第8章　网络应用的安全

8.1　认证应用
8.2　电子商务安全
8.3　E-mail的安全
8.4　DNS的安全性
8.5　网络管理安全

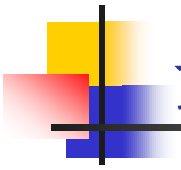
网络与信息安全的构成

- 物理安全性
  - 设备的物理安全：防火、防盗、防破坏等
- 通信网络安全性
  - 防止入侵和信息泄露
- 系统安全性
  - 计算机系统不被入侵和破坏
- 用户访问安全性
  - 通过身份鉴别和访问控制，阻止资源被非法用户访问
- 数据安全性
  - 数据的完整、可用
- 数据保密性
  - 信息的加密存储和传输

安全的分层结构和主要技术

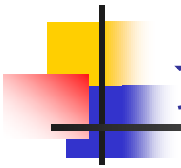
安全协议的含义

- 在网络协议中使用加密技术、认证技术等密码技术以保证信息交换安全的网络协议
- 具体地说就是建立在密码体系上的一种互通协议，为需要安全的各方提供一系列的密钥管理、身份认证及信息完整性等措施以保证通信的安全完成



安全协议的理解

- 首先是协议
  - 标准程序
  - 语义、语法规则
  - 双方或多方
  - 数据交换
- 安全协议是
  - 协议中与安全相关的部分
  - 同安全相关的协议

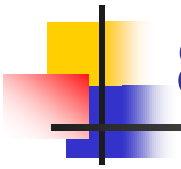


安全协议的基石

- 通信技术
- 密码技术
  - 对称密钥算法
  - 公开密钥算法
- 密码技术的应用
  - 保密：机密性
  - 认证：完整性和真实性
- 相关问题
  - 密钥的分发和交换

安全协议的种类

- 根据功能分
  - 密钥安全协议
    - 指参与协议的双方或多方之间建立的通信中的会话密钥
  - 认证协议
    - 主要用来在信息交换过程中防止信息的假冒、篡改或否认
  - 密钥交换和认证协议
    - 将密钥技术与认证技术相结合，同时完成信息的加密与认证的功能

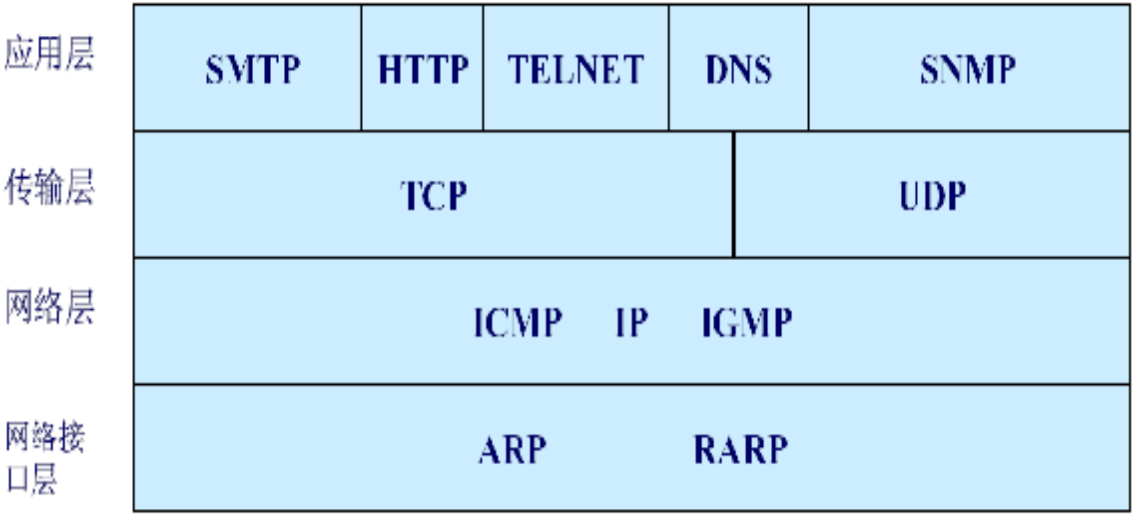


8.1 认证应用

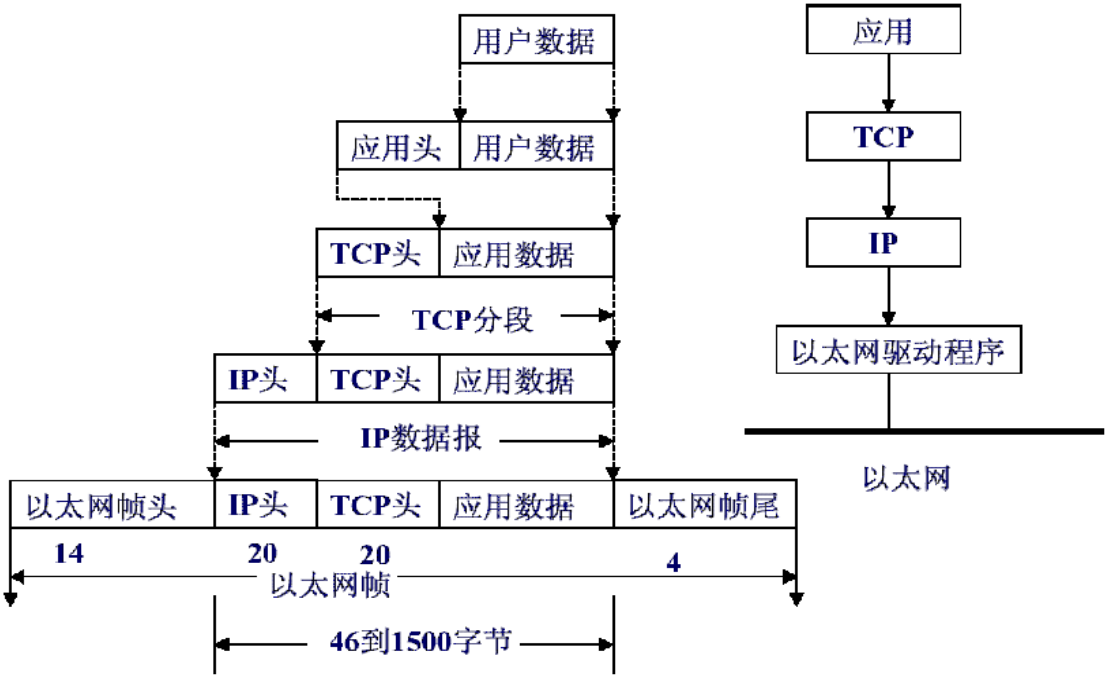
安全协议 – 标准

- **IPSec**
- **SSL**

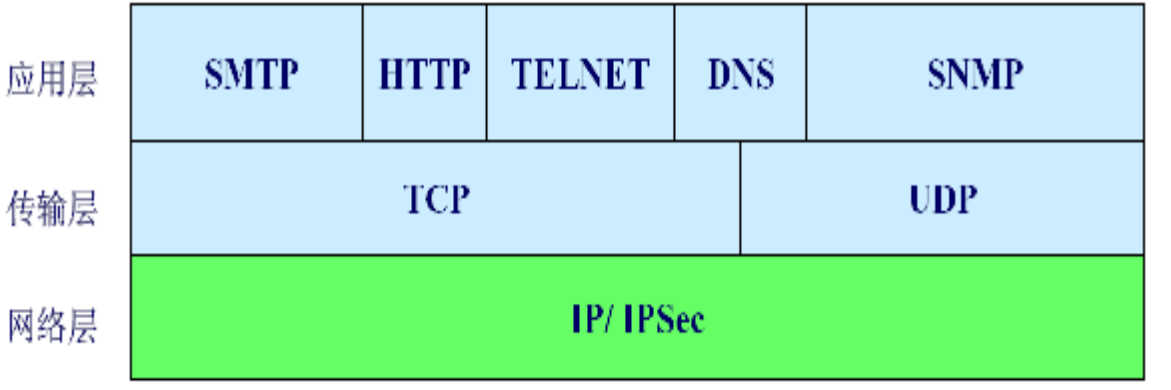
TCP/IP协议栈


应用层
SMTP
HTTP
TELNET
DNS
SNMP
传输层
TCP
UDP
网络层
ICMP
IP
IGMP
网络接口层
ARP
RARP

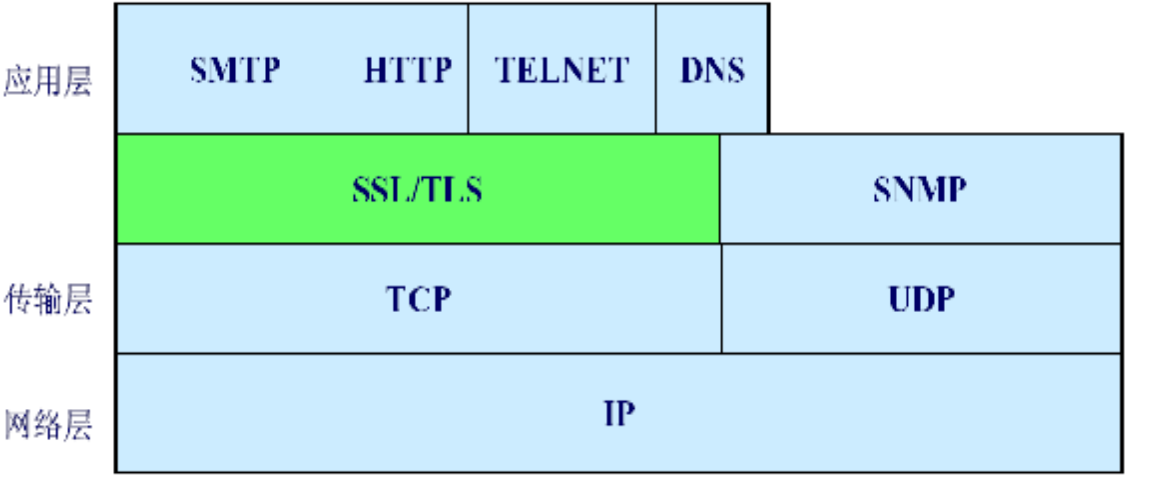
协议栈的封装过程


用户数据
应用头
用户数据
TCP头
应用数据
TCP分段
IP头
TCP头
应用数据
IP数据报
以太网帧头
IP头
TCP头
应用数据
以太网帧尾
14
20
20
4
以太网帧
46到1500字节
应用
TCP
IP
以太网驱动程序
以太网

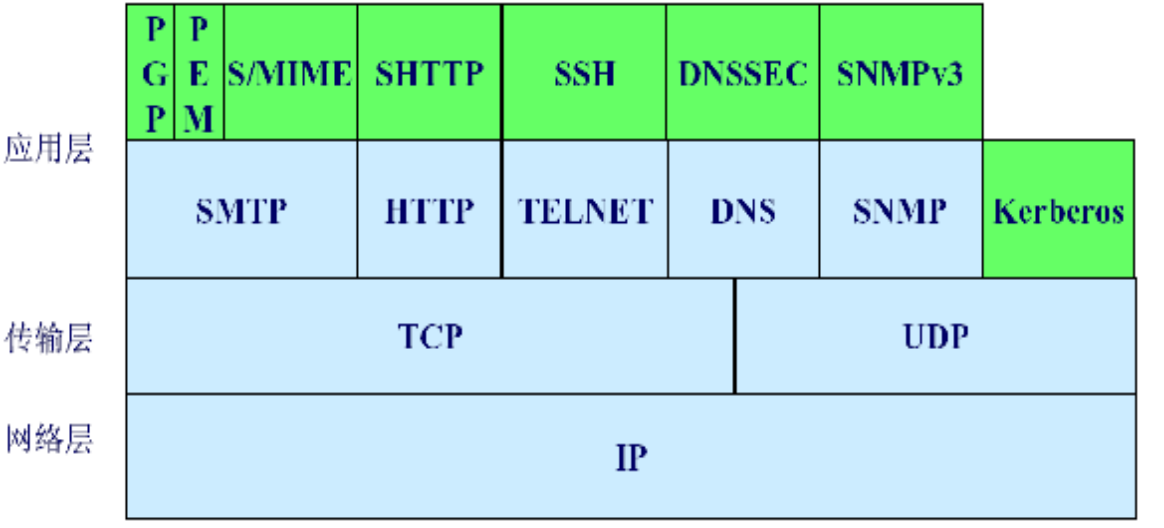
网络层安全



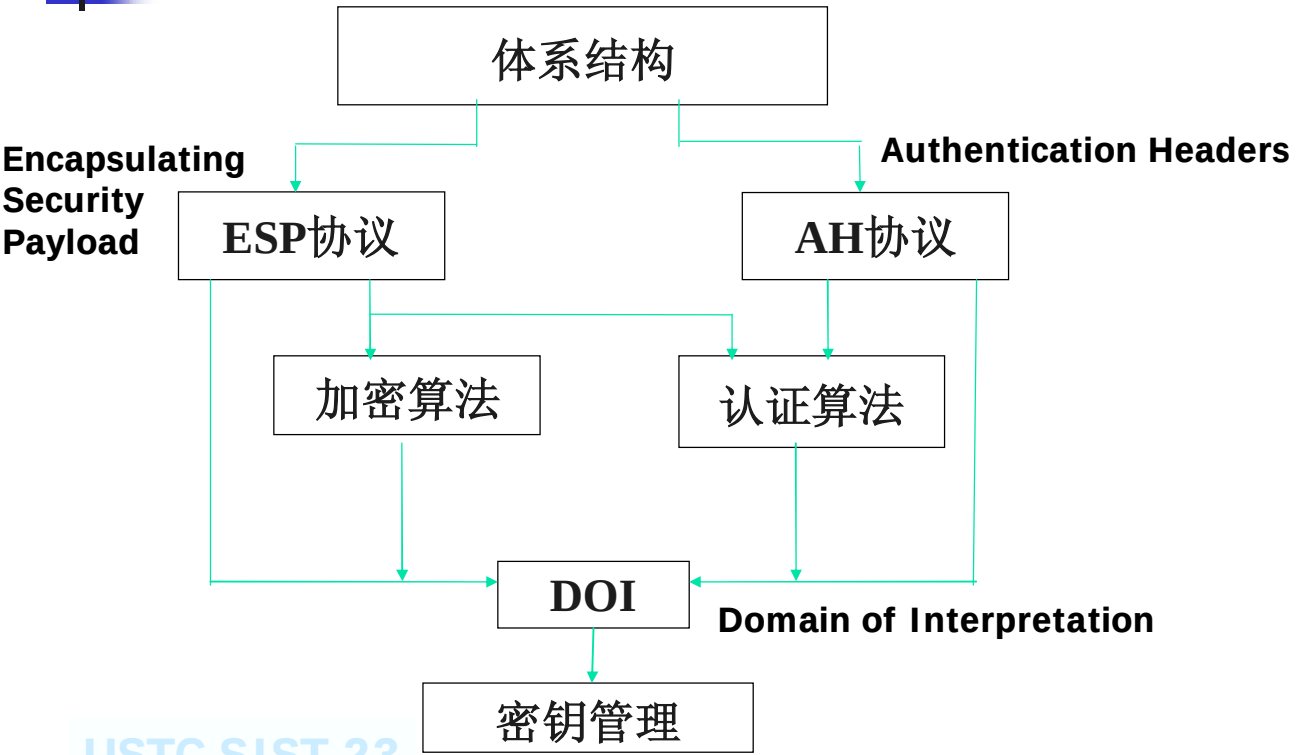
传输层安全



应用层安全

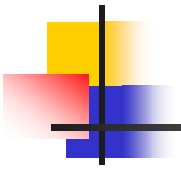

应用层
传输层
网络层
PGP
PEM
S/MIME
SHTTP
SSH
DNSSEC
SNMPv3
SMTP
HTTP
TELNET
DNS
SNMP
Kerberos
TCP
UDP
IP

IPSec框架


体系结构
Encapsulating Security Payload
ESP协议
Authentication Headers
AH协议
加密算法
认证算法
DOI
Domain of Interpretation
密钥管理

