

Chapter1 概论

常见安全协议

安全协议分析

攻击的类型

网络安全需求

密码学回顾

逻辑学回顾

Chapter2 安全协议

安全协议

无可信第三方参与的对称密钥协议

RPC协议分析

有可信第三方参与的对称密钥协议

Needham Schroeder协议

Otway-Rees协议

大嘴青蛙协议

有可信第三方参与的公钥协议

Needham Schroeder公钥协议

Denning Sacco密钥分配协议

协议设计原则：

Chapter3 Kerberos

密钥管理问题

SSO：单点登录

Kerberos认证服务协议

Kerberos的加密体制

主要功能

基本概念

动机

V4流程

Kerberos和多个域

跨域认证

V5流程

小结

Chapter4 IPSEC

IP安全问题

离散对数困难

HMAC

IPSEC概述

体系结构

基本概念

安全联盟 (SA)

安全关联数据库 (SAD)

安全策略库 (SPD)

AH—Authentication Header

传输模式

隧道模式

ESP- Encapsulating Security Payload

传输模式

隧道模式

AH vs ESP

IKE

阶段1

阶段2

IKE的协商过程

两个阶段

交换信息

阶段一协商过程

阶段二协商过程

Chapter5 SSL

不同协议层的安全

SSL概念

安全机制

SSL的分层结构

SSL基本过程

握手协议

阶段1：建立安全能力

阶段2：服务器认证与密钥交换

阶段3：客户机认证与密钥交换

Cipher Change

Alert Protocols

传输应用数据

记录协议

SSL安全性分析

SSL脆弱性分析

Chapter6 SET

电子交易的主要模式

支付系统无安全措施的模式

通过第三方代理人支付的模式

数字现金支付模式

简单加密支付系统模式

安全电子交易SET支付模式

SET概述

交易中的主体

主体证书

电子商务的安全架构需求

电子商务的典型场景

电子支付的流程

双重数字签名

SET 交易流程

支付过程初始化

下单过程

PReq

PRes

支付网关认证过程

AuthReq

AuthRes

支付完成

SET 核心技术

SET 协议交易过程
SET相比SSL的不同处

Chapter7 PGP

Email工作原理
PGP公钥的分发
PGP的证书
存在的风险
证书的撤销
信任的水平
PGP加密处理过程
PGP压缩
PGP密钥环
信任模型

Chapter8 BAN

“Dolev-Yao”攻击者模型
基本术语
规则
message-meaning rules
nonce-verification rule
jurisdiction rule
接受规则
新鲜性规则
BAN 逻辑的假定
关于协议运作的前提与假定
密钥分发与认证的逻辑目标
Ban 逻辑分析的过程
BAN 逻辑的缺陷
Example
Kerberos 协议分析
RPC 协议
多重会话攻击

Chapter9 CSP

Agents
CSP 概述
CSP 常见符号
CSP 形式化定义
CSP 语法
CSP 分析
消息刻画
使用CSP进行安全协议建模

Chapter1 概论

常见安全协议

- Kerberos协议：
 - Kerberos协议的基本概念和过程

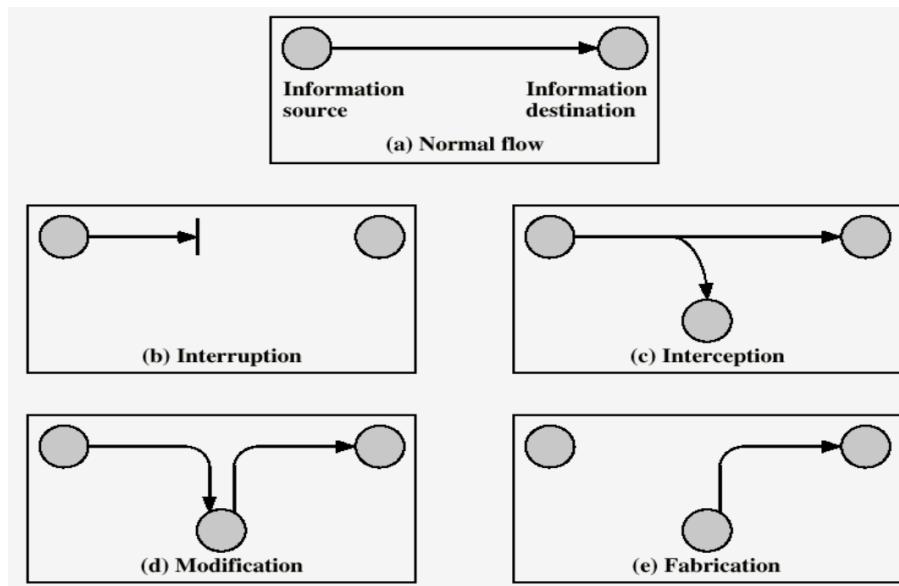
- 解决分布式场景下的安全认证与授权问题
- IPSEC:
 - 理解掌握IPsec的基本概念
 - SA, SPD, AH, ESP, 隧道模式, 传输模式, Outbound & Inbound 处理
 - 解决IP层的安全加固问题
- SSL/TLS协议:
 - 基本概念, 协议过程
 - 解决传输层的安全加固问题
- SET协议:
 - 电子支付基本流程, 双重数字签名, SET
 - 提供安全支付的一种经典问题解决方案
- PGP协议:
 - 公钥环, 私钥环, PGP的处理流程
 - 提供一种无政府状态下的互联网信任安全问题解决方案

安全协议分析

- 基于推理结构性方法: Ban逻辑分析的过程, SVO逻辑分析的过程
- 基于攻击型分析方法: 使用CSP进行协议分析
- 基于证明的结构性方法: 串, 串空间, Bundle, 串空间分析安全协议

攻击的类型

- 拦截、窃听、篡改、冒充



网络安全需求

- 身份鉴别、访问控制、数据机密性、数据完整性、抗否认、可用性

密码学回顾

数字签名的安全属性: 不可抵赖性、不可伪造性、发生争执时可以由第三方仲裁

密钥分配：

- 对称密码体制密钥的分配
- 可逆公钥密码体制
- Diffie-Hellman密钥分配
- 公钥密码体制：公钥证书

判断证书有效性：

1. 验证证书是否在有效期内
2. 验证证书是否被吊销
3. 验证证书的真实性，是否是上级CA签发的

逻辑学回顾

命题：表示知识的陈述性形式

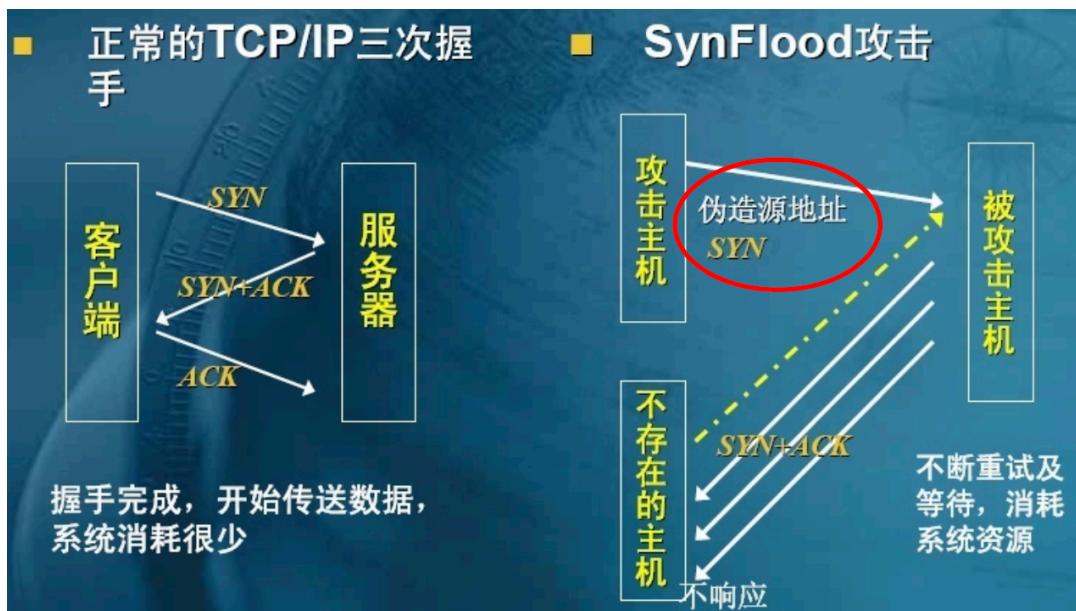
我们可以很容易地把客观世界的各种事实表示为逻辑命题，用命题逻辑把各种命题写成合适公式，也称“谓词公式”

联结词：合取、析取、否定、蕴含

一阶谓词逻辑是对命题逻辑的抽象：student(X)

Chapter2 安全协议

拒绝服务攻击的发生机理：



安全协议

协议：两个或两个以上的参与者采取一系列的步骤，以完成某项特定的任务。

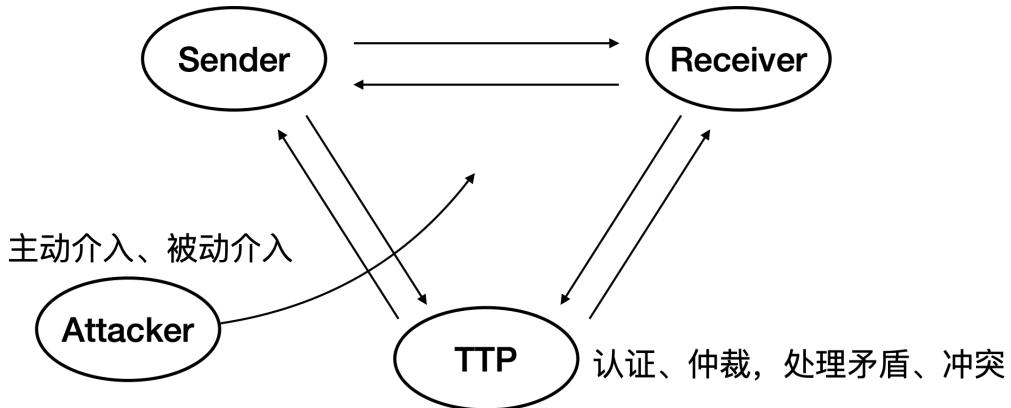
安全协议：基于密码体制，在分布式系统，通信网中，目标：密钥分配，身份认证，信息保密，电子商务。

安全协议类型：

- 密钥交换协议：完成会话密钥建立

- 认证协议：身份认证，消息认证，数据源\目的认证
- 认证和密钥交换协议：Kerberos
- 电子商务：公平性，set协议

安全协议系统模型：



安全协议的缺陷：

- 协议设计要素：主体身份标识，签名，随机数，时戳
- 缺陷分类：基本协议缺陷，陈旧消息(重放攻击)，并行会话

常用术语：

Term	Description
A, B, P, Q, R	主体，协议参与者
K_{ij}	对称密钥
K_i, K_j	公钥
R_i	随机数
N_i	i产生的新鲜数，不会重复，只有i知道，无法伪造，特有随机数
T_i	i产生的时戳
$[m_1 m_2]$	
$E(K:m)$	加密
TEXT ₁ , TEXT ₂	
Z	攻击者
$f_{Kab}(X)$	

良好密钥的标准：

- 新鲜产生，或者来自于可信第三方
- 仅仅被参与协议的指定合法主体所拥有

认证的目的：

- A 知道 B active 且想开启协议

$$\begin{aligned}A &\rightarrow B : N_a \\B &\rightarrow A : \text{Sign}_B(A, N_a)\end{aligned}$$

挑战应答：

- 判断对方是否收到消息
- 判断对方收到消息后沟通的意愿

无可信第三方参与的对称密钥协议

RPC协议分析

协议目的：利用已有的共享密钥建立新的会话密钥

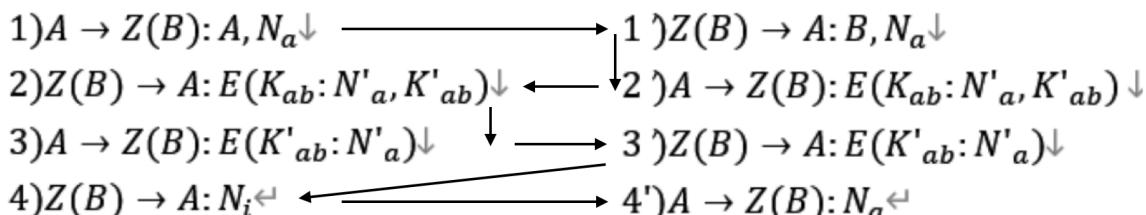
$$\begin{aligned}A &\rightarrow B : A, E(K_{ab} : N_a) \\B &\rightarrow A : E(K_{ab} : N_a + 1, N_b) \\A &\rightarrow B : E(K_{ab} : N_b + 1) \\B &\rightarrow A : E(K_{ab} : K'_{ab}, N'_b)\end{aligned}$$

- A客户端，B服务器
- 第一步中的A为身份标识，用于B寻找对应的会话密钥
- 第四步中 K'_{ab} 为新密钥， N'_b 此处无意义

简化版本：

$$\begin{aligned}A &\rightarrow B : A, N_a \\B &\rightarrow A : E(K_{ab} : N'_a, K'_{ab}) \\A &\rightarrow B : A, E(K'_{ab} : N'_a) \\B &\rightarrow A : N_b\end{aligned}$$

- 第三步B得知A是否接受新的会话密钥
- 第四步用于A判断B是否收到消息
- 存在多重会话攻击(重放攻击)，Z劫持B的会话，把B屏蔽了
 - 达成目标：协议双方未达成目标，B被屏蔽了，导致协议运行的非法状态，合法参与者无法判断，因此协议设计不够健壮



加固：

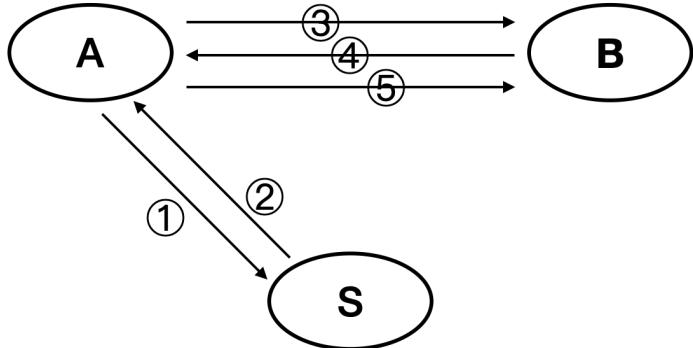
- 一般在密文中引入身份标识，可解决前述重放问题

$A \rightarrow B : A, N_a$
 $B \rightarrow A : E(K_{ab} : \textcolor{red}{B}, N'_a, K'_{ab})$
 $A \rightarrow B : A, E(K'_{ab} : N'_a)$
 $B \rightarrow A : N_b$

有可信第三方参与的对称密钥协议

Needham Schroeder协议

目的：获得会话密钥



1) $A \rightarrow S : A, B, N_a$

2) $S \rightarrow A : E(K_{as} : N_a, B, K_{ab}, E(K_{bs} : K_{ab}, A))$

3) $A \rightarrow B : E(K_{bs} : K_{ab}, A).....(***)$

4) $B \rightarrow A : E(K_{ab} : N_b)$

5) $A \rightarrow B : E(K_{ab} : N_b - 1)$

- S为TTP

攻击：

$A \rightarrow S : A, B, N_a$
 $S \rightarrow A : E(K_{as} : N_a, B, K_{ab}, E(K_{bs} : K_{ab}, A))$
 $A \rightarrow Z(B) : E(K_{bs} : K_{ab}, A)$
 $Z(B) \rightarrow A : N_z$
 $A \rightarrow Z(B) : E(K_{ab} : Decrypt(K_{ab} : N_b) - 1)$

- 利用类型缺陷，使得 N_z 匹配 $\{N_b\}_{K_{ab}}$ 的格式
- A把 N_z 当成 $\{N_b\}_{K_{ab}}$ ，解密得到的结果认为是正常的 N_b
- B无法得到 K_{ab} ，攻击者也不知道

加固：

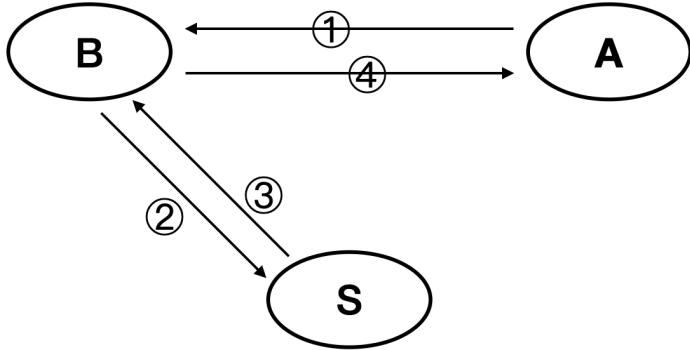
- 加入身份标识

$A \rightarrow S : A, B, N_a$
 $S \rightarrow A : E(K_{as} : N_a, B, K_{ab}, E(K_{bs} : K_{ab}, A))$
 $A \rightarrow B : E(K_{bs} : K_{ab}, A)$
 $B \rightarrow A : E(K_{ab} : \textcolor{red}{B}, N_b)$
 $A \rightarrow B : E(K_{ab} : N_b - 1)$

Otway-Rees协议

轮识别符、类型缺陷、重放攻击

M: 轮识别符，第M轮的消息



- 1) $A \rightarrow B : M, A, B, E(Kas : Na, M, A, B)$
- 2) $B \rightarrow S : M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B)$
- 3) $S \rightarrow B : M, E(Kas : Na, Kab), E(Kbs : Nb, Kab)$
- 4) $B \rightarrow A : M, E(Kas : Na, Kab)$

- 第三步中的 N_a 与 N_b 可防TP冒充

重放攻击：

- 1) $A \rightarrow Z(B) : M, A, B, E(Kas : Na, M, A, B)$
- 2) $Z(B) \rightarrow A : M, E(Kas : Na, M, A, B)$

- 导致A把{M,A,B}当作 K_{ab}
- 类型缺陷的攻击
- 由于{M,A,B}是第一步中的明文，因此攻击者知道了会话密钥

加固：

- 1) $A \rightarrow B : M, A, B, E(Kas : Na, M, A, B)$
- 2) $B \rightarrow S : M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B)$
- 3) $S \rightarrow B : M, E(Kas : Na + 1, Kab), E(Kbs : Nb + 1, Kab)$
- 4) $B \rightarrow A : M, E(Kas : Na + 1, Kab)$

另一版本：

```

A → B : A, B, Na
B → S : A, B, Na, Nb
S → B : E(Kas : Na, A, B, Kab), E(Kbs : Nb, A, B, Kab)
B → A : E(Kas : Na, A, B, Kab)
  
```

攻击：

- (1) $A \rightarrow B : A, B, Na$
- (2) $B \rightarrow S : A, B, Na, Nb$
- (2') $S \rightarrow P(B) : \{Na, A, B, Kab\} Kas, \{Nb, A, B, Kab\} Kbs$
- (3') $P(B) \rightarrow S : A, B, Na, Nb$
- (3'') $S \rightarrow P(B) : \{Na, A, B, K'ab\} Kas, \{Nb, A, B, K'ab\} Kbs$
- (3) $P(S) \rightarrow B : \{Na, A, B, K'ab\} Kas, \{Nb, A, B, Kab\} Kbs$
- (4) $B \rightarrow A : \{Na, A, B, K'ab\} Kas$

- 交叉组合，违背了原子性

大嘴青蛙协议

判断时间新鲜性：

- M：轮识别符
- N_a ：只对A有判断价值，反映当前消息先后顺序， N_a (先)， N_a+1 (后)
- T_a ：时戳，全系统公认，分布式系统内维护一个时钟， T_a 发出后，B收到计算是否在时间窗口内，判断有效性

