



第5章

实用的网络安全协议



第5章 内容概要

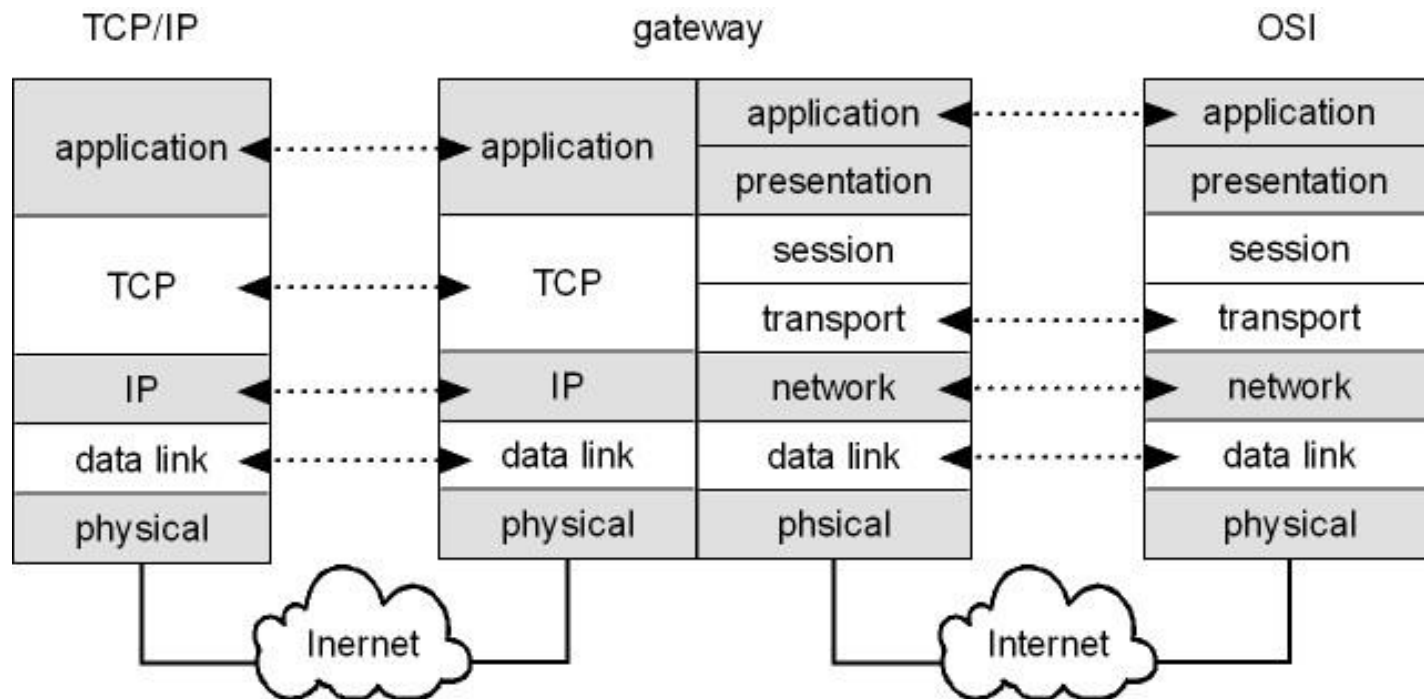
- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议



网络安全协议的构造

- 加密和认证算法是构造网络安全协议的基本模块
- 在不同的层次实施密码算法具有不同的效果
- 我们应该在网络体系的哪一个层次部署实施安全协议？

TCP/IP 协议和OSI模型



TCP/IP 协议层



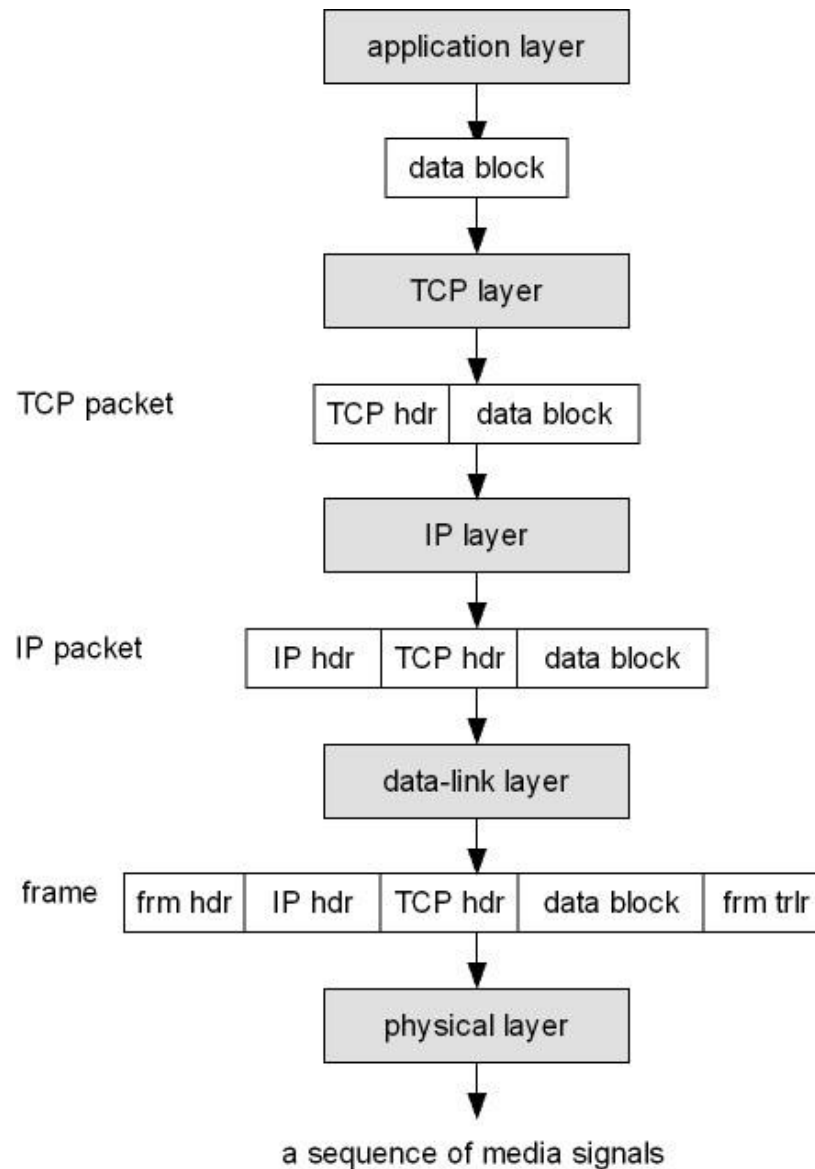
逻辑的(软件)

- 应用层
 - 网页, 电子邮件
- 传输层
 - TCP, UDP
- 网络层
 - IP

物理的(硬件)

- 数据链路层
 - 以太网, 802.11
- 物理层

TCP/IP 包生成





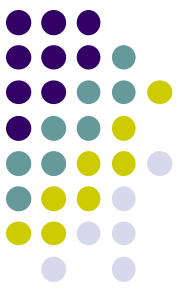
不同层次实施的优缺点？

● 应用层

- 提供端对端安全保护；数据在这一层加密和认证
- 不需要解密数据或验证签名（在其他层）
- 攻击者能够分析流量以及修改报头（**TCP/IP**包头没有加密或认证）

● 传输层

- 提供**TCP**包的安全保护（**TCP**载荷或整个包可以被加密或认证）
- 不需要修改应用程序（不影响接收应用层的数据）
- 攻击者可以通过**IP**包头分析网络流量（**IP**包头没有加密）



