安全，本部分主要讨论后者。

安全协议

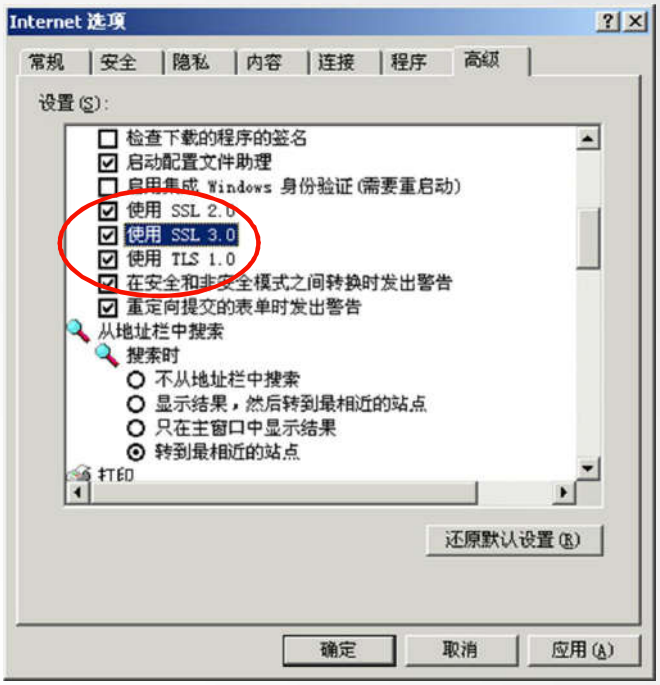
1 SSL/TLS

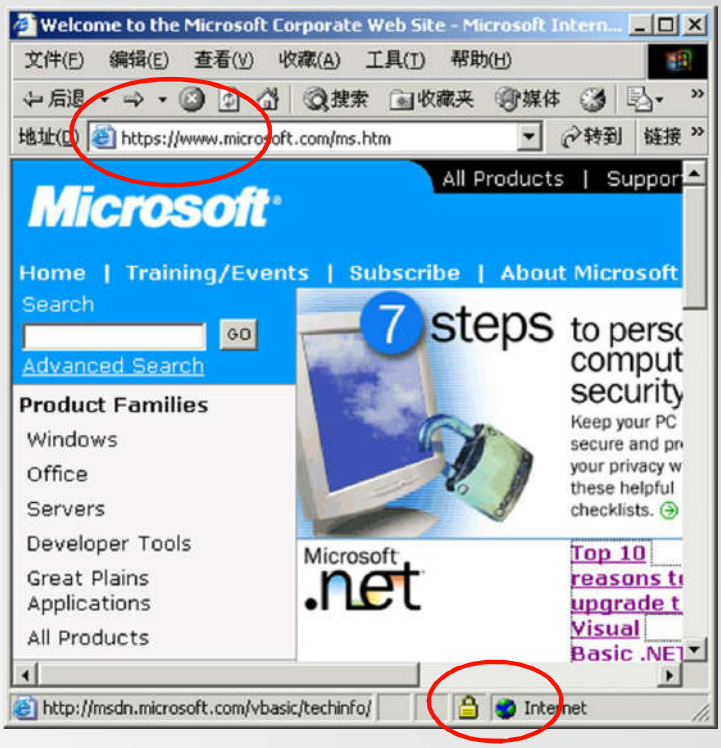
SSL/TLS 协议

- 1994年Netscape开发了安全套接层协议SSL(Secure Socket Layer)，专门用于保护Web通讯
- 版本和历史
  - 1.0，不成熟
  - 2.0，基本上解决了Web通讯的安全问题
    - Microsoft公司发布了PCT(Private Communication Technology)，并在IE中支持
  - 3.0，1996年发布，增加了一些算法，修改了一些缺陷
  - IETF内部成立传输层安全（TLS）工作组制定通用标准时，达成一致选择SSL。TLS 1.0(Transport Layer Security, 也被称为SSL 3.1)，1997年IETF发布Draft，同时，Microsoft宣布放弃PCT，与Netscape一起支持TLS 1.0。1999年正式发布RFC 2246(The TLS Protocol v1.0)。