1) $A \rightarrow S : A, E(Kas : Ta, B, Kab)$

2) $S \rightarrow B : E(Kbs : Ts, A, Kab)$

问题：

- 缺乏对A的response
- B是否愿意？

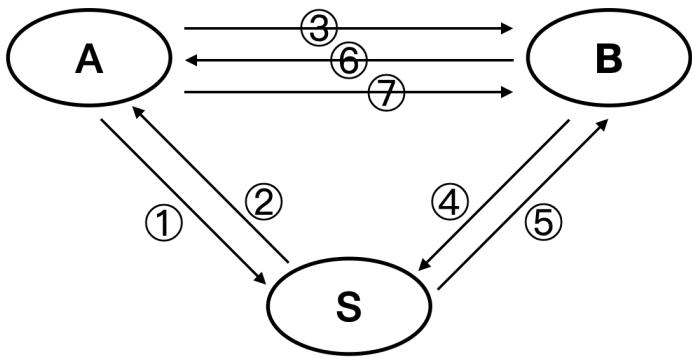
攻击

1. $T_{window}=300s$, 20s时正常发送，200s攻击者重放， K_{ab} 多次分发，违背设计意愿
2. 第二步被Z(B)截获

- 1) $A \rightarrow S : A, E(Kas : Ta, B, Kab)$
- 2) $S \rightarrow B : E(Kbs : Ts, A, Kab)$
- 3) $B \rightarrow A : E(Kab : B, Nb)$
- 4) $A \rightarrow B : E(Kab : Nb + 1)$

有可信第三方参与的公钥协议

Needham Schroeder公钥协议



- 1) $A \rightarrow S : A, B$
- 2) $S \rightarrow A : E(Ks^{-1} : Kb, B)$
- 3) $A \rightarrow B : E(Kb : Na, A)$
- 4) $B \rightarrow S : B, A$
- 5) $S \rightarrow B : E(Ks^{-1} : Ka, A)$
- 6) $B \rightarrow A : E(Ka : Nb, B)$
- 7) $A \rightarrow B : E(Kb : Nb)$

- 第二步的实质为S向A发放B的证书

重放攻击：

- 1.3. $A \rightarrow I : \{Na, A\}_{Ki}$
- 2.3. $I(A) \rightarrow B : \{Na, A\}_{Kb}$
- 2.6. $B \rightarrow I(A) : \{Na, Nb\}_{Ka}$
- 1.6. $I \rightarrow A : \{Na, Nb\}_{Ka}$
- 1.7. $A \rightarrow I : \{Nb\}_{Ki}$
- 2.7. $I(A) \rightarrow B : \{Nb\}_{Kb}.$

- 第一轮A和I正常通信，第二类I冒充A与B通信
- B以为在和A通信，其实在和I通信

加固：

- 1) $A \rightarrow S : A, B$
- 2) $S \rightarrow A : E(Ks^{-1} : Kb, B)$
- 3) $A \rightarrow B : E(Kb : Na, A)$
- 4) $B \rightarrow S : B, A$
- 5) $S \rightarrow B : E(Ks^{-1} : Ka, A)$
- 6) $B \rightarrow A : E(Ka : Nb, B)$
- 7) $A \rightarrow B : E(Kb : Nb)$

Denning Sacco密钥分配协议

可能考对的分析：

- 1) $A \rightarrow S : A, B$
- 2) $S \rightarrow A : CertA, CertB$
- 3) $A \rightarrow B : CertA, CertB, E(K_b : E(K_a^{-1} : Kab, Ta))$

协议设计原则：

- 认证原则：通过挑战应答机制实现对于身份与共识的认证。基于随机数的挑战应答，建立在其自身被有效保护的基础上；挑战应答的封闭性(大嘴青蛙问题)
- 新鲜性原则：通过新鲜性识别机制区分消息轮次和先后顺序，包括轮识别符，Nonce和时戳等。时戳的使用前提是可靠的分布式时钟机制
- 原子性原则：密文及消息的原子性(绑定)，尽量避免使用长周期的密钥
- 效率原则：不要引入不必要的加密和消息字段
- 消息内容明确原则：每条消息应该力求意义明确，例如加入身份标识

Chapter3 Kerberos

密钥管理问题

- 所有的密码系统都存在这样的问题：如何安全/可靠地分配密钥
- 理想的情况是，密钥分配协议应该得到形式化验证

SSO：单点登录

- 用户只需要登录一次，就可以访问多个系统，不需要记忆多个口令密码
- 优点
 - 用户可以快速访问网络，提高工作效率，也能帮助提高系统的安全性
 - 有利于进行账户密码管理、用户审计
 - 方便进行企业应用部署

Kerberos认证服务协议

- 提供一个在客户端跟服务器端之间或服务器与服务器之间的身份验证机制（并且是相互的身份验证机制）
- 解决的问题
 - 在公开的分布式环境中，工作站上的用户希望访问分布在网络中的服务器上的服务
 - 服务器希望能够限制授权用户的访问，并对服务请求进行鉴别

Kerberos的加密体制

- Kerberos提供一个中心认证服务器，提供用户和服务器之间的认证服务
- 采用传统加密算法，无公钥体制
- 常用版本：Kerberos Version 4 和 Kerberos Version 5

主要功能

- 在分布式的client/server体系结构中，采用Kerberos服务器提供认证服务
- 总体方案是提供一个可信第三方的认证服务
 - 用tickets验证
 - 避免本地保存密码和在互联网上传输密码
 - 包含可信第三方
 - 使用对称加密

- 客户端与服务器之间能够相互验证

基本概念

- Principle: 安全个体，被认证的个体，有名字和口令
- KDC: 密钥分发中心，提供ticket和临时会话密钥
- Ticket:
 - 客户可以用它来向服务器证明自己的身份
 - Ticket 中的大多数信息都被加密，密钥为服务器的密钥
- Authenticator: 包含最近产生的信息的记录，产生这些信息需要用到客户和服务器之间共享的会话密钥
- Credentials: 一个 ticket 加上秘密的会话密钥
- AS: Authentication Server
 - 通过 long-term key 认证用户
 - AS 给予客户 ticket granting ticket 和 short-term key
 - 认证服务
- TGS: Ticket Granting Server
 - 通过 short-term key 和 ticket granting ticket 认证用户
 - TGS 发放 tickets 给客户以访问其他的服务器
 - 授权和访问控制服务

动机

- 认证和授权的逻辑分离，虽然AS和TGS物理上常在一起
- 不同的生命周期：
 - ticket granting tickets (typically 10 hours)
 - session tickets for actual access to services (typically 5 minutes)
- 方便客户
- 降低密钥的暴露时间
- 弹性、可伸缩性

V4流程

- 引入可信第三方的认证服务，基于Needham & Schroeder协议
- 采用DES加密算法，提供认证服务

1. 认证服务交换：获得票据许可票据(ticket granting ticket)

$$\begin{aligned} C \rightarrow AS : & ID_c || ID_{tgs} || TS_1 \\ AS \rightarrow C : & E(K_c[K_{c,tgs} || ID_{tgs} || TS_2 || Lifetime_2 || Ticket_{tgs}]) \end{aligned}$$

$$Ticket_{tgs} = E(K_{tgs}[K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2])$$

- AD_c : 客户所在工作站的网络地址
- $Ticket_{tgs}$: 用于验证客户身份，客户不能打开

2. 票据许可服务交换：获得服务许可票据(service granting ticket)

$$\begin{aligned} C \rightarrow TGS : & ID_v || Ticket_{tgs} || Authenticator_c \\ TGS \rightarrow C : & E(K_{c,tgs}[K_{c,v} || ID_v || TS_4 || Ticket_v]) \end{aligned}$$

$$Ticket_{tgs} = E(K_{tgs}[K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2])$$

$$Ticket_v = E(K_v[K_{c,v} || ID_c || AD_c || ID_v || TS_4 || Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}[ID_c || AD_c || TS_3])$$

- ID_v : 应用服务器的ID
- TGS解开 $Ticket_{tgs}$ 、 $Authenticator_c$, 对比验证身份

3. 客户/服务器认证交换：获得服务

$$\begin{aligned} C \rightarrow V : & Ticket_v || Authenticator_c \\ V \rightarrow C : & E(K_{c,v}[TS_5 + 1]) \end{aligned}$$

$$Ticket_v = E(K_v[K_{c,v} || ID_c || AD_c || ID_v || TS_4 || Lifetime_4])$$

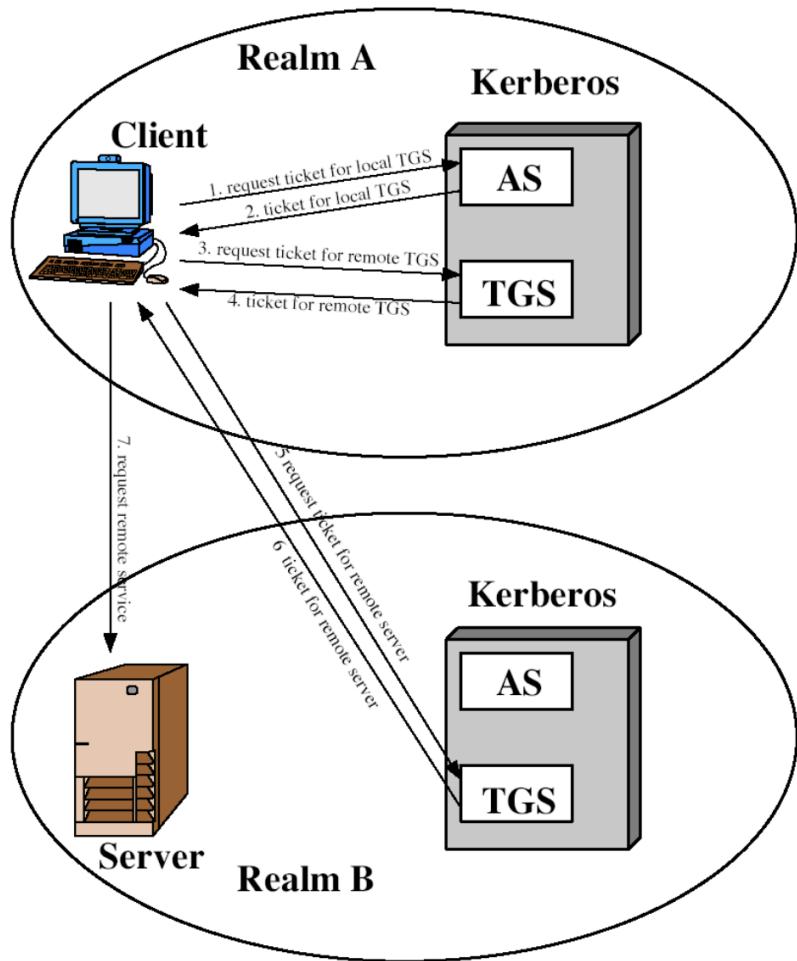
$$Authenticator_c = E(K_{c,v}[ID_c || AD_c || TS_5])$$

Kerberos和多个域

- 完整的Kerberos环境包括Kerberos服务器、一组工作站和一组应用服务器
 - 所有用户和服务器均在Kerberos服务器上注册
 - Kerberos服务器必须在数据库中拥有所有用户的ID和口令散列表
 - Kerberos服务器必须与每一个服务器之间共享一个保密密钥
- 对于不同的域
 - 每个辖区的Kerberos服务器与其他辖区的Kerberos服务器之间共享一个保密密钥，两个服务器互相注册

跨域认证

1. 获得本地TGS访问权
2. 请求一张远程TGS的票据许可票据
3. 向远程TGS请求其域内的服务



V5流程

基于v4的改进

通用性：

	v4	v5
加密算法	DES	扩展
网络协议地址	IP	OSI
票据生命周期	最大1280min	不限制
认证转发	不支持	允许服务器在事务中代表客户端访问另一台服务器

技术改进：

- 双重加密
 - v4中的票据被重复加密
- 消息重放
 - AS → Client、TGS → Client 消息在票据生命周期中或可被重放，v5 采用新鲜数
 - 采用同一票据的多个cs连接使用相同的会话密钥，因而会遭受重放，v5 使用 Subkey 机制

1. 认证服务交换：获得票据许可票据(ticket granting ticket)

$$C \rightarrow AS : Options || ID_c || Realm_c || ID_{tgs} || Times || Nonce_1$$

$$AS \rightarrow C : Realm_c || ID_c || Ticket_{tgs} || E(K_c[K_{c,tgs} || ID_{tgs} || Times || Nonce_1 || Realm_{tgs}])$$

$$Ticket_{tgs} = E(K_{tgs}[Flags || K_{c,tgs} || ID_c || AD_c || Realm_c || Times])$$

- $Realm_c$: 域
- $Times$: 时戳
- $Ticket_{tgs}$: 相比 v4 不加密
- 原子性: 避免交叉组合不同轮次消息, 然而 $Nonce$ 和 $Times$ 的存在解决了此问题, 因此 $Ticket$ 可以放外面, 效率也得以提升(密文里的 $Times$ 与 $Ticket_{tgs}$ 中的 $Times$ 一致, 保证无法交叉被重放)

2. 票据许可服务交换: 获得服务许可票据(service granting ticket)

$$C \rightarrow TGS : Options || ID_v || Times || Nonce_2 || Ticket_{tgs} || Authenticator_c || Realm_v$$

$$TGS \rightarrow C : Realm_c || ID_c || Ticket_v || E(K_{c,v}[K_{c,v} || ID_v || Times || Nonce_2 || Realm_v])$$

$$Ticket_{tgs} = E(K_{tgs}[Flags || K_{c,tgs} || ID_c || AD_c || Realm_c || Times])$$

$$Ticket_v = E(K_v[flags || K_{c,v} || ID_c || AD_c || Realm_c || Times])$$

$$Authenticator_c = E(K_{c,tgs}[ID_c || Realm_c || TS_1])$$

- $Ticket_v$: 相比 v4 不加密

3. 客户/服务器认证交换: 获得服务

$$C \rightarrow V : Options || Ticket_v || Authenticator_c$$

$$V \rightarrow C : E(K_{c,v}[TS_2 || Subkey || Seq\#])$$

$$Ticket_v = E(K_v[flags || K_{c,v} || ID_c || AD_c || Realm_c || Times])$$

$$Authenticator_c = E(K_{c,v}[ID_c || Realm_c || TS_2 || Subkey || Seq\#])$$

- $Subkey$ 可用于不同 session 的加密, 使用不同 $Subkey$ 避免被重放, 由 C 创建

小结

- 认证方法:
 - 本地机器录入密码
 - 经由中央 KDC 认证(1times/day)
 - 网上不传输密码
- Single Sign-on (via TGS):
 - KDC 给予票据 TGT, 其作用一般维持一天的时间
 - TGT 可以用于获取其他的服务票据
- Kerberos优点:
 - 密码不易被窃听
 - 密码不在网上传输
 - 密码猜测更困难
 - Single Sign-on
 - 更便捷: 一次使用口令登录

- 不用记忆多个口令
- 票据被盗之后难以使用，因为需要配合认证头来使用
- 缺点
 - 缺乏撤销机制
 - 密钥管理复杂，特别是跨域的情况
 - 跨域认证复杂
 - 需要始终同步
 - 需要始终在线的AS和TGS
 - 口令安全、软件安全

Chapter4 IPSEC

IP安全问题

IP协议本质上是不安全的，仅仅依靠IP头部的校验和字段无法保证IP包的安全

- IP地址欺骗
- 缺乏对通信双方真实身份的验证
- 包重放
- 无数据完整性和机密性
- IPv4无连接、不可靠

离散对数困难

Diffie-Hellman算法的有效性依赖于计算离散对数的难度，其含义是：当已知大素数p和它的一个原根a后，对给定的b，要计算i，被认为是很困难的，而给定i计算b却相对容易

$$b = a^i \bmod p$$

HMAC

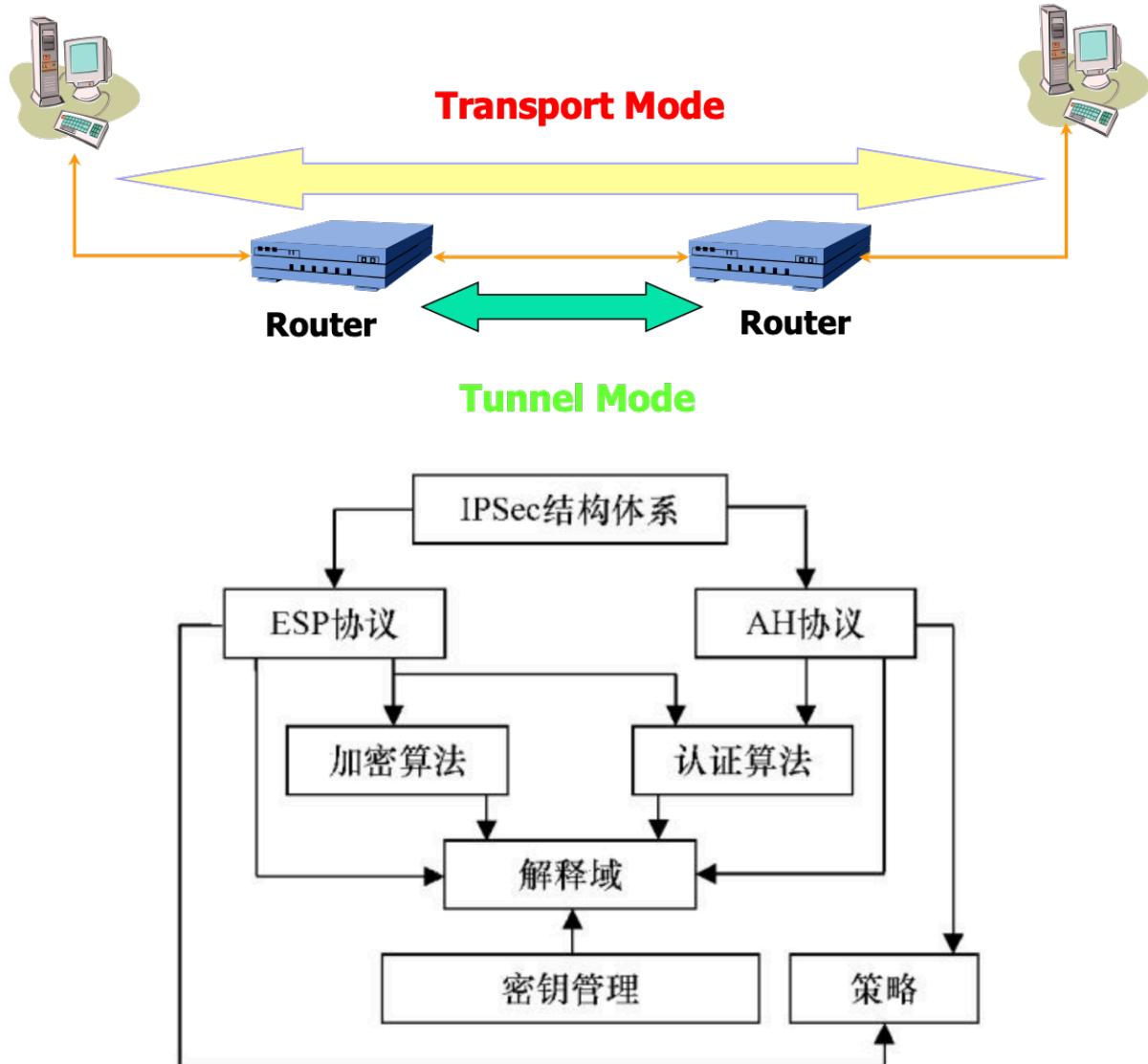
$$\begin{aligned} ipad &= \text{the byte } 0x36 \text{ repeated } B \text{ times} & '6' \\ opad &= \text{the byte } 0x5C \text{ repeated } B \text{ times} & '\\' \\ H(K \text{ XOR } opad, H(K \text{ XOR } ipad, \text{text})) \end{aligned}$$