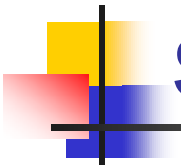
IPSec密钥管理

- 手工管理
- 自动管理
  - **Photuris -** 基于**Diffie-Hellman** 算法，要求客户首先发送一个**cookie (**一个随机数**)**，然后服务器给予应答，这样减少了否认服务攻击的威胁**(**否认服务攻击是由攻击者伪造源地址而导致的**)**。**Photuris**也要求通信各方都必须对协商好的密钥签名，以减少中间者攻击
  - 简单**Internet**密钥管理协议 **- SKIP** 以**Diffie-Hellman**密钥交换为基础，但是它并不要求通信各方使用随机数来计算其密钥，而是要求使用静态的密码表。各方查找密码表中的秘密值，然后基于查到的秘密值来计算，并传送所算出的值。
  - **Internet**密钥交换协议**- IKE**
- **IPSec**默认的协议
  - **IKE (Internet Key Exchange Protocol)**



IKE

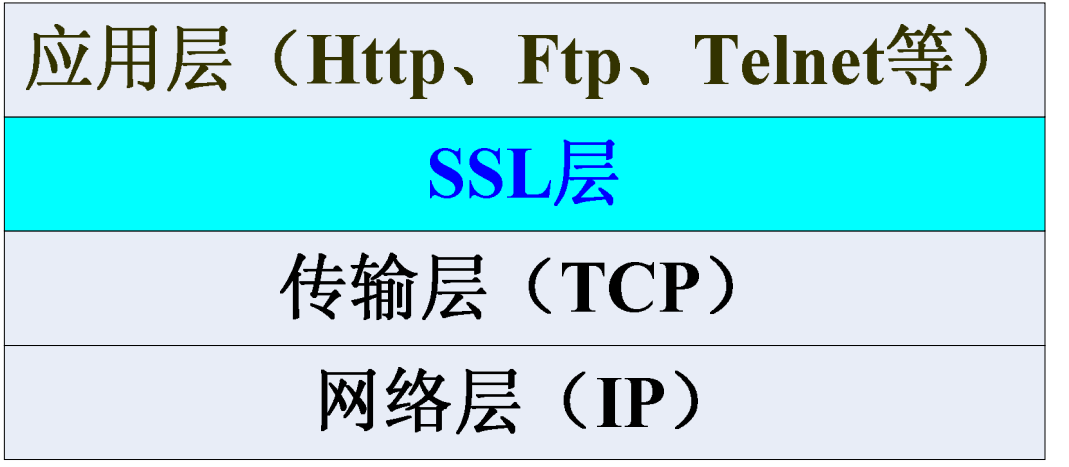
- **AH & ESP**
  - 安全关联**SA**和会话密钥建立以后的工作过程
- **IKE**
  - 用于双向认证、建立共享密钥和会话关联**SA**的协议
  - 包括三个部分
    - **IKE**
    - **ISAKMP （Internet Security Association and Key Management Protocol）**
    - **DOI** 定义**IKE**所没有定义的协商的内容



SSL简介

- SSL（Secure Socket Layer，安全套接层）协议
  - Netscape公司于1994年最先提出来的
  - 被设计成使用TCP来提供一种可靠的端到端的安全服务，是一种基于会话的加密和认证的协议
  - 在客户和服务器之间提供了一个安全的管道
    - 为了防止客户/服务器应用中的监听、篡改、消息伪造等，SSL提供了服务器认证和可选的客户端认证
    - 通过在两个实体间建立一个共享的秘密，SSL提供保密性
- SSL工作在传输层（TCP）和应用层之间
  - SSL独立于应用层协议，高层协议可基于SSL进行透明的传输

SSL协议的协议层次



SSL协议提供的主要服务

- 加密处理
  - 加密数据以防止数据中途被窃取；
- 维护数据的完整性
  - 确保数据在传输过程中不被改变。
- 认证服务
  - 认证用户（可选）和服务器，确保数据发送到正确的客户机（可选）和服务器；

SSL的两个重要概念 - 1

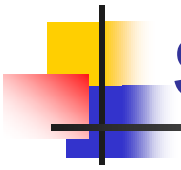
- SSL会话（session）
  - 一个SSL会话指客户与服务器之间的联系
  - 会话由SSL握手协议创建，定义了一套安全加密参数，为多个连接所共享
  - 握手协议的职责是协调客户和服务器的状态，使得双方在不能精确地并行的情况下，也能工作一致
- 会话状态
  - 会话标识：是由服务器选择的任意字节序列，用于标识活动的会话或可恢复的会话状态
  - Peer 证书：peer的x509证书
  - 压缩方法：在加密之前用来压缩数据的算法
  - 密码规范：加密算法；消息摘要算法；有关密码的一些属性
  - 主密钥：客户和服务器共享的主密钥
  - 恢复 ，可重新开始的标识：用一个标志位表示该会话是否可以初始化一个新的连接。

SSL的两个重要概念 - 2

- **SSL连接（connection)**
  - 用于实现特定类型的服务数据的安全传输
  - **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
  - 一个**SSL**会话可包括多种安全连接
- 连接状态
  - 服务器和客户端随机序列
  - 服务器端写摘要协议
  - 客户端写摘要协议
  - 服务器端写密钥：用于数据加密的密钥，长度为**128 bit**
  - 客户端写密钥：用于数据加密的密钥，长度为**128 bit**
  - 初始化向量：分组密文算法
  - 序列号：发送和接收的报文都维持着一个序列号

SSL安全连接的特性

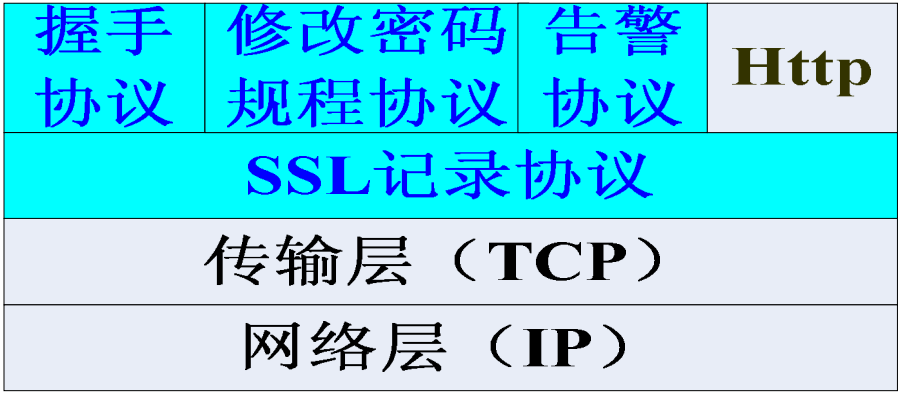
- 连接具有私有性
  - 在初始化连接后，协商密钥，基于对称密码体制对数据进行加密（如**DES**、**RC4**等）
- 对端实体鉴别
  - 可采用非对称密码体制或公开密钥密码体制（如**RSA**、**DSS**）
- 连接是可靠的
  - 消息传输使用加密**MAC**算法进行消息完整性检查



SSL安全协议

- 记录协议
- 修改密码规程协议
- 告警协议
- 握手协议

SSL安全协议的组成

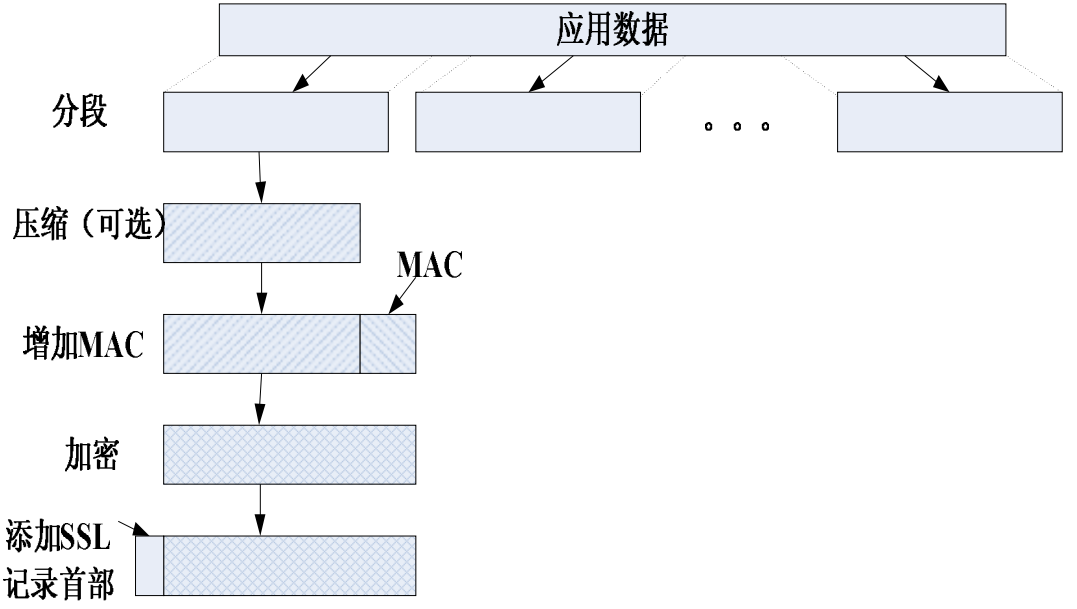


- SSL不是单个的协议，而是两层协议
  - SSL记录协议基于可靠的传输层协议，用来封装高层协议
  - 高层协议主要包括SSL握手协议、修改密码参数协议、报警协议、应用数据协议（如HTTP）等。

SSL记录协议层的功能

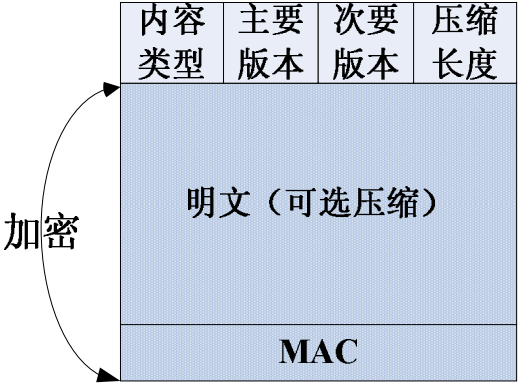
- 为高层协议提供基本的安全服务
- 记录层封装各种高层协议
  - 以任意大小的非空块从高层接收尚未解释的数据
  - 分块
    - 将高层数据分割成**SSL**明文记录
  - 压缩和解压
    - 所有记录采用会话状态中定义的压缩算法进行压缩
  - 记录的保护
    - 所有记录通过会话状态中定义的加密算法和**MAC**算法进行保护

SSL记录协议的操作过程



SSL记录首部的组成

- 主要类型：用来说明封装的数据段的更高层的协议。已定义的内容类型有：修改密码规程协议、告警协议、握手协议和应用数据；
- 主要版本：指示使用SSL的主要版本号；
- 次要版本：指示使用的次要版本号；
- 压缩长度：明文数据段以字节为单位的长度（如果使用压缩就是压缩数据段）。



握手协议

- 握手协议在SSL记录层之上，负责建立当前会话状态的参数
  - 客户端和服务器相互鉴别对方身份（利用证书）
  - 协商安全参数（加密和MAC算法，加密密钥等）
- 由一系列在客户和服务器间交换的消息组成
- 每个消息由3个字段组成：消息类型、以字节为单位的消息长度以及这个消息有关的参数内容

握手协议定义的消息类型 - 1

消息类型	说明	参数
hello_ request	握手请求，服务器可在任何时候向客户端发送该消息。若客户端正在进行握手过程就可忽略该消息。否则客户端发送cleint_hello消息，启动握手过程。	无
client_ hello	客户启动握手请求，该消息时当客户第一次连接服务器时向服务器发送的第一条消息。该消息中包括了客户端支持的各种算法。若服务器端不能支持，则本次会话可能失败。	版本、随机数、会话ID、密文族、压缩方法
server_ hello	其结构与client_hello消息，该消息是服务器对客户端client_hello消息的恢复。	版本、随机数、会话ID、密文族、压缩方法
server_ certificate	服务器提供的证书。如果客户要求对服务器进行认证，则服务器在发送server_hello消息后，向客户端发送该消息。证书的类型一般是X.509v3。	X．509v3证书链
server_key_ exchange	服务器密钥交换。当服务器不使用证书，或其证书中仅提供签名而不提供密钥时，需要使用本消息来交换密钥。	参数、签名

握手协议定义的消息类型 - 2

消息类型	说明	参数
certificate_ request	用于服务器向客户端要求一个客户证书。	类型、授权
server_hello_ done	该消息表明服务器端的握手请求报文已经发送完毕，正在等待客户端的响应。客户端在收到该消息时，将检查服务器提供的证书及其他参数是否是有效、可以接受的。	无
client_ certificate	客户端对服务器certificate_request消息的响应，只有在服务器端要求客户证书的时候使用。一般该消息是客户端收到server_hello_done消息后所发送的第一条消息。若客户端没有合适的证书，则向服务器端发送no_certificate的告警消息（无证书可能导致握手失败）	X．509v3证书链
client_key_ exchange	客户密钥交换。当客户不使用证书，或其证书中仅提供签名而不提供密钥时，需要使用本消息来交换密钥。	参数、签名
certificate_ verify	该消息用于向服务器提供对客户证书的验证。	签名
finished	该消息在“加密规约修改”（Change Cipher Spec）消息之后发送，以证实握手过程已经成功完成。本消息发送后，发送方开始使用协商的新参数来执行操作。该消息需要在两个方向上传送。	散列值

握手协议过程 - 1

- 第一阶段 安全能力的建立
  - (1) 客户 → 服务器： client_hello
  - (2) 服务器 → 客户： server_hello
- 第二阶段 服务器认证和密钥交换
  - (3) 服务器 → 客户： server_certificate
  - (4) 服务器 → 客户： server_key_exchange
  - (5) 服务器 → 客户： certificate_request
  - (6) 服务器 → 客户： server_hello_done

握手协议过程 - 2

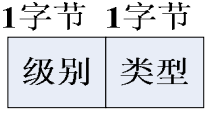
- 第三阶段 客户认证和密钥交换
  - (7) 客户 → 服务器： client_certificate
  - (8) 客户 → 服务器： client_key_exchange
  - (9) 客户 → 服务器： certificate_verify
- 第四阶段 结束阶段
  - (10) 客户 → 服务器： change_cipher_spec
  - (11) 客户 → 服务器： finished
  - (12) 服务器 → 客户： change_cipher_spec
  - (13) 服务器 → 客户： finished

修改密码规程协议

- 位于**SSL**记录协议之上
- 仅定义了一个由单个字节“**1**”构成的消息报文
- 该消息将改变连接所使用的加密规约
- 用途：切换状态
  - 把密码参数设置为当前状态。在握手协议后，当安全参数协商一致后，发送此消息，以通知接收方下面的记录将受到刚达成的密码规范和密钥的保护

告警协议

- 一种通过SSL记录协议进行传输的特定类型的消息
- 主要作用
  - 规定了告警的级别和告警的类型，在SSL协议执行过程中通过告警协议来显示信息交换过程中所发生的错误
- 组成：



告警级别

级别	告警名称	含义
1	警告	表示一个一般警告信息.
2	致命错误	表示出现了致命的错误,立即终止当前连接,同一会话的其它连接还可继续,不会再产生其它新的连接.

告警类型

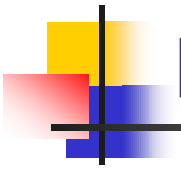
类型	告警名称	含义
0	**Close_notify**	通知接受方，发送方在本连接中不再发送消息.
10	**Unexpected_message**	收到不适当的消息.
20	**Bad_recode_mac**	接收到的记录的MAC有错误（致命错误）.
30	**Decompression_failure**	解压缩失败（致命错误）.
40	**Handshake_failure**	发送方无法进行的安全参数设置（致命错误）.
41	**No_certificate**	认证中心没有证书.
42	**Bad_certificate**	证书已破坏.
43	**Unsupported_certificate**	不支持接收的证书类型.
44	**Certificate_revoked**	证书已经撤销.
45	**Certificate_expired**	证书过期.
46	**Certificate_unknown**	在产生证书时有不明问题.
47	**Illegal_parameter**	握手过程某个字段超出范围.