- 网络层

- 提供链对链的安全保护
 - 传输模式: 只加密载荷
 - 隧道模式: 加密包头和载荷, 需要网关
- 不需要修改任何应用程序

- 数据链路层

- 提供数据帧的安全保护 (帧的载荷被加密或认证)
- 不需要修改任何应用程序
- 流量分析不会泄露太多信息



第5章 内容概要

- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议



PKI公钥密码基础设施

- PKI 是使用公钥密码体系的机制
- PKI 负责签发、管理公钥证书:
 - 确定用户的合法性
 - 根据用户的需求颁发公钥证书
 - 根据用户的需求延长证书的有效期
 - 根据用户的需求或当私钥泄露时，撤销证书
 - 存储和管理公钥证书
 - 防止签名者抵赖自己的签名
 - 支持CA 中心之间实现互相认证

X.509 PKI体系 (PKIX)



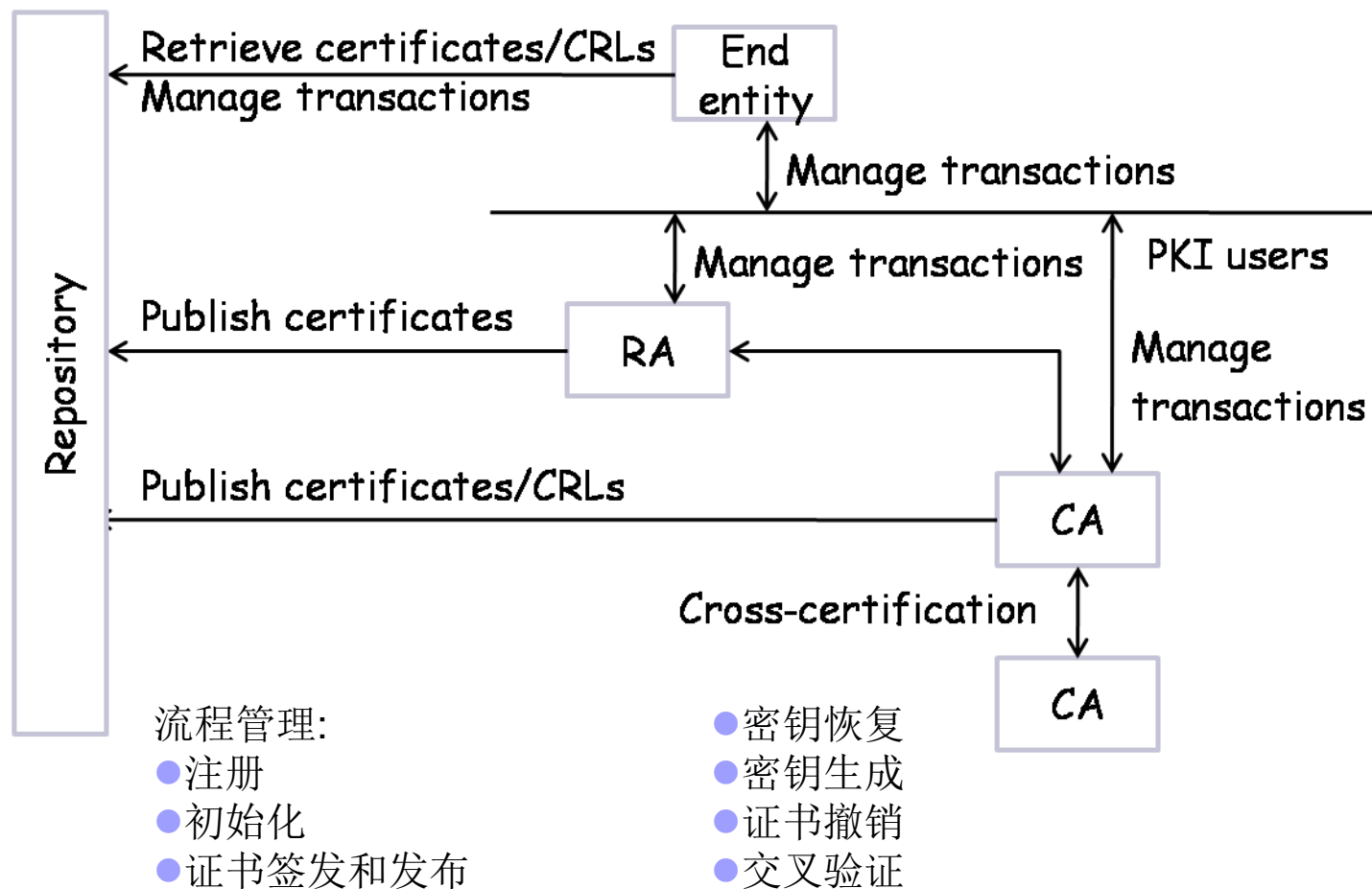
- IETF (Internet Engineering Task Force) 推荐
- 四个基本的组成:
 1. 终端实体(end entity)
 2. 证书机构(CA)
 3. 登记机构(RA)
 4. 证书库(repository)

X.509 PKI体系 (PKIX)



- 主要功能:
 - CA 负责签发和撤销公钥证书
 - RA负责验证公钥证书所有者的身份
 - 证书库负责存储和管理公钥证书和证书撤销列表 (CRLs)

PKIX 架构



X.509 证书格式

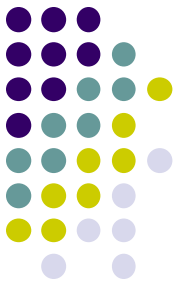


- *版本*: 证书使用的版本
- *序列号*: 证书唯一的编号
- *算法*: 密码散列函数和公钥密码算法的名称
- *签发者*: 签发者的名称
- *有效期*: 证书有效的时间段
- *用户名*: 证书拥有者的名字
- *公钥*: 用户的公钥和参数信息
- *扩展项*: 其它信息(版本3中使用)
- *数字签名*: 证书机构对证书散列值的签名



第5章 内容概要

- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 **IPSec协议: 网络层的安全协议**
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议



IPsec: 网络层协议

- IPsec 实现对IP包的加密和认证
- 包括3个协议:
 - 首部协议(AH)
 - 认证IP包的来源和完整性（同时认证IP包的首部和载荷）
 - 用滑动窗口防御消息重放攻击
 - 载荷安全封装协议(ESP)
 - 规定加密格式，用于加密和认证IP包
 - 互联网密钥交换协议(IKE)
 - 规定密钥交换格式，用于通信双方协商密钥
- 两种运行模式:
 - 传输模式
 - 隧道模式(需要网关)

IPsec 安全联盟(SA)



- 当 Alice 要与 Bob 建立 IPsec 连接, 双方首先需要协商使用的算法和密钥
- 安全联盟就是为实现以上目的
- 一个安全联盟(SA)在通信的发起者和响应者之间建立, 在一个会话阶段内有效
- 一个安全联盟(SA) 可以用于加密或者认证, 但不能同时用于两者
- 如果一个连接既需要加密又需要认证, 则需要建立两个安全联结(SA), 一个用于加密, 另一个用于认证



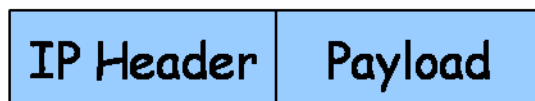
安全联盟(SA) 组成


- 三个参数:
 - 安全参数索引(SPI)：出现在IPsec 包头里，索引SADB中的SA
 - 目标IP 地址
 - 安全协议标识符：标明是为AH还是为ESP而设立的
- 安全联盟数据库(SAD)
 - 在本地主机上存储安全联盟
- 安全策略数据库(SPD)
 - 一组对IP包进行加密或认证的策略
- SA 选择器(SAS)
 - 指定每个安全联盟用于哪些IP包的规则