IPSEC概述

- IPsec是一个框架，一套协议和标准，支撑IP层的安全
- 在IP包上为IP业务提供保护的安全协议标准，弥补IPv4的安全性，对IPV4是可选的，对IPV6是必须的
- 目的：把安全机制引入IP协议，使用现代密码学方法支持机密性和认证性服务，确保公网上数据通信的可靠性和完整性
- IPsec实现两个基本目标：
 - 保护IP数据包安全
 - 为抵御网络攻击提供防护措施
- IPsec由三种机制共同保障：认证、信息机密性和密钥管理
- IPsec提供：

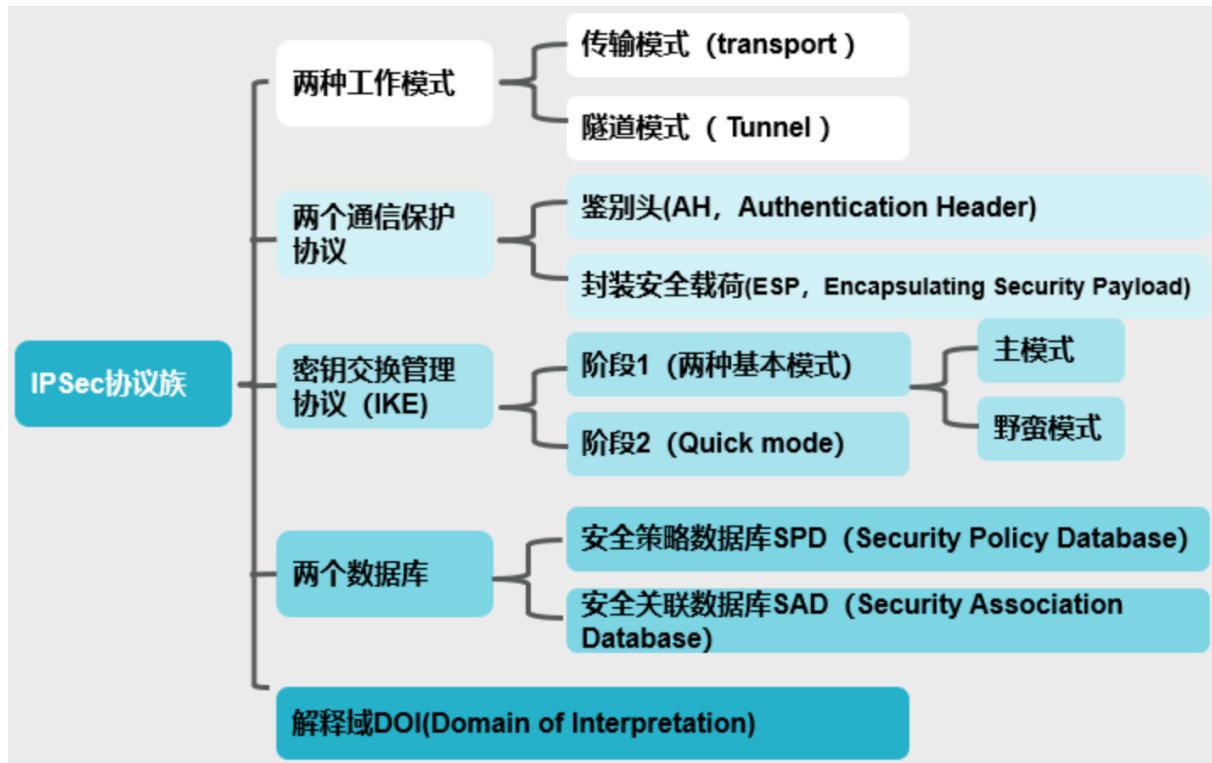
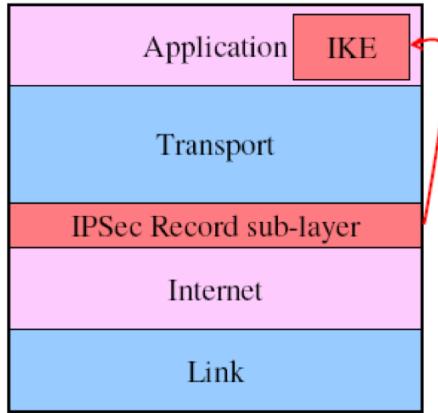
- 访问控制
- 无连接完整性
- 数据源认证
- 载荷机密性
- 有限流量机密
- 重放攻击、DOS保护

体系结构



两个分离的层：

- IKE: Internet Key Exchange
- IP-Sec record sub-layer: 流量封装与保护



基本概念

安全联盟 (SA)

- Security Association
- 由SPI (Security Parameter Index) 和目标地址组成
- 单个IPSec连接至少需要两个SA
- 在网络上传输数据，进行密钥协商的抽象个体

安全关联数据库 (SAD)

- Security Association Data
- SAD包含了所有活跃的SA的所有参数信息
- 流出数据、流入数据
- 手工创建或者经由IKE生成

安全策略库 (SPD)

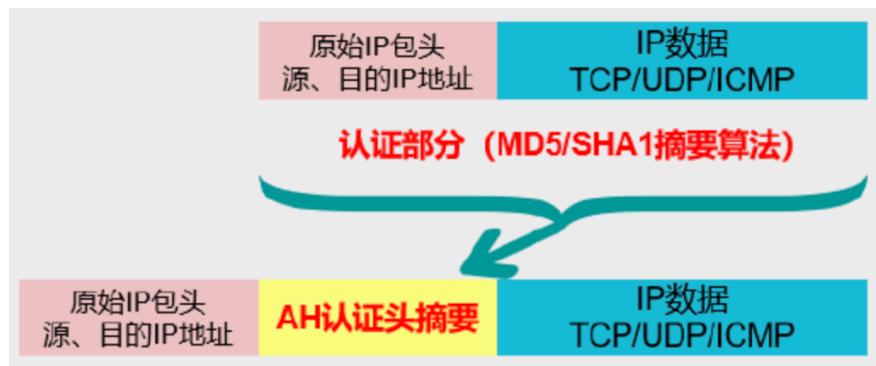
- Security Policy Data
- 含有用户定义的策略：每个报文的安全服务及其水平
- 比如：哪些要IPSEC处理，哪些不用，哪些拦截

AH—Authentication Header

- 为IP通信提供数据源认证、数据完整性和反重播保证，它能保护通信免受篡改
- 不能防止窃听，适合用于传输非机密数据，无机密性保护
- 在每一个数据包上添加一个身份验证报头
 - 此报头包含一个带密钥的hash散列 (HMAC)
 - 此hash散列在整个数据包中计算，因此对数据的任何更改将致使散列无效
 - 因此提供了完整性保护

传输模式

- AH被插在IP头之后但在所有的传输层协议(如TCP、UDP)之前，或其它IPSec协议头之前
- 使用原始的明文IP头，即源/目的地址不变，所有安全信息在AH中
- 被AH验证的区域是整个IP包（可变字段除外），包括IP包头部，因此源IP地址、目的IP地址是不能修改的，否则会被检测出来



- 然而，如果该包在传送的过程中经过NAT网关，其源/目的IP地址将被改变，将造成到达目的地址后的完整性验证失败
- 因此，AH在传输模式下和NAT是冲突的，不能同时使用，或者说AH不能穿越NAT

隧道模式

- AH插在原始的IP头之前，另外生成一个新的IP头放在AH之前



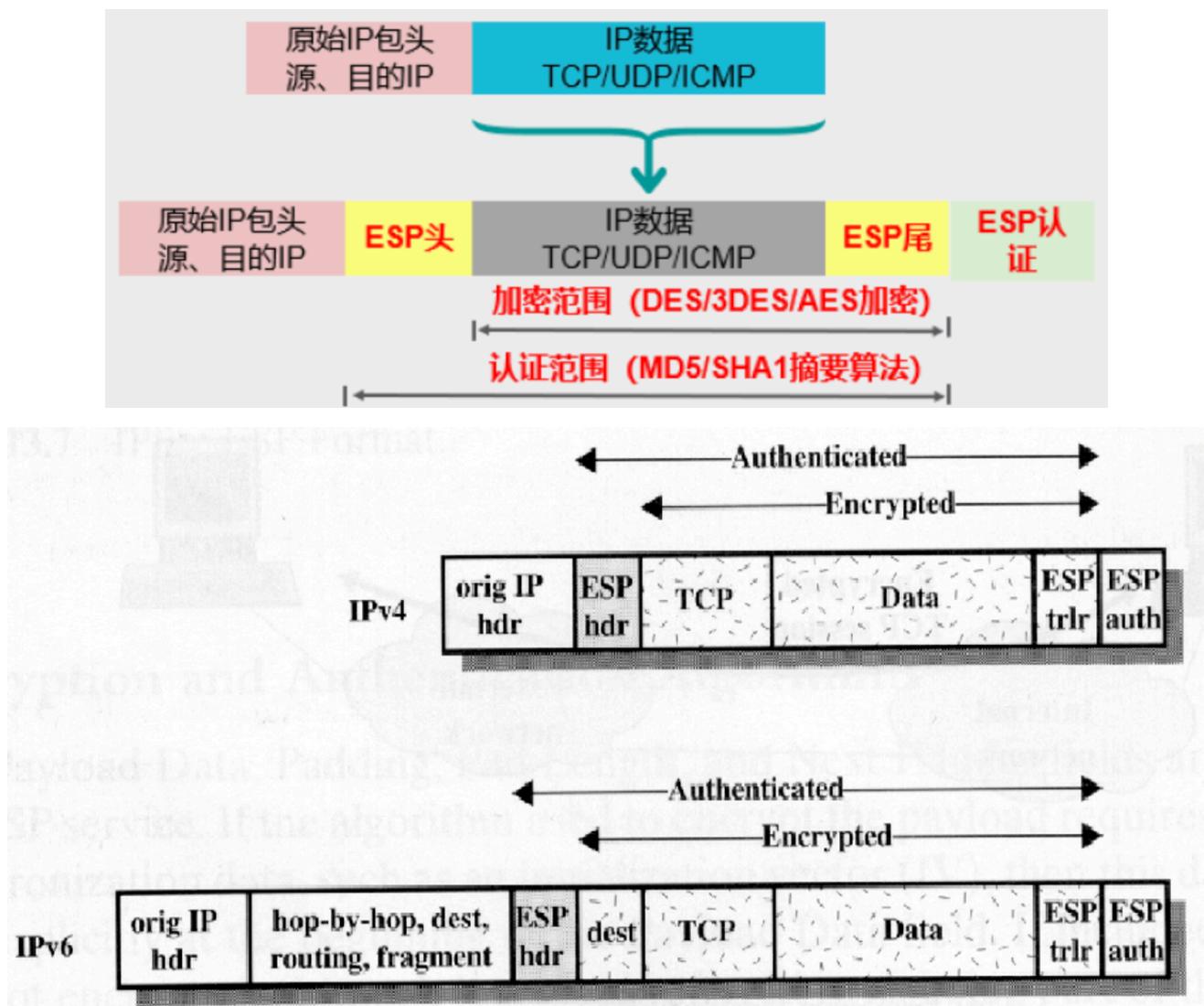
- 加密整个IP数据报，包括全部TCP/IP或UDP/IP头和数据，并用自己的地址作为源地址加入到新的IP头
- AH验证的范围也是整个IP包，因此AH和NAT的冲突在隧道模式下也存在

ESP- Encapsulating Security Payload

- 提供机密性、完整性、认证、抗抵赖性
- ESP的验证不会对整个IP包进行，IP包头部（含选项字段）不会被验证。因此，ESP不存在像AH那样的和NAT模式冲突的问题
- 如果通信的任何一方具有私有地址或者在安全网关背后，双方的通信仍然可以用ESP来保护其安全，因为IP头部中的源/目的IP地址和其他字段不被验证，可以被NAT网关或者安全网关修改
- 原IP地址被当作有效载荷的一部分受到IPSec的安全保护，另外，通过对数据加密，还可以将数据包目的地址隐藏起来，这样更有助于保护端对端隧道通信中数据的安全性
- ESP隧道模式的验证和加密能够提供数据流加密服务

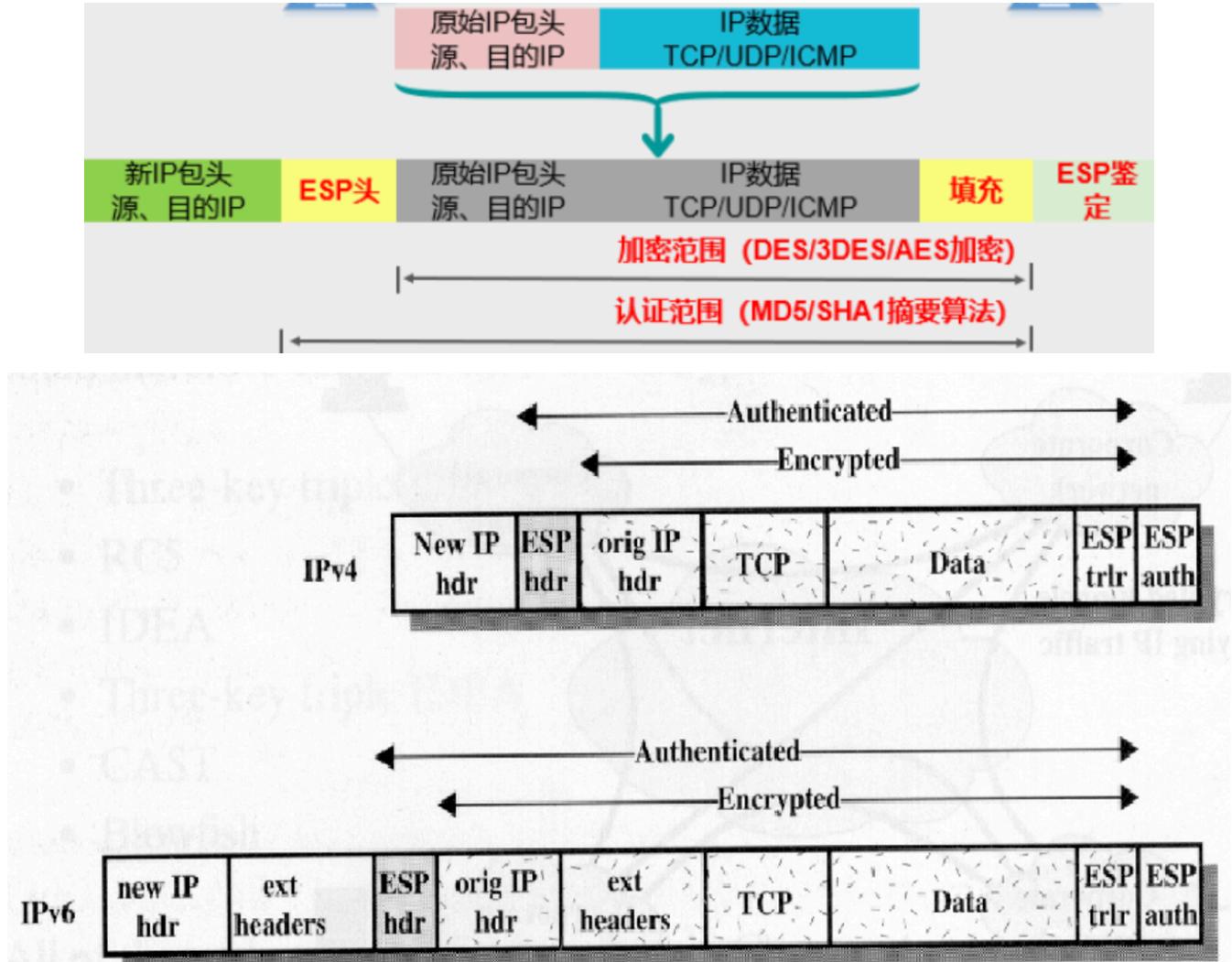
传输模式

- 加密范围与认证范围



隧道模式

- 内部IP报头（原IP报头）指定最终的信源和信宿地址
- 外部IP报头（新IP报头）中包含的常常是做中间处理的安全网关地址

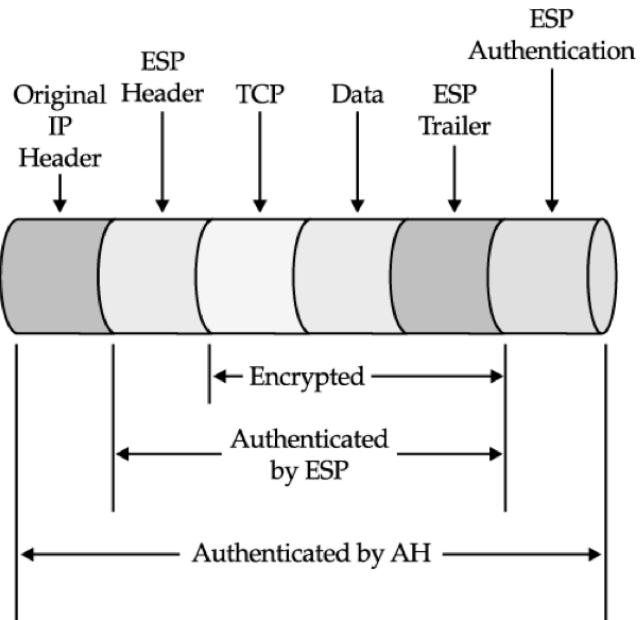


AH vs ESP

安全特性	AH	ESP
完整性	√	√ (不验证IP头)
数据源验证	√	√
数据加解密	✗	√
抗重放	√	√
抗抵赖性	√	✗
NAT	✗	√

- 一般组合使用AH、ESP

- 先使用ESP，再在外部使用AH
- 优点：ESP头部也能由AH保护

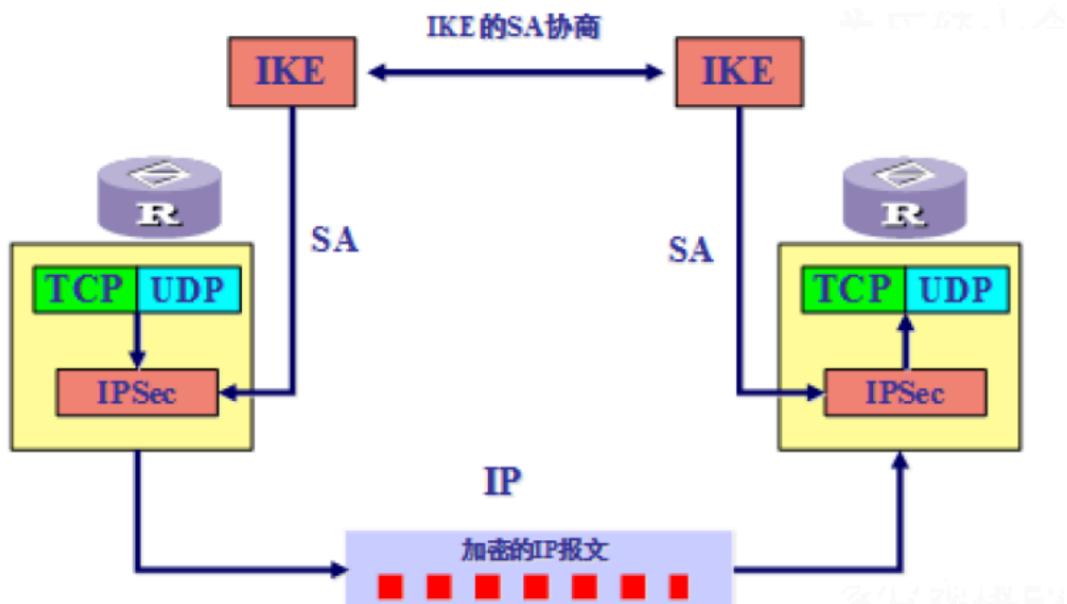


IKE

- Internet Key Exchange (IKE) 用于配置IPSec，使用DH
- ISAKMP : Internet Security Association and Key Management Protocol
- IPSec的信令协议，为IPSec提供自动协商交换密钥、建立安全联盟的服务，能够简化IPSec的使用和管理，大大简化IPSec的配置和维护工作