SSL工作过程

- 发送方的工作过程
  - 从上层接收要发送的数据
  - 对信息进行分段，成若干记录
  - 使用指定的压缩算法进行数据压缩数据（可选）；
  - 使用指定的**MAC**算法生成**MAC**;
  - 使用指定的加密算法进行数据加密；
  - 发送数据
- 接收方的工作过程
  - 接收数据；
  - 使用指定的解密算法解密数据；
  - 使用指定的**MAC**算法校验**MAC**;
  - 使用压缩算法对数据解压缩（在需要时进行）；
  - 将记录进行数据重组；
  - 将数据发送给高层。

HTTPS（Hypertext Transfer Protocol Secure）

- 由Netscape开发并内置于其浏览器中，用于对数据进行压缩和解压操作，并返回网络上传送回的结果。HTTPS实际上应用了Netscape的安全套接字层（SSL）作为HTTP应用层的子层。（HTTPS缺省使用端口443，而不是HTTP的缺省端口80来和TCP/IP进行通信。）IE、Chrome等
- HTTPS使用40位-128位密码作为RC4流加密算法，支持使用X.509数字认证，可以确认发送者是谁。
- HTTPS是以安全为目标的HTTP通道，简单讲是HTTP的安全版，即HTTP下加入SSL层，HTTPS的安全基础是SSL
- 安全保护依赖浏览器的正确实现以及服务器软件、实际加密算法的支持
- 确保服务器是安全的，防止钓鱼攻击，常见例子是模仿银行域名。



HTTPS服务器

- Tomcat 5.x
- Nginx
- IIS
- Apache 2.x
- IBM HTTP SERVER 6.0

HTTPS 要比 HTTP 多用多少服务器资源？

- **http**使用**TCP** 三次握手建立连接，客户端和服务器需要交换**3**个包，**https**除了 **TCP** 的三个包，还要加上 **ssl**握手需要的**9**个包，所以一共是**12**个包。**http** 建立连接，按照下面链接中针对**Computer Science House**的测试，是**114**毫秒；**https**建立连接，耗费**436**毫秒。**ssl** 部分花费**322**毫秒，包括网络延时和**ssl** 本身加解密的开销（服务器根据客户端的信息确定是否需要生成新的主密钥；服务器回复该主密钥，并返回给客户端一个用主密钥认证的信息；服务器向客户端请求数字签名和公开密钥）。
- **SSL handshake latency and HTTPS optimizations. :: semicomplete.com**
- 来源：知乎

SSL协议运行实例 - 1

1. 客户机使用IE浏览器向服务器发送客户端的SSL版本号、密码设置、随机数和需要服务器使用SSL协议与客户机进行通信的其它信息；IE浏览器使用https协议向服务器申请建立SSL会话；
2. 服务器向客户端发送服务器端的SSL版本号、密码设置、随机数和客户端使用SSL协议与服务器通信需要的其它信息。同时服务器端发送它自己的数字证书供客户认证，如果认为客户端需要身份认证则要求客户发送证书，该操作是可选的；
3. 客户端利用服务器发送信息认证服务器的真实身份并取得公开密钥等，如果服务器不被认证，用户被告警发生了问题，通知不能建立带有加密和认证的连接。若服务器能被成功地认证，客户机将继续下一步。

SSL协议运行实例 - 2

4. 客户机利用数字信封技术为将要进行的会话创建会话预密钥**pre_master_secret**，并用服务器的公钥加密它，然后向服务器发送加密的会话预密钥；
5. 当客户能被成功认证后，服务器会使用它的私人密钥解密从客户端得到的会话预密钥**pre_master_secret**， 并生成真正的会话密钥**master_secret**。同时客户端也从相同的**pre_master_secret**开始得到相同的**master_secret** 。会话密钥**master_secret**是对称密钥用于加密和解密在**SSL**会话期间交换的信息，并验证信息的完整性 ；
6. 会话密钥生成后，客户端向服务器发送消息，通知服务器以后从客户端来的消息将用会话密钥加密，这表明握手的客户端部分已经完成；

SSL协议运行实例 - 3

7. 服务器向客户端发送相同的消息，通知从服务器来的消息将用会话密钥加密，这表明握手的服务器部分已经完成；
8. SSL会话开始，使用安全通道发送消息，客户端和服务器使用会话密钥加密和解密它们彼此发送的数据和验证数据完整性；
9. 当通信完成后，一般情况会话密钥会被丢弃。

TLS协议

- 安全传输层协议（**TLS：Transport Layer Security Protocol**）用于在两个通信应用程序之间提供保密性和数据完整性。

- **1997**年，**IETF**基于**SSLv3**协议发布了**TLS(Transport Layer Security)v1**传输层安全协议的草案 使用的版本号为**SSLv3.1**。
- **1999**年，正式发布了**RFC2246**
- 从**TLSv1**成为工业标准以后，**TLSv1**在因特网上已得到了广泛的应用
- 除了如**S／HTTP**，**S／MIME**，**SSL-Telnet**，**SSL-SMTP**和**SSL-POP3**等常用的协议以外，目前许多电子商务和电子政务系统也基于**TLS**来确保其安全性

TLS协议

- 该协议由两层组成： **TLS** 记录协议（**TLS Record**）和 **TLS** 握手协议（**TLS Handshake**）。

- **TLS** 记录协议是一种分层协议。每一层中的信息可能包含长度、描述和内容等字段。记录协议支持信息传输、将数据分段到可处理块、压缩数据、应用 **MAC** 、加密以及传输结果等。对接收到的数据进行解密、校验、解压缩、重组等，然后将它们传送到高层客户机。
- **TLS** 握手协议由三个子协议组构成，允许对等双方在记录层的安全参数上达成一致、自我认证、例示协商安全参数、互相报告出错条件。

TLS协议

- TLS建立会话协商的参数、握手协议过程等与SSL一致，其基本的工作流程分为如下两个阶段：

服务器认证阶段。

- 客户端向服务器发送一个开始信息“Hello”以便开始一个新的会话连接；
- 服务器根据客户的信息确定是否需要生成新的主密钥，如需要则服务器在响应客户的“Hello”信息时将包含生成主密钥所需的信息；
- 客户根据收到的服务器响应信息，产生一个主密钥，并用服务器的公开密钥加密后传给服务器；
- 服务器恢复该主密钥，并返回给客户一个用主密钥认证的信息，以此让客户认证服务器。

用户认证阶段

TLS协议

TLS的特点：

- **TLS** 记录协议用于封装各种高层协议
- 可以使用非对称的，或公共密钥的密码术来认证对等方的身份
- 共享加密密钥的协商是安全的
- 协商是可靠的
- **TLS** 的最大优势就在于：**TLS** 是独立于应用协议

TLS的不足：

- **TLS** 标准并没有规定应用程序如何在 **TLS** 上增加安全性
- 把如何启动 **TLS** 握手协议以及如何解释交换的认证证书的决定权留给协议的设计者和实施者来判断

OpenSSL安全漏洞

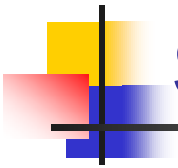
- “心脏出血”漏洞事件
  - 2014年4月，OpenSSL一个重大漏洞被曝出，因为危害极大，因此该漏洞被称为“Heartbleed”（心脏出血），特定版本的OpenSSL由于未能在memcpy()调用受害用户输入内容作为长度参数之前正确进行边界检查。攻击者可以追踪OpenSSL所分配的64KB缓存、将超出必要范围的字节信息复制到缓存当中再返回缓存内容，受害者的内存内容就会以每次64KB的速度进行泄露。
- DROWN漏洞（水牢漏洞）
  - 2016年3月被发现，可以允许攻击者破坏使用SSLv2协议进行加密的HTTPS网站，攻击者可以监听加密流量，通过“中间人劫持攻击”读取诸如密码、信用卡账号、商业机密和金融数据等加密信息。
  - 与“心脏出血”直接攻破资料存储的服务器不同，“水牢漏洞”需要黑客与被攻击者在同一个局域网下，例如入侵对方所处的WiFi网络，或者和对方在一个公司等。
  - 另外，“水牢漏洞”利用难度较高，需要租用计算集群并花费8个小时才能破解密钥。而租用计算集群的成本在400美元左右。

什么是SSO

- SSO英文全称Single Sign On，单点登录(又称为单一签入/登入)。一种对于许多相互关连，但是又是各自独立的应用系统，提供存取控制的属性。当拥有这项属性时，当使用者登入时，就可以取得所有系统的存取权限，不用对每个单一系统都逐一登入。它是目前比较流行的企业业务整合的解决方案之一

- 相反的SSO也可也以理解为Single Sign Off，单点登出，也就是一处登出动作，就可以结束对多个系统的存取权限

- 一处登录、全网漫游



SSO的优势

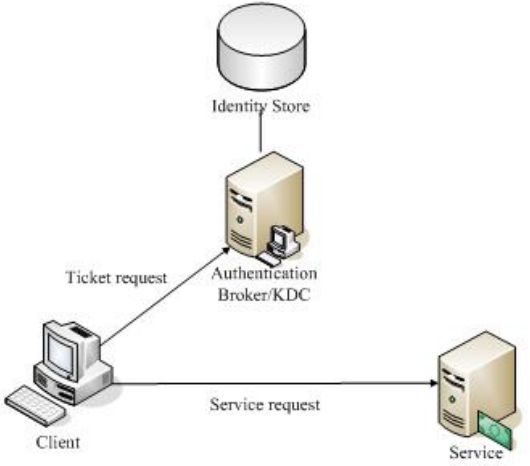
- 减少用户在不同系统中登录耗费的时间，减少用户登录出错的可能性
- 实现安全的同时避免了处理和保存多套系统用户的认证信息
- 减少了系统管理员增加、删除用户和修改用户权限的时间
- 增加了安全性：系统管理员有了更好的方法管理用户，包括可以通过直接禁止和删除用户来取消该用户对所有系统资源的访问权限

Kerberos协议

- Kerberos协议主要用于计算机网络的身份认证(Authentication), 其特点是用户只需输入一次身份验证信息就可以凭借此验证获得的票据(ticket-granting ticket)访问多个服务，即SSO(Single Sign On)。
- Kerberos 是一种网络认证协议，其设计目标是通过密钥系统为客户机 / 服务器应用程序提供强大的认证服务。
- 认证过程的实现不依赖于主机操作系统的认证，无需基于主机地址的信任，不要求网络上所有主机的物理安全，并假定网络上传送的数据包可以被任意地读取、修改和插入数据。
- Kerberos 作为一种可信任的第三方认证服务，是通过传统的密码技术（如：共享密钥）执行认证服务的。

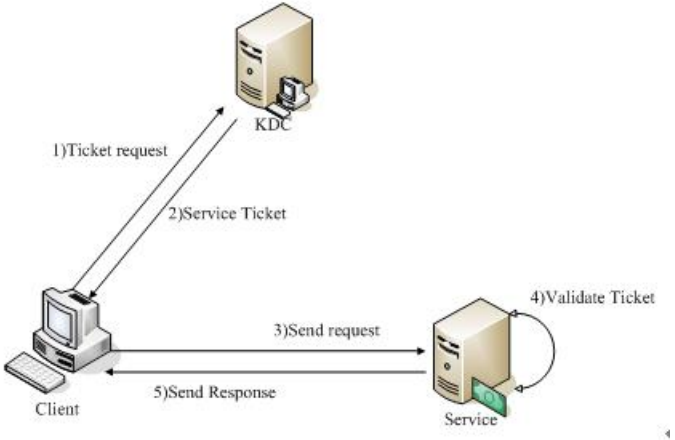
Kerberos协议

1. Client向KDC发送自己的身份信息，KDC从Ticket Granting Service得到TGT；
2. Client利用之前获得的TGT向KDC请求其他Service的Ticket，从而通过其他Service的身份鉴别。



Kerberos协议

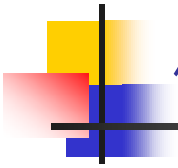
- Client将之前获得TGT和要请求的服务信息(服务名等)发送给KDC，KDC中的Ticket Granting Service将为Client和Service之间生成一个Session Key用于Service对Client的身份鉴别；
- 此时KDC将刚才的Ticket转发给Client；
- 为了完成Ticket的传递，Client将刚才收到的Ticket转发到Service；
- Service 收到Ticket后利用它与KDC之间的密钥将Ticket中的信息解密出来，从而获得Session Key和用户名，用户地址（IP），服务名，有效期。
- 如果Service有返回结果，将其返回给Client。



Kerberos协议

- Kerberos的特点：
- 保密性
  - 完成ticket的安全传递。
- 完整性
  - 完成了Session Key的安全发布。

- Kerberos的不足：
- 由于kerberos协议的消息无法穿透防火墙，因此限制了该协议往往应用于一个组织内部，这一点使得其的应用范围受到一定局限性。



什么是CAS

- **CAS = Central Authentication Service**，中央认证服务，一种独立开始指令协议。**CAS** 是 **Yale** 大学发起的一个开源项目，旨在为 **Web** 应用系统提供一种可靠的单点登录方法，**CAS** 在 **2004** 年 **12** 月正式成为 **JA-SIG** 的一个项目。

- **CAS**可以认为是一款针对 **Web** 应用的单点登录框架，也可以认为是一种协议

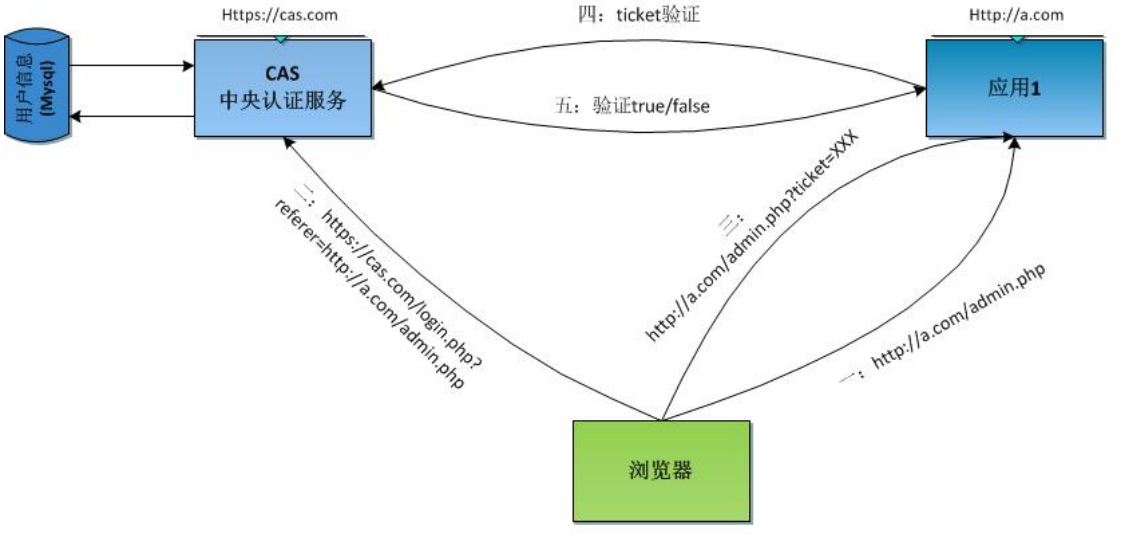
- **CAS**特点
  - 开源的企业级单点登录解决方案。
  - **CAS Server** 为需要独立部署的 **Web** 应用
  - **CAS Client** 支持非常多的客户端**(**这里指单点登录系统中的各个 **Web** 应用**)**，包括 **Java**、**php**、**python**等