SSL/TLS 协议

- 协议的设计目标
  - 为两个通讯个体之间提供保密性和完整性(身份认证)
  - 考虑互操作性、可扩展性、相对效率等
- 协议的使用





SSL体系结构

- SSL协议的目标就是在通信双方利用加密的SSL信道建立安全的连接。它不是一个单独的协议，而是两层协议
  - 一次会话从握手开始，协商双方密钥等参数信息；
  - 握手完成时用密码变更规则协议发送密码变更规格消息；
  - 开始利用记录协议封装应用数据进行通信；
  - 通信中出现错误时，利用报警协议进行协调。

<table>
<tr><td>SSL握手协议</td><td>SSL更改密码规则协议</td><td>SSL警报协议</td><td>HTTP</td></tr>
<tr><td colspan="4">SSL记录协议</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

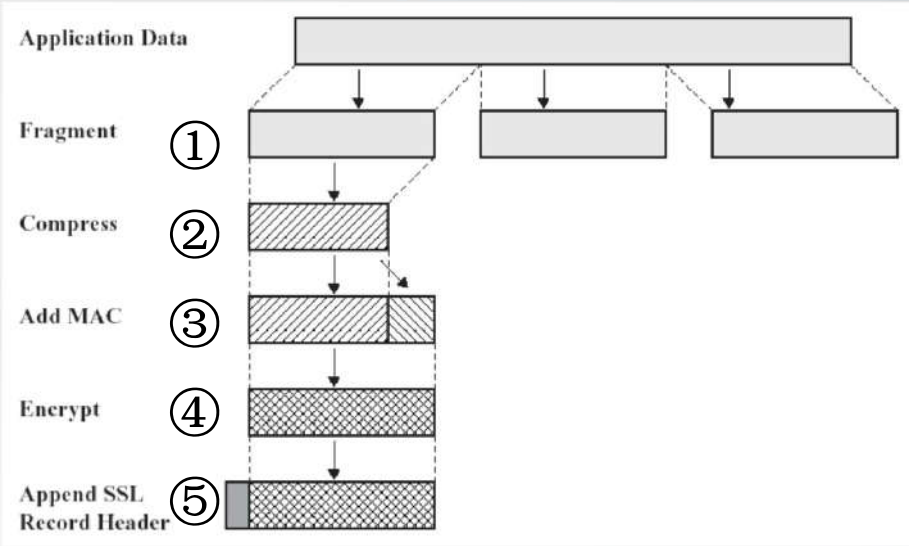
SSL两个主要的协议

- SSL记录协议
  - 建立在可靠的传输协议(如TCP)之上
  - 它提供连接安全性，有两个特点
    - 保密性，使用了对称加密算法
    - 完整性，使用HMAC算法
  - 用来封装高层的协议
- SSL握手协议
  - 客户和服务器之间相互鉴别
  - 协商加密算法和密钥
  - 它提供连接安全性，有三个特点
    - 身份鉴别，至少对一方实现鉴别，也可以是双向鉴别
    - 协商得到的共享密钥是安全的，中间人不能够知道
    - 协商过程是可靠的

1）SSL记录协议

- ①fragmentation
  - 上层消息的数据被分片成2^{14}字节大小的块，或者更小
- ②compression(可选)
  - 必须是无损压缩。有时候数据量很少，压缩反而会增加数据，则增加部分的长度不超过1024字节
- ③MAC 计算：
- ④ 加密

  可供选择的加密算法
  - IDEA 128 RC-40 40
  - RC2-40 40 RC4-128 128
  - DES-40 40
  - DES 56
  - 3DES 168
  - Fortezza 80
- ⑤最后添加SSL控制头