IKE与IPSec关系：

- IKE位于UDP之上，属于应用层协议
- IKE为IPSec协商建立SA，并将参数与密钥交给IPSec
- IPSec使用IKE建立的SA对IP报文加密或验证处理
- AH和ESP的协议号是51和50



IKE的作用：

- 降低手工配置的复杂度
- SA和密钥定时更新
- 允许IPSec提供反重放服务
- 允许在端与端之间动态认证

阶段1

- 在两台计算机间建立一个安全、认证的信道
- 对通信双方进行认证和保护
- 协商拟使用的 ISAKMP SA策略：加密、认证算法，共享密钥
- 实施共享密钥交换
- 为第二阶段建立一个安全的通道
- 两种工作模式：Main mode or Aggressive mode
 - Main Mode IKE
 - 协商加密和哈希的算法
 - 使用DH算法生成共享密钥
 - 身份校验
 - Aggressive Mode IKE
 - 用更少的数据包来传送简化后的算法和密钥交换的协商过程
 - 优点：低带宽更快捷
 - 不足：可能被嗅探

阶段2

- 协商 IPSec SA 参数
- 为特定连接（如 FTP、telnet 等）建立 IPSec SA
- 定期重新协商 IPSec SAs
- 可选择执行额外的 Diffie-Hellman 交换

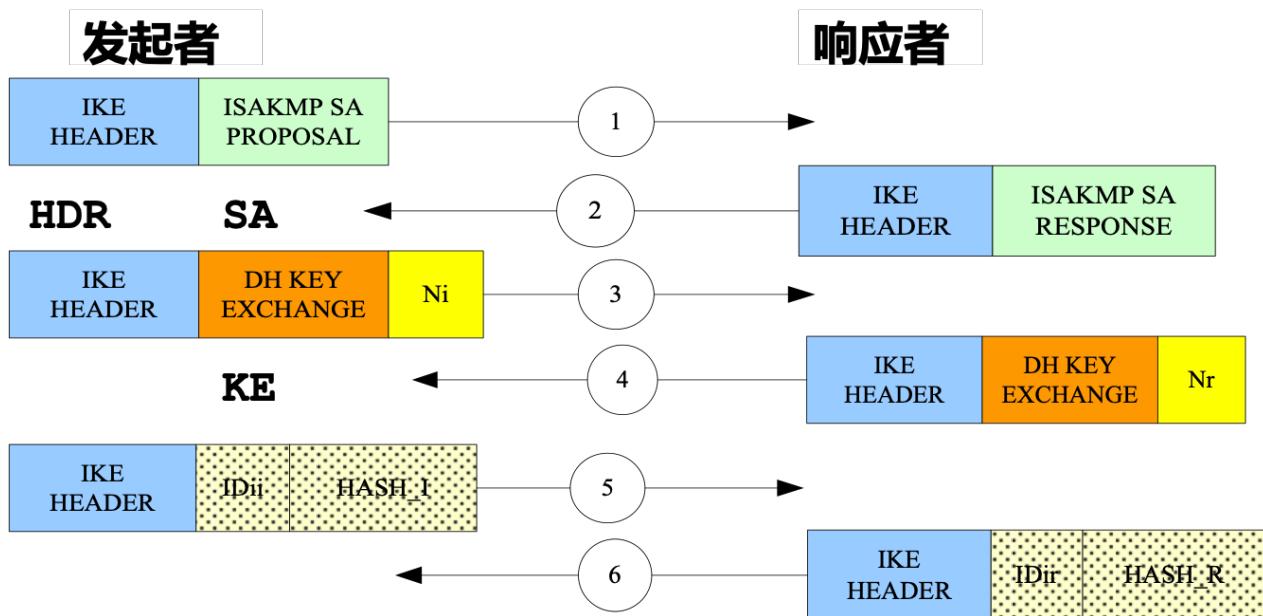
PFS (Perfect Forward Secrecy) :

完美前向机密性，一个key失窃了，只会影响到此key加密的数据。如果通过此key产生新的key，则受它加密的内容不会受到影响

当在备忘录中使用时，完美前向保密 (PFS) 是指这样一种概念，即由单个密钥组成将只允许访问受单个密钥保护的数据。为了使 PFS 存在，用于保护数据传输的密钥不得用于派生任何其他密钥，并且如果用于保护数据传输的密钥是从某些其他密钥材料派生的，则该材料不得用于派生更多密钥。

When used in the memo Perfect Forward Secrecy (PFS) refers to the notion that composed of a single key will permit access to only data protected by a single key. For PFS to exist the key used to protect transmission of data MUST NOT be used to derive any additional keys, and if the key used to protect transmission of data was derived from some other keying material, that material MUST NOT be used to derive any more keys.

主模式：用预共享密钥认证



- HDR contains CKY-I | CKY-R , 是ISAKMP头标
- SA带有一个或多个建议的安全关联载荷。Ni/Nr是nonce值
- KE = g^i (Initiator) or g^r (Responder) , 是密钥交换载荷
- IDii/IDir是发起者/响应者的标识载荷
- HASH是杂凑载荷

1. 发起者向响应者发送一个封装有建议（如加密算法、认证算法及认证方式等）的SA载荷
2. 响应者发送一个SA载荷，表明它所接受的正在协商的SA建议
- 1、2. 协商了CKY-I, CKY-R; SA带有一个或多个建议载荷、变换载荷，确定了加密、认证算法等
- 3、4. 发起者和响应者交换D-H公开值和随机数（nonce），该nonce是计算共享密钥所必须的

交换了 g^i , g^r 和Ni,Nr,生成密钥:

```

SKEYID=PRF(preshared key, Ni|Nr)
SKEYID_d = PRF(SKEYID, g^ir|CKY-i|CKY-r|0)
SKEYID_a = prf(SKEYID, SKEYID_d|g^ir|CKY-I|CKY-R|1)
SKEYID_e = prf(SKEYID, SKEYID_a|g^ir|CKY-I|CKY-R|2)

```

- 5、6. 发起者和响应者交换标识数据（如节点的IP地址、域名等）并认证D-H交换，这两个消息是加密的，采用3和4中交换的信息生成密钥

```

HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b )
HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAr_b | IDir_b )

```

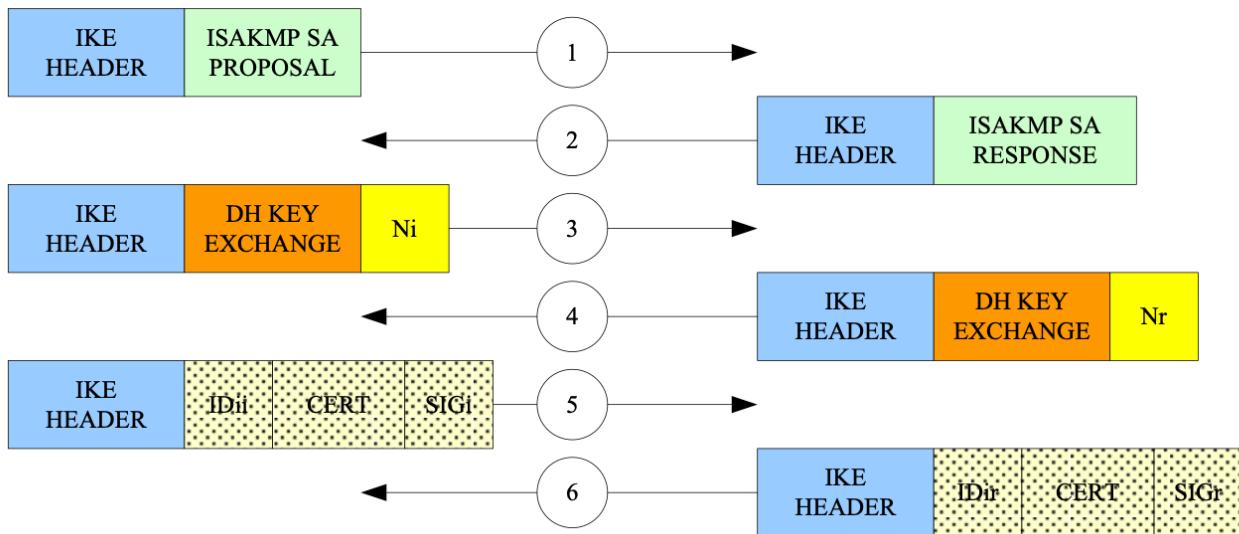
主模式：用数字签名认证

IKE Phase 1 Authenticated With Signatures

Main Mode

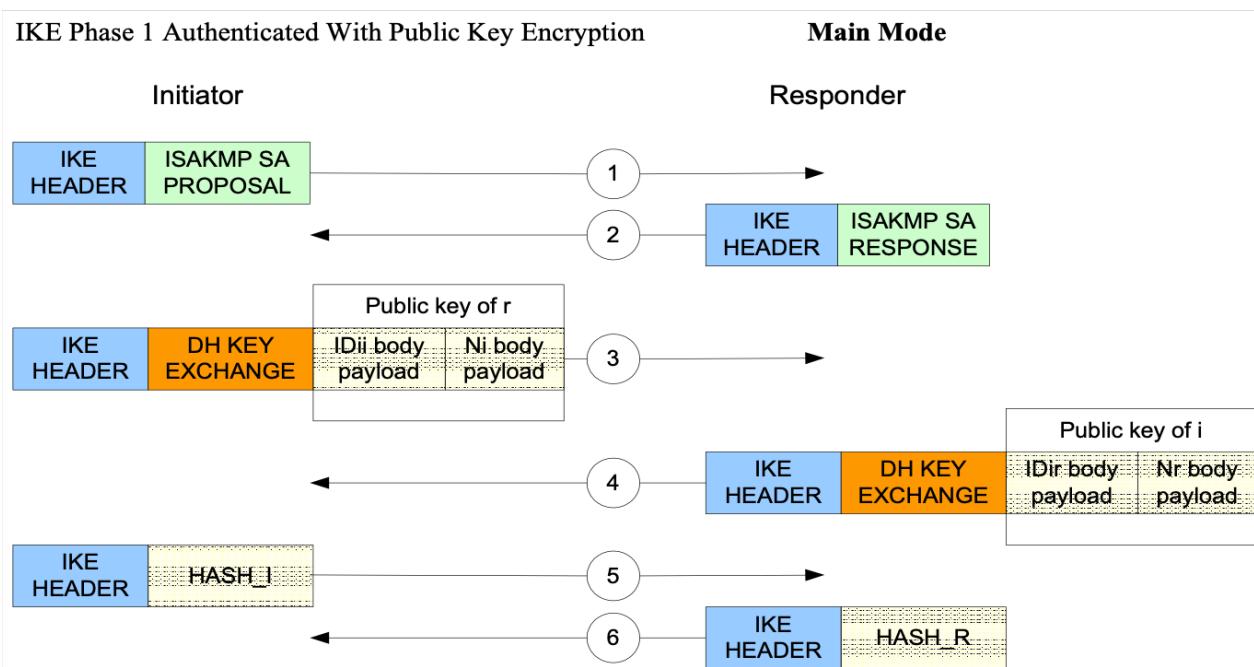
Initiator

Responder



- HDR contains CKY-I | CKY-R
- KE = g^i (Initiator) or g^r (Responder)
- SIG_I/SIG_R = digital sig of HASH_I/HASH_R
- CERT是证书载荷

主模式：用公钥加密认证



- HDR contains CKY-I | CKY-R , HASH (1) 是响应方证书的HASH值
- KE = g^i (Initiator) or g^r (Responder)
- 这里nonce是通过加密传输的，因此可作为共享秘密，HASH值就可以直接用来认证对方的身份。

IKE的协商过程

两个阶段

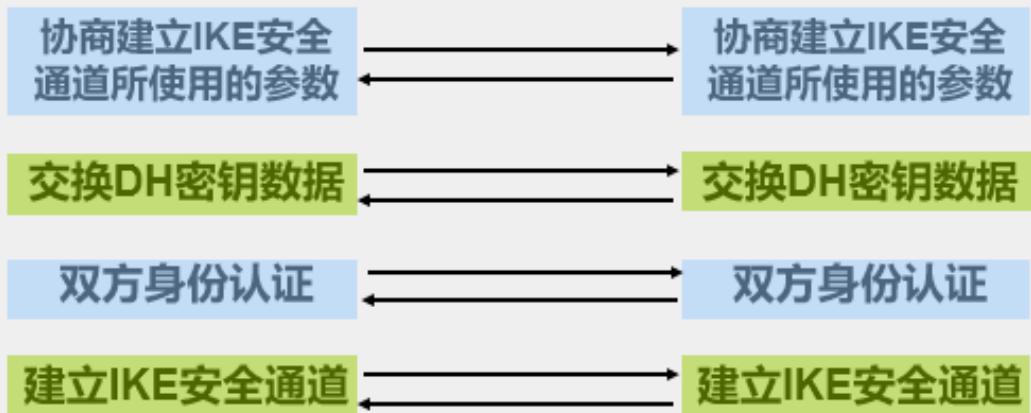
- 阶段一：在网络上建立IKE SA，为阶段二提供保护和快速协商。通过协商创建一个通信信道，并对其进行认证，为通信提供机密性、消息完整性以及消息源认证服务。
- 阶段二：在IKE SA的保护下完成IPSec的协商。

交换信息

- SA交换，协商确认有关安全策略的过程。
- 密钥交换，交换Diffie-Hellman公共值和辅助数据，产生加密密钥。
- ID交换和验证数据交换，进行身份验证和对整个SA交换进行验证。

阶段一协商过程

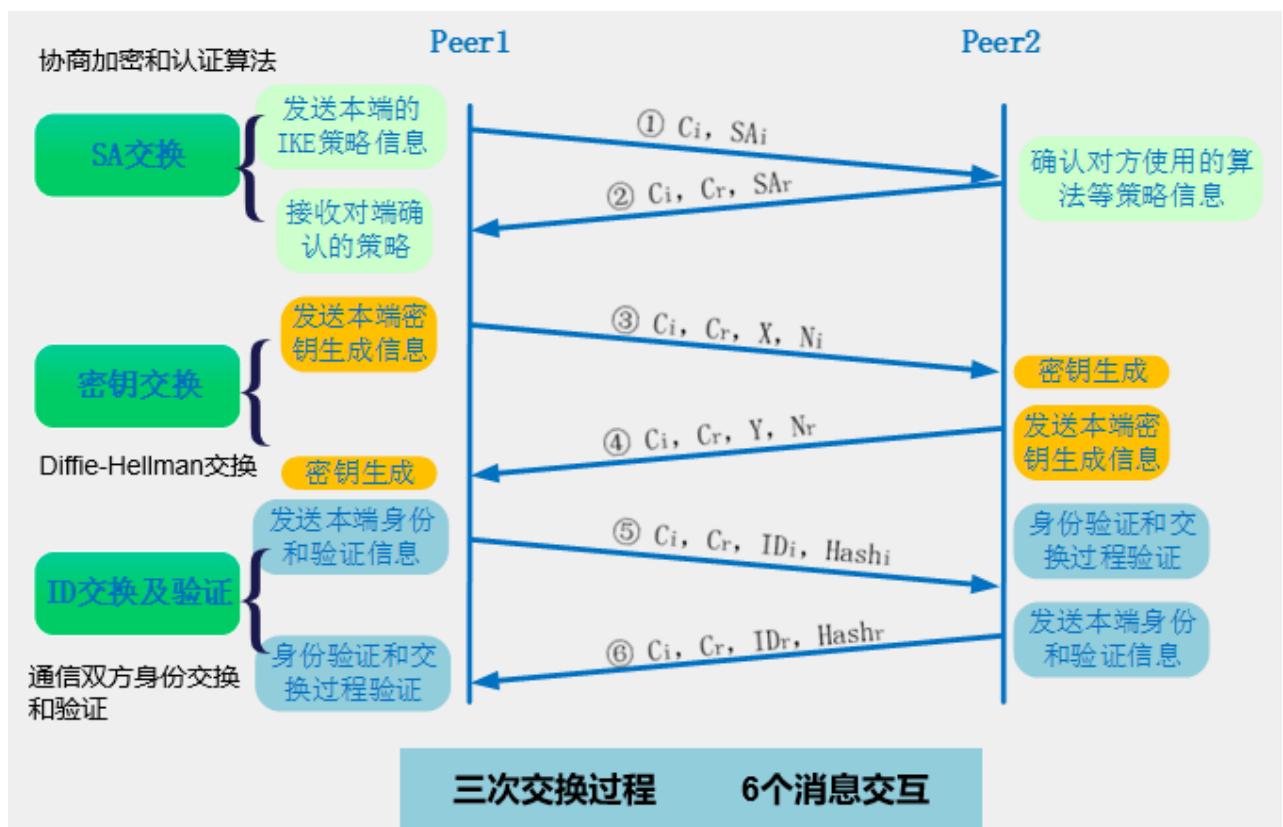
双方建立了一个已通过身份验证和安全保护的通道，此阶段建立了一个ISAKMP。



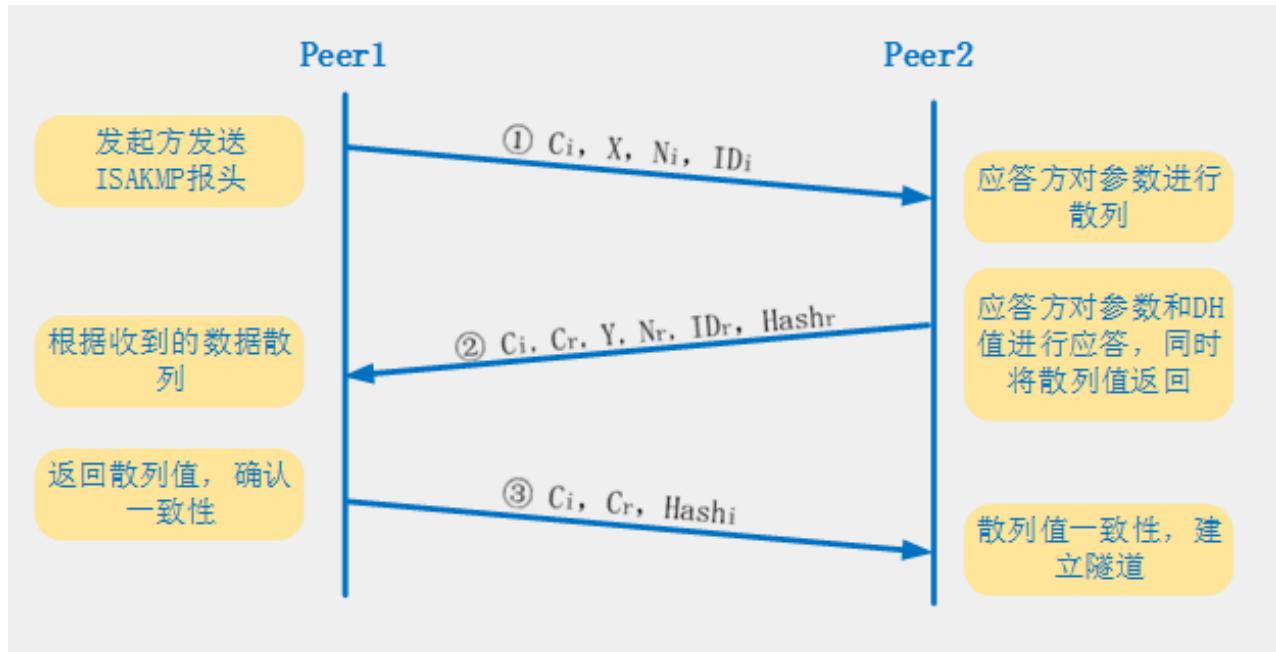
- 两种协商模式

- 主模式协商

- 适合两设备的公网IP固定，实现设备之间点对点的环境。



- 野蛮模式协商



	主模式	野蛮模式
消息交互	交互6个消息	交互3个消息
身份ID	以IP地址作为身份ID，自动生成本端身份ID和对端身份ID	可以以多种形式（IP，字符串等）手动或自动的生成本端和对端的身份ID
域共享密钥	只能基于IP地址来确定预共享密钥。	基于ID信息（主机名和IP地址）来确定预共享密钥。
安全性	较高 前4个消息以明文传输，最后两个消息加密，对对端身份进行了保护	较低 前两个消息以明文传输，最后一个消息进行加密，不保护对端身份
速度	较慢	较快