SSO实现机制

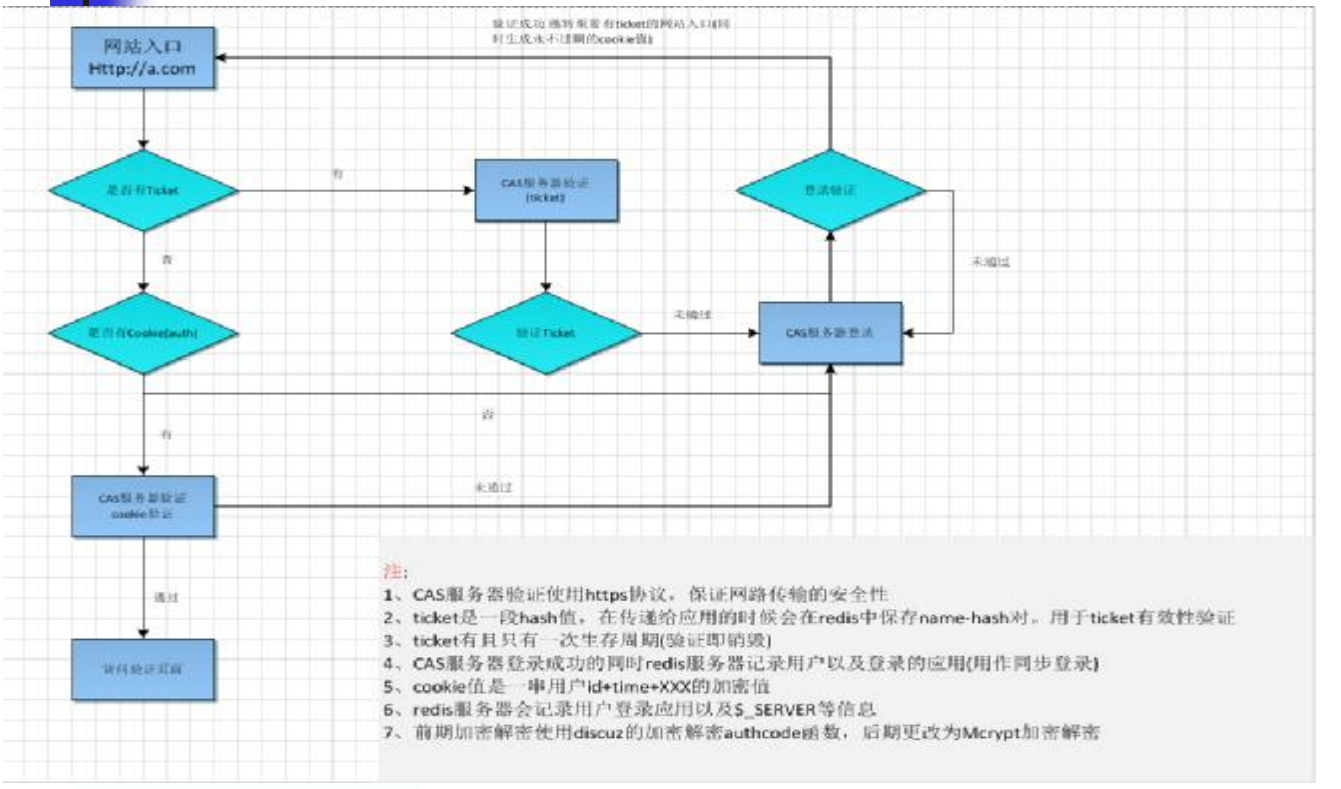
- SSO的实现机制不尽相同，大体分为Cookie(客户端)验证机制和Session(服务器端)验证机制两大类
- Cookie认证是大众使用的用户验证方案(通过接口对每个域名下写cookie)-广播机制
  - demo(ucenter)：
  - <iframe width="0" height="0" src="https://b.com/sso.php?sessid=xxxxxxxxxxxx&sfkey=xxxxxxxxxxx"></iframe>
  - <iframe width="0" height="0" src="https://c.com/sso.php?sessid=xxxxxxxxxxxx&sfkey=xxxxxxxxxxx"></iframe>
- 共享Session(db，nosql等)验证。-共享机制

CAS中央验证架构

SSO单点登录系统



CAS验证流程


网站入口
Http://a.com
是否有Ticket
有
CAS服务器验证
(ticket)
登录验证
不通过
否
是否有Cookie(auth)
验证Ticket
不通过
CAS服务器登录
否
有
CAS服务器验证
cookie验证
未通过
通过
注:
1、CAS服务器验证使用https协议，保证网路传输的安全性
2、ticket是一段hash值，在传递给应用的时候会在redis中保存name-hash对，用于ticket有效性验证
3、ticket有且只有一次生存周期(验证即销毁)
4、CAS服务器登录成功的同时redis服务器记录用户以及登录的应用(用作同步登录)
5、cookie值是一串用户id+time+XXX的加密值
6、redis服务器会记录用户登录应用以及$_SERVER等信息
7、前期加密解密使用discuz的加密解密authcode函数，后期更改为Mcrypt加密解密

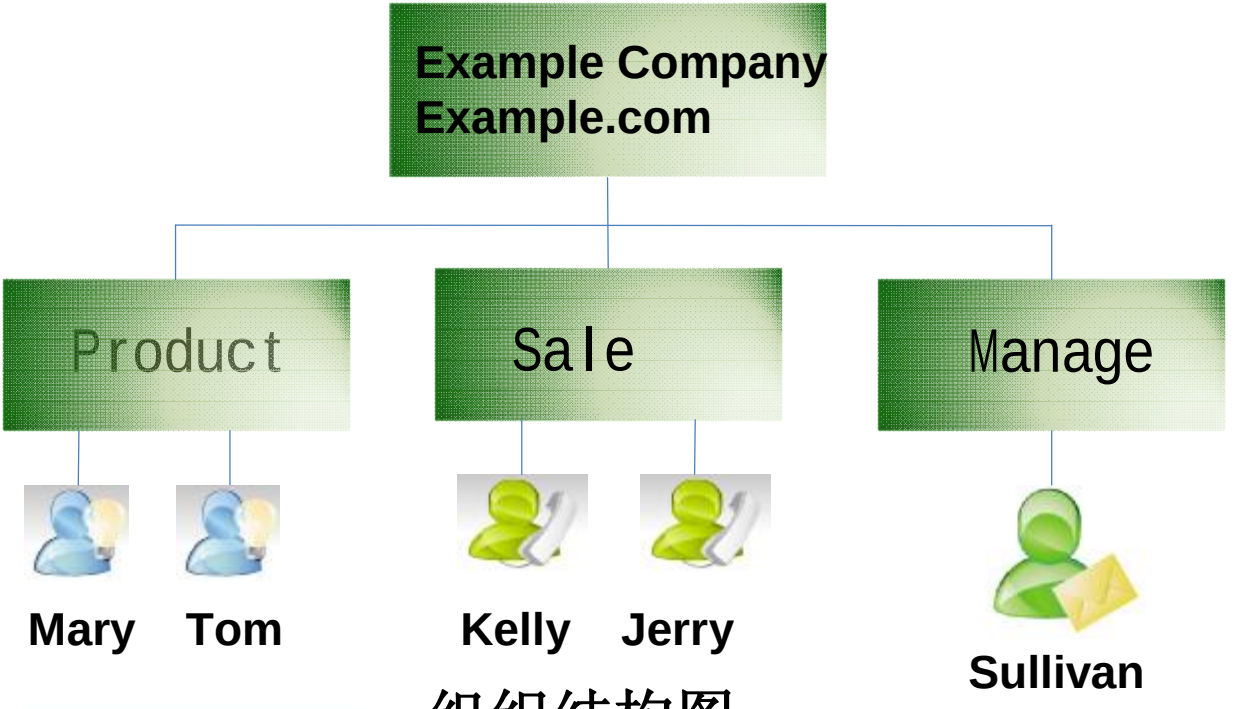
安全论证

- SSO是所有应用的入口，所以SSO的安全性比所有应用安全性都重要
- 不以SSL方式登录的系统都是不安全的。CAS 从 v1 到 v3 ，都很依赖于SSL ，它假定用户在一个非常不安全的网络环境中使用SSO，Hacker的Sniffer会很容易抓住所有的 Http Traffic，包括通过 Http 传送的密码甚至 Ticket 票据。一旦ticket被第三方获取，所有的权限都会被攻破
- CAS的传输安全性紧紧依赖SSL（CAS认证中心必须使用https）
- Service Ticket 只能使用一次
- Service Ticket 在一段时间内失效
- Service Ticket 是基于随机数生成的(如果 Service Ticket 生成规则被猜出，Hacker 就等于绕过 CAS 认证，直接访问所有服务)

LDAP协议及其特点

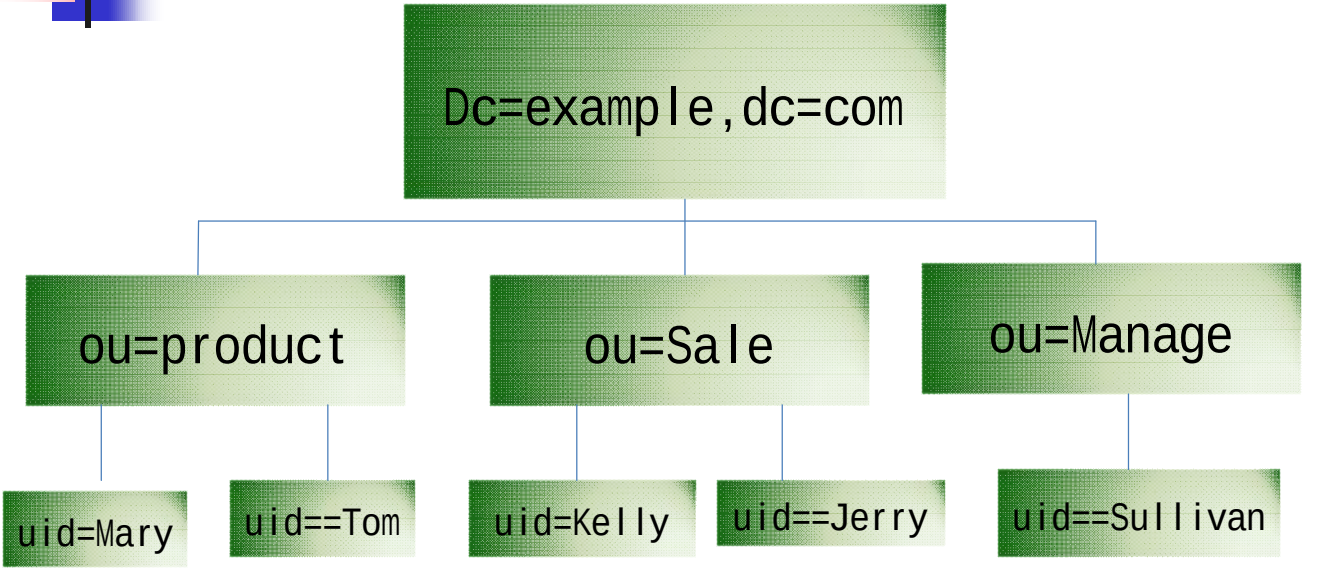
- **LDAP**是轻量目录访问协议**(Lightweight Directory Access Protocol)**的缩写，也是一种目录服务，类似于文件系统中所使用的目录，查询电话号码使用的电话号码簿。
- **LDAP**特点：
  - **LDAP**是一个跨平台，标准的协议，得到了业界广泛的认可；
  - **LDAP**服务器可以用“推”或“拉”的方法复制部分或全部数据，提高数据访问安全；
  - **LDAP**是一个安全协议支持多种认证，如**TLS,SSL,SASL;**
  - 支持异类数据存储；
  - 安装装装简单，易于维护和优化，降低成本；
  - 支持**TCP/IP**协议。可以不依赖作保特定的软硬件平台；
  - 允许使用**ACL**。例如，设备管理员可以有权改变员工的工作地点和办公室号码， 但是不允许改变记录中其它的域。**acl**可以根据谁访问数据、访问什么数据、数据存在什么地方以及其它对数据进行访问控制。因为这些都是由**LDAP**目录服务器完成的，所以不用担心在客户端的应用程序上是否要进行安全检查。

LDAP目录结构 （例）



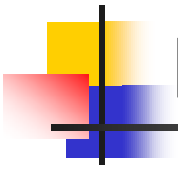
组织结构图

LDAP目录树


Dc=example,dc=com
ou=product
ou=Sale
ou=Manage
uid=Mary
uid==Tom
uid=Kelly
uid==Jerry
uid==Sullivan

LDAP目录树中常见的关键字

关键字	说明
dc	域名，其格式是将完整的域名分成几部分，如域名example.com变成dc=example,dc=com
uid	用户ID，如“Kelly”
ou	组织机构类似于linux文件系统中的子目录，是一个容器对象，组织机构可以包含其他组织机构，如”Sales”
cn	全称，如“Kelly King”
sn	姓，如King
dn	唯一标识，类似于linux文件系统中的绝对路径，每个对象都有一个唯一的名称 如”uid=kelly,ou=sales,dc=example,dc=com”，在一个目录树中DN总是唯一的
rdn	相对唯一标识，类似于文件系统中的相对路径，它是与目录树结构无关的部分，如”uid=kelly”或“cn=Kelly King”
c	国家，如“CN 或US”等
o	组织名称，如”Example,Inc”



LDAP安装

桌面图形化安装

手动编译安装

YUM安装

LDAP配置

LDAP主配置文件：/etc/openldap/slapd.conf

#vi /etc/openldap/slapd.conf

suffix “dc=example.com” //设置主机名前缀

rootdn “cn=Manager,dc=example,dc=com” //设置域

rootpw admin123 //设置管理员密码

#slaptest -f slapd.conf //测试配置文件是否正确

config file testing succeeded

LDAP数据库文件：/var/lib/ldap/DB_CONFIG.example

#cp /usr/share/doc/openldap-servers-2.4.19/DB_CONFIG.example /var/lib/ldap/ DB_CONFIG

启动/停止LDAP服务： #/etc/init.d/slapd start/stop

或者用 #service slapd start/stop

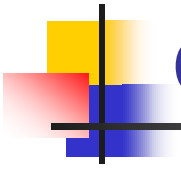
检查搜索域：#ldapsearch -x -b “dc=example.com”

或者#ldapsearch -x -b “dc=example,dc=com”

创建待认证的用户:#useradd ldapuser1

#echo "123456"|passwd --stdin ldapuser1

使用迁移工具：migrationtools



OpenLDAP

- OpenLDAP在Windows上的安装配置

  http://www.micmiu.com/enterprise-app/sso/openldap-windows-config/
- CAS+LDAP实现单点登录认证

  http://www.micmiu.com/enterprise-app/sso/sso-cas-ldap-auth/

8.2 电子商务安全

电子商务的概念

- 利用计算机网络，通过电子数据交换来完成某种与商务或服务相关的工作
- 将传统商务移植到互联网上
- 基础：以电子化的形式处理和传输商务数据
- 电子商务三要素
  - 信息
  - 电子数据交换
  - 电子资金转账

电子商务分类

- 按照应用领域
  - 企业对消费者（**Business to Customer, B2C**）
    - 电子零售
    - 如：网上书店
  - 企业对企业（**Business to Business, B2B)**
  - 企业对政府机构（**Business to Government, B2G**）
  - 消费者对政府机构（**Customer to Government, C2G)**
- 按照付款方式
  - 电子数据交换
  - 信用卡
  - 电子支票
  - 电子现金
  - 记账卡
  - 物物交换

电子商务的支撑环境

- 网络环境
  - Internet
- 支付系统
  - 网上银行、网上支付系统
- 安全认证系统
- 配送系统

电子商务面临的安全威胁

- 对销售者的威胁
  - 中央系统被破坏
  - 客户资料被竞争者获悉
  - 被竞争者假冒
  - 消费者提交订单而不付款
  - ……
- 销售者对电子商务的要求
  - 鉴别消费者身份
  - 争议解决机制
  - 信息保密
- 对消费者的威胁
  - 被他人假冒
  - 付款后不能收到商品
  - 机密信息泄露
  - ……
- 消费者对电子商务的要求
  - 鉴别销售者身份
  - 信息保密
  - 争议解决机制

电子商务安全的技术要求

- 真实性
  - 对消息和实体身份进行鉴别
- 机密性
  - 信息不被泄露给非授权者
- 完整性
  - 保证数据一致性
- 可用性
  - 合法用户的使用不被不正当地拒绝
- 不可否认性
  - 防止实体的抵赖行为
- 可控性
  - 控制资源的使用方式

电子商务系统所需安全服务

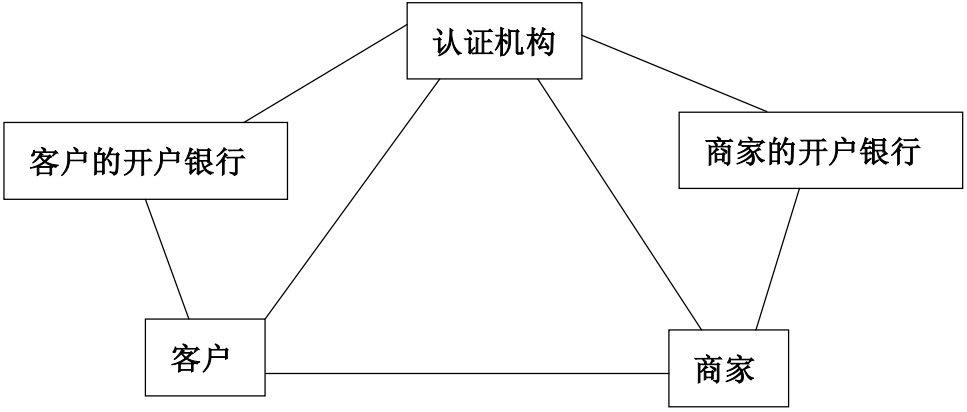
- 鉴别服务
  - 对实体身份进行鉴别
- 访问控制服务
  - 通过授权控制使用资源的方式
- 机密性服务
  - 为电子信息提供机密性保证
- 不可否认服务
  - 为交易双方提供不可否认的证据