IPsec 包的组成



Normal IP Packet



 Unauthenticated Plain Text

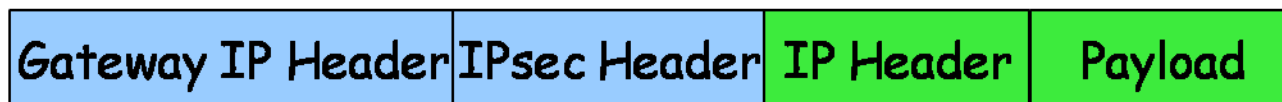
IPsec in Transport Mode



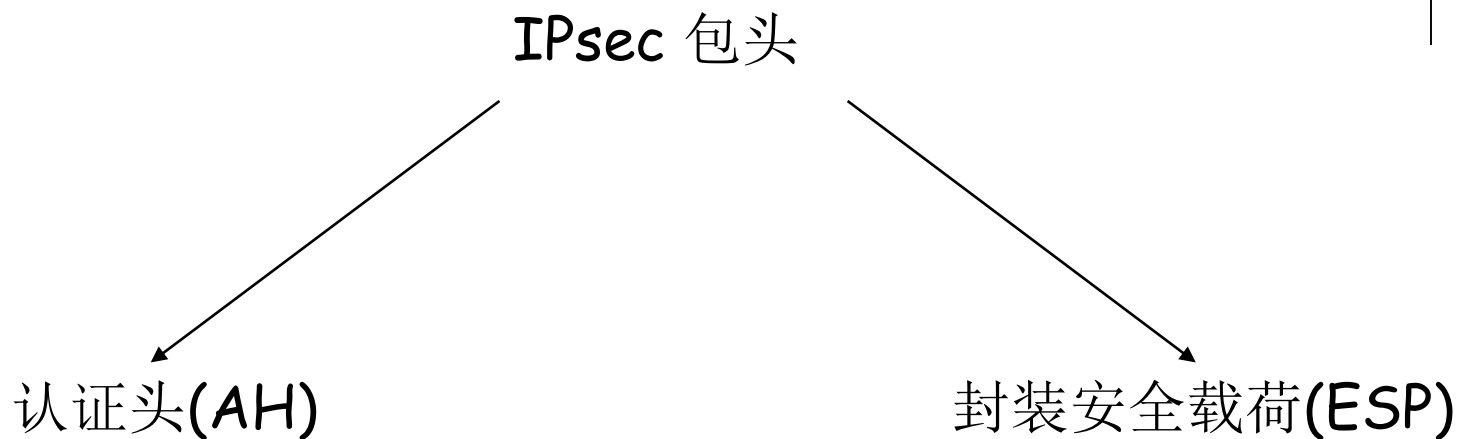
 Authenticated and/or Encrypted

IPsec in Tunnel Mode

- Single tunnel
- Nested tunnel



IPSec包头



认证和加密使用不同的安全联盟(SA)

认证头



0	8	16	31
next header	payload length	RESERVED	
security parameters index (SPI)			
sequence number			
integrity check value (variable length)			

抵御消息重放攻击



滑动窗口和序列号机制用于抵御消息重放攻击



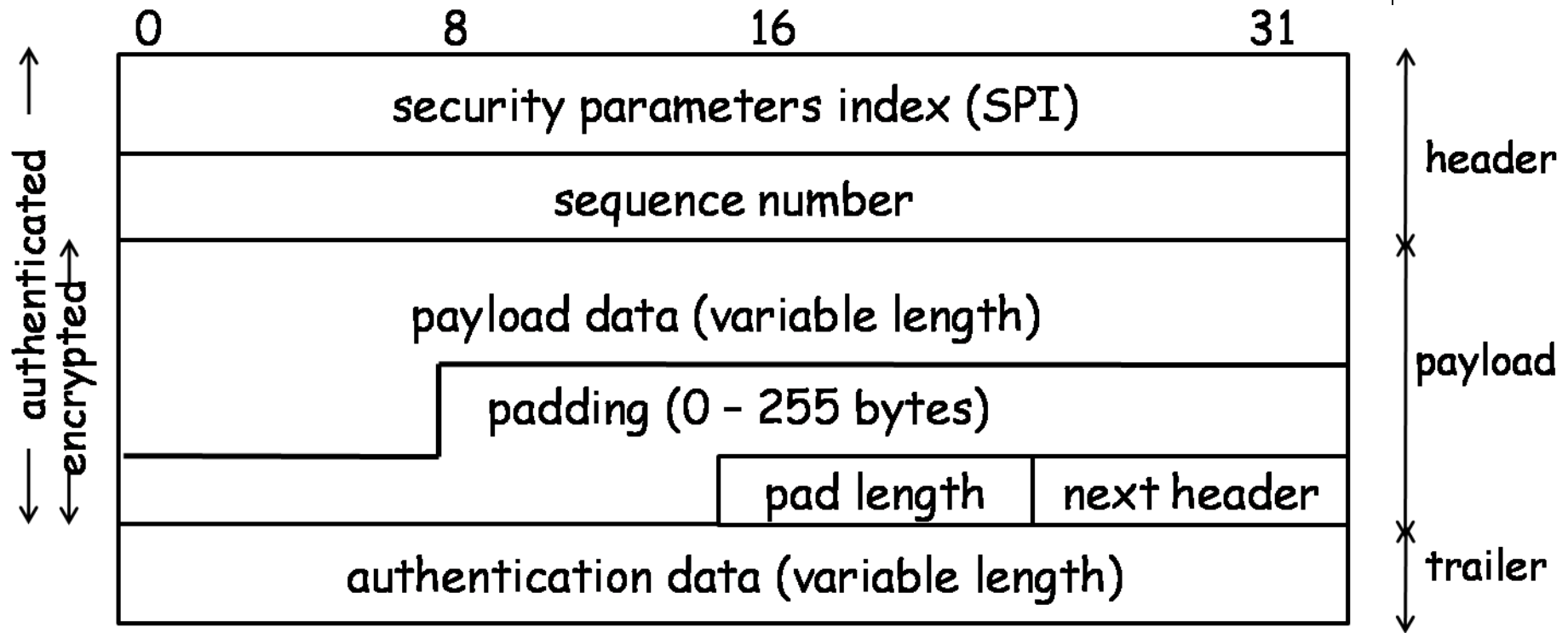
给定一个序列号为# s 的IP包, 有3种情况:

s 在A中 – 此IP包已经过时, 应丢弃

s 在B中 – 此IP包在窗口中, 检查其是否已经接收过

s 在C中 – 移动窗口后, 类似上面的窗口B的方法进行处理

封装安全载荷





密钥确定和分发

- Oakley 密钥确定协议(KDP)
 - Diffie-Hellman 密钥交换+认证 & cookies
 - 认证用于抵御中间人攻击
 - Cookies用于抵御阻塞攻击
 - Nonce(现时数技术)用于抵御消息重放攻击



阻塞攻击

- 一种拒绝服务攻击
- 攻击者在欺诈的IP包中发送大量的公钥 Y_i ，使得被攻击主机忙于进行大量的计算秘密密钥的运算 $K_i = Y_i^X \bmod p$
 - Diffie-Hellman密钥交换协议中的模幂运算是非常耗时的
- Cookies技术
 - 在计算之前, 接收方向发起方发送一个cookie (随机数), 并等待发起方发回包含此cookie的确认信息
 - 这可以防止攻击者通过修改源IP地址来伪造大量的DH请求包