阶段二协商过程

- 使用“快速模式”交换，实现两个主要功能
 - 协商安全参数保护数据连接
 - 周期性地更新密钥信息
- 协商出IPSec单向SA，保护数据流。
- 协商过程受第一阶段ISAKMP/IKE SA保护。



Chapter5 SSL

不同协议层的安全

HTTP	FTP	SMTP
TCP		
IP/IPSEC		

At the Network Level

HTTP	FTP	SMTP
SSL/TLS		
TCP		
IP		

At the Transport Level

	S/MIME	PGP	SET
Kerberos	SMTP	HTTP	
UDP	TCP		
IP			

At the Application Level

SSL概念

- 用于加固http层安全，主要用于web的安全传输协议
- TLS (IETF进行了标准化) 向后兼容 SSLv3
- 目的是使得TCP实现可靠、端对端的服务
- 两部分组成：
 - SSL 记录协议
 - SSL 管理协议 (Handshake/Cipher Change/Alert Protocols)

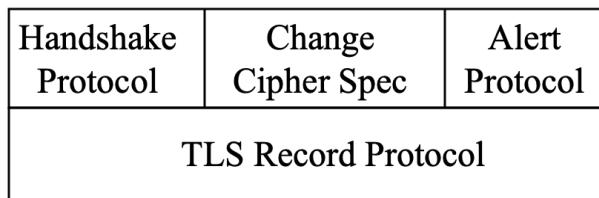
安全机制

- 机密性：使用对称密钥算法对传输的数据进行加密。
- 身份验证：基于证书利用数字签名对server和client进行身份验证。

- 消息完整性验证：使用MAC算法检验消息的完整性

SSL的分层结构

- 上层协议
 - Handshake：协商密钥
 - Cipher Change：中途改变密钥
 - Alert Protocols：处理错误，纠错协议，遇到非正常状态的处理
- 下层协议
 - SSL 记录协议



SSL基本过程

- 建立一个会话
 - 协商算法
 - Share secrets
 - 实施认证
- 传输应用数据
 - 确保机密性和完整性

握手协议

SSL的核心协议部分

完成在传输应用数据之前进行的准备工作：

- 相互认证
- 协商加密算法
- 建立共享密钥

阶段1：建立安全能力

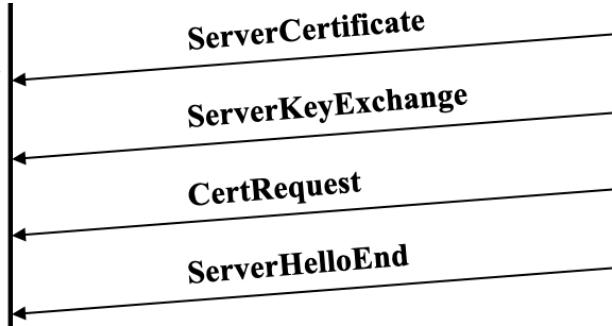


$A \rightarrow G : \{vers\#, r_A, SessID, CiphList, CompList\}$
 $G \rightarrow A : \{vers\#, r_G, SessID, CiphChoice, CompChoice\}$

- 明文传输
- vers#：支持的协议版本
- r_A ：生成的随机数，用于生成“会话密钥”

- CipherList: 支持的加密方法
- CompList: 支持的压缩方法

阶段2：服务器认证与密钥交换



$G \rightarrow A : \{G_X509Cert\}$
 $G \rightarrow A : \{(n_G, e_G) || E_{K_G}[\text{hash}(r_A || r_G || (n_G, e_G))]\}$
 $G \rightarrow A : \{\text{CertType} || \text{ValidCertAuthorities}\}$
 $G \rightarrow A : \{\text{EndHello}\}$

- server发送证书，包含一个X.509证书，或一条证书链
- server发送server_key_exchange消息
 - 可选，服务器证书没有包含必需数据时发送
 - 包含签名，签名内容包括两个随机数以及服务器参数
- server发送certificate_request消息
 - 非匿名server可以像client请求一个证书
 - 包含证书类型和CAs
- 服务器发送server_hello_done，等待应答

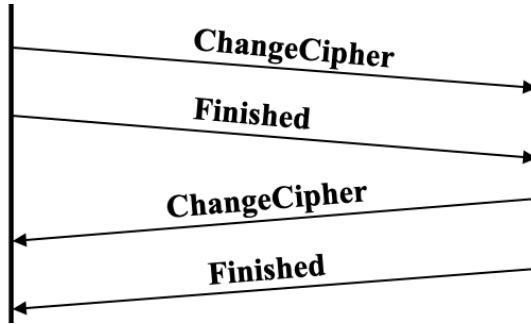
阶段3：客户机认证与密钥交换



$A \rightarrow G : \{A_X509Cert\}$
 $A \rightarrow G : \{E_{K_G}[S_{PM}]\}$
 $A \rightarrow G : \{\text{hash}(MS || r_G || \text{hash(Messages1to8 || MS || r_A)})\}$
 $MS = MD5(s_{PM} || SHA1('A' || s_{PM} || r_A || r_G)) ||$
 $MD5(s_{PM} || SHA1('BB' || s_{PM} || r_A || r_G)) ||$
 $MD5(s_{PM} || SHA1('CCC' || s_{PM} || r_A || r_G))$

- 客户校验证书有效性和参数有效性
- 若服务器请求了证书，客户则发送自己的证书
- 服务器生成PreMasterSecret: SPM
- MS is master secret and Step 9 is for verification

Cipher Change



$A \rightarrow G : \{ChangeCipher\}$

$A \rightarrow G : \{hash(MS||r_G||hash(Messages1to9||Client||MS||r_A)\}$

$G \rightarrow A : \{CipherChanged\}$

$G \rightarrow A : \{hash(MS||r_G||hash(Messages1to9||Server||MS||r_A)\}$

- 客户发出 change cipher 请求 (via the Change Cipher Protocol)
- 服务器响应该请求
- 收尾的消息：用于校验的hash值
- 在新的密码参数达成一致后发送一个字节

Alert Protocols

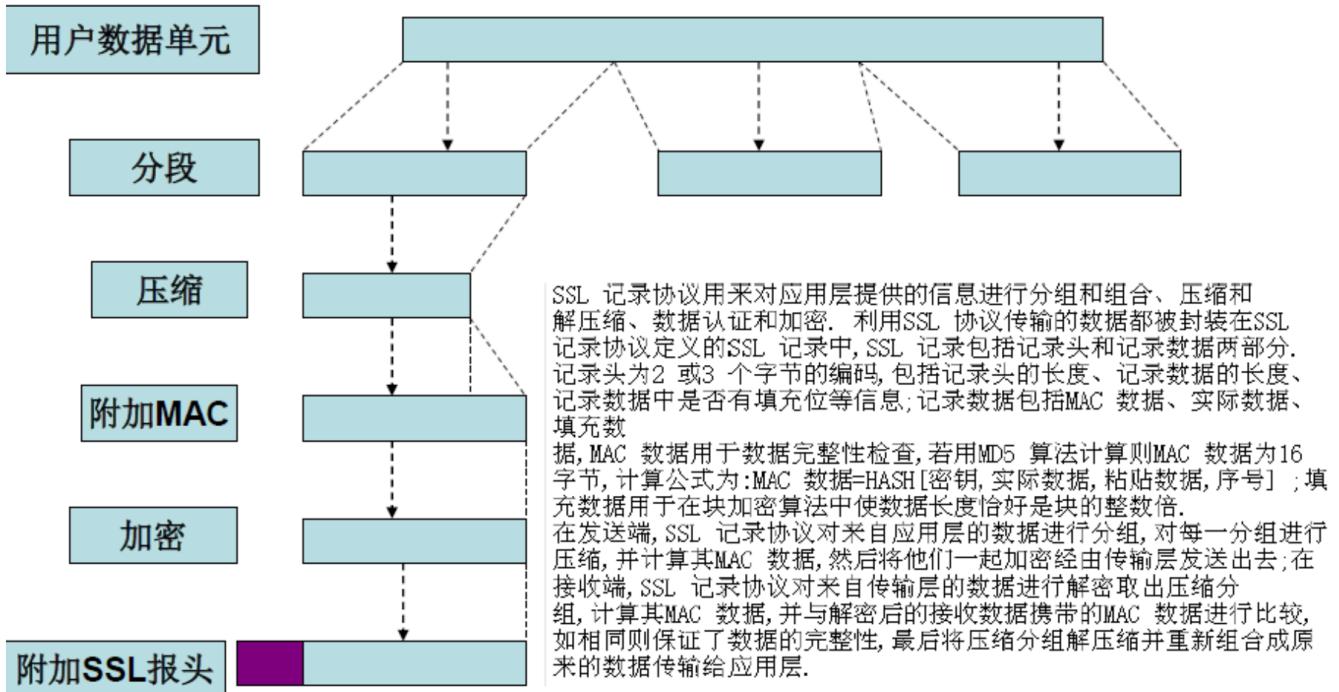
当握手过程或者数据加密等操作出错或者发生异常情况时，向对方发出警告或中止当前连接

传输应用数据

- 通过握手协议建立一个SSL会话，拥有一组安全参数
- 一个SSL会话可以对应多个SSL连接，这些连接可以使用相同的安全参数
- SSL连接
 - 点对点
 - 连接是暂时的，每个连接和一个会话关联
- SSL会话
 - 会话是在server和client之间的一个关联，由握手协议建立，定义了一组密码安全参数
 - 避免为每一个连接提供新的安全参数所需昂贵的协商代价

记录协议

SSL数据单元的形成过程



SSL安全性分析

- 鉴别机制：客户端与服务器交换了证书
- 加密机制：对称加密保护数据传输，非对称加密协商会话密钥
- 完整性机制：数据分组压缩后，产生MAC
- 抗重放攻击：使用序列号，传输中被加密

SSL脆弱性分析

- 客户端假冒
- 无法提供基于UDP应用的安全保护
- 不能对抗通信流量分析
- 进程中主密钥泄露
- handshake、数据加解密耗时

Chapter6 SET

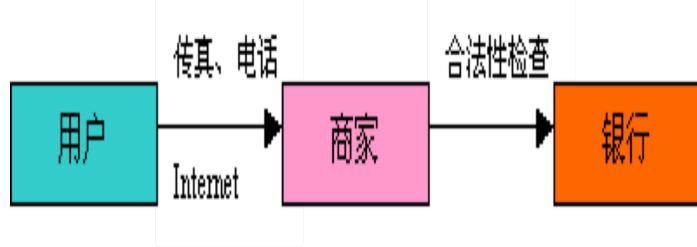
信用卡的安全支付，**信息的按需分配**

问题.信用卡，交易数据，信任关系：

- 买家和商家的认证
- 数据传输的保密性

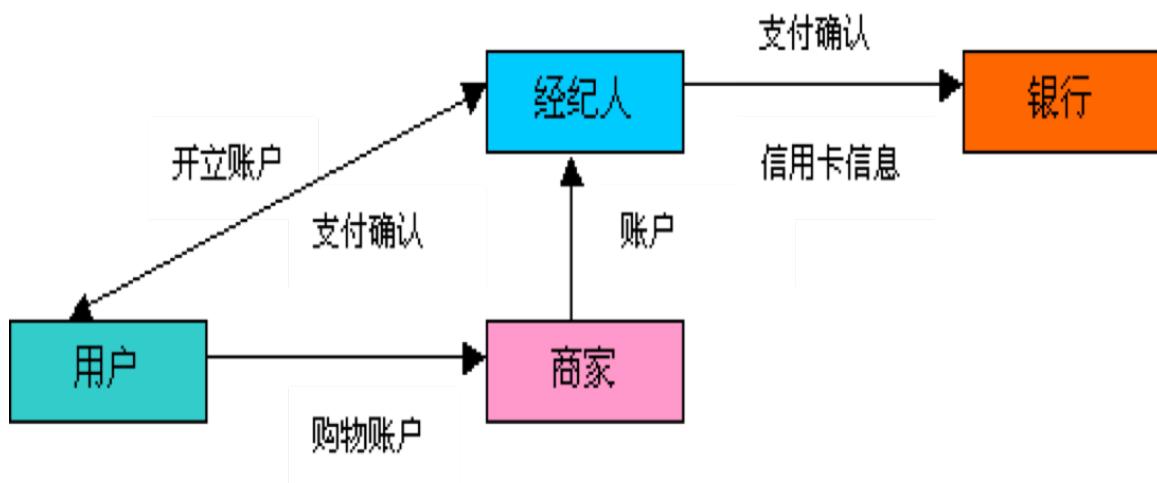
电子交易的主要模式

支付系统无安全措施的模式



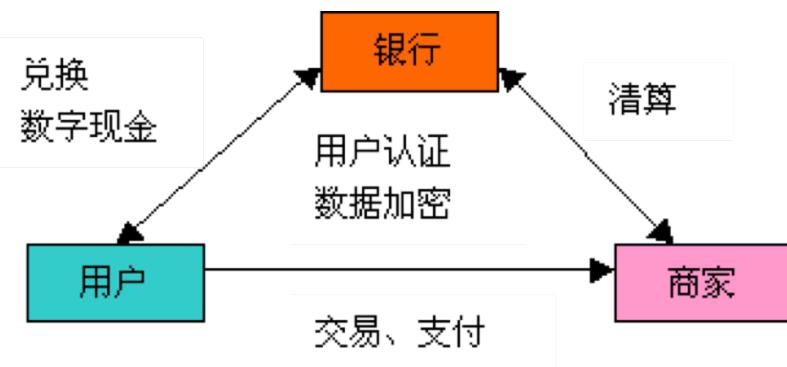
- 风险由商家承担
- 商家完全掌握用户的信用卡信息
- 信用卡信息的传递无安全保障

通过第三方代理人支付的模式



- 用户账户的开设不通过网络
- 信用卡信息不在开放的网络上传送
- 通过电子邮件来确认用户身份
- 商家自由度大，风险小
- 支付是通过双方都信任的第三方(经纪人)完成的

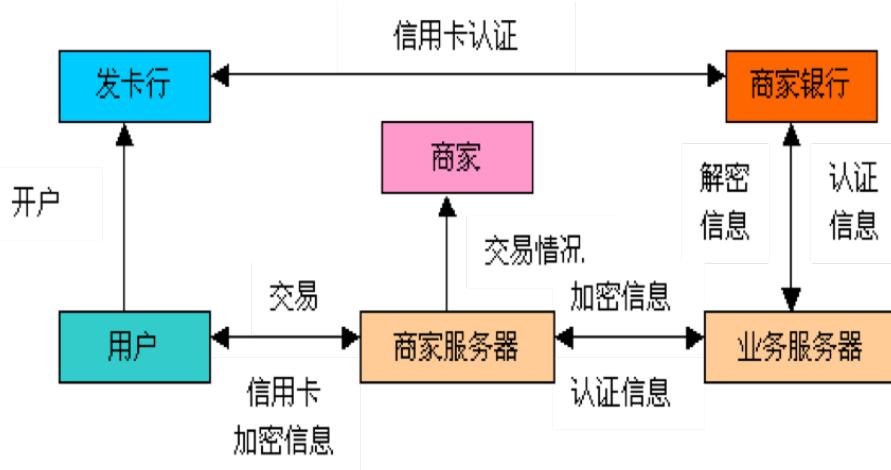
数字现金支付模式



- 银行和商家之间应有协议和授权关系
- 用户、商家和数字现金的发行都需要使用数字现金软件
- 适用于小额交易

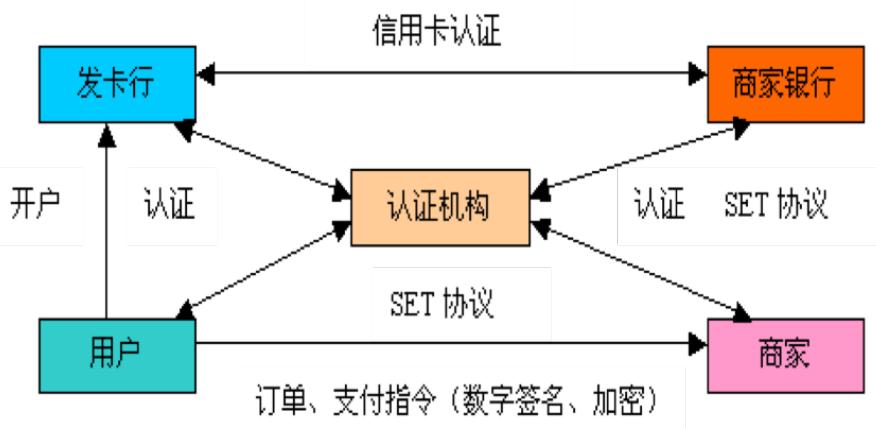
- 身份验证是由数字现金本身完成的
- 数字现金的发行负责用户和商家之间实际资金的转移
- 数字现金与普通现金一样，可以存、取和转让

简单加密支付系统模式



- 信用卡等关键信息需要加密
- 使用对称和非对称加密技术
- 可能要启用身份认证系统
- 以数字签名确认信息的真实性
- 需要业务服务器和服务软件的支持

安全电子交易SET支付模式



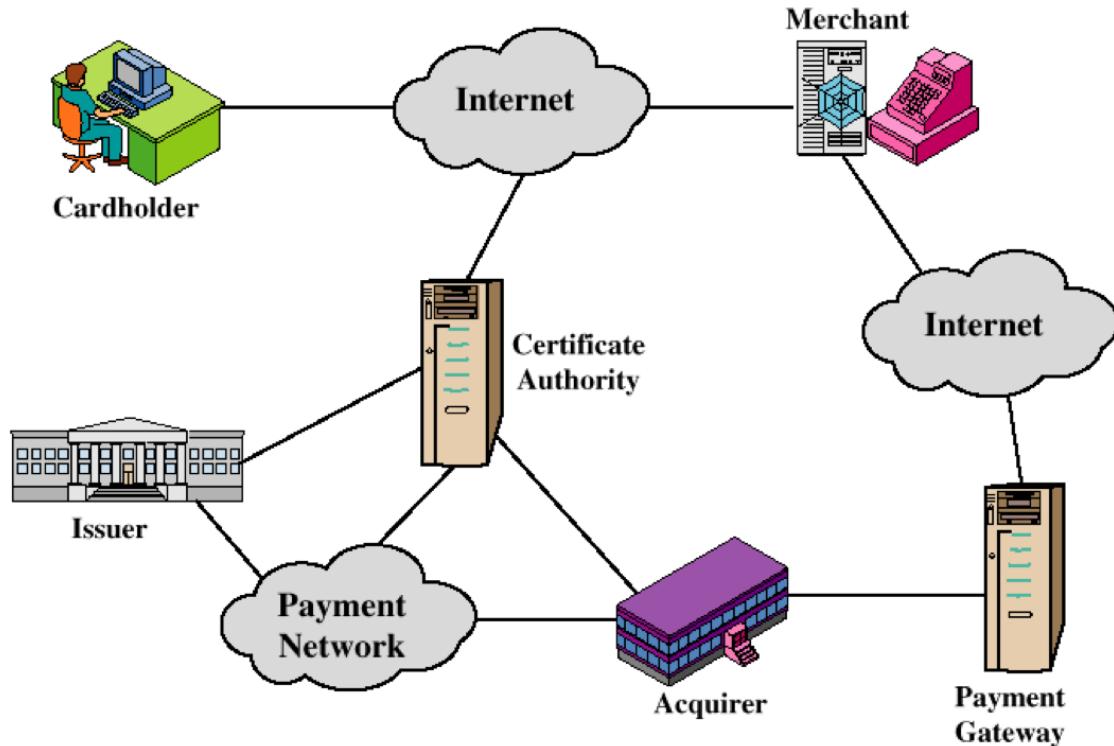
- 信息在互联网上安全传输，不能被窃听或篡改
- 用户资料要妥善保护，商家只能看到订货信息，看不到用户的账户信息
- 持卡人和商家相互认证，以确定对方身份
- 软件遵循相同的协议和消息格式，具有兼容性和互操作性