电子商务的安全体系结构

电子支付系统的安全

- 电子支付
  - 通过网络进行货币支付
  - 本质是试图把现有支付结构转化为电子形式
- 电子支付手段
  - 电子信用卡
  - 电子支票
  - 电子现金
- 对电子支付系统的安全要求
  - 认证：对实体身份进行鉴别，X.509证书和数字签名
  - 保密：加密算法对业务进行加密
  - 业务完整性：消息摘要算法以保证业务完整性
  - 业务的不可否认性
  - 多方支付协议：处理多方贸易

网上支付系统的基本组成



- 保密和完整性：SSL协议和S-HTTP协议，加密和消息摘要
- 不可否认：公钥体制和X.509证书
- 多支付协议：
  - 商家只能读取订单信息而不是信用卡信息
  - 接收银行只是读取支付信息而不是订单信息

SET协议

SET概述

- **1996年2月，由VISA和MasterCard两大信用卡公司提出了安全电子交易 SET (Secure Electronic Transaction)** 协议，该协议是在因特网环境中解决用户、商家和银行之间通过信用卡支付交易而设计的安全规范。
- 众多厂商和机构参与了**SET**的制定
  - 信用卡机构
  - 计算机厂商
  - 网络安全提供商
- **SET**提供三种服务
  - 交易各方的安全通信
  - 基于**X.509**证书的信任
  - 消息的机密性保证

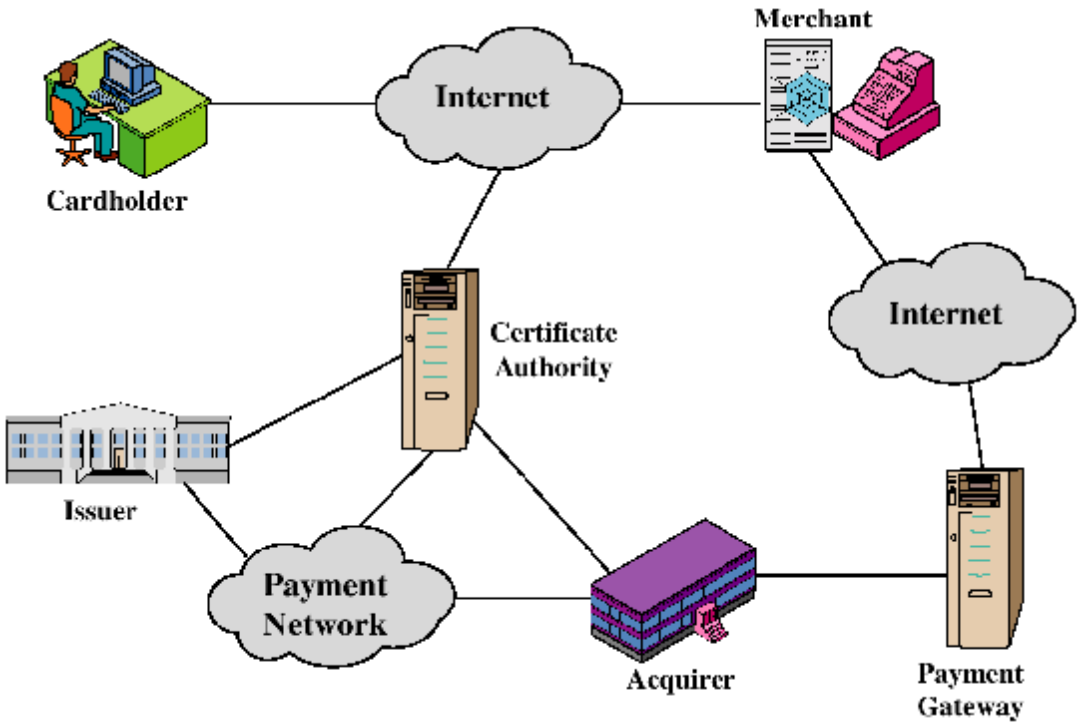
SET协议

- SET是一组安全协议的集合，将现有信用卡支付的基础设施安全地配置在开放的Internet上

采用SET协议进行网上交易支付时，主要涉及：

- 持卡人，即消费者，他们通过Web浏览器或客户端软件购物；
- 商家，在Intemet上提供在线商店；
- 发卡人，通常为银行，为持卡人开设账户，并且发放用于网上支付的信用卡；
- 支付者，通常为银行，为商家建立账户，并且处理支付卡的认证和支付的事务；
- 支付网关，实现对支付信息从Intemet到银行内部网络的转换，用来处理商家支付。

SET Participants



SET商业模型的需求

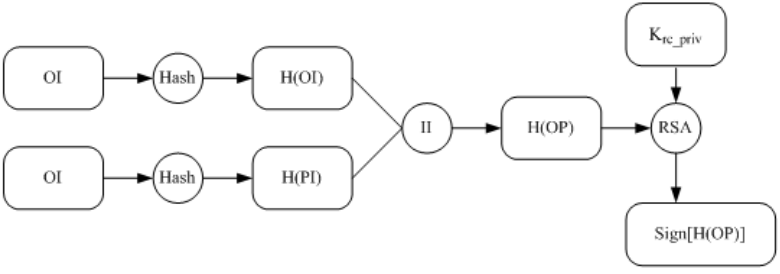
- 对订购信息和支付信息提供保密性
- 保证信息传输的完整性
- 对持卡人的身份认证
  - 是否信用卡帐户的合法用户
- 对商家的身份认证
- 建立不依赖于任何安全协议的安全通信机制
- 方便和鼓励互操作性

SET的关键特征

- 信息机密性
  - 使用**DES**对信息进行加密
- 数据完整性
  - 使用**SHA-1＋RSA**数字签名来提供完整性
- 制卡用户帐户鉴别
- 商家身份鉴别

SET协议

- **SET安全功能**
  - **SET协议的安全措施十分完善，它把对称密钥体制和公开密钥体制完美地结合了起来，利用数字信封技术进行密钥的安全可靠交换，通过DES，RSA等进行高效率数据加密，利用数字证书、消息摘要、时间戳、数字签名和双重签名等技术确保信息的完整性和不可抵赖性等，具有安全性高、密钥管理简便等优点。**



SET协议

SET和TLS协议的比较

- **SET与TLS同样提供了电子安全交易的机制，但是运行方式有所差别。**

(1) SET是一个多方的报文协议，它定义了银行、商店和消费者之间必需遵守的报文规范；TLS只是简单地在交易双方之间建立了安全连接；

(2) TLS是面向连接的；而SET允许各方之间的报文交换不是实时的；

(3) SET报文能够在银行内部网或者其他网络上传输；而TLS之上的交付系统只能与WEB浏览器捆绑在一起工作；

(4) SET的安全需求较高，因此所有参与SET交易的成员（消费者、商店、支付网关等）都必须先申请数字证书来识别身份，并且整个交易过程都受到严密的保护；而在TLS中只有商店端的服务器需要认证，客户端认证是可选的，同时交易过程中的安全范围只限于消费者到商店的信息交换；

(5) SET交易，除了申请数字证书之外，还必须安装符合SET规范的电子钱包软件；而TLS交易不需要；

(6) 由于SET的成本较高，并且引入中国的时间较短，目前是以TLS的普及率较高，大约有7-8成，但是网上交易的安全性需求不断提高，可以预料SET将会成为日后市场的主流。

SET协议

SET工作过程

- 在进行**SET**交易之前，消费者首先需要向支持**SET**的银行申请开户，获得一个用于网上支付的信用卡账户。同时，消费者和商家还必须从**CA**认证中心那里申请相应的**X.509**数字证书。此后，商家就可以在因特网上开设超市，持卡人**(**消费者**)**也就可以通过浏览器在因特网实现购物。

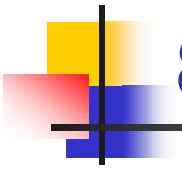
- 订购商品
- 购买商品
- 处理订单
- 支付
- 购买应答

SET交易过程 - 1

- 消费者申请信用卡
- 持卡用户获得X.509证书
  - 证书包含了用户的RSA公钥
- 商家获得证书
- 两个证书，一个用于消息签名，一个用于密钥交换
- 持卡用户订购商品
- 持卡用户对商家进行身份认证
  - 商家向用户提供自己的证书

SET交易过程 - 2

- 用户发送订购、支付信息以及自己的证书
  - 订单确认对商品的购买
  - 支付信息包含信用卡细节，被加密，商家不可读
  - 证书用于商家对用户的认证
- 商家请求支付认可
  - 商家将用户的支付信息发给支付网关，请求核准用户的余款是否足以支持本次交易
- 商家确认订购
  - 支付请求被确认，商家给用户一个订单确认
- 供货
- 商家请求支付
  - 商家将支付请求发给支付网关，支付网关将货款从用户的信用卡帐户转到商家帐户



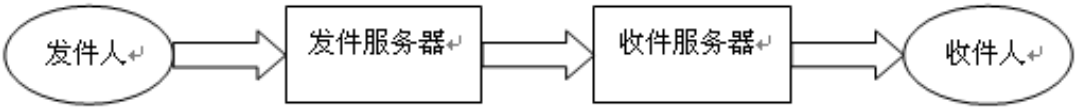
8.3 E-mail的安全

- 电子邮件安全的基本特性
- 电子邮件面临的安全问题
- 电子邮件安全性的安全技术
- 电子邮件安全防护措施

电子邮件基本概念

- 电子邮件就是我们通常所说的**E-mail**。在英语中，**mail**是邮件的意思，而**E**则是英语**Electronic**的缩写。
- 电子邮件是**Internet**上应用最广同时也是最基本的服务之一。只要能够连接到因特网，拥有一个**E-mail**账号，就可以通过电子邮件系统，用非常低廉的价格、非常快的速度，与世界上任何一个角落的网络用户联络。

电子邮件的工作原理



- 电子邮件的工作过程是：发信方通过客户端程序将编辑好了的电子邮件向服务器发送，服务器识别到收信人的地址，并向管理该地址的服务器发送消息，邮件服务器识别后将消息存放在接收者的电子信箱内，并告知收信人有新邮件到来。收信人通过邮件客户程序连接到服务器后，就会看到服务器的通知，打开电子信箱来查收邮件。
- 电子邮件安全隐患的原因主要有三个方面：
  - 一是电子邮件传送协议自身的先天安全隐患。
  - 二是邮件接收客户端软件的设计缺陷。
  - 三是用户个人的使用问题。

常见的电子邮件协议

（1）SMTP（简单的邮件传输协议）

- Internet上传输电子邮件的标准协议，用于提交和传送电子邮件，规定了主机之间传输电子邮件的标准交换格式以及邮件在链路层上的传输机制。

（2）POP3（邮局协议）

- Post Office Protocol - Version 3，是TCP/IP协议族中的一员，由RFC1939 定义，主要用于客户端远程管理在服务器上的电子邮件。POP不支持对服务器邮件进行扩展操作，此过程需要更高级的IMAP4协议来完成。

常见的电子邮件协议

（3）IMAP4（交互式数据消息访问协议第四个版本）

- IMAP4支持协议客户机在线或者离开访问并阅读服务器上的邮件，还能交互式的操作服务器上的邮件。不需要像POP3协议那样把邮件下载到本地，用户可以直接对服务器上的邮件进行操作还可以在服务器上维护自己邮件目录

（4）MIME（多功能的网际扩展协议）

- MIME增强了在RFC 822中定义的电子邮件报文的能力，允许传输二进制数据。MIME编码技术用于将数据从使用8位的格式转换为使用7位的ASCII码格式。

电子邮件的特点

（1）方便：足不出户就可以和远在万里之外的其他人通信。只要能连上Internet就能随时随地读取和发送邮件。

（2）快捷：Internet上的信息是在光纤中是以光速传播的，因此E-mail比普通的邮政信件快得多，甚至比普通的电报还要快。

（3）便宜：由于收发E-mail所占用的上网时间很短，所以相对寄送普通邮件来说，E-mail是很便宜的。

（4）信息多样：能把用数字表示的所有信息以附加文件的方式发给收件人，可以是文字、图像，也可以是声音甚至动画等。

（5）功能强大：不仅可以用来向网上的亲朋好友发邮件，还可以参加专题讨论组，订阅电子期刊，完成文件传输等。

电子邮件安全问题

（1）邮件病毒

- 邮件病毒一般是通过邮件中的附件进行扩散，一旦运行了附件中的病毒程序，就能导致电脑染毒。然而一些新型邮件病毒只需在打开邮件正文或浏览标题时就能够感染电脑。邮件已成为计算机病毒传播的一个主要途径。目前多数蠕虫病毒都可以通过邮件传播。一些邮件服务器每小时能收到上万封病毒邮件， 有时甚至会造成网络堵塞或邮件服务器瘫痪。还有些蠕虫病毒能利用人工漏洞传染病毒。

电子邮件安全问题

（2）垃圾邮件

- 中国互联网协会在《中国互联网协会反垃圾邮件规范》中定义的垃圾邮件包括如下属性：
  1）收件人事先没有提出要求或同意接收的广告、电子刊物、各种形式的宣传品等宣传性的电子邮件；
  2）收件人无法拒收的电子邮件隐藏；
  3）隐藏发件人身份、地址、标题等信息的电子邮件；
  4）含有虚假的信息源、发件人、路由等信息的电子邮件。
- 垃圾邮件不仅占用了大量的网络资源，而且浪费了邮件服务使用者宝贵的时间和精力。

电子邮件安全问题

（3）监听

- 一种是局域网内的监听。通常使用嗅探器对局域网内传输的数据进行监听，由于协议通常都是明文传输，嗅探器很容易嗅探到用户的邮箱密码，使用浏览器收发邮件就显得安全一些。
- 另一种监听来自邮箱内部。用户密码被破解之后，攻击者并不会修改密码， 而是将邮件先发送到攻击者的信箱，再将邮件转发到用户邮箱。从而完全控制用户信箱的流量，选择其能够接收到的邮件。这种监听方法相当隐蔽，危害很大。

几种电子邮件安全技术

- **PGP**
- **PGP（Pretty Good Privacy）是由Phil Zimmermann主持开发的网络应用，能为电子邮件系统和文件存储应用过程提供认证业务和保密业务。其工作主要有：**

（1）选择了较好的加密算法来创建数据块。

（2）将密码算法集成在应用程序中，与操作系统和处理器独立、且其命令集易于使用。

（3）软件包或源代码等文档可通过因特网、公告板或其他商用网络免费使用。

（4）用户可与公司建立协议，以获得完全兼容的、低成本的商用版本。

PGP

- PGP被广为使用的原因有：

（1）世界各地都可免费使用的几种版本都可用于各种平台，此外，商用版支持商家开发自己的产品。

（2）所用的算法具有很高的安全性，其软件包中的公钥加密算法有RSA、DSS以及ElGamal，单钥算法有CAST_128、IDEA、三重DES和SHA。

（3）适用范围极为广泛。

（4）它的开发未受任何政府组织和标准化组织的控制。

PGP的认证业务

- PGP中通过数字签名提供认证的过程分为5步：

1）发方产生消息M。

2）用SHA产生160比特长的消息摘要H(M)。

3）发方用自己的密钥SKA按RSA算法对H(M)进行加密，并将加密结果与M连接在一起发送。

4）收方用发方的公钥对得到消息的摘要进行解密得到H(M)。

5）收方对收到的M计算消息摘要，并与(4)中的H(M)比较，如果一致，则认为M是真实的。

PGP的保密业务

1）发方产生消息M及一次性会话密钥KS。

2）用密钥KS按CAST-128/IDEA/3DES加密M。

3）用收方的公钥PKB按RSA算法加密一次性会话密钥KS，将2）、3）中的两个加密结果连接起来一起发往收方。

4）收方用自己的私钥按RSA算法恢复一次性会话密钥。

5）收方用一次性会话密钥恢复收到的消息。

- 优点：一、由于分组加密速度远快于公钥加密算法速度，大大地减少加密时间。二、由于会话密钥是一次性的，因此没必要使用会话密钥的交换协议。三、一次性会话密钥的使用进一步加强了本来就很强的分组加密算法。

PGP的保密性与认证性

- 如果采用对同一消息同时提供保密性和认证性，发方首先用自己的私钥对消息签字，将明文消息和签字连接在一起，再使用一次性会话密钥按**CAST-128/IDEA/3DES**对其加密，同时用**RSA**对会话密钥加密，最后将两个加密结果一同发往收方。这一过程中，先对消息签字再对签字进行加密。这一顺序优于先加密、再对加密结果签字。因为将签字与明文消息一起存储会带来很多方便，同时也给第三方签字的认证带来方便。