ISAKMP



- ISAKMP: 互联网安全联结和密钥管理协议
 - 确定密钥交换的格式
 - 每一种载荷具有相同的载荷头的格式

64-bit initiator's cookie				
64-bit responder's cookie				
8-bit next payload	4-bit major ver	4-bit minor ver	8-bit exchange type	8-bit flags
32-bit message ID				
32-bit length				

ISAKMP 头



ISAKMP载荷类型

- **SA:** 建立安全联盟
- **提议:** 用于协商安全联盟的参数和算法等
- **传递:** 确定加密和认证的算法
- **密钥交换:** 确定密钥交换算法
- **身份认证:** 识别通信对方的身份
- **证书请求:** 请求公钥证书
- **证书:** 包含公钥证书
- **散列:** 包含散列值
- **签名:** 包含签名值
- **Nonce (现时数型):** 包含随机数 (现时数)
- **通知:** 通知其它类型负载的状态
- **删除:** 通知接收端, 发送端已删除一个或几个SA

8-bit 下一载荷	8-bit 保留	16-bit 载荷长度
---------------	-------------	----------------



第5章

实用的网络安全协议

Part II



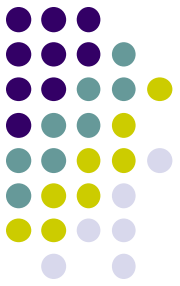
第5章 内容概要

- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 **SSL/TLS协议: 传输层的安全协议**
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议

SSL/TLS协议



- Secure Socket Layer Protocol (SSL)
 - 1994年由Netscape公司设计
 - 用于保护 WWW 应用和电子交易
 - Transport layer security protocol (TLS)
 - SSLv3修订版本
 - 两个主要组成部分:
 - 记录协议：在传输层协议的上方
 - 握手协议、密码更换协议、提醒协议：位于应用层协议和记录协议之间



SSL例子

- SSL上的HTTP协议 (https)
 - 在OSI模型的应用层
 - 用SSL实现
 - 加密HTTP包
 - 实现服务器与客户端之间的认证

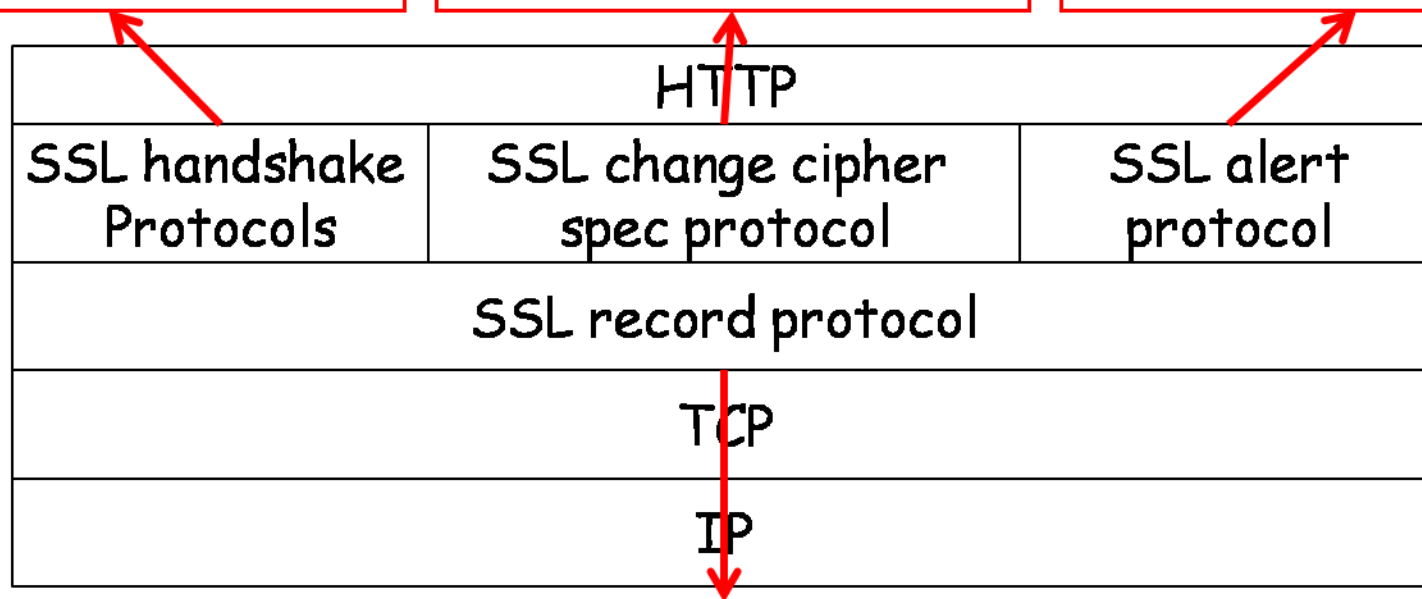
SSL结构



- Cryptographic algorithms
- A compression algorithm
- Parameters during exchange

Allow communicating parties to change algorithms or parameters during a communication session

- A management protocol
- Notify communicating parties when problems occur

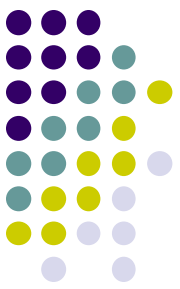


- Divide M into blocks
- Compress each block
- Authenticate, encrypt, add a record header to each block
- Transmit the resulting blocks

SSL握手协议



- 客户端和服务端协商将使用的密码算法，并进行密钥协商
- 实现相互的身份认证
- 四个阶段：
 - 选择密码算法
 - 客户端问候消息
 - 服务端问候消息
 - 服务器认证和密钥交换
 - 客户端认证和密钥交换
 - 完成握手



第1阶段(a): 客户端问候消息

包括以下信息:

1. 版本号, V_C :
 - 在客户端主机安装的SSL的最高版本号
 - 例如 $V_C = 3$
2. 伪随机串, r_c
 - 32字节字符串, 包括
 - 4 字节时间戳
 - 28 字节现时数
3. 会话ID, S_C
 - 如果 $S_C=0$, 则是新会话的新SSL连接
 - 如果 S_C 不等于0, 则是一个存在的会话的新SSL连接或是更新当前SSL连接的参数
4. 密码组: (公钥算法, 对称算法, 散列算法)
 - 例如 <RSA, ECC, Elgamal, AES-128, 3DES, Whirlpool, SHA-384, SHA-1>
 - 客户端支持的公钥密码算法、对称密码算法和密码散列函数的列表
5. 压缩算法
 - 例如 <WINZIP, ZIP, PKZIP>
 - 客户端支持的压缩算法的列表

第1阶段(b): 服务器问候消息



包括以下信息:

1. 版本号, V_s :

- $V_s = \min \{V_{\text{Client}}, V\}$
- 服务器安装的SSL的最高版本

2. 伪随机串, r_s

- 32字节字符串
 - 4 字节时间戳
 - 28 字节nonce

3. 会话ID, S_s

- 如果 $S_c=0$, 则 S_s = 新会话 ID
- 如果 $S_c \neq 0$, 则 $S_s=S_c$

4. 密码组: (公钥算法, 对称算法, 散列算法)

- 例如 <RSA,AES-128,Whirpool>
- 服务器支持的公钥密码算法、对称密码算法和密码散列函数的列表

5. 压缩算法

- 例如 <WINZIP>
- 服务器从客户端的压缩算法列表中选择

第2阶段（服务器认证和密钥交换）



服务器发送下面的信息给客户端:

1. 服务器的公钥证书
2. 服务器的密钥交换信息
3. 服务器对客户端证书的请求
4. 完成服务器问候

注意: 认证部分通常不实现（即第3步）

第3阶段



- 客户端回复以下信息给服务器:
 - 客户端的公钥证书
 - 客户端的密钥交换信息
 - 客户端的公钥证书的验证值
- 密钥交换信息用于生成主密钥
- 例如, 在第1阶段中, 服务器选择**RSA**作为密钥交换手段, 则客户端按下面方法进行密钥交换:
 - 验证服务器的公钥证书的签名
 - 获得服务器的公钥 K_s^u
 - 生成一个48字节的伪随机串 s_{pm} (预主秘密, pre-master secret)
 - 用 K_s^u 加密 s_{pm} , 并发送密文作为密钥交换信息给服务器

第3阶段 (继续)



- 第3阶段完成后，双方拥有 r_c , r_s , s_{pm} ，则服务器和客户端可以计算预主秘密 s_m ：

$$s_m = H_1(s_{pm} \parallel H_2('A' \parallel s_{pm} \parallel r_c \parallel r_s)) \parallel \\ H_1(s_{pm} \parallel H_2('BB' \parallel s_{pm} \parallel r_c \parallel r_s)) \parallel \\ H_1(s_{pm} \parallel H_2('CCC' \parallel s_{pm} \parallel r_c \parallel r_s))$$

第4阶段



- 客户端和服务端完成了握手协议.

- 双方用类似的方法计算秘密密钥块 K_b

$$\begin{aligned} K_b = & H_1(S_m \parallel H_2('A' \parallel S_m \parallel R_c \parallel R_s)) \parallel \\ & H_1(S_m \parallel H_2('BB' \parallel S_m \parallel R_c \parallel R_s)) \parallel \\ & H_1(S_m \parallel H_2('CCC' \parallel S_m \parallel R_c \parallel R_s)) \\ & \dots 'DDDD' \dots \\ & \dots \end{aligned}$$

- K_b 被分成6段, 每一段形成一个秘密密钥

$$K_b = K_{c1} \parallel K_{c2} \parallel K_{c3} \parallel K_{s1} \parallel K_{s2} \parallel K_{s3} \parallel Z \text{ (其中} Z \text{是剩余的字符串)}$$

- 将秘密密钥分成两组:

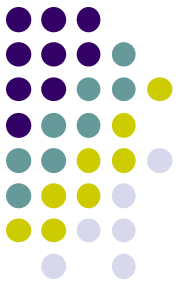
Group I: $(K_{c1}, K_{c2}, K_{c3}) = (K_{c,HMAC}, K_{c,E}, IV_c)$ (保护从客户端到服务器的通信)

Group II: $(K_{s1}, K_{s2}, K_{s3}) = (K_{s,HMAC}, K_{s,E}, IV_s)$ (保护从服务器到客户端的通信)

SSL记录协议-客户端



- 当建立了一个安全的通信会话后, 服务器和客户端将用SSL记录协议来保护通信
- 客户端进行以下步骤:
 - 将消息 M 分成一系列数据块 M_1, M_2, \dots, M_k
 - 对 M_i 进行压缩, 得到 $M_i' = CX(M_i)$
 - 对 M_i' 进行认证, 得到 $M_i'' = M_i' \parallel H_{Kc, HMAC}(M_i')$
 - 加密 M_i'' , 得到 $C_i = E_{Kc, HMAC}(M_i'')$
 - 封装 C_i , 得到 $P_i = [\text{SSL record header}] \parallel C_i$
 - 发送 P_i 给服务器

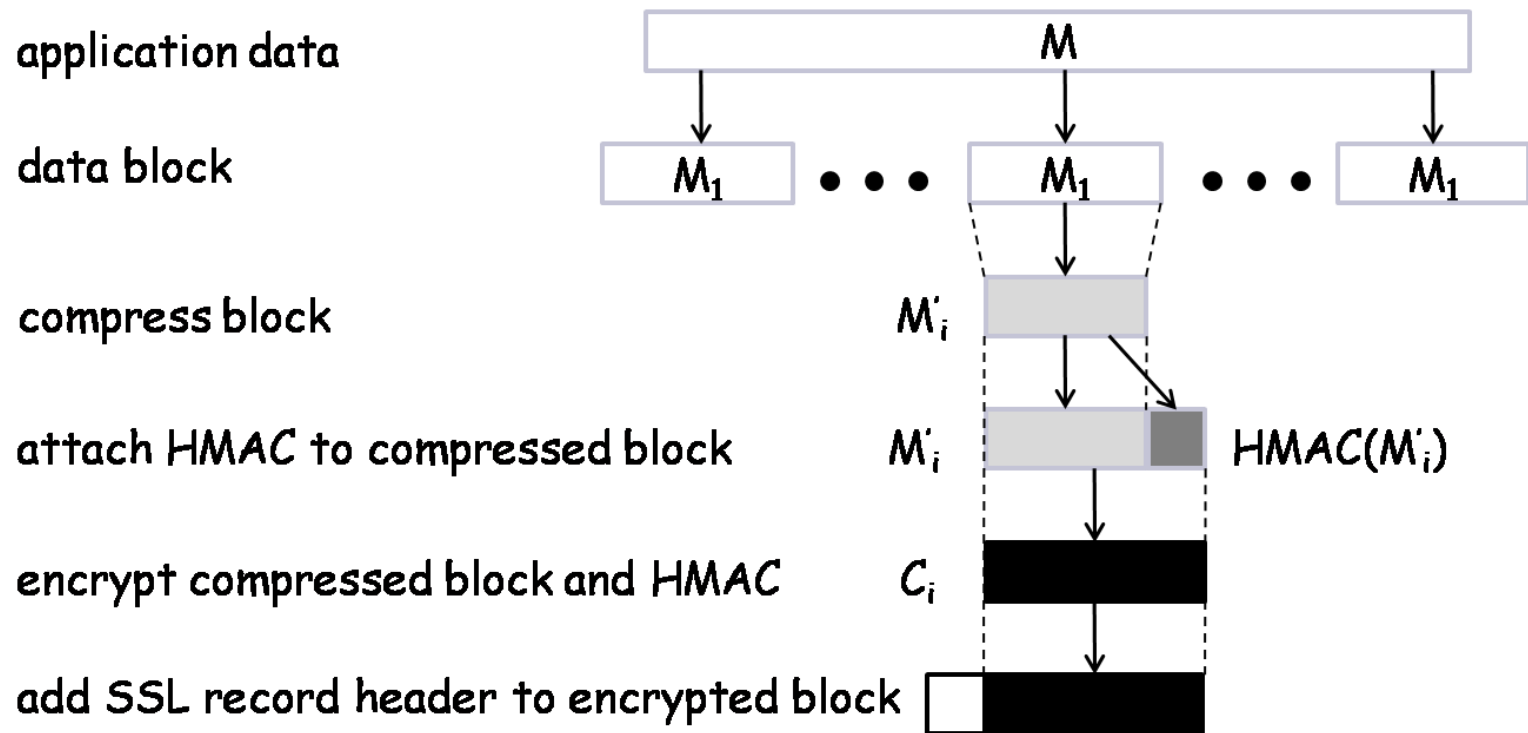


SSL记录协议-服务器

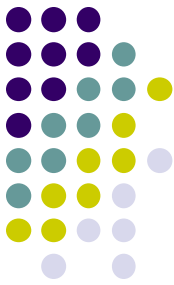
- 服务器进行以下步骤:
 - 从 P_i 中得到 C_i
 - 解密 C_i 得到 M_i'
 - 得到 M_i' 和 $H_{Kc,HMAC}(M_i')$
 - 验证认证码
 - 解压 M_i' 得到 M_i