SET概述

- 使用信用卡交易的官方规范
- 消息加密、各方持有数字证书、信息按需供应
- SET 仅仅关心支付问题

交易中的主体

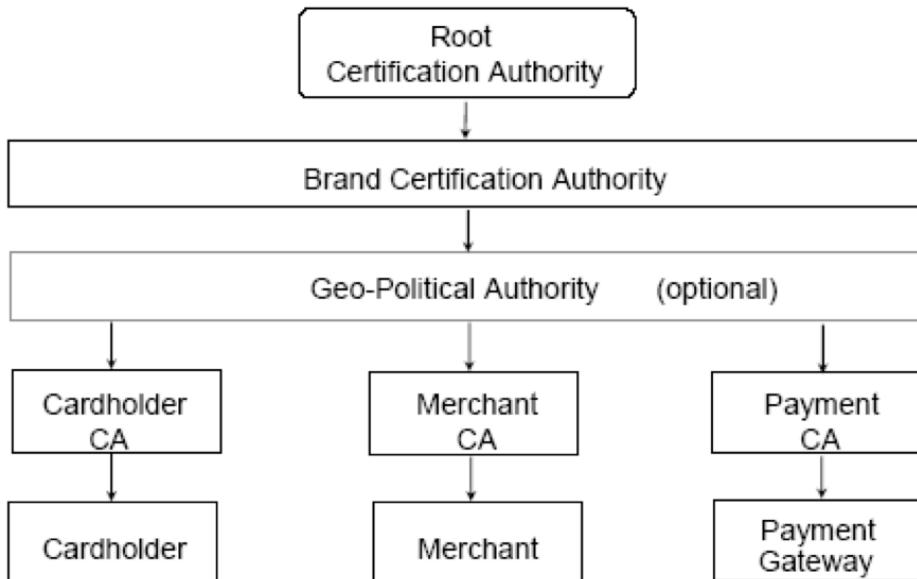
- 持卡人 (cardholder)
- 发卡机构 (Issuer)
- 商家 (Merchant)
- 收单银行 (Acquirer)
- 支付网关 (Payment Gateway)
- 认证中心 (CA)



Payment Network: 支付网络，相对封闭，安全性高

主体证书

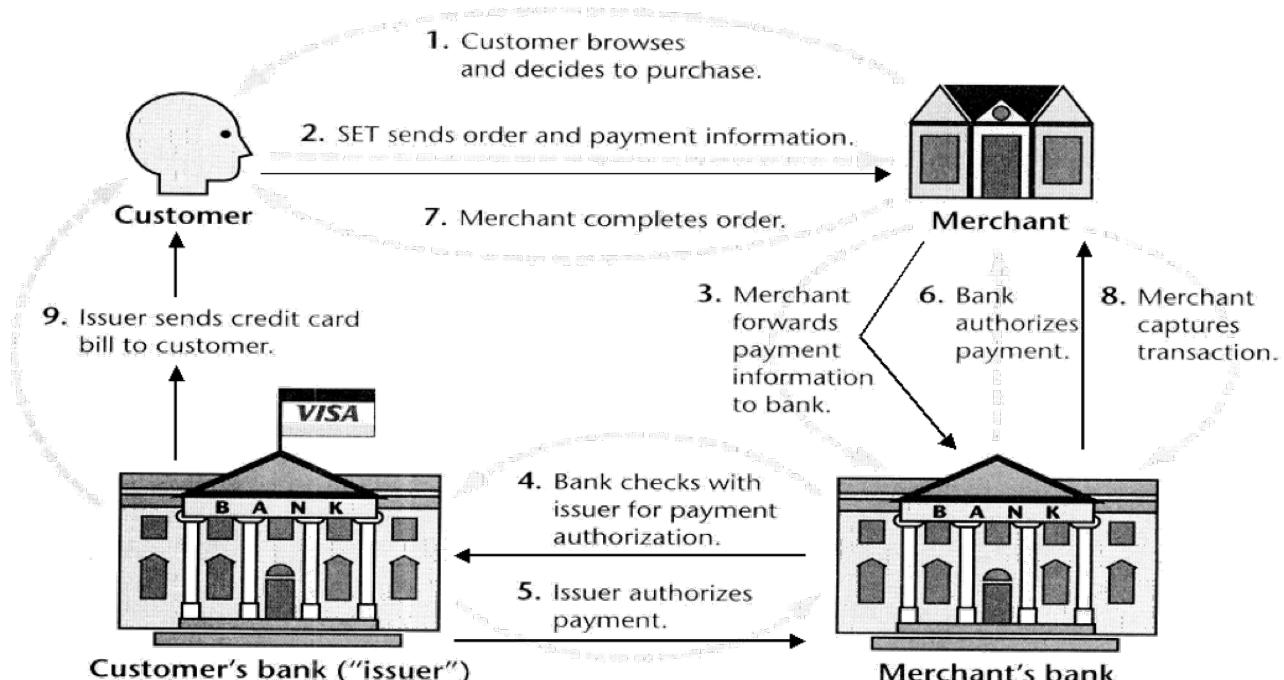
- 协议各方持有名字和密钥对
- 身份使用 X.509v3 证书和密钥关联



电子商务的安全架构需求

- 支付订单信息的机密性
- 传输数据的完整性
- 卡持有者身份的合法性认证
- 商家的身份认证
- 保证参与方的利益
- 独立于传输安全
- 软件架构沟通性

电子商务的典型场景



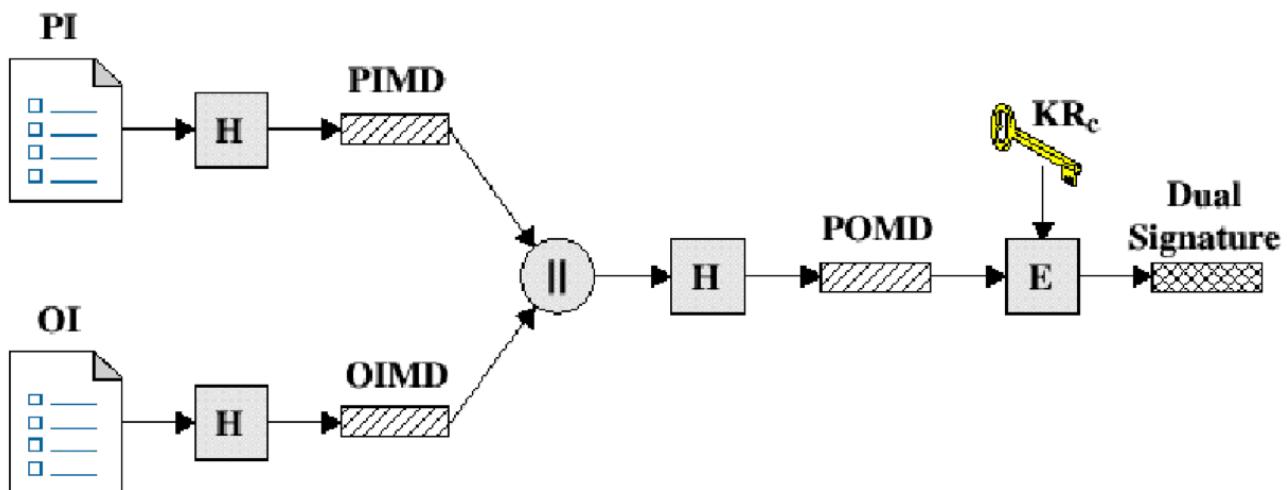
3. 检验有无那么多存款
4. 授权支付
5. 商家把钱转到自己账户，一段时间统一次

电子支付的流程

- 客户在发卡行开户
- 客户持有银行签发的 X.509 V3 证书
- 商家持有两个同类品牌的证书X.509 V3
 - 一个用于签名 &一个用于密钥交换
- 客户向商家发订单
- 商家发送证书拷贝向客户出示自己身份
- 客户发送订单和支付信息给商家
- 商家向支付网关请求支付授权
- 商家确认向客户订单
- 商家向客户提供商品或者服务
- 商家向支付网关请求支付

双重数字签名

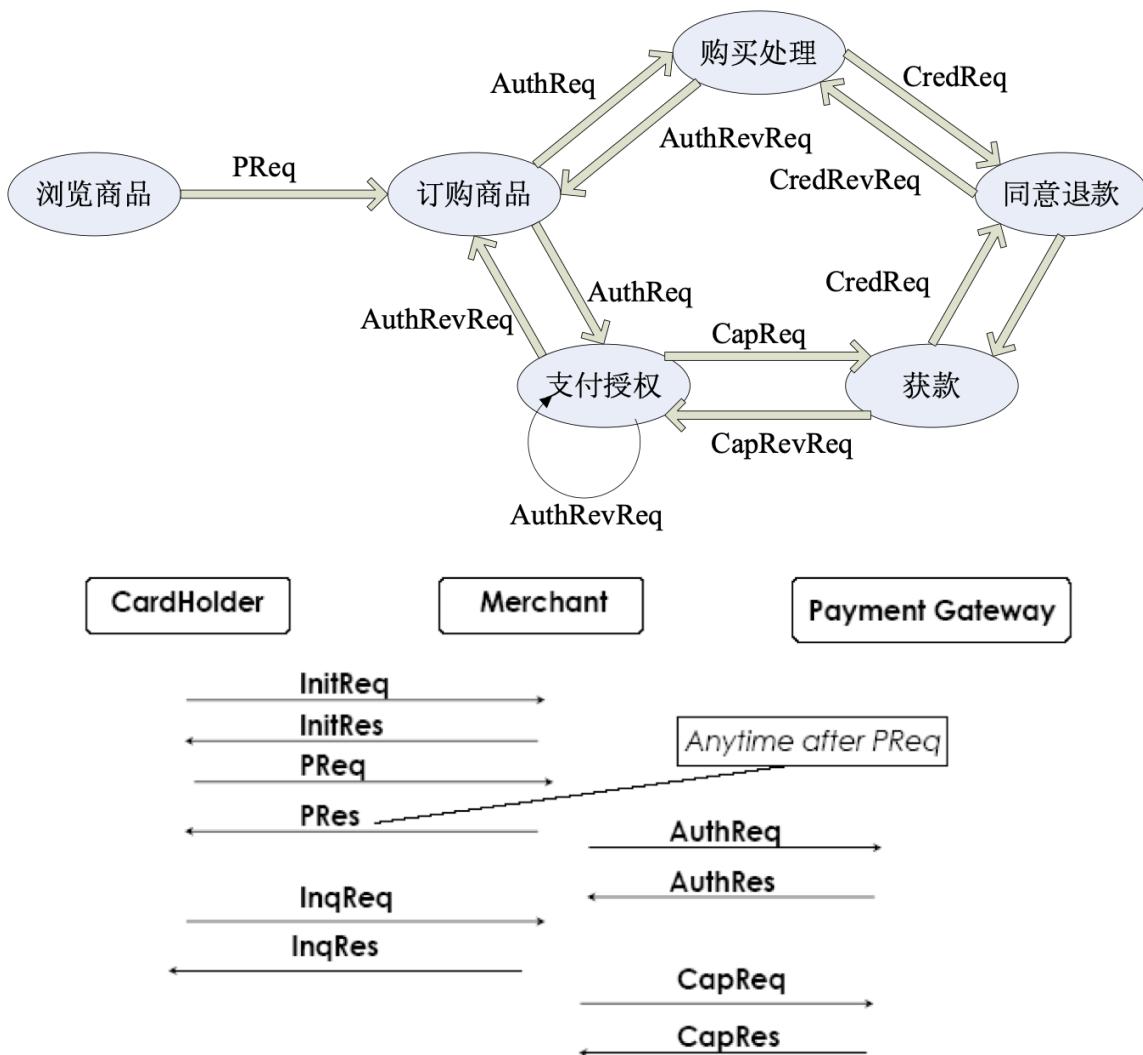
- **原因和机理**
- 概念：将两个消息连接在一起，这两个消息面向的对象不同
 - Order Information (OI): 客户给商家，订单信息（买什么，几件，价格）
 - Payment Information (PI): 客户给银行，支付信息（多少钱，从哪里扣）
- 目标：按需分发消息
- 商家不需要支付信息
- 银行不需要订单信息
- 保护客户隐私
- 确保信息不被非法使用
- 否则的话，商家有可能将多个支付信息和订单信息交叉组合，牟利



- 具体操作：
 - 将PI和OI分别初次hash

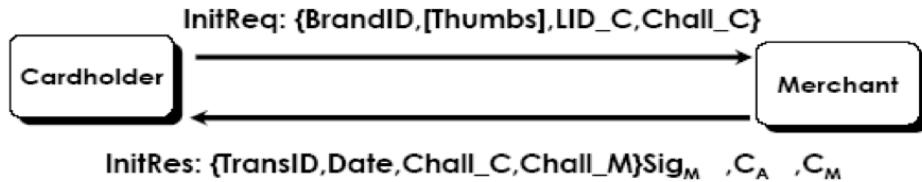
- 连接成 $[H(PI) || H(OI)]$ 再hash
- 客户私钥加密产生双重签名： $DS = E_{KRC} [H(H(PI) || H(OI))]$
- 商家收到 OI 校验签名
- 银行收到 PI 校验签名
- 客户连接OI和PI，证明该关联
 - 特约商店计算出两个数： $H(PIMD||H(OI))$ 和 $DK_{Uc}[DS]$
 - 其中 DK_{Uc} 为消费者的公开密钥。如果这两个数计算出的结果相同，则特约商店就可核准这个签名。
 - 相同地，如果银行拥有 DS , PI , 与 OI 的消息摘要($OIMD$)，及消费者的公开密钥，则银行可计算： $H(H(PI)||OIMD)$ 和 $DK_{Uc}[DS]$

SET 交易流程



支付过程初始化

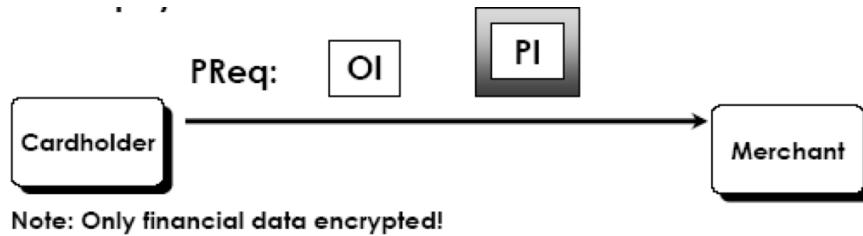
InitReq/InitRes: 明文



- 持卡人向商家发送初始请求，包括
 - 交易卡ID、身份标识、挑战随机数
- 商家接收初始请求，产生初始应答，对初始应答生成消息摘要，并进行数字签名，包括
 - 摘要签名、支付网关证书、商家证书

下单过程

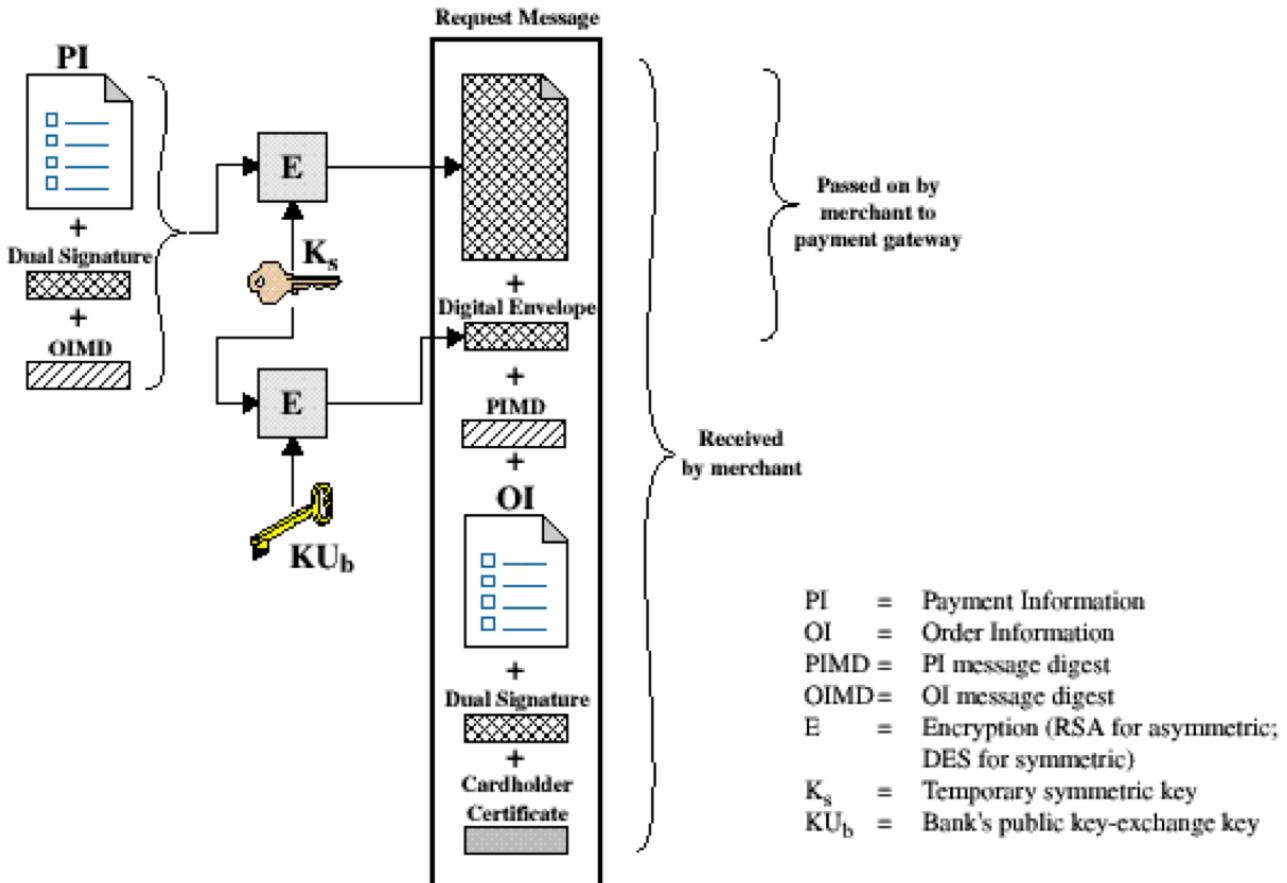
PReq/PRes



- 持卡人接收初始应答，检查商家证书和网关证书，用商家公钥解开数字签名，验证数据未被篡改，否则丢弃
- 持卡人发出购物请求，包含了真正的交易行为，包括
 - 发往商家的订单信息（OI）
 - 通过商家转发往网关的支付信息（PI）
- 通过双重数字签名将OI与PI进行关联。
- PI被加密，商家只能看到OI