SSL记录格式

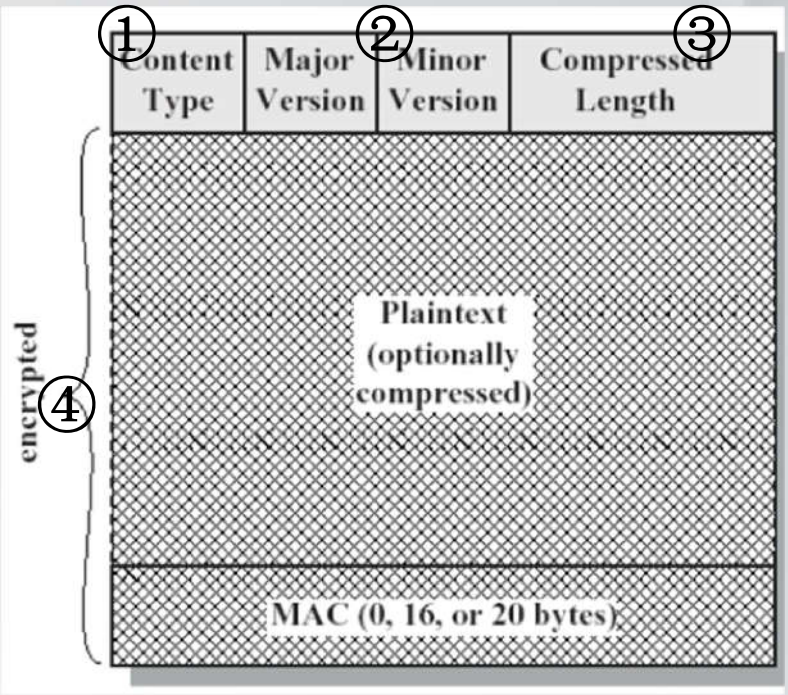
① **ContentType**—— 8位，上层协议类型

【4种】：密码变更规格、报警、握手、应用数据

② **Major version Minnor version**——16位，主次版本

③ 压缩长度：16位— 加密后数据的长度，不超过**214+2048**字节

④ **EncryptedData fragment;** —— 密文数据



SSL的有效载荷——封装的数据

SSL握手协议	SSL更改密码规则协议	SSL警报协议	HTTP
SSL记录协议			

有效载荷——即SSL记录里封装的上层数据有【4种】

①Change Cipher Spec

- 只有1字节，值为1。用以将握手时的延迟状态转至当前状态，更新了在当前连接上的密码机制。

②Alert

- 2字节。第一个字节表明报警的等级（1——warning；2——fatal）；第二个字节则是具体警告的编码。

③handshake

- 1字节（握手消息的类型）+3字节（握手消息的长度）+n字节（本条握手消息的相关参数）

④应用数据

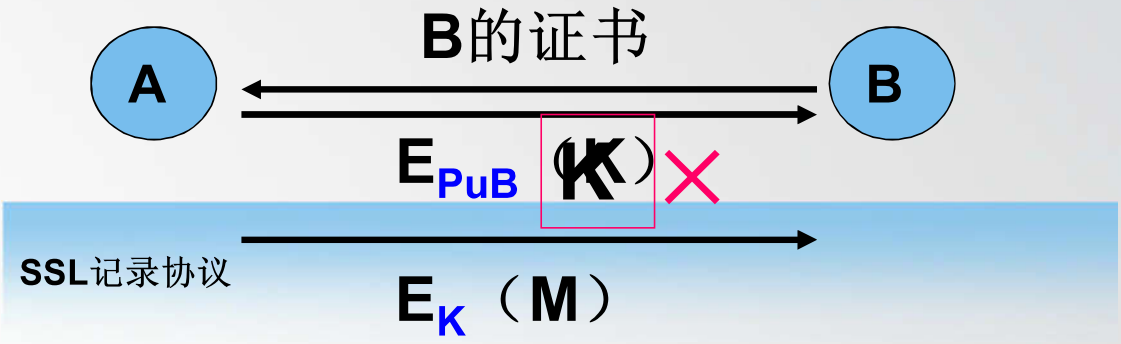
- 上层应用数据

2）SSL握手协议

- SSL握手协议在记录协议之上，记录协议是一种保密的数据封装，而记录协议加密用的密钥来自握手协议。

  问：SSL握手消息是明文还是密文？

  - SSL握手消息前期是明文，握手协商过程中逐渐变为密文。SSL记录协议有了密钥后才能开始密文封装。



握手协议使用的消息

消息	参数
hello_request	Null
client_hello	版本，随机数，会话id，密码参数，压缩方法
server_hello	
certificate	X.509 v3证书链
server_key_exchange	参数，签名
certificate_request	类型，CAs
server_done	Null
certificate_verify	签名
client_key_exchange	参数，签名
finished	Hash值