SSL记录协议示意图



SSL记录协议



第5章 内容概要

- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议



基本的电子邮件安全机制

- Alice应向Bob证明消息 M 是她发送的
 - 发送 $M \parallel \hat{E}_{K_A^r}(H(M)) \parallel \text{CA}\langle K_A^u \rangle$ 给Bob用于认证, 其中 \hat{E} 表示公钥加密
- Alice应保证消息 M 在传输过程中是保密的
 - 发送 $E_{K_A}(M) \parallel \hat{E}_{K_B^u}(K_A)$ 给Bob
 - 接收到消息后, Bob首先解密 $\hat{E}_{K_B^u}(K_A)$ 得到 K_A
 - 然后, 用 K_A 解密 $E_{K_A}(M)$ 得到 M

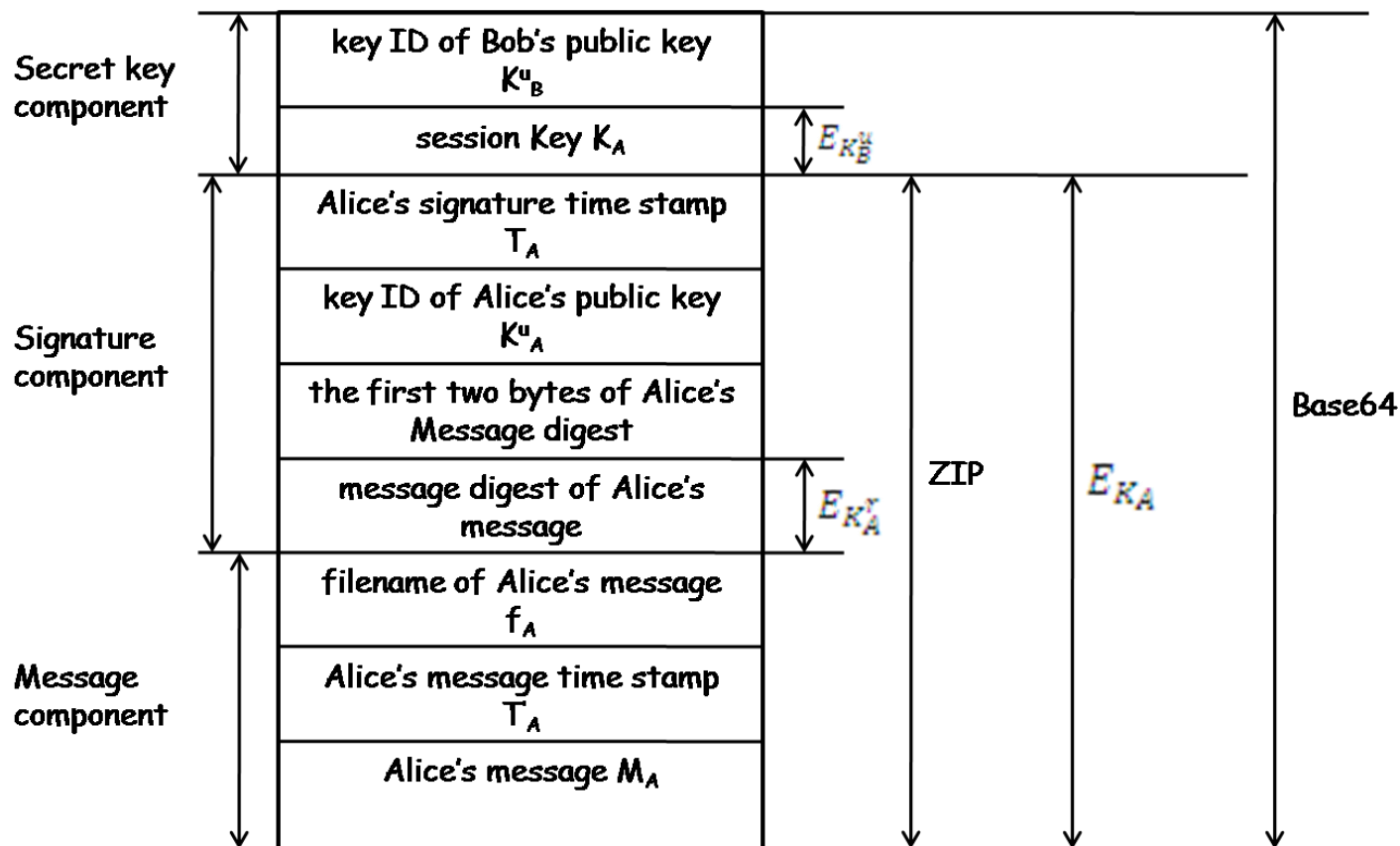
PGP



- PGP (Pretty Good Privacy)
 - 实现了主要的密码算法、ZIP压缩算法和Base64 编码算法
 - 可以用于认证和加密
 - 一般形式:
 - 认证
 - ZIP压缩
 - 加密
 - Base64 编码 (用于 SMTP传输)

PGP消息格式

发送方: **Alice**; 接收方: **Bob**



S/MIME



- Secure Multipurpose Internet Mail Extension
- 解决了PGP的不足
 - 支持多种格式的消息, 不只是ASCII码文本
 - 支持IMAP协议 (Internet Mail Access Protocol)
 - 支持多媒体
- 类似于PGP, 可以进行认证和加密;但要求签名者必须持有公钥证书
- 使用X.509 PKI 和公钥证书
- 支持标准的对称密码算法、公钥密码算法、数字签名算法、密码散列算法和压缩算法



第5章 内容概要

- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 SSH: 远程登录安全协议

Kerberos基础



- 目标:
 - 不使用**PKI**的方式，实现局域网中的用户认证
 - 允许用户访问服务，而不必每次都重新输入登录密码
- 使用对称加密和电子通行证， 又称为票据(tickets)
- 使用两种类型的票据:
 - **TGS**票据: 由**AS**颁发给用户
 - **V-票据** (服务器票据): 由**TGS**颁发给用户



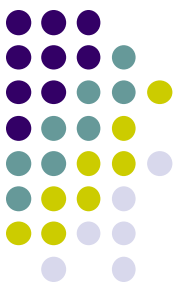
Kerberos服务器

- 需要两个特殊的服务器向用户发放票据:
 - AS: 认证服务器, 用于管理用户和用户认证.
 - TGS: 票据授予服务器, 用于管理服务器.
- 两个Kerberos协议(单网络 vs. 多网络)
 - 单域Kerberos
 - 多域Kerberos