PReq

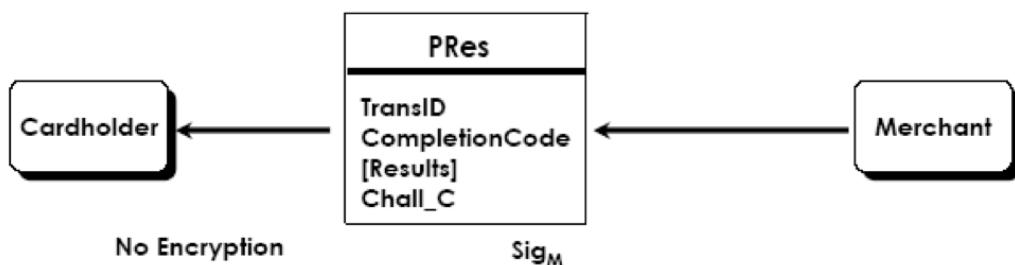
详细视图：



商家验证卡用户的身份和授权：

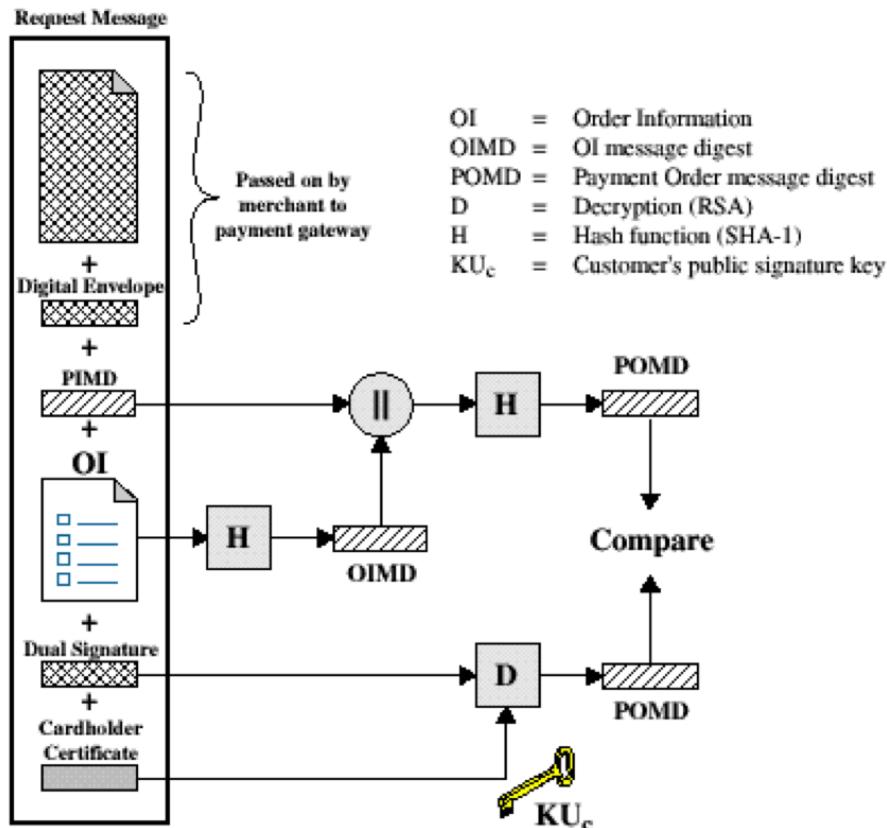
- 存储 PI 以转发给 acquirer
- 遍历信任链，校验持卡人证书
- 校验双重签名
- 从支付网关那里获取授权
- 发送响应给持卡人，确认订单
- 若授权延迟 PRes 给持卡人“请稍后查询”消息

PRes



详细视图：

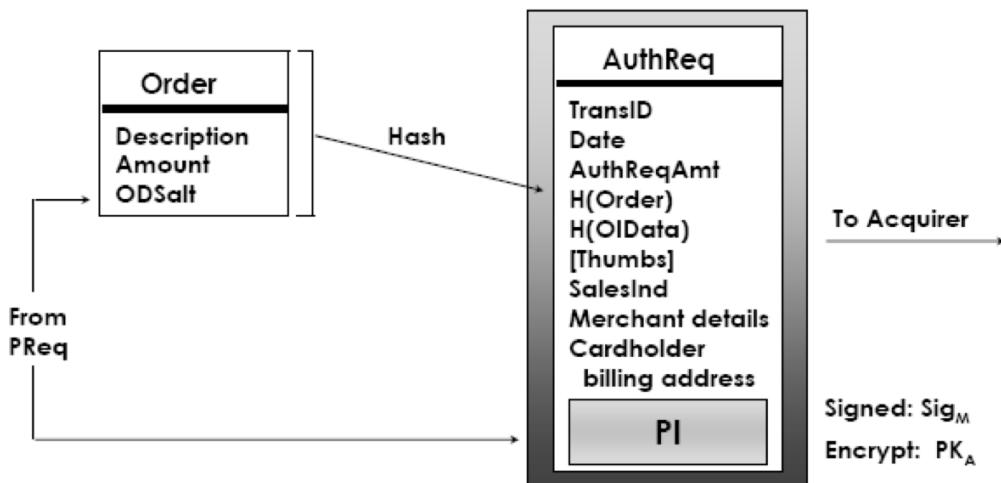
- 验证OI完整性以及与PIMD的捆绑



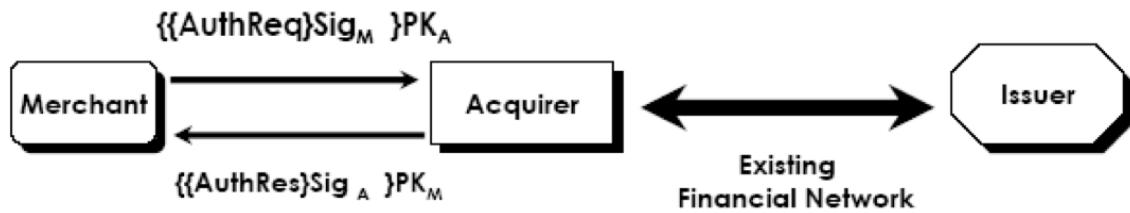
支付网关认证过程

AuthReq/AuthRes

AuthReq



- 校验持卡人的信用
- 含有 PI (来自 PReq)
- 含有 $H(\text{Order})$ ，表明和PI的一致性
- 订单信息明文不发送给 acquirer
- 商家签名并加密，信息是密文
- 结合授权和捕获 = 销售交易
- 根据AuthRes结果运输商品

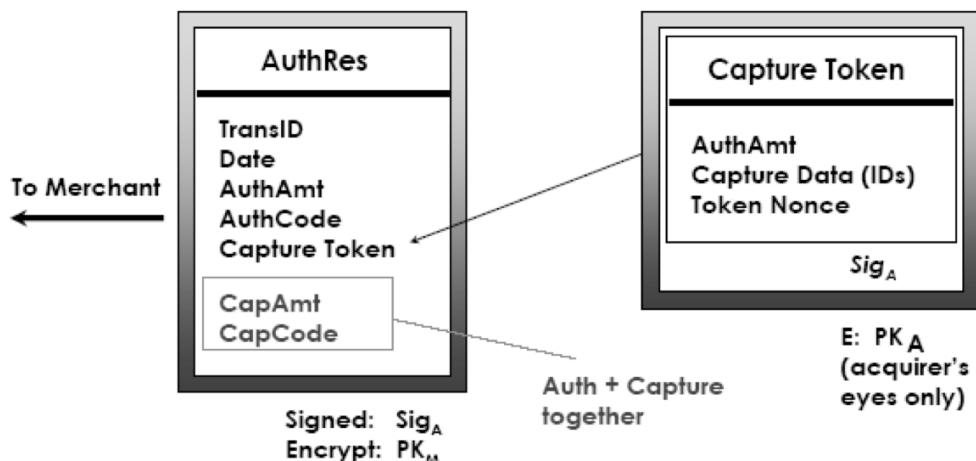


- 左为互联网，右为支付网络

Acquirer处理：

- 解密 AuthReq
- 校验商家签名
- 解密来自于持卡人的 PI
- 校验双重签名
- 从PI中抽取卡数据
- 确保 PI 和 AuthReq 的一致性
- 校验持卡人和商家对于订购行为的一致性：H(Order)，PI 和 AuthReq
- 通过金融网络获取授权信息
- 生成 AuthRes 及其 Capture Token

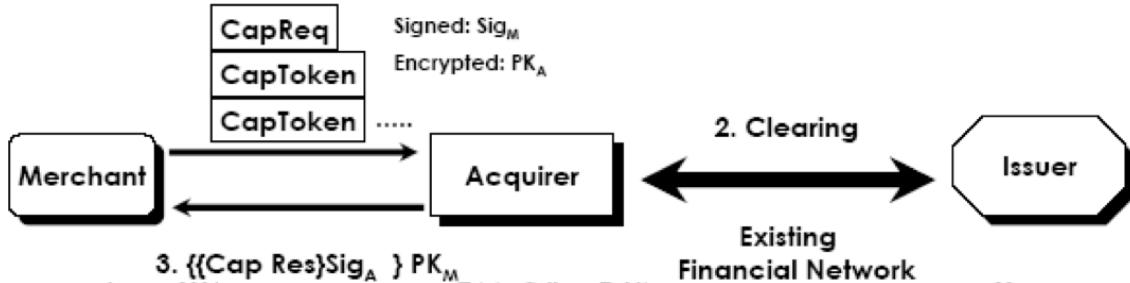
AuthRes



- Capture Token：数字令牌，用于结算时兑现
 - Token Nonce：保持新鲜性

支付完成

- 完成授权，交易的支付
- 通过捕获令牌来完成支付
- 可能多次 AuthResponses 的令牌累积后完成
- Capture Token = 金额证据



SET核心技术

公钥加密、数字签名、数字信封、电子安全证书

SET协议交易过程

- 验证电子证书9次
- 验证数字签名6次
- 传递证书7次
- 进行签名5次
- 对称加密和非对称加密4次

SET相比SSL的不同处

- SET远不止是一个技术方面的协议，它还说明了每一方所持有的数字证书的含义，希望得到数字证书以及响应信息的各方应有的动作，与一笔交易紧密相关的责任分担
- SET实现非常复杂，商家和银行都需要改造系统以实现互操作，并且还需要认证中心的支持
- SET是一个多方的报文协议，它定义了银行、商家、持卡人之间的必须的报文规范
- SSL只是简单地在两方之间建立一条安全连接
- SSL是面向连接的，而SET允许各方之间的报文交换不是实时的
- SET报文能够在银行内部网或者其他网络上传输，而SSL之上的卡支付系统只能与Web浏览器捆绑在一起
- 抛开算力，SET更优秀

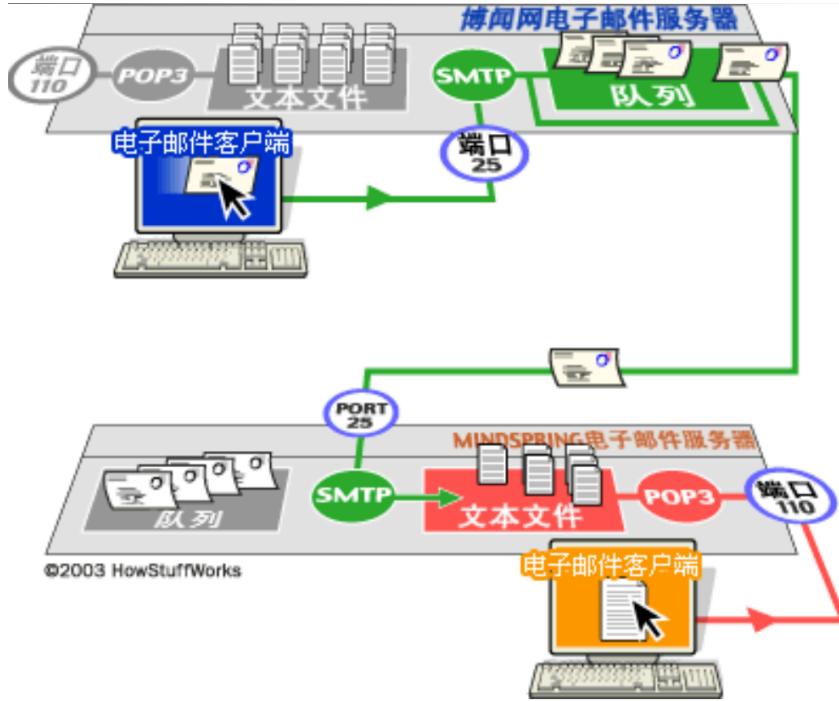
Chapter7 PGP

如何在无政府状态的网络上构造一种信任模型？

如何基于上述模型分享信息？

Email工作原理

- 无连接
- 原始协议都是明文



- 安全的电子邮件主要是解决身份鉴别和保密性的安全问题
- PGP：提供可用于电子邮件和文件存储应用的保密与鉴别服务
- 邮件的存储转发特性不适合用DH算法交换密钥
- 使用公钥算法对每个消息生成一次性会话密钥

PGP公钥的分发

PGP：无政府状态，用户决定信任与否

- 获取某人公钥并信任它
- 将其加入自己的PGP 系统
- 公钥服务器

PGP的证书

证书可选：

- 彼此之间可以颁发证书
- 若你信任A并且获得A签名的C的公钥，则，你信任C的公钥

所以PGP灵活易用

存在的风险

存在的问题PGP无政府结构：

- A被贿赂为C颁发证书
- A疏于检查(密钥和身份的对应)，签发证书

答案：

- 用不用在你
- 给公钥信任度

- 给证书信任度

证书的撤销

- 可以在任意时间撤销(删除)公钥
- 可以通过私钥来撤销公钥
- 证书的颁发者可以撤销证书
- 证书或者密钥的有效期

信任的水平

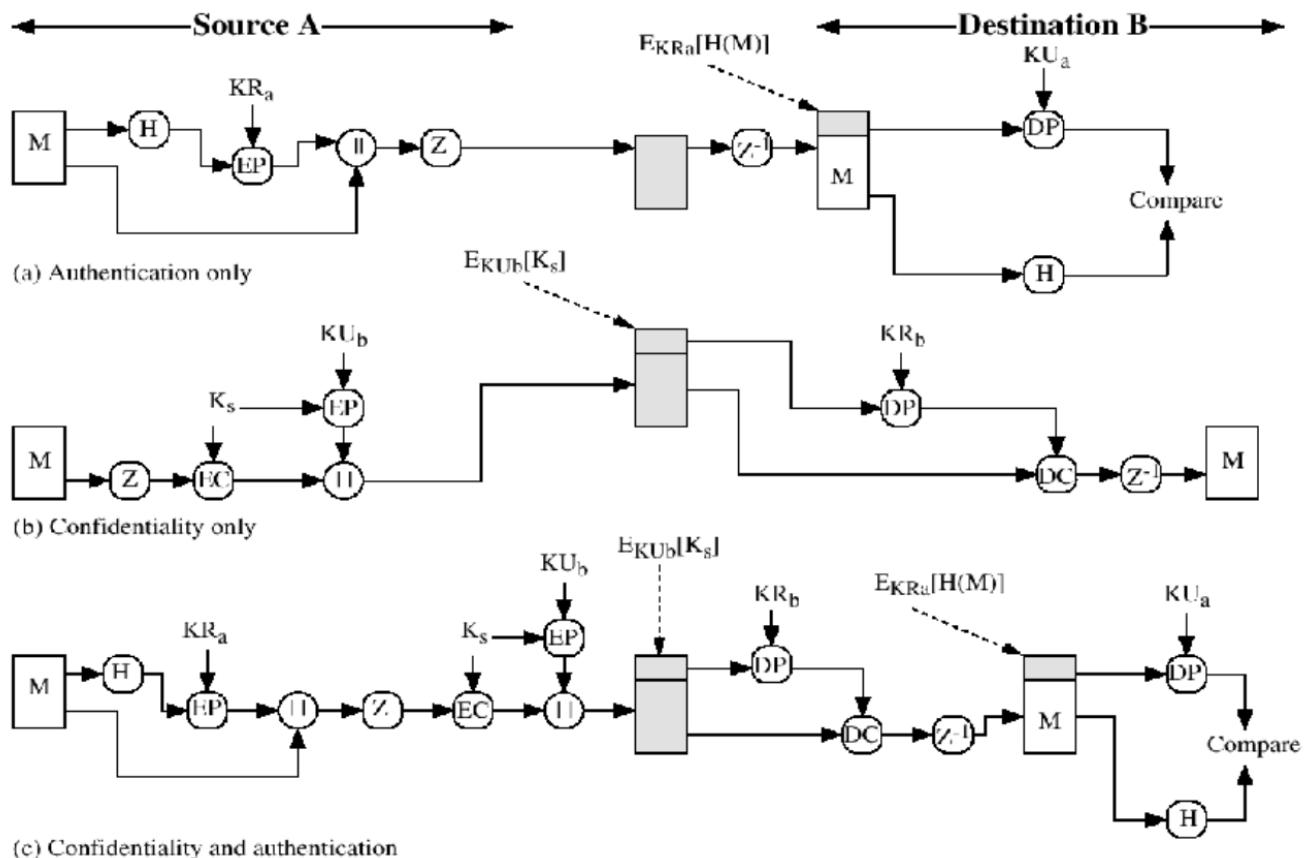
数据结构：

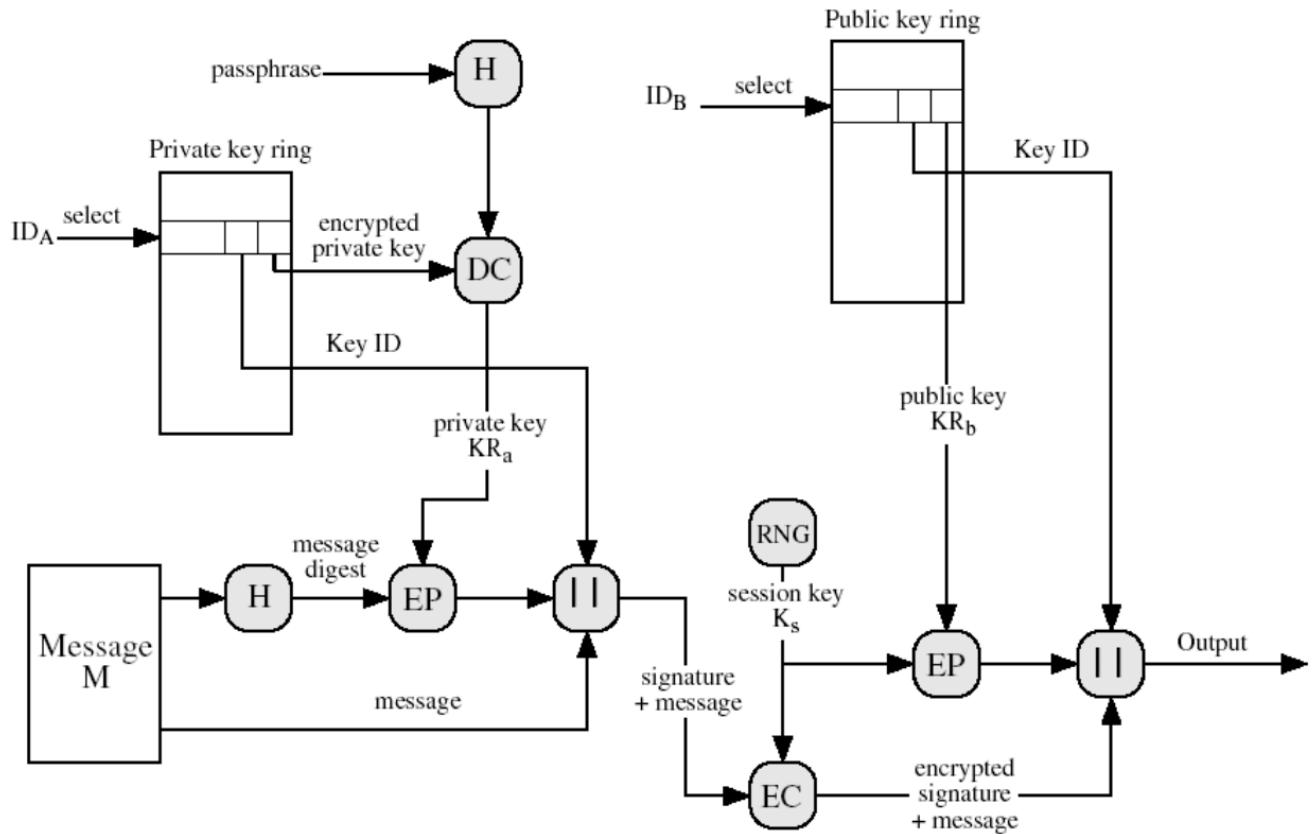
- pubring.pgp：公钥和证书
- secring.pgp：私钥

三种信任水平：none、partial、complete

个体的信任水平决定其签发的证书的信任水平

PGP加密处理过程





PGP发送方：

- 签名消息：
 - PGP 使用用户的id作为索引获取发送者的私钥
 - PGP 提示用户输入口令来解密私钥
 - PGP 创建签名
- 加密消息：
 - PGP 生成会话密钥，加密消息
 - PGP 使用用户ID作为索引获取接受方的公钥
 - PGP 创建会话消息