PGP的压缩

- 压缩的目的是为邮件的传输或文件的存储节约空间。压缩运算应在签名之后、加密以前，这是因为：

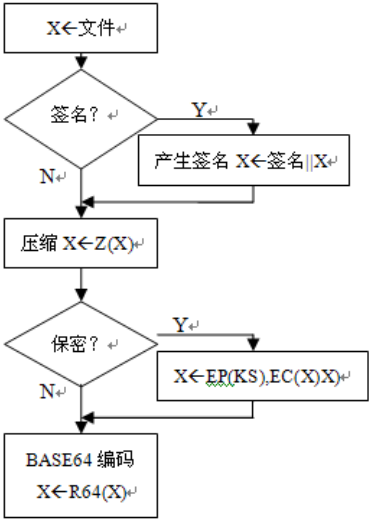
1）对不压缩的的消息签字，可便于以后对签字的验证。如果对压缩后的消息签字，以后要对签字的验证时，需存储压缩后的消息或在验证签字时对消息重做压缩。

2）对消息压缩后再进行加密可加强其安全性，这是因为消息压缩后比压缩前的冗余度要小，因此会使密码分析更为困难。

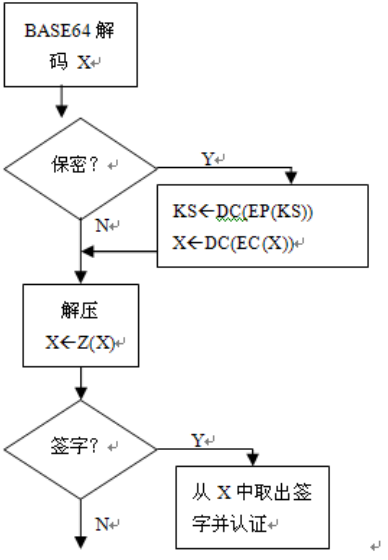
电子邮件的兼容性

- PGP变换具有“盲目性”，即不管输入变换的消息内容是不是ASCII文件，都将变换为BASE64格式。作为一种配置选择，PGP可以只将消息的签字部分转换为BASE64格式，从而使得接收方不使用PGP就可以阅读消息，但对签字的验证仍然需要使用PGP。

PGP的消息处理过程



(a)发送方处理框图



(b)接收方处理框图

分段与重组

- 电子邮件通常都对最大可用的消息长度有所限制，如果消息长度大于最大可用长度，则将消息分为若干子段并分别发送。分段操作在**BASE64**变换以后进行，会话密钥报头和签字报头仅在第一子段的开头处出现一次。
- 接收方在处理过程之前，先去掉第一子段开头处的报头再将各自段拼接在一起。

分段与重组

- 电子邮件通常都对最大可用的消息长度有所限制，如果消息长度大于最大可用长度，则将消息分为若干子段并分别发送。分段操作在**BASE64**变换以后进行，会话密钥报头和签字报头仅在第一子段的开头处出现一次。接收方在处理过程之前，先去掉第一子段开头处的报头再将各自段拼接在一起。

密钥和密钥环

（1）会话密钥的产生

- 会话密钥的使用是一次性的，其中CAST-128和IDEA所用会话密钥长为128比特，3DES所用的会话密钥长为168比特。CAST-128密钥的随机数产生器由CAST-128加密算法构成。其输入为一个128比特的密钥和两个64比特的明文，采用CFB模式，对两个明文分组加密，再将得到的两个64比特密文分组连接在一起形成所要产生的128比特密钥。

密钥和密钥环

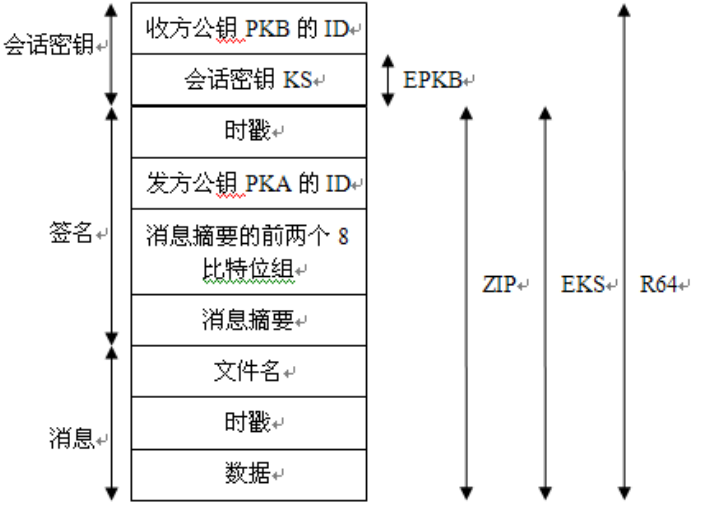
（2）密钥识别符

- PGP在对消息加密的同时，还需用收方的公钥加密一次性会话密钥。从而使得只有收方能恢复会话密钥，进而恢复加密的消息。
- 如果收方只有一个密钥对(即公钥，私钥对)，即可直接恢复会话密钥。当有多个密钥对，如何知道会话密钥是用哪一个公钥加密呢?
- PGP采用的方式是使用公钥中64个最低有效位表示该密钥的ID，即公钥PKA的ID是PKAmod 2的64次方。由于64位足够长，因此不同密钥的ID相重的概率很小。PGP在数字签名时也将密钥加上识别符，收方必须知道使用发方的哪个公钥来验证签名。

密钥和密钥环

（3）PGP的消息格式

- PGP的消息由三部分组成：消息、消息的签名（可选）会话密钥（可选）
- 消息部分包括被存储或被发送的实际数据、文件名以及时戳，时戳表示产生消息的时间。



密钥和密钥环

（3）PGP的消息格式

- 签字部分包括以下成分：

1）时戳：产生签字的时间。

2）消息摘要：消息摘要是SHA对消息签字的时戳与消息本身连接得到160比特输出后再由发方用私钥签字的结果。

3）消息摘要的前两个8比特位组：接收方利用解密消息摘要后得到的前两个8比特为组来确定自己在验证发方的数字签名时是否正确地使用了发方的公钥。

4）发方公钥的ID：用于表示解密消息摘要(即验证签名)的公钥，相应地也代表了签字的私钥。

密钥和密钥环

（4）密钥环

- 为有效存储、组织密钥，同时也为了便于用户的使用，PGP为每个节点(即用户)都提供了两个表型的数据结构，一个用于存储用户自己的密钥对(即公私钥对)，另一个用于存储该用户所知道的其他各用户的公钥，这两个数据结构分别称为私钥环和公钥环。

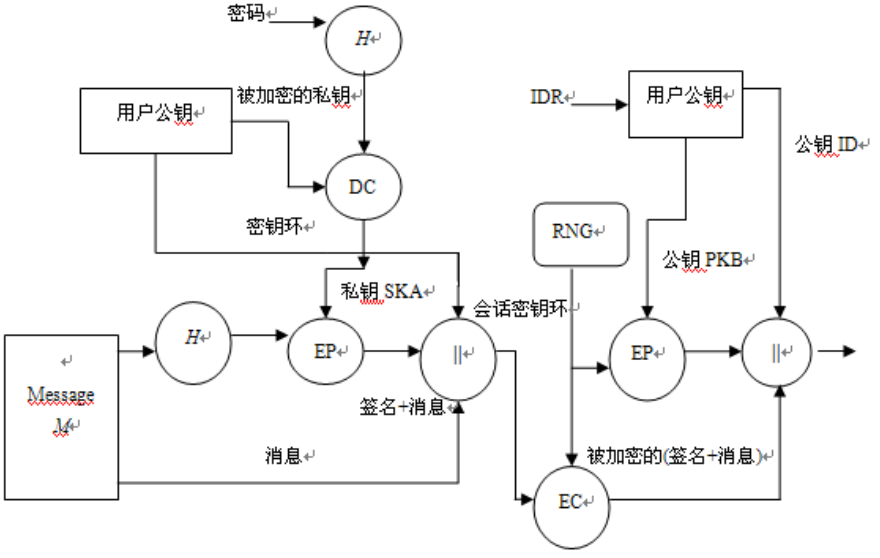
时截	公钥 ID*	公钥	被加密的私钥	用户 ID*
Ti	Pki mod 2 的 64 次方	PKi	Ski	USERi

(a) 私钥环

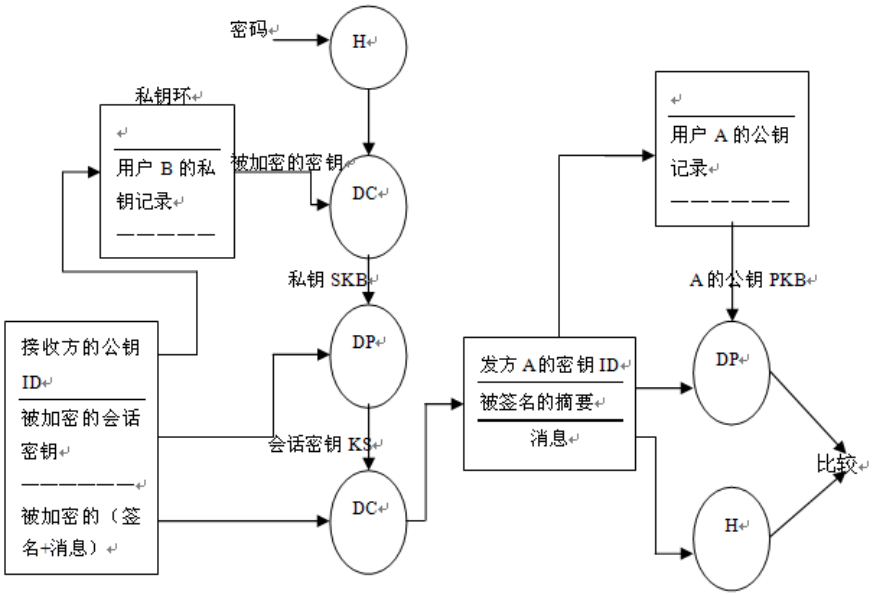
时戳	密钥 ID*	公钥	拥有者可信字段	用户 ID*	密钥合法性字段	签字	签字可信字段
Ti	Pki mod 2 的 64 次方	PKi	Trust flagi	USERi	Trust flagi		

(b) 公钥环

PGP消息的产生过程


密码
H
用户公钥
被加密的私钥
DC
密钥环
私钥 SKA
H
EP
||
Message
M
签名+消息
消息
会话密钥环
IDR
用户公钥
公钥 ID
RNG
公钥 PKB
EP
||
被加密的(签名+消息)
EC

PGP消息的接收过程


密码
H
私钥环
用户 B 的私钥记录
被加密的密钥
DC
私钥 SKB
DP
接收方的公钥 ID
被加密的会话密钥
被加密的（签名+消息）
会话密钥 KS
DC
发方 A的密钥 ID
被签名的摘要
消息
用户 A 的公钥记录
A 的公钥 PKB
DP
比较
H

S／MIME

- S／MIME(安全／多用途因特网邮件扩展)是基于RSA数据安全公司技术、MIME邮件格式标准的安全增强版。
- 其安全功能有：

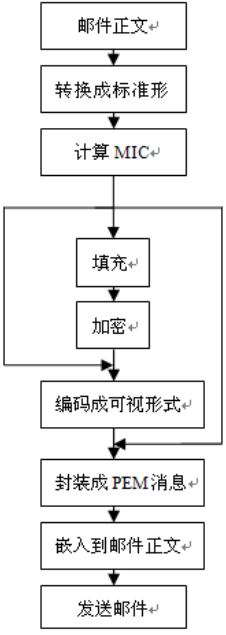
（1）数据的封装：被封装的数据包括被加密的消息内容和被加密的消息加密密钥，封装过程与PGP类似，只是没有压缩过程。

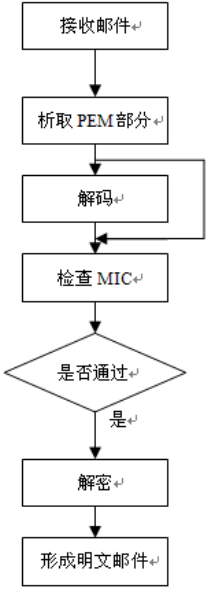
（2）数据的签字：发方用自己的私钥对消息摘要进行签字。然后，将签字和消息内容一起经BASE64编码后发给对方，仅当收方也具有S／MIME的安全功能时，才能解读消息。

PEM(Privacy Enhanced Mail)

- PEM是基于X.509v1提出的一个专用于加密E-mail通信的正式因特网标准。
- PEM使用两级密钥：数据加密密钥(DEK)和交换密钥(IK)。
- DEK用来加密消息正文和计算消息集成校验(MIC)，同时用来加密MIC的签名表示。DEK在每次会话中重新生成一个，从而达到一次一密的效果。而IK用来加密DEK，以便在每次会话的初始段对DEK进行加密交换。加密DEK的IK是收方的公钥。加密MIC的IK是发方的私钥，实现对MIC的签名。

PEM(Privacy Enhanced Mail)





电子邮件安全的防范措施

- 防范欺诈邮件
- 欺诈性邮件一般多出现在**HTML**格式的电子邮件中。如果接收到的邮件格式是**HTML**，那么它就会以**HTML**格式在您的电子邮件客户端显示出来，这就意味着电子邮件客户端也存在着和**Web**浏览器一样受到某种**HTML**威胁的隐患。
- 有些欺诈性的电子邮件内容是某个网站需要您查询一些信息，并给您一个链接，点击这个链接之后会出现可疑的登录界面。在这类邮件里，发件人通常是假冒的邮件地址。

电子邮件安全的防范措施

- 防范危险附件
- 每一份电子邮件附件都是对计算机安全性的潜在威胁。因此用户应该对预料之外的附件保持警惕，无论该附件来自何处，即使看上去像某个熟悉并可靠的联系人发送的。无论附件的来源是什么，除非它通过了最新的反病毒程序检查，否则，不要在电子邮件程序中运行或打开该附件。应该把自己反病毒程序设置成当文件被创建或保存时扫描该文件，也可以在打开附件之前手动地用反病毒程序对附件进行扫描。

电子邮件安全的防范措施

- 安全使用基于Web的电子邮件
- HOTMAIL，YAHOO、163等这些基于Web的电子邮件系统为广大用户提供了一个方便发送和接收电子邮件的途径。许多基于Web的电子邮件系统都提供了一个“记住”用户名和密码的功能。如果在公用计算机上错误地选择了简易登录选项，那么其他人都会很容易地访问到用户的密码和账号。所以要注意两点：确保系统不会把用户的登录证书保存在缓存中；不使用电子邮件系统，要确保退出登录。

电子邮件安全的防范措施

- 加强电子邮件保密性
- 在发送邮件时，电子邮件并不是直接发送到了对方的电子邮件信箱里，而是会经过数量不可预知的中间服务器。任何人只要能访问到该路径上的任何服务器，就都可以读到正在传输的消息内容。
- 和纸质邮件一样，电子邮件的传送也与距离有关，两个电子邮件信箱之间的中间服务器节点越少，被人偷看的可能性就越低。因此对于需要保密的邮件，采用数字证书帮助安全发送是最常见的方法。

电子邮件安全的防范措施

- 邮箱炸弹的防范
- 邮件炸弹的原理是向有限容量的信箱投入足够多或足够大的邮件使邮箱崩溃。这类邮件炸弹很多，如**Nimingxin**、**Quickfyre**、**Amail**、雪崩等，炸弹邮件的使用也很简单填上收信人的**E- mail**地址、输入要发送的次数、选择**SMTP**主机、随意填上自己的地址，按“发信”就开始发送炸弹了。
- 使用如下方法来尽可能地避免邮件炸弹的袭击：