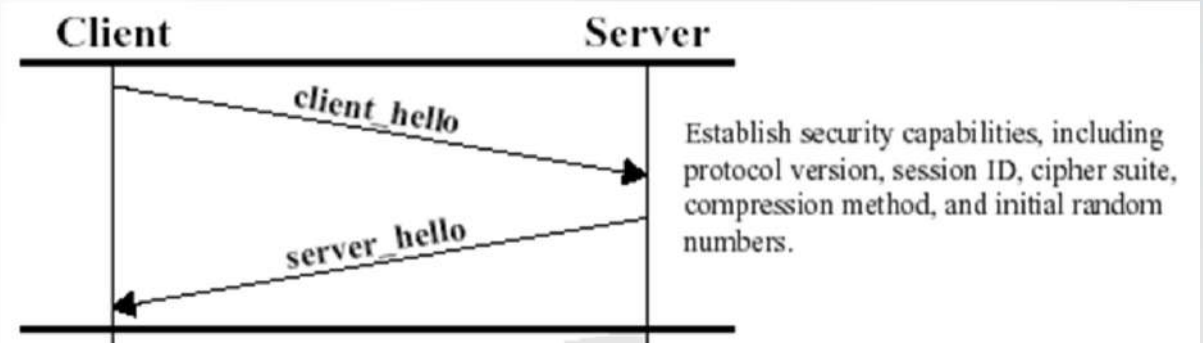
SSL握手协议的流程

第一阶段：建立起安全协商

①客户发送一个client_hello消息：

- 版本（version）：客户端SSL最高版本
- 随机数（random）：32位时间戳+28字节随机序列
- 会话ID（session ID）：一个会话状态对应一组参数
- 密码构件（CipherSuite）：客户支持的密码算法列表，包括密钥交换算法和密码算法等。以优先选用递减顺序给出。
- 压缩方法（Compression Mehod）：客户支持的压缩方法表