Kerberos如何工作？

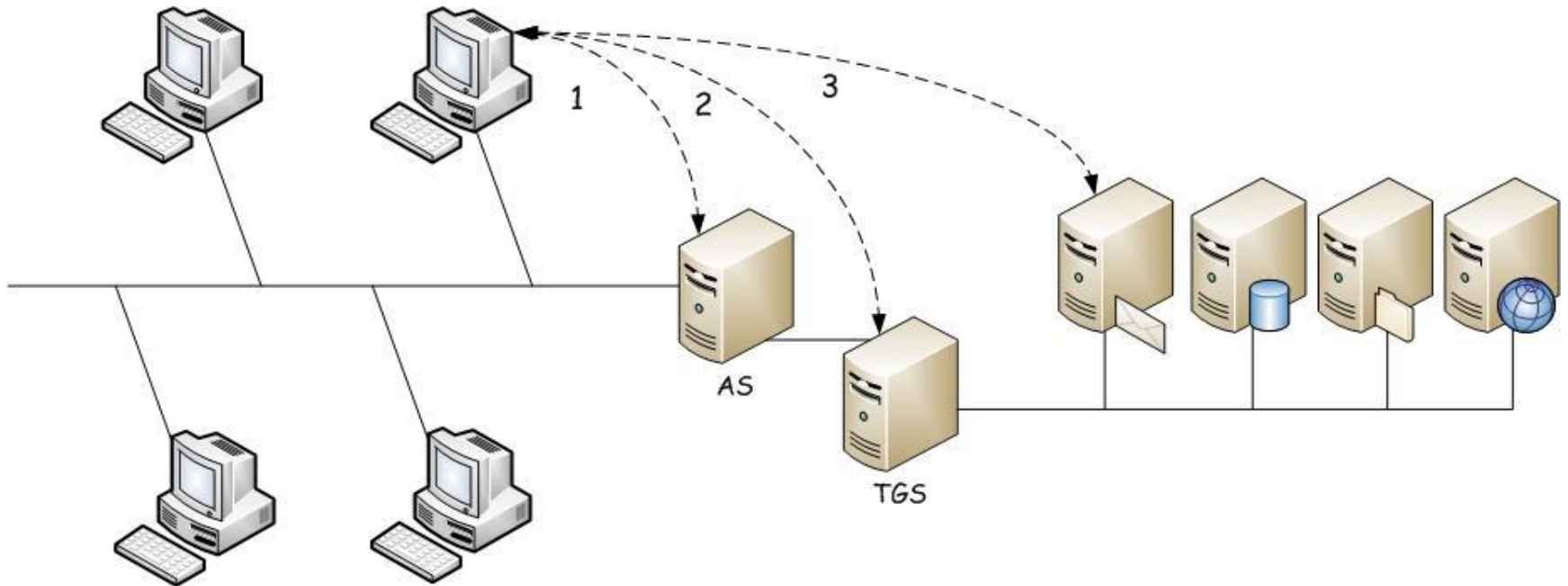
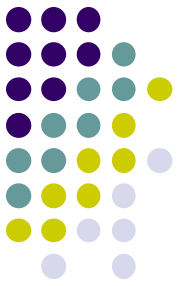
- 登录时，用户向认证服务器**AS**提供用户名和密码
- **AS**对用户进行认证，并提供一个**TGS**票据给用户
- 当用户要访问服务器**V**提供的某种服务时，用户向**TGS**服务器提供他的**TGS**票据
- **TGS**服务器对用户的**TGS**票据进行认证，然后向用户发放一个**V**票据 (服务器票据) 给用户
- 用户向服务器提供**V**票据，通过后访问服务器提供的服务



Kerberos协议的符号定义

Notation	Meaning
U	User
V	Server
ID_U	U 's ID
ID_{TGS}	TGS's ID
t_i	Time stamp
E_K	Symmetric-key encryption with secret key K
K_U	The secret key derived from user U 's password
$K_{U,TGS}$	The session key generated by AS to be used by U and TGS
K_{TGS}	The master key shared by AS and TGS
K_V	The master key shared by TGS and V
$K_{U,V}$	The session key generated by TGS to be used by U and V
LT_i	Expiration time
$Ticket_{TGS}$	TGS-ticket issued to U by AS
$Ticket_V$	Server ticket for using server V issued to U by TGS
AD_U	U 's MAC address
$Auth_{U,TGS}$	Authentication code generated using secret key $K_{U,TGS}$
$Auth_{U,V}$	Authentication code generated using secret key $K_{U,V}$

单域 Kerberos

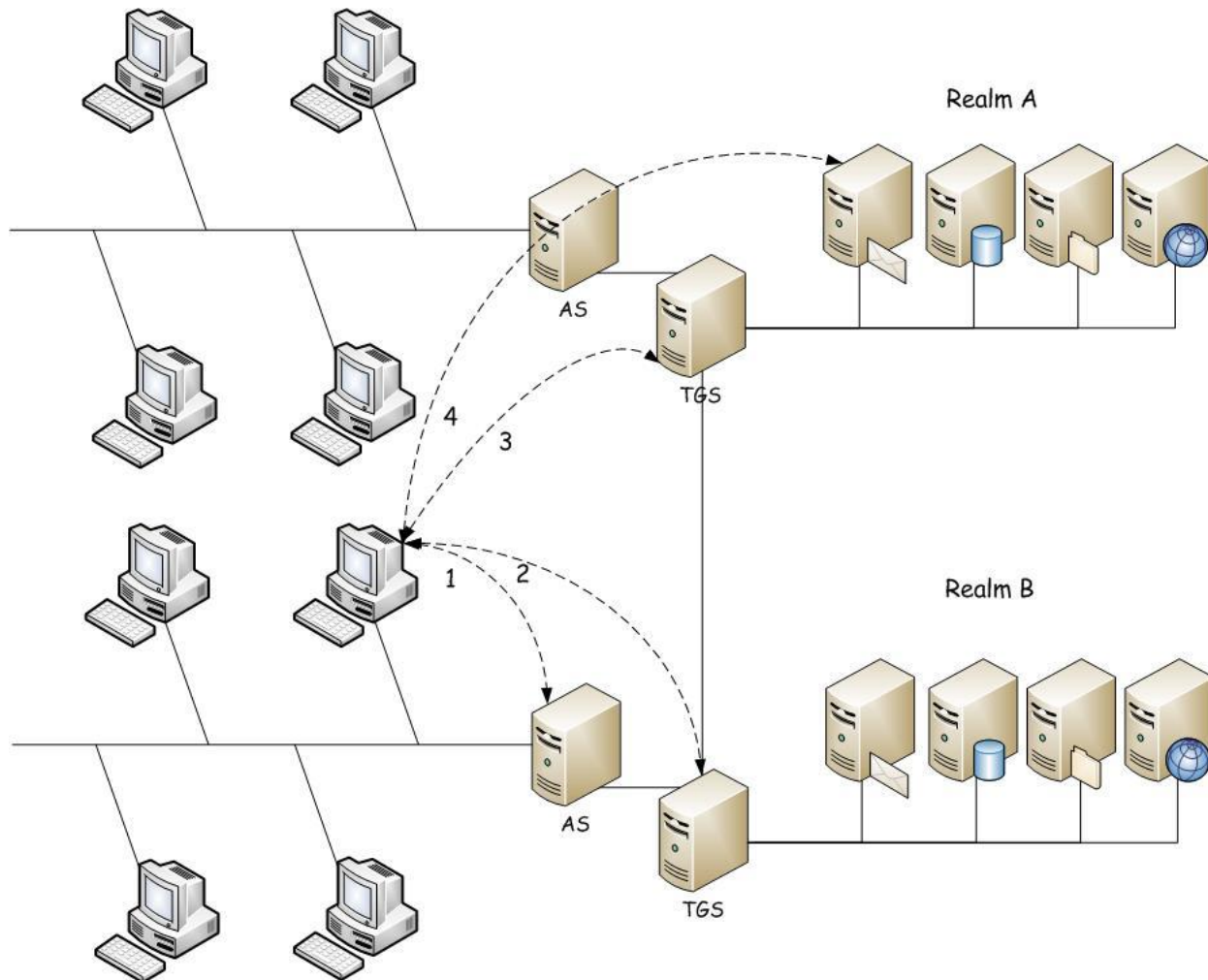
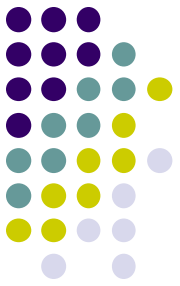