1）不随意公开自己的信箱地址；

2）隐藏自己的电子邮件地址。

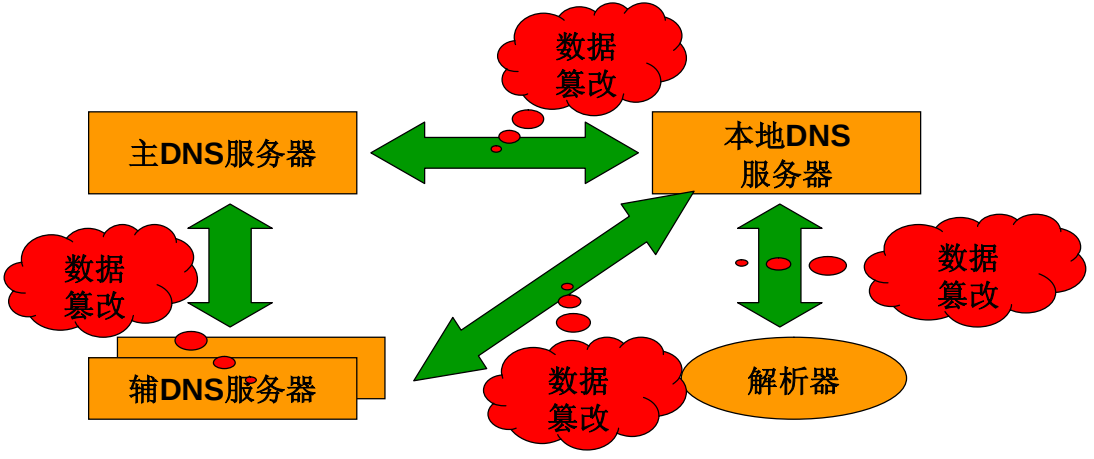
3）谨慎使用自动回信功能

8.4 DNS的安全性

- DNS劫持又称域名劫持，是指在劫持的网络范围内拦截域名解析的请求，分析请求的域名，把审查范围以外的请求放行，否则返回假的IP地址或者什么都不做使请求失去响应，其效果就是对特定的网络不能反应或访问的是假网址。
  - 2009年 巴西最大银行遭遇DNS攻击，1%用户被钓鱼
  - 2010年1月12日 “百度域名被劫持”事件
  - 2012年 日本邮储银行、三井住友银行和三菱东京日联银行各自提供的网上银行服务都被钓鱼网站劫持

DNSSEC应用背景

- DNS是Internet基础协议中至关重要的一个；
- DNS设计时没有考虑安全问题；
- 互联网的迅速发展，安全性很重要。



DNSSEC概念与原理

DNSSEC（DNS Security Extension）通过为DNS中的数据添加数字签名信息，使得客户端在得到应答消息后可以通过检查此签名信息来判断应答数据是否权威和真实，从而为DNS数据提供数据来源验证和数据完整性检验，可以防止针对DNS的相关攻击。

DNSSEC功能：

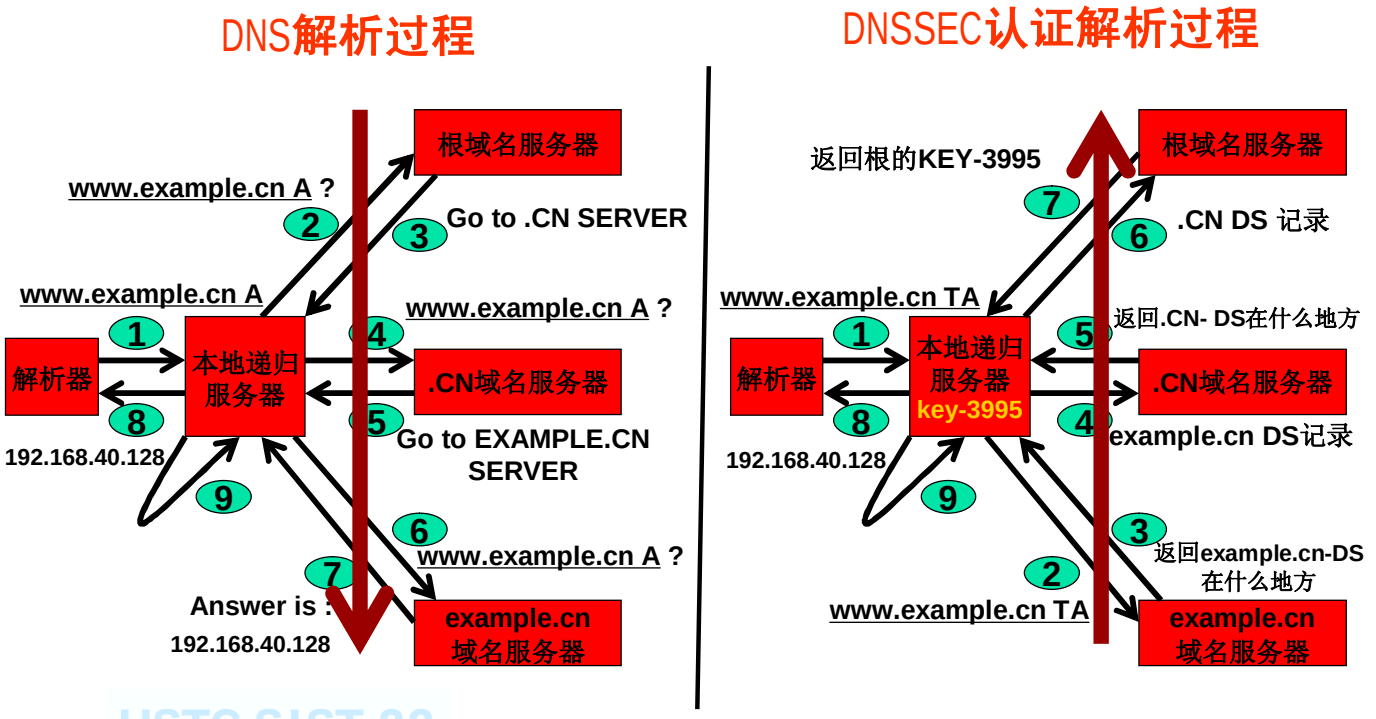
- 为DNS数据提供来源验证
- 为数据提供完整性验证
- 为查询提供否定存在验证
  - 即为否定应答消息提供验证，确认授权服务器上不存在所查询的资源记录

DNSSEC引入的新资源记录

- **DNSKEY**，用于存储验证**DNS**数据的公钥
- **RRSIG**，用于存储**DNS**资源记录的签名信息
- **NSEC**，存储和对应的所有者相邻的下一个资源记录；主要用于否定存在验证。
- **DS**（**Delegation Signer**，授权签名者），用于**DNSKEY**验证过程，存储密钥标签，加密算法和对应的**DNSKEY**的摘要信息。

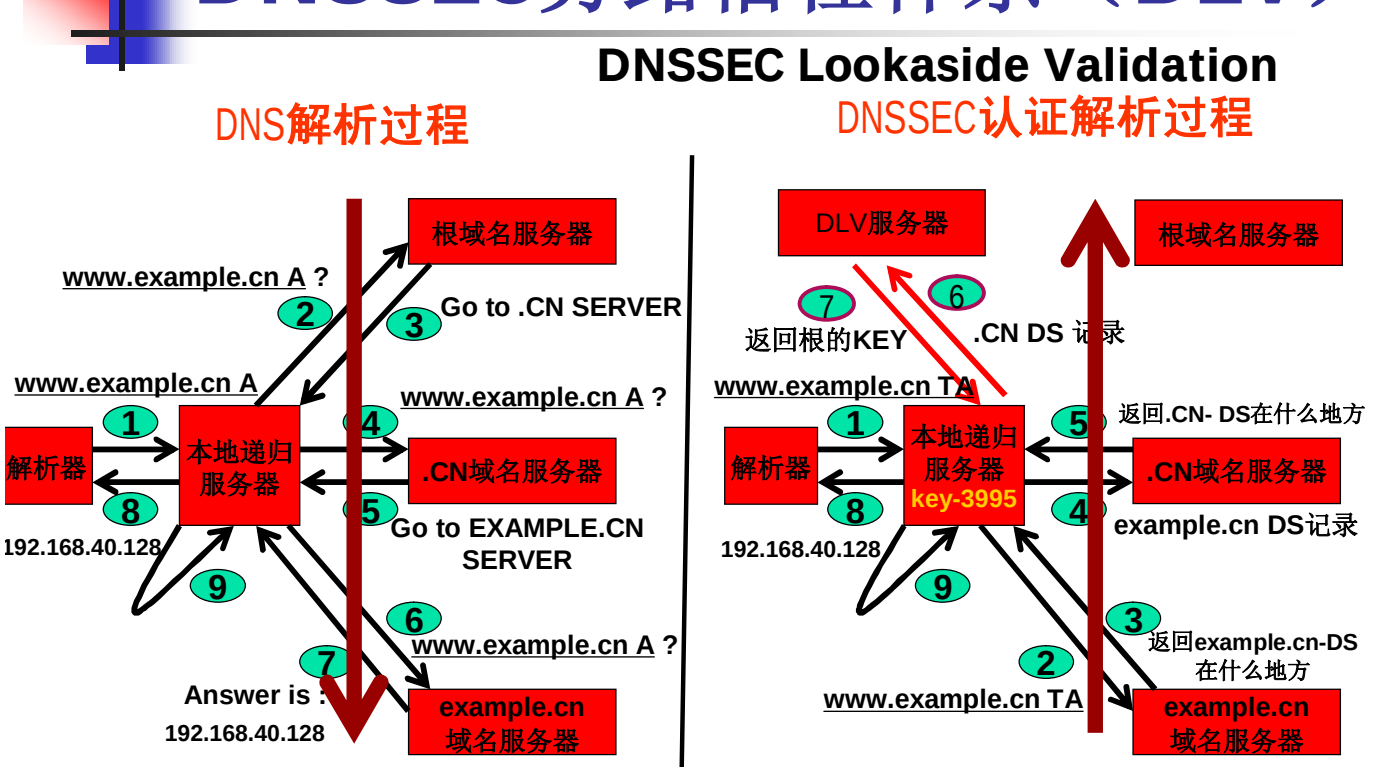
```
example.cn.              21600 DNSKEY  256 3 5 (
                               AwEAAdiwTbM4np7Qbi3la/HHs6tuHs06BN21
                               GeIOUGkmhg9F89YNGVqnEwSC3+3QU9dD
                               ) ; key id = 3525
test1.example.cn.   21600      IN A     192.168.40.10
                    21600      RRSIG    A 5 3 21600 20090502050309 (
                               20090402050309 3525 example.cn.
                               EsOUYz2nwxmeHbQpHZqUR4mLfA2jHPW0qrAd
                               VJlbnRwFZ/XjKVHGdUssvFYTyHQ= )
test1.example.cn.   21600      NSEC     test2.example.cn. A RRSIG NSEC
                    21600      RRSIG    NSEC 5 3 21600 20090502050309 (
                               20090402050309 3525 example.cn.
                               mfLTL9oAErJytYRmKrrXfGjyCrt8Hhr91dxO
                               52Q6f5zRoiRYIrFij6gzFAM72dVcnHvrA2U4
                               S2lFZVB9tyELoeKCBvl/Y0NrU1A= )
example.cn.  IN  DS  36379 5 1 6AC6B2B09994F446266BC8EFB817522BC6DB6597
example.cn.  IN  DS  36379 5 2 100EF6435DD98ECC499D19172AD349C7DC0EB08A
                               8C0D61F865FFCD8E B19F6569
```

DNSSEC的完整域名解析过程



DNSSEC旁路信任体系（DLV）

DNSSEC Lookaside Validation



DNSSEC部署过程

1、生成公/私密钥对

2、ZONE FILE的签署

3、DNS服务器配置

4、DNSSEC验证

（以BIND软件、example.cn域名举例说明）

DNSSEC部署过程—密钥对生成

dnssec- keygen -a RSASHA1 -b 1024 -n ZONE *zonename*

- **DNSSEC标准中指定使用非对称密钥来生成和验证签名;**
- **参数a表示使用的加密算法（三种算法:DSA/RSA/椭圆曲线DSA）**
- **参数b用来制定密钥长度;**
  密钥长度需要考虑到密钥的可靠性和性能，
  以及如何根据需要在两者之间取得平衡。
- **参数n指定密钥类型（ZONE/HOST）**
- **“*zonename*”密钥的名字（密钥的所有者）**

```
[root@linux1 geh]# dnssec-keygen -a RSASHA1 -b 1024 -n ZONE example.cn
Kexample.cn.+005+03525
[root@linux1 geh]# ls -rtl
-rw-------  1 root root  937  4月  2 09:48 Kexample.cn.+005+03525.private
-rw-r--r--  1 root root  210  4月  2 09:48 Kexample.cn.+005+03525.key
```

.key公有密钥

.private私有密钥

DNSSEC部署过程—密钥对生成

- 根据DNSSEC部署经验，至少需要两种类型密钥才能地对 DNSSEC域区进行安全的管理
- ZSK（Zone-Signing Key）和KSK（ Key-Signing Key）

1、区签名密钥（ZSK）——用于签名域区内的数据

2、密钥签名密钥（KSK）——用于签名ZSK并创建区的“安全入口点”

```
dnssec- keygen  -a RSASHA1  -b 1024  -n ZONE  – f KSK  zonename
```

DNSSEC部署过程—文件签署

签署之前，先将KSK和ZSK公有密钥添加到域区文件中！

$INCLUDE到*zonefile*或者

cat K*zonename*+<alg>+<fing>.key >> zonefile

dnssec- signzone -o *zonename* –f *result.file* [-N INCREMENT] [-k KSKfile] [-t] *zonefile* [ZSKfile]

- **签名工具BIND自带**
- **参数o指定所有域区文件起始**
- **-N INCREMENT序列号自增**
- **参数-K指定KSK文件名称**
- **“*zonefile*”被签名的zone文件**
- **ZSKfile制定ZSK文件名称**
- **-f 指定输出文件名称**
- **-t显示统计信息**

DNSSEC部署过程—文件签署

进行签名操作：

```
[root@linux1 geh]# dnssec-signzone -o example.cn. -N INCREMENT \
-k Kexample.cn.+005+36379.key example.zone Kexample.cn.+005+03525.key
example.zone.signed
Signatures generated:                       17
Signatures retained:                         0
Signatures dropped:                          0
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Runtime in seconds:                      0.090
Signatures per second:                 187.579
```

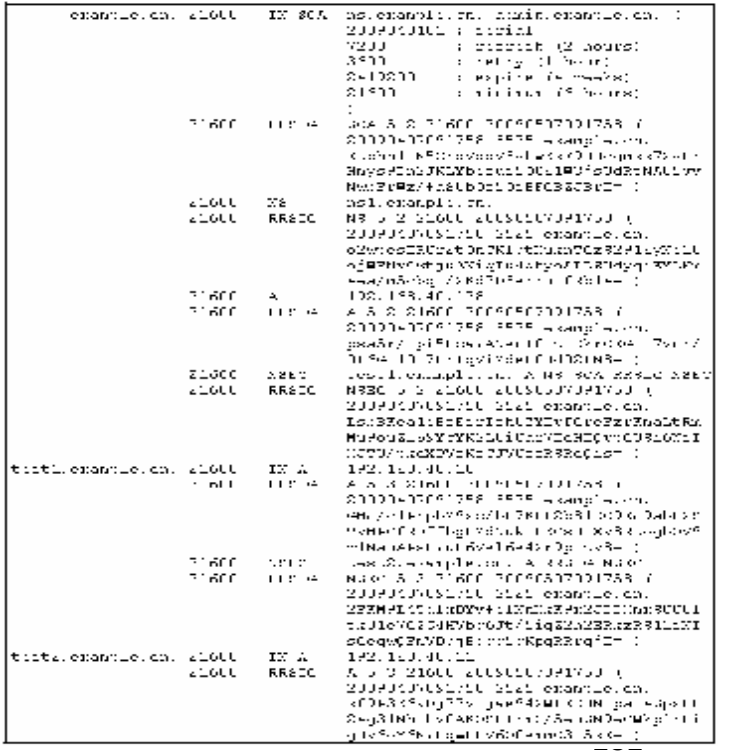
- 签名后输出文件为***zonefile*.signed**
- 签署域区的过程主要包括：
  - 将域区文件中的资源记录按照规则进行排序
  - 为域区中的每个资源记录生成一个**NSEC**记录
- 将生成的签名信息存储在相应的**RRSIG**记录中
- 签名后文件比原域区文件更大

DNSSEC部署过程—文件签署

原域区数据形式

```
$TTL 6h
@  IN SOA ns.example.cn. admin.example.cn. (
            2009040100    ; serial
            7200    ; refresh
            3600    ; retry
            2419200 ; expire
            21600   ; minimum
            )
example.cn.   IN      NS      ns1.example.cn.
example.cn.   IN      A       192.168.40.128
$ORIGIN example.cn.
test1   IN A 192.168.40.10
test2   IN A 192.168.40.11
test3   IN A 192.168.40.12
test4   IN A 192.168.40.13
test5   IN NS ns1.example.cn.
test6   IN CNAME 192.168.40.20
```