然后，客户等待服务器的server_hello消息



②服务器发送server_hello消息：

包括客户建议的低版本以及服务器支持的最高版本、服务器产生的随机数、会话ID、密钥构件域、压缩域

- 密钥构建域（服务器从客户建议的密码算法中挑出一套，包括包以下内容）

①密钥交换方法：RSA、固定/暂态/匿名Diffie-Hellman、以及Fortezza方法

②CipherSpec：密码算法、MAC算法、密码类型（流/分组）、可否出口、散列码大小、密钥材料（用于产生写密钥的数据）、IV大小

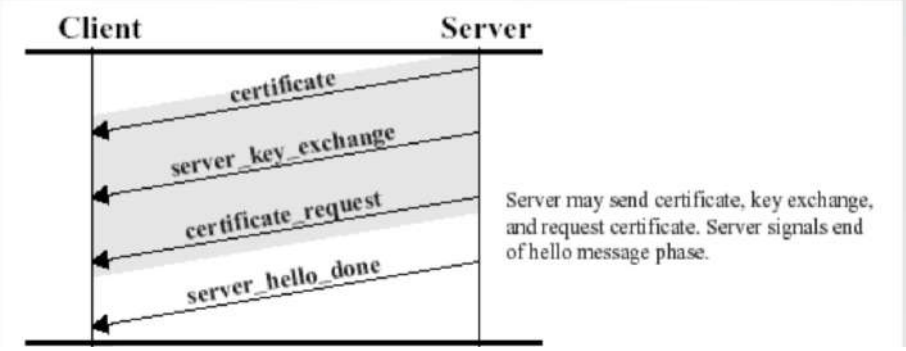
- 压缩域包括

（服务器从客户建议的压缩方法中挑出一个）

SSL握手协议的流程

第二阶段：服务器鉴别和密钥交换

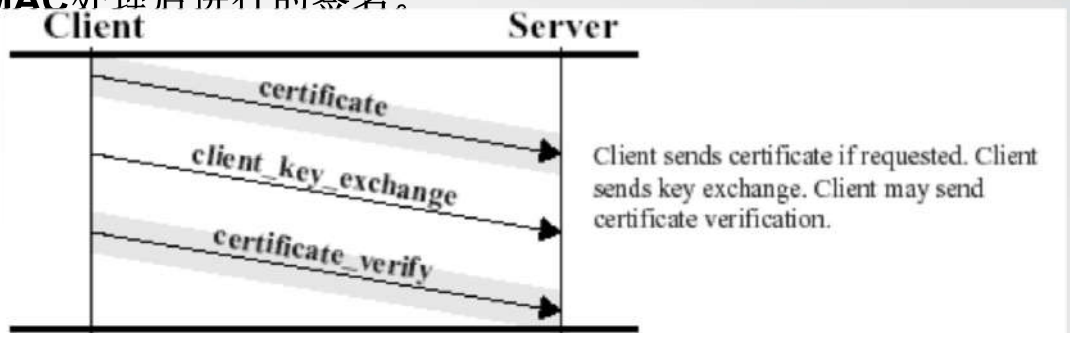
- 以服务器发送自己的证书为标志，开始第二阶段，该消息包含一个**X.509**证书，或者一条证书链（可选的，除匿名**Diffie-Hellman**交换之外的密钥交换方法都需要证书）
- 服务器发送**server_key_exchange**消息
  - 消息包含签名，被签名的内容包括两个随机数以及服务器参数。
  - 可选的，只有当服务器的证书没有包含必需的数据的时候才发送此消息
- 服务器发送**certificate_request**消息
  - – 可选的，非匿名**server**可以向客户请求一个证书
  - – 包含证书类型和**Cas**（可接收的认证机构名称列表）
- 服务器发送**server_hello_done**, 然后等待应答



SSL握手协议的流程

第三阶段：客户鉴别和密钥交换

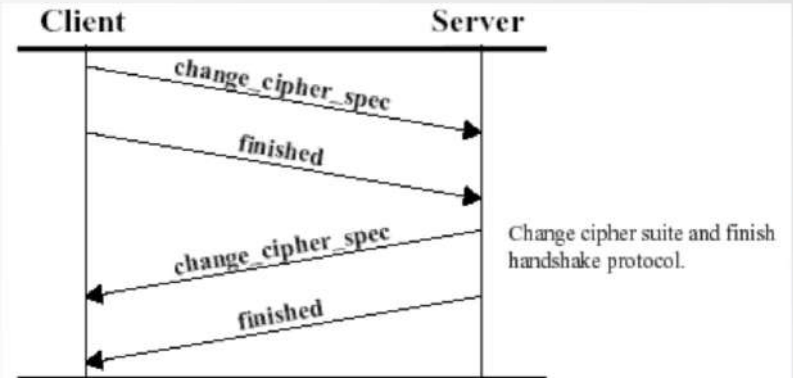
- 客户收到**server_done**消息后，它根据需要检查服务器提供的证书，并判断**server_hello**的参数是否可以接受，如果都没有问题的话，发送一个或多个消息给服务器
- 如果服务器请求证书的话，则客户首先发送一个**certificate**消息，若客户没有证书，则发送一个**no_certificate**警告
- 然后客户发送**client_key_exchange**消息，消息的内容取决于密钥交换的类型（若是**RSA**方式，客户端产生一个**48**字节的预备主密钥**Premaster key**）
- 最后，若客户端证书具有签名功能，发送一个**certificate_verify**消息，该消息是对客户端启动**client_hello**后发送或接收到的所有握手协议消息填充**MAC**处理后进行的签名。