单域Kerberos的三个步骤

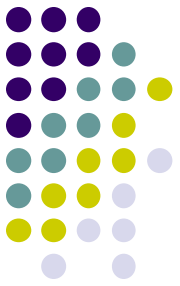


- Phase 1: AS 颁发一个TGS票据给用户
 1. $U \rightarrow AS: ID_U \parallel ID_{TGS} \parallel t_1$
 2. $AS \rightarrow U: E_{K_U}(K_{U,TGS} \parallel ID_{TGS} \parallel t_2 \parallel LT_2 \parallel Ticket_{TGS})$
 $Ticket_{TGS} = E_{K_{TGS}}(K_{U,TGS} \parallel ID_U \parallel AD_U \parallel ID_{TGS} \parallel t_2 \parallel LT_2)$
- Phase 2: TGS 颁发一个服务器票据给用户
 3. $U \rightarrow TGS: ID_V \parallel Ticket_{TGS} \parallel Auth_{U,TGS}$
 $Auth_{U,TGS} = E_{K_{U,TGS}}(ID_U \parallel AD_U \parallel t_3)$
 4. $TGS \rightarrow U: E_{K_{U,TGS}}(K_{U,V} \parallel ID_V \parallel t_4 \parallel Ticket_V)$
 $Ticket_V = E_{K_V}(K_{U,V} \parallel ID_U \parallel AD_U \parallel ID_V \parallel t_4 \parallel LT_4)$
- Phase 3: 用户获得请求的服务
 5. $U \rightarrow V: Ticket_V \parallel Auth_{U,V}$
 $Auth_{U,V} = E_{K_{U,V}}(ID_U \parallel AD_U \parallel t_5)$
 6. $V \rightarrow E_{K_{U,V}}(t_5+1)$

多域 Kerberos



《计算机网络的理论与实践（第2版）》. 【美】王杰, 高等教育出版社, 2011年.



多域Kerberos的四个步骤

- **Phase 1: 本域AS 颁发一个本域的TGS票据给用户**
 1. $U \rightarrow AS: ID_U \parallel ID_{TGS} \parallel t_1$
 2. $AS \rightarrow U:$
 $E_{K_U}(K_{U,TGS} \parallel ID_{TGS} \parallel t_2 \parallel LT_2 \parallel Ticket_{TGS})$
 $Ticket_{TGS} = E_{K_{TGS}}(K_{U,TGS} \parallel ID_U \parallel AD_U \parallel ID_{TGS} \parallel t_2 \parallel LT_2)$
- **Phase 2: 本域TGS 颁发一个邻域TGS票据给用户**
 3. $U \rightarrow TGS: ID_V \parallel Ticket_{TGS} \parallel Auth_{U,TGS}$
 $Auth_{U,TGS} = E_{K_{U,TGS}}(ID_U \parallel AD_U \parallel t_3)$
 4. $TGS \rightarrow U:$
 $E_{K_{U,TGS}}(K_{U,TGS'} \parallel ID_{TGS'} \parallel t_4 \parallel Ticket_{TGS'})$
 $Ticket_{TGS'} = E_{K_{TGS'}}(K_{U,TGS'} \parallel ID_U \parallel AD_U \parallel ID_{TGS'} \parallel t_4 \parallel LT_4)$

- **Phase 3: 邻域TGS颁发一个服务器票据给用户**
 5. $U \rightarrow TGS':$
 $ID_V \parallel Ticket_{TGS'} \parallel Auth_{U,TGS'}$
 $Auth_{U,TGS'} = E_{K_{U,TGS'}}(ID_U \parallel AD_U \parallel t_5)$
 6. $TGS' \rightarrow U:$
 $E_{K_{U,TGS'}}(K_{U,V} \parallel ID_V \parallel t_6 \parallel Ticket_V)$
 $Ticket_V = E_{K_V}(K_{U,V} \parallel ID_U \parallel AD_U \parallel ID_V \parallel t_6 \parallel LT_6)$
- **Phase 4: 用户从邻域服务器请求服务**
 7. $U \rightarrow V:$
 $Tickey_V \parallel Auth_{U,V}$
 $Auth_{U,V} = E_{K_{U,V}}(ID_U \parallel AD_U \parallel t_7)$
 8. $V \rightarrow U: E_{K_{U,V}}(t_7 + 1)$



第5章 内容概要

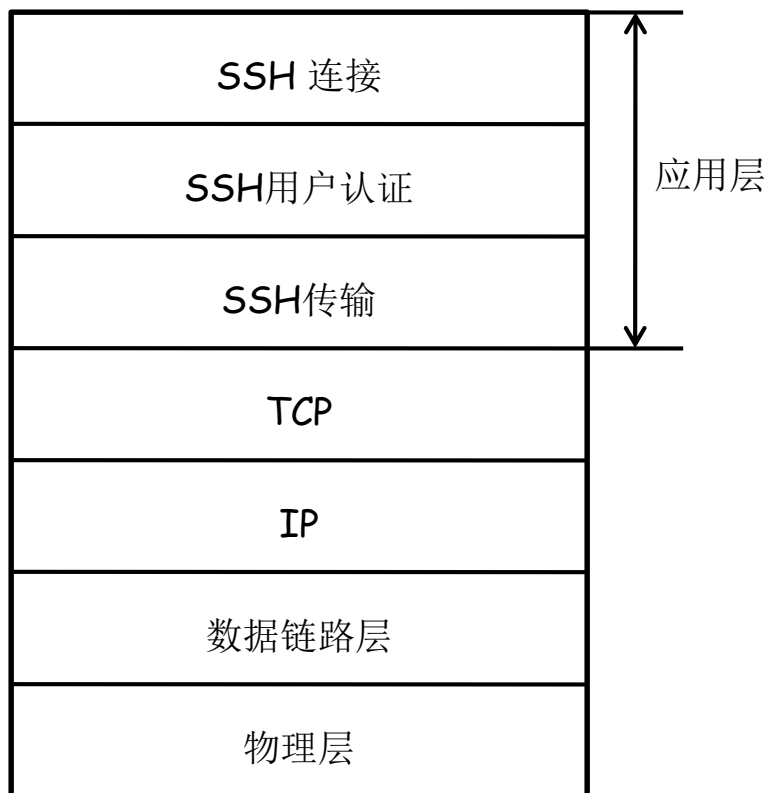
- 5.1 密码算法在网络各层中的部署
- 5.2 公钥密码基础设施
- 5.3 IPSec协议: 网络层的安全协议
- 5.4 SSL/TLS协议: 传输层的安全协议
- 5.5 PGP and S/MIME: 电子邮件安全协议
- 5.6 Kerberos: 认证协议
- 5.7 **SSH: 远程登录安全协议**

SSH简介



- SSH: 安全外壳(Secure Shell)
- 用于替代不安全的登录工具，例如 RCP, FTP, RSH, Telnet, rlogin等
- 用认证和加密算法在两台计算机之间建立安全连接
- 支持数据压缩
- 为文件传输(SFTP)和文件拷贝(SCP)提供安全保护
- SSH协议包括3个部分

SSH的3层



SSH 架构

- SSH连接:
 - 在一个SSH连接中为不同的应用建立多个通道
- SSH用户认证:
 - 认证用户的身份
 - 用登录口令或公钥密码体系
- SSH传输
 - 处理初始化建立，服务器认证和密钥交换
 - 设定加密算法和压缩算法