签名后域区数据形式



DNSSEC部署过程—解析软件配置

DNS解析软件配置——权威服务器

```
options {
        dnssec-enable yes;
};
```

加载签名后的区数据

- 更新**named.conf**文件

用:

```
zone "example.cn" {
        file "dir/example.zone.signed";
};
```

代替:

```
zone "zonename" {
        file "dir/example.zone";
};
```

- 告诉**named**重新读入配置

```
rndc reconfig
rndc flush
```

DNSSEC部署过程—解析软件配置

DNS解析软件配置——递归服务器

```
options {
        dnssec-enable yes;
        dnssec-validation yes;
};
```

```
trusted-keys { example.cn. 257 3 5 "AwEAAaRYHoXDS3yuqLaUR3JF2LFRCk3Ytb5V
                                    HibBiGjsKsiqpVeBUtzG9c1lM06n1Ka6F5rz
                                    xx85oI8bYYR2ISzOZsqnAFhANANwokonq363
                                    +9YXx82EeZYeG2VNDscnDNXsjRHvLuMRUWWy
                                    dj4ZSLSUyAbhE+XINvvjHCXkfEYxXd/+p+Ly
                                    FjjtrAUE0T6QvAd4JoNh97MNl2dLS3lsfk8M
                                    EY6gFXz1nNLJdS4DcYXS8AUewAdEi4sFpbVn
                                    +rORgGu4XlRHk++z4UCz7mM=";
};
```

DNSSEC部署过程—DNSSEC验证

用dig工具测试DNSSEC

- 用dig工具运行DNSSEC查询

```
; <<>> DiG 9.4.2-P2 <<>> @218.241.99.50 +dnssec gs.cn
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24963
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;gs.cn.                          IN      A

;; AUTHORITY SECTION:
gs.cn.                                                  a.dns.cn. root.cnnic.cn. 2009040708 7200 3600 2419200 21600
gs.cn.                  10800   IN      RRSIG   SOA 5 2 21600 20090507061105 20090407061105 25290 gs.cn. a/QUPkEgYF/B9YoEoZd6ACJs4NWOvu1F3YybhgSmtnzivSXw/yCtdl/6 9hYNyyIxsf/ZKTSgoMmUYUWU1qvsZd+Z5X8K453H1m6SQ9j2cWtJ6W7Z fsrElujtjVErOr1M3vvSAXahZx7uYG1mBqhGIBlG6UplemXUfqe7PdJx FqM=
gs.cn.                  10800   IN      NSEC    0.gs.cn. NS SOA RRSIG NSEC DNSKEY
gs.cn.                  10800   IN      RRSIG   NSEC 5 2 21600 20090507061105 20090407061105 25290 gs.cn. FjN8b/jd3vFSu8KGiOtiMXL3oiOokrCIZDUwZghcI1a5cb5G6C+b3DzP EhnXVAIi3GbR3UeO0XrAmPpLeMnmdTD2KLYdVxVapSt0AoeICfTo/u70 nJnFgFcUDXgGssAEZoJF7eUmQ/mtFIJaPNiwj4rOePl5MNA1KiH+V9AV dqo=

;; Query time: 12 msec
;; SERVER: 218.241.99.50#53(218.241.99.50)
;; WHEN: Tue Apr  7 18:26:11 2009
```

加上“+ dnssec”选项

在flags字段有“AD”选项

DNSSEC部署过程—DNSSEC验证

用dig工具调试DNSSEC

加上“+ cdflag”

```
; <<>> DiG 9.4.2-P2 <<>> @218.241.99.50 +dnssec +cdflag gs.cn
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49642
;; flags: qr rd ra ad cd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDIT
IONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;gs.cn.                          IN      A

;; AUTHORITY SECTION:
gs.cn.                  10517   IN      SOA     dns.cn. root.cnn
ic.cn. 2009040708 7200 3600 2419200 21600
gs.cn.                  10517   IN      RRSIG   SOA 5 2 21600 2009
0507061105 20090407061105 25290 gs.cn. a/QUPkEgYF/B9YoEoZd6ACJs4NW
Ovu1F3YybhgSmtnzivSXw/yCtdl/6 9hYNyyIxsf/ZKTSgoMmUYUWU1qvsZd+Z5X8K
453H1m6SQ9j2cWtJ6W7Z fsrElujtjVErOr1M3vvSAXahZx7uYG1mBqhGIBlG6Uple
mXUfqe7PdJx FqM=
gs.cn.                  10517   IN      NSEC    0.gs.cn. NS SOA RR
SIG NSEC DNSKEY
gs.cn.                  10517   IN      RRSIG   NSEC 5 2 21600 200
90507061105 20090407061105 25290 gs.cn. FjN8b/jd3vFSu8KGiOtiMXL3oi
OokrCIZDUwZghcI1a5cb5G6C+b3DzP EhnXVAIi3GbR3UeO0XrAmPpLeMnmdTD2KLY
dVxVapSt0AoeICfTo/u70 nJnFgFcUDXgGssAEZoJF7eUmQ/mtFIJaPNiwj4rOePl5
MNA1KiHHV9AV dgo=
```

签名信息

DLV旁路认证

- DLV——DNSSEC Lookaside Validation
- DLV是一个DNS服务器
- 提供DNSSEC签名认证的信任链解决方案
- 递归服务器配置的信任锚点是DLV，就可以认证DLV服务器的权威域，进而认证权威域授权的信任的权威域。

DNSSEC验证流程

1、在验证时，解析器在父区查找被验证区的DS记录

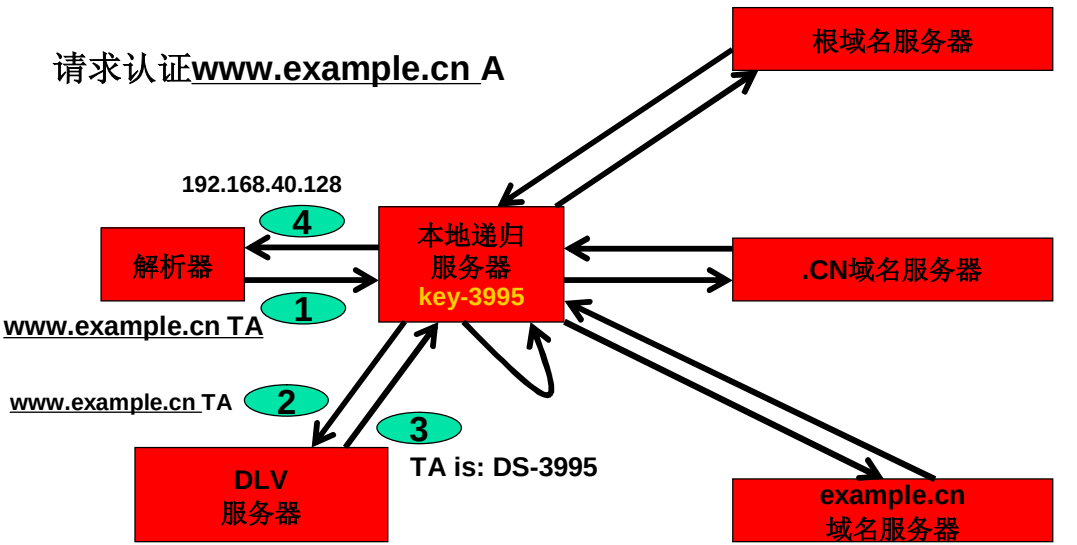
2、如果不存在，向DLV注册机构发出一个对DLV记录的请求

3、如果成功，DLV资源记录被当做这个区的DS记录

4、在递归服务器上进行验证

DLV旁路认证

DLV旁路认证过程



DLV旁路认证

- DLV服务器的配置
- 递归服务器的配置

```
options {
    dnssec-enable yes;
    dnssec-validation yes;
    dnssec-lookaside "." trust-anchor dlv.cnnic.cn.;
};
```

```
trusted-keys {  dlv.cnnic.cn 256 3 5 "AwEAAaCcjMurEO7zuIuNyNclEbcKNrQSLMJg
                                      utyLWlX2gzYB/kP/EVaNb0Cpji7ojvH3QTeE
                                      lxT0ja/ZVKKRavd74hdIH35Ihb7R4tl7W9SZ
                                      G4etqWdBQFQvvgEYiH4Dt1hWmREskBvnUmqt
                                      KYfXULfPHCIJilA0fQ68lab8pe4HzRZn";
};
```

面临的挑战

- 密钥更换、管理的挑战
  - **DNSSEC** 技术的目的之一是通过对数据进行数字“签名”来抵御此类攻击，从而使您确信数据有效。但是，为了从互联网中消除该漏洞，必须在从根区域到最终域名（例如， **www. icann. org** ）的查找过程中的每一步部署该项技术。对根区域进行签名（在根区域部署 **DNSSEC** ）是整个过程中的 必要步骤。需要说明的是，该技术并不对数据进行加密。它只是验证您所访问的站点地址是否有效。
- 网络设备和服务器性能的挑战
  - **DNSSEC** 为记录提供了验证途径。它不会加密数据或更改数据的管理方式，并且与当前的 **DNS** 和应用程序“向后兼容”。这意味着它不会更改 互联网 的寻址系统所基于的现有协议。它将一系列数字签名结合到 **DNS** 层次结构中，并使每个级别都拥有其自己的签名生成密钥。

相关参考资料

- **RFC3008 RFC3225 RFC3226 RFC4641 RFC4431 RFC5074**
  http://www.rfc-archive.org/getrfc.php?rfc=XXX
- **DNSSEC Training Course by www.ripe.net**
  http://www.ripe.org/training/dnssec/material/dnssec.pdf
- **The DNSSEC deployment Working Group home page**
  http://www.dnssec-deployment.org
- **DNSSEC in NL**
  http://www.miek.nl/publication/dnsseccn/index.html
- **DNS and BIND Talk Notes**
  http://www.tfug.org/helpdesk/general/dnsnotes.html
- **DLV Introduction**
  http://www.isc.org/ops/dlv
- http://www.dnssec.net
- http://www.nlnetlabs.nl/projects/dnssec/

8.5 网络管理安全

- SNMP全称是简单网络管理协议，安装这个组件之后，就可以通过MRTG等流量监控软件来监控服务器的流量、CPU、内存等关键信息。
- Multi Router Traffic Grapher - MRTG是一个监控网络链路流量负载的工具软件，通过SNMP协议得到设备的流量信息，并将流量负载以包含PNG格式图形的HTML 文档方式显示给用户，非常直观。

简单网络管理协议（SNMP）

- SNMP（Simple Network Management Protocol）是IETF的研究小组为了解决Internet上的路由器管理问题而提出的。IETF成立两个工作组，从两个方面定义了因特网管理的标准。管理信息库（MIB）工作组负责定义MIB内的元素及结构。网络管理协议（SNMP）工作组定义了不同厂商的设备之间交换的协议，为网络管理系统提供了底层网络管理的框架。目前的SNMP存在三个版本，即SNMPv1、SNMPv2 和SNMPv3。
- SNMP以轮询和应答的方式进行工作，采用集中或者集中分布式的控制方法对整个网络进行控制和管理。整个网络管理系统包括SNMP管理者、SNMP代理、管理信息库（MIB）和管理协议四个部分。

简单网络管理协议（SNMP）

- SNMP位于应用层，利用UDP的两个端口（161和162）实现管理员和代理之间的管理信息交换。UDP端口161用于数据收发，UDP端口162用于代理报警（即发送Trap报文）。
- SNMP协议可以使用在IP、IPX、AppleTalk以及传输协议上，为网络管理系统提供了底层网络管理的框架和统一协调机制。只要支持SNMP协议的设备或软件都可纳入管理的范围。
- SNMP协议的特点：
  （1）SNMP是分布式的管理协议。一个系统可以只作为SNMP管理器或SNMP代理中的单一角色，也可以同时完成这两者的功能。
  （2）SNMP是一种无连接协议。它通过使用请求报文和返回响应的方式，在SNMP代理和管理器之间传送信息。这种机制减轻了SNMP代理的负担，提供了一种独有的方式来处理可靠性和故障检测方面的问题。

简单网络管理协议（SNMP）

（3）SNMP协议应用广泛，可用于控制各种设备。比如说电话系统、环境控制设备，以及其它可接入网络且需要控制的设备等，这些非传统装备都可以使用SNMP协议。

（4）SNMP协议易实现。一个SNMP管理代理组件在运行时不需要很大的内存空间， SNMP协议在目标系统中开发迅速，它设计简单、扩展灵活、易于使用。

（5）SNMP协议有很多详细的文档资料，网络业界对这个协议也有着较深入的理解。这些都是SNMP协议进一步发展和改进的基础。

- 正是由于有了上述这些特点，SNMP协议已经被认为是网络设备厂商、应用软件开发者及终端用户的首选管理协议。

简单网络管理协议（SNMP）

SNMP的命令和报文

（1）SNMP的命令

SNMP协议定义了数据包的格式，及网络管理员和管理代理之间的信息交换，它还控制着管理代理的MIB数据对象。因此，可用于处理管理代理定义的各种任务。SNMP协议对外提供了三种用于控制MIB对象的基本操作命令，分别是Set 、Get 和 Trap 。

Set：是一个特权命令，可以通过它来改动设备的配置或控制设备的运转状态。

Get：是SNMP协议中使用率最高的一个命令，该命令是从网络设备中获得管理信息的基本方式。

Trap：功能就是在网络管理系统没有明确要求的前提下，由管理代理通知网络管理系统有一些特别的情况或问题发生。

简单网络管理协议（SNMP）

（2）SNMP的消息报文

- 在SNMP的三条命令的基础上，SNMP中定义了如下所示的五种消息类型（协议数据单元PDU）：
  - GetRequest
  - GetNextRequest
  - GetResponse
  - SetRequest
  - Trap

  SNMPv2中增加了两种PDU：GetBulkRequest和InformRequest 。

简单网络管理协议（SNMP）

- SNMP管理员使用GetRequest从拥有SNMP代理的网络设备中检索信息，
- SNMP代理以GetResponse消息响应GetRequest。
- 可以交换的信息很多，如系统的名字，系统自启动后正常运行的时间，系统中的网络接口数等。GetRequest和GetNextRequest结合起来使用可以获得一个表中的对象。GetRequest取回一个特定对象；而使用GetNextRequest则请求表中的下一个对象。使用SetRequest可以对一个设备中的参数进行远程配置。SetRequest可以设置设备的名字，关掉一个端口或清除一个地址解析表中的项目。
- Trap即SNMP陷阱，是SNMP代理发送给管理端的非请求消息。这些消息告知管理端本设备发生了一个特定事件，如端口失败，掉电重启等，管理端可相应的作出处理。

简单网络管理协议（SNMP）

（3）SNMP报文的基本格式

- 一条SNMP报文由三个部分组成: Version域（Version Field）、Community域（Community Field）和SNMP协议数据单元域（SNMP Protocol Data Unit Field）
  Version域：版本号域。用于说明现在使用的是哪个版本的SNMP协议。目前，Version1是使用最广泛的SNMP协议。

简单网络管理协议（SNMP）

- **Community域**：团体名或社区名域。用于实现**SNMP**网络管理员访问**SNMP**管理代理时的身份验证。**Community**（**community name**）是管理代理的口令，管理员被允许访问数据对象的前提就是网络管理员知道网络代理的口令。如果把配置管理代理成可以执行**Trap**命令，当网络管理员用一个错误的分区名查询管理代理时，系统就发送一个**autenticationFailure trap**报文。
- **SNMP PDU域**：协议数据单元域。指明了**SNMP**的消息类型及其相关参数。

SNMP网管设置及服务设置

SNMP网管代理设置

- 设置PC支持SNMP协议的网管代理，需要在机器中安装SNMP的服务。在Windows 2000以上的操作系统中，可作为标准服务安装。是否安装可以启动【控制面板】中的【管理工具】进行查看。
- Windows 7 开启SNMP服务
- snmp协议在windows下的安装与配置
- Configuring Net-SNMP

SNMP 应用工具

- **WebNMS SNMP Utilities**
  - Adventnet SNMP Utilities
- **iReasoning MIB browser**
  - Windows，MAC OS X 10.9 or later
  - Linux and other UNIX platforms
- **MIB Browser**
  - Windows
- **Net-SNMP**
  - available for many Unix and Unix-like operating systems and also for Microsoft Windows
- **MRTG - The Multi Router Traffic Grapher**
- **Linux SNMP Network Management Tools**