SSL握手协议的流程

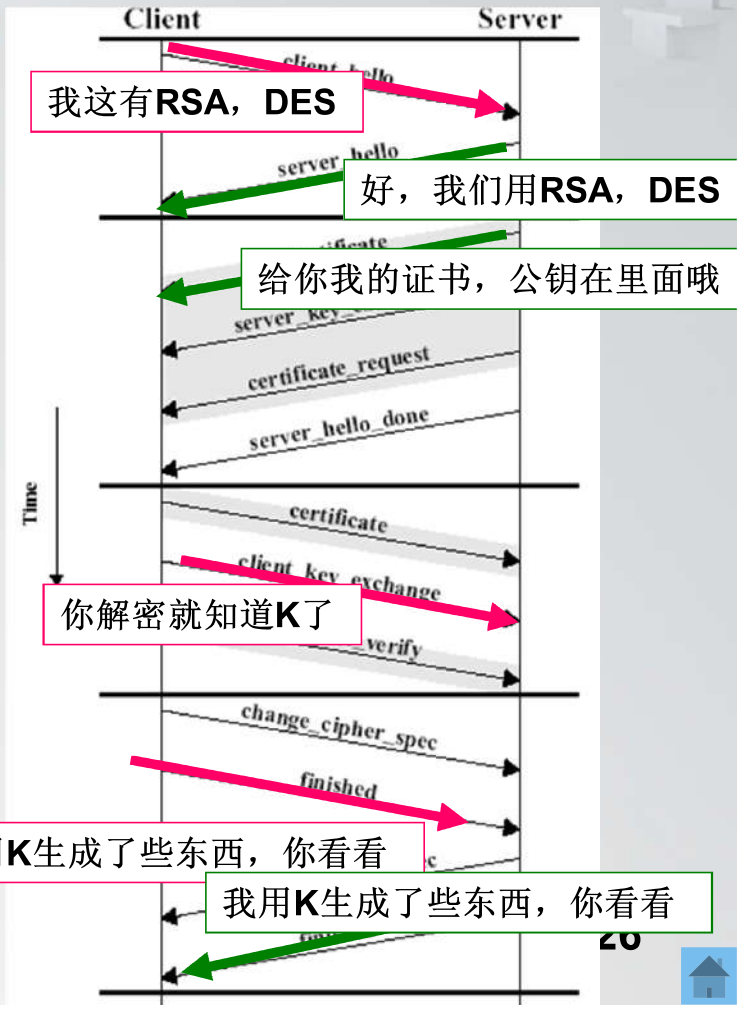
第四阶段：结束

- 第四阶段建立起一个安全的连接
- 客户发送一个**change_cipher_spec**（密码变更规格）消息，并把协商得到的**CipherSuite**拷贝到当前连接的状态之中
- 客户用新的算法、密钥参数发送一个**finished**消息，包括一个校验值，对所有握手以来的消息（不包括本消息）进行校验。该消息可以检查密钥交换和鉴别过程是否已经成功。
- 服务器同样发送**change_cipher_spec**消息和**finished**消息。
- 握手过程完成，客户和服务器可以交换应用层数据。



如何协商实现保密通信的

- **第一阶段**，设**client_hello**给服务器提供的密钥交换方法有**RSA**方式：即用接收者公钥加密密钥。服务器也选择并回送**server_hello**。
- **第二阶段**，服务器传递自己的证书（这以后就可以单方向加密了）
- **第三阶段**，客户端的密钥交换消息中就可以用服务器的公钥将生成的密钥（预备主密钥**Kpre**）加密后发给服务器。（至此，双方都掌握了一个密秘信息K。这以后就可双方加密）
- **第四阶段**，发送密码规格变更消息，双方确认一下。



关于双方共享的密钥

- 加密、鉴别很多地方需要密钥和秘密初始向量等，一个秘密**Kpre**是不够的。
- 第三阶段双方都知道**Kpre**后，各自会根据**Kpre**利用**SSL**设定的算法计算得到主密钥。然后按顺序从主密钥生成客户端写**MAC**密钥，服务器写**MAC**密钥，客户端写密钥，服务器端写密钥、客户端写初始向量**IV**，服务器端写初始向量**IV**。

如何实现双方消息和身份鉴别的

除匿名D-H交换方式，web服务和client都需要给对方发送证书。

①如何信任消息不是冒充的实体发的？

- 接收者信任权威机构颁发的证书中的公钥
- 消息用私钥签名，用户可以用其公钥认证实体

②双方互相发送的消息如何鉴别是完整的？

- 第二阶段，服务器发送的密钥交换消息是被签名的，被签名数据是客户端随机数、服务器端随机数及服务器参数的哈希值。没有服务器私钥，很难伪造这种消息；消息中使用了随机数，对随机数使用合适的话能防止消息被重放。
- 第三阶段，客户端同样会发送被签名的消息，该消息中包含计算得到的主密钥、以及全部握手协议消息的哈希值。
- 第四阶段，双方用都知道的主密钥、发送者标识、握手以来的全部消息计算散列值，互相验证。

思考整个过程

- 一次会话从握手开始，协商双方密钥等参数信息；
- 握手完成时用密码变更规则协议发送密码变更规格消息（双方都把会话、链接相关的参数存储在合适的位置）；
- 开始利用记录协议封装应用数据进行通信；
- 通信中出现错误时，利用报警协议进行协调。

SSL的两个重要概念

- **SSL会话（session）**
  - 一个SSL会话是在客户与服务器之间的一个关联。会话由SSL握手协议创建。会话定义了一组可供多个连接共享的密码安全参数。
  - 会话用以避免为每一个连接提供新的安全参数所需昂贵的协商代价。
- **SSL连接（connection)**
  - 一个连接是一个提供一种合适类型服务的传输（OSI分层的定义）。
  - SSL的连接是点对点的关系。
  - 连接是暂时的，每一个连接和一个会话关联。

SSL会话

- SSL会话由握手协议创建，定义了一系列相应的安全参数，最终建立客户机和服务器之间的一个关联。对于每个SSL连接，可利用SSL会话避免对新的安全参数进行代码繁多协商。
- 每个SSL会话都有许多与之相关的状态。一旦建立了会话，就有一个当前操作状态。SSL会话状态参数包括：
  （1）会话标志符（Session Identifier）用来确定活动或可恢复的会话状态；
  （2）对等实体证书（Peer Certificate），是对等实体X.509 v3证书；
  （3）压缩方法（Compression Method）；
  （4）加密规范（Cipher Spec）包括加密算法DES，3DES和IDEA等、消息摘要算法MD5和SHA-1等，以及相关参数；
  （5）主密码（Master Secret），由客户机和服务器共享的密码；
  （6）是否可恢复（is Resumable）会话是否可用于初始化新连接的标志。

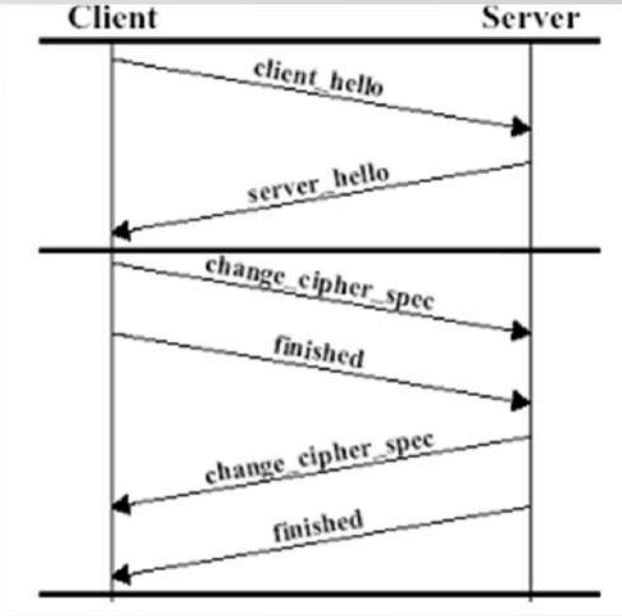
SSL连接

- SSL连接是一个双向连接，每个连接都和一个SSL会话相关。SSL连接成功后，可以进行安全保密通信。SSL连接状态的参数包括7个：
  （1）服务器和客户机随机数（Server and Client Random）：Server和Client 为每一个连接所选择的字节序列。
  （2）服务器写MAC秘密（Server Write MAC Secret）：一个密钥，用来对Serv
  （3）客户
  （4）服务
  （5）客户
  （6）初始化向量（Initialization Vectors）：当数据加密采用CBC方式时，每一个密钥保持一个IV。该字段首先由SSL Handshake Protocol，以后保留每次最后的密文数据块作为IV。
  （7）序列号（Sequence Number）：每一方为每一个连接的数据发送与接收维护单独的顺序号。当一方发送或接收一个改变的cipher spec message时，序号置为0, 最大$2^{64}-1$

总的来说，包括由主密码生成的子密码，用于在记录协议中生成MAC、加密；还包括若干随机数。

重用一个会话

- 客户方client_hello消息中指定会话ID（如35）
  - 服务器检查cache中的会话状态，若已有该会话，则基于该会话更新已有连接的安全参数；否则可新建一个连接；
- 客户方client_hello消息中指定会话ID为0
  - 表示客户希望在一个新的会话上建立一个新的连接，服务器返回一个新的会话ID。
- 重用会话可跳过第二第三阶段，直接把会话中的参数传递给记录层



client_hello消息中的会话ID (Session ID)

- client_hello消息中客户指定的会话ID非0，服务器采用相同的取值。一般表示基于一个会话更新已有连接的安全参数，或者创建一个新的连接。服务器同意客户指定的会话ID，检查cache中的会话状态。
- 如果会话ID等于0，则表示客户希望在一个新的会话上建立连接。服务器返回一个新的会话ID。

对SSL攻击的启示

- 历史上有不少针对**SSL**的攻击，有的并不是**SSL**协议本身的缺陷，而是实现上导致的缺陷，一些启示：
  - 随机数对于安全协议或者安全系统的重要性
  - 对待错误消息如何响应
    - **Contiune?** 会不会招致**DOS**？
    - 返回精确的错误容易被分析
  - 为防止明文模式被分析，可进行随机数填充

* TLS与SSL的区别

- 版本号
- 实际使用的MAC算法和计算范围
- 伪随机函数
- 除支持SSLv3中定义的所有报警码之外还定义了更多的报警码
- 密钥构件里不支持Fortezza
- 客户端证书类型有小不同
- 消息、密码计算、填充等处也有部分细节不同

OpenSSL
——开发自己的支持SSL的应用

- 目前实现SSL/TLS的软件虽然不多，但都很优秀。除了SSL标准提出者Netscape实现的，OpenSSL是一个非常优秀的实现SSL/TLS的开放源代码软件包，主要是作为提供SSL算法的函数库供其他软件调用而出现的，可给任何TCP/IP应用提供SSL功能。

- 1995年，Eric A. Young和Tim J. Hudson开始开发OpenSSL，后来不断发展更新，直到现在，SSL还在不断的修改和完善，新版本也不断的推出。最新的版本可以从OpenSSL的官方网站http://www.openssl.org下载。

SSL安全协议的应用

- SSL安全协议是国际上最早应用于电子商务的一种网络安全协议。目前，我国开发的电子支付系统未采用SSL协议，而是广泛使用SET协议。
  - SSL协议已被浏览器和WEB服务器内置，无需安装专门软件，可快速实现安全的web应用；
  - 但在电子商务领域，虽然在SSL3.0中可以通过数字签名和数字证书实现浏览器和Web服务器之间的身份验证，但仍不能实现多方认证，而且SSL中只有商家服务器的认证是必须的，客户端认证则是可选的。相比之下，SET协议的认证要求较高，所有参与SET交易的成员都必须申请数字证书，并且解决了客户与银行、客户与商家、商家与银行之间的多方认证问题。但SET协议中客户端需安装专门的电子钱包软件，在商家服务器和银行网络上也需安装相应的软件。

WEB安全相关的协议

传输层

- SSL\TLS

应用层

- SET
- HTTPS
- S-HTTP