PGP 接受方：

- 解密消息：
 - PGP 使用消息内密钥ID字段作为索引，获取私钥
 - PGP 提示用户输入密钥解密私钥
 - PGP 恢复会话密钥，解密消息
- 认证消息：
 - PGP 使用签名密钥的密钥id作为索引，获取公钥
 - PGP 恢复消息摘要
 - PGP 计算消息摘要并和传输版本比较认证

PGP压缩

- 使用ZIP算法
- 先签名后压缩

- 只需要存储原始报文和签名
- 易于更换压缩算法

PGP密钥环

PGP在每个结点提供一对数据结构

- 私钥环——存储该节点拥有的公开/私有密钥对
- 公钥环——存储该节点知道的其他所有用户的公开密钥

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·
T _i	KU _i mod 2 ⁶⁴	KU _i	E _{H(Pi)} [KR _i]	User i
·	·	·	·	·
·	·	·	·	·
·	·	·	·	·

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
T _i	KU _i mod 2 ⁶⁴	KU _i	trust_flag _i	User i	trust_flag _i		
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·

信任模型

- PGP 采用 web of trust 信任模型
- 无集中授权机构
- 个体签名他人公钥，存入公钥环
- PGP 计算公钥环内每个公钥的信任度
- 用户解释信任水平

公钥信任水平决定于：

- 密钥的签名数目
- 每个签名的信任度

信任水平要不断进行计算

- PGP中没有认证机构，而是由用户互相对对方的公钥进行数字签名
- 确认是否信任
 - 通过自己的数字签名
 - 通过自己完全信任的人的数字签名

- 通过自己有限信任的多个人的数字签名

Chapter8 BAN

基于推理结构性方法

问题：如何针对协议的安全性进行自动化分析？

- 选择一种形式化的描述方式进行建模
- 使用某种模型和工具进行分析运算
- 对于所得结果进行解释

“Dolev-Yao”攻击者模型

- DY模型，用于界定攻击者的能力
- 攻击者是一个不确定的过程
 - 可以读取任何消息，将其分解成部分并重新组装
 - 无法获得部分知识，无法进行统计测试
- “黑盒”密码学
 - 当且仅当攻击者知道正确的密钥时，他才能解密
 - 假设加密算法没有特殊属性
- 大多数用于安全分析的机械化形式方法都使用此模型的某个版本

基本术语

P、Q：主体

X：消息的语义

K：密钥

{X}K：用密钥K对X加密

P->Q:(X)：主体P发送消息X给Q

谓词	解释
bel(P, X)	主体P相信消息X, 在整个协议运行中都相信
sees(P, X)	P接收到X
said(P, X)	P发送过X, 发送不意味创造, 可能只是传播
cont(P, X)	P拥有对X正确与否非的判决权
fresh(X)	X是新鲜的, 在之前没有出现过, 例如Nonce
$\langle X \rangle y$	y是一个secret, 能由此判断发出X的人的身份
skey(P, K, Q)	K是P、Q共享密钥
goodkey(P, K, Q)	K是P、Q共享的良好密钥
pubkey(P, Q)	K是P的公开密钥
secret(P, X, Q)	X是P、Q共享秘密

- $P \xrightarrow{K} Q$: P and Q may use the *shared key* K to communicate. The key K is good, in that it will never be discovered by any principal except P or Q, or a principal trusted by either P or Q.
- $\xrightarrow{K} P$: P has K as a *public key*. The matching *secret key* (denoted K^{-1}) will never be discovered by any principal except P or a principal trusted by P.
- $P \xrightarrow{X} Q$: The formula X is a *secret* known only to P and Q, and possibly to principals trusted by them. Only P and Q may use X to prove their identities to one another. An example of a secret is a password.
- $\{X\}_K$: This represents the formula X encrypted under the key K. Formally, $\{X\}_K$ is a convenient abbreviation for an expression of the form $\{X\}_K$ from P. We make the realistic assumption that each principal is able to recognize and ignore his own messages; the originator of each message is mentioned for this purpose.

规则

规则解读

分子: if, 分母: then

逗号是"AND"

message-meaning rules

- 主体判断消息来源

$$\frac{P \text{ believes } Q \xleftrightarrow{K} P, P \text{ sees } \{X\}_K}{P \text{ believes } Q \text{ said } X}.$$

$$\frac{P \text{ believes } \xleftrightarrow{K} Q, P \text{ sees } \{X\}_{K^{-1}}}{P \text{ believes } Q \text{ said } X}.$$

$$\frac{P \text{ believes } Q \xrightarrow{Y} P, P \text{ sees } \langle X \rangle_Y}{P \text{ believes } Q \text{ said } X}$$

nonce-verification rule

- 消息新鲜性
- 如X是一个key，本轮第一次出现，那么P认为Q认可此key
- 这是诚实性假定

$$\frac{P \text{ believes } \text{fresh}(X), P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}$$

jurisdiction rule

- 仲裁规则
- P相信Q生成key的能力，P相信Q是认可此key的，那么P相信此key

$$\frac{P \text{ believes } Q \text{ controls } X, P \text{ believes } Q \text{ believes } X}{P \text{ believes } X}$$

接受规则

- 拆分

$$\begin{aligned} & \frac{P \text{ sees } (X, Y)}{P \text{ sees } X}, \quad \frac{P \text{ sees } \langle X \rangle_Y}{P \text{ sees } X}, \quad \frac{P \text{ believes } Q \xleftrightarrow{K} P, P \text{ sees } \{X\}_K}{P \text{ sees } X}, \\ & \frac{P \text{ believes } \xrightarrow{K} P, P \text{ sees } \{X\}_K}{P \text{ sees } X}, \quad \frac{P \text{ believes } \xleftrightarrow{K} Q, P \text{ sees } \{X\}_{K^{-1}}}{P \text{ sees } X}. \end{aligned}$$

新鲜性规则

$$\frac{P \text{ believes } \text{fresh}(X)}{P \text{ believes } \text{fresh}(X, Y)}$$

BAN 逻辑的假定

- 分布式环境的假定：消息可以任意获取
- 分布式特征：会话
- 相信，得到，发送
- 一个基本的语句反映一个公式的若干属性

关于协议运作的前提与假定

- Nonces
- 时戳
- 已有密钥
- 密钥创建能力

密钥分发与认证的逻辑目标

- 3、4. 确定对方是否愿意
- 任何一条欠缺，在BAN逻辑下都是不安全的

$$\begin{array}{ll} A \text{ believes } A \xleftrightarrow{K_{ab}} B & B \text{ believes } A \xleftrightarrow{K_{ab}} B \\ A \text{ believes } B \text{ believes } A \xleftrightarrow{K_{ab}} B & B \text{ believes } A \text{ believes } A \xleftrightarrow{K_{ab}} B \end{array}$$

Ban 逻辑分析的过程

- 理想化协议
- 初始状态及其假设
- 协议目标
- 逻辑推理

BAN 逻辑的缺陷

- 非标准的理想化协议过程：明文忽略问题
- 不合理的假设：主体诚实性问题
- 违规现象：重放攻击

Example

给一协议，描述BAN逻辑假定

Kerberos 协议分析

Message 1. $A \rightarrow S: A, B.$
Message 2. $S \rightarrow A: \{T_s, L, K_{ab}, B, \{T_s, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}.$
Message 3. $A \rightarrow B: \{T_s, L, K_{ab}, A\}_{K_{bs}}, \{A, T_a\}_{K_{ab}}.$
Message 4. $B \rightarrow A: \{T_a + 1\}_{K_{ab}}.$

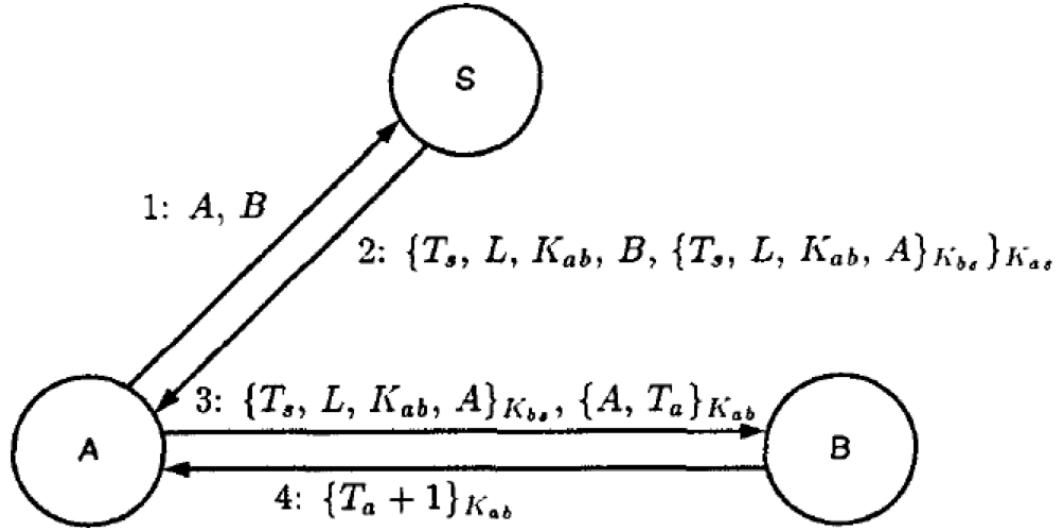


Fig. 1. The Kerberos Protocol.

理想化：忽略消息中的明文部分

Message 2. $S \rightarrow A: \{T_s, A \xleftrightarrow{K_{ab}} B, \{T_s, A \xleftrightarrow{K_{ab}} B\}_{K_{bs}}\}_{K_{as}}.$

Message 3. $A \rightarrow B: \{T_s, A \xleftrightarrow{K_{ab}} B\}_{K_{bs}}, \{T_a, A \xleftrightarrow{K_{ab}} B\}_{K_{ab}}$ from A.

Message 4. $B \rightarrow A: \{T_a, A \xleftrightarrow{K_{ab}} B\}_{K_{ab}}$ from B.

假定：

A believes $A \xleftrightarrow{K_{as}} S,$

S believes $A \xleftrightarrow{K_{as}} S,$

S believes $A \xleftrightarrow{K_{ab}} B,$

A believes (**S controls** $A \xleftrightarrow{K} B),$

A believes **fresh**(T_s),

B believes $B \xleftrightarrow{K_{bs}} S,$

S believes $B \xleftrightarrow{K_{bs}} S,$

B believes (**S controls** $A \xleftrightarrow{K} B),$

B believes **fresh**(T_s),

B believes **fresh**(T_a).

RPC 协议

- 反例：不安全

Message 1. $A \rightarrow B: A, \{N_a\}_{K_{ab}}.$

Message 2. $B \rightarrow A: \{N_a + 1, N_b\}_{K_{ab}}.$

Message 3. $A \rightarrow B: \{N_b + 1\}_{K_{ab}}.$

Message 4. $B \rightarrow A: \{K'_{ab}, N'_b\}_{K_{ab}}.$

理想化：

- BAN下 N_a 与 N_a+1 等同
- 给个协议，写一下BAN逻辑下的理想化过程

Message 1. $A \rightarrow B: \{N_a\}_{K_{ab}}.$
Message 2. $B \rightarrow A: \{N_a, N_b\}_{K_{ab}}.$
Message 3. $A \rightarrow B: \{N_b\}_{K_{ab}}.$
Message 4. $B \rightarrow A: \{A \xleftrightarrow{K'_{ab}} B, N'^b\}_{K_{ab}}.$

假定：

- 左二：A相信B创建key的能力
- 时戳都相信，Nonce只有创建者相信

A believes $A \xleftrightarrow{K_{ab}} B$	B believes $A \xleftrightarrow{K_{ab}} B$
A believes (B controls $A \xleftrightarrow{\kappa} B$)	B believes $A \xleftrightarrow{K'_{ab}} B$
A believes $\text{fresh}(N_a)$	B believes $\text{fresh}(N_b)$

A believes $\text{fresh}(N'_b)$	B believes $\text{fresh}(N'_b)$
--	--

多重会话攻击

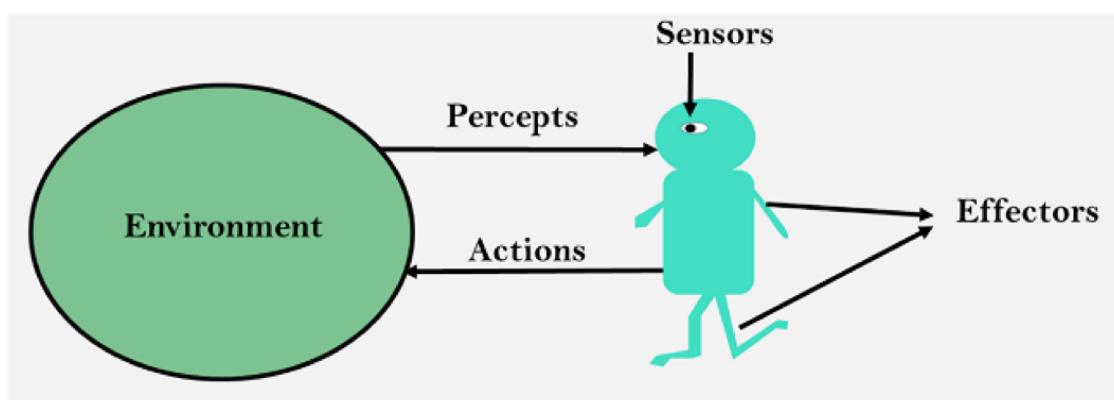
Message 1. $A \rightarrow B: A, N_a.$
Message 2. $B \rightarrow A: \{N_a, K'_{ab}\}_{K_{ab}}.$
Message 3. $A \rightarrow B: \{N_a\}_{K'_{ab}}.$
Message 4. $B \rightarrow A: N'_b.$

Chapter9 CSP

- 基于攻击结构性方法
- 自动售货机VMS的工作机理，使用并行处理的结构
- CSP：一种代数结构

Agents

- Agent有感知环境的能力，并且可以做出反馈，Agent与Agent之间进行交互
- Agent是智能体



CSP 概述

- CSP 是用于刻画分布式场景下agents交互的一种模型

CSP 常见符号

每一个agent可以刻画为一个进程P, P与P通过信道通信

αP : 进程P的字母表

αc : 信道c上传输的消息的集合

$a \rightarrow P$: a一般是一个event, 遇到a, 触发进程P (条件触发的过程, a then P)

$(a \rightarrow P \mid b \rightarrow Q)$: 逻辑的分支结构, 选a或者b, 触发P或者Q

$(x : A \rightarrow P(x))$: 选择A中的x, 作为进程P的输入进行处理

$\mu X : A \cdot F(X)$: 即 $X = F(X)$, $F(X)$ 为迭代表达式, A为X的定义域

$P \parallel Q$: P和Q并行, 代表着同步的动作

$P \parallel\parallel Q$: P和Q交织, 一个人做事, 另一人在等待, 没有同步的行为

$P \sqcap Q$

P or Q (non-deterministic)

$*P$: 无限循环

$b * P$: 带条件的循环, while b repeat P

$x := e$: 赋值

$b!e$: 在信道b上输出值e

$b?x$: 在信道b上输入值x

$l!e?x$

call of shared subroutine named l
with value parameter e and results to x

$P \text{ sat } S$: 进程P符合规约S, 模型检查的思想

tr

an arbitrary trace of the specified process

CSP 形式化定义

- 一门数学语言, 包含用于指定并发性、并行性、通信、选择等的主要结构
- 进程的演化基于events、actions
 - Visible actions \sum : 与其它进程的交互
 - Invisible action τ : 进程内部计算步骤

CSP 语法

- Inaction: Stop
 - 终止、死锁、无法执行任何内部或外部操作
- Input: $in ? x : A \rightarrow P(x)$

- 从信道in获得信息x, x属于type A, 把x交给进程P作为输入
- Output: $out ! m \rightarrow P(x)$
 - 向信道out输出m, 接下来执行P(x)
 - 例如发出消息后等待对方reply
- Recursion: $P(y_1, \dots, y_n) = Body(y_1, \dots, y_n)$

```
Copy = in ? x | out ! m → Copy
copy and paste的过程
```

- External choice: $P \sqcup Q$
 - Execute a choice between P and Q. Do not choose a process which cannot proceed

```
(a ? x → P(x)) [] (b ? x → Q(x))
Execute one and only one input action. If only one is available then choose that one. If both are available then choose arbitrarily. If none are available then block. The unchosen branch is discarded
```

- Internal choice: $P + Q$
 - Execute an arbitrary choice between P and Q. It is possible to choose a process which cannot proceed
 - 执行过程不受制于外界的输入, 系统内部的随机性
- Parallel operator with synchronization: $P || Q$
 - P 和 Q 并行进行, 并且必须同步所有常见操作

```
(c ? x → P(x)) || (c ! m → Q)
左边收, 右边发, m的内容赋值给了x
同步: 两个进程只有在它们的动作对应时才能进行
握手: 发送和接收是同时进行的
Communication: m is transmitted to the first process, which continues as P(m).
Broadcasting: c ! m is available for other parallel procs
```

可能考

```
((c?x → P(x)) [] (d?y → Q(y))) || (c!m → R)
```

在信道c上同步

左边server, 可能接受A的请求, 也可能接受B的请求

右边为client

- Parallel operator with synchronization and interleaving: $P ||_A Q$
 - $P ||| Q$, $P || \emptyset$: 永远没有交集
 - P 和 Q 必须仅对 A 中的公共操作进行同步
 - 他们在所有不在 A 中的动作上交织
 - Teacher || 上课Students

$(c ? x \rightarrow P(x)) \parallel \{c\} ((c ! m \rightarrow Q) [] (d ! n \rightarrow R))$

这两个进程可以在通道 c 上同步操作，或者第二个进程可以在 d 上执行操作。在第二种情况下，第一个进程将

保持阻塞状态，直到第二个进程决定对 c 执行（如果有的话）输出操作

信道 c 上同步，其它交织

X2 A foolish customer wants a large biscuit, so he puts his coin in the vending machine VMC . He does not notice whether he has inserted a large coin or a small one; nevertheless, he is determined on a large biscuit

$$\begin{aligned}FOOLCUST = & (in2p \rightarrow large \rightarrow FOOLCUST \\& | in1p \rightarrow large \rightarrow FOOLCUST)\end{aligned}$$

Unfortunately, the vending machine is not prepared to yield a large biscuit for only a small coin

$$(FOOLCUST \parallel VMC) = \mu X \bullet (in2p \rightarrow large \rightarrow X | in1p \rightarrow STOP)$$

The $STOP$ that supervenes after the first $in1p$ is known as *deadlock*. Although each component process is prepared to engage in some further action, these actions are different; since the processes cannot agree on what the next action shall be, nothing further can happen. \square

CSP 分析

NS公钥协议：

- Message 1. $A \rightarrow B : A.B.\{N_a.A\}_{PK(B)}$
- Message 2. $B \rightarrow A : B.A.\{N_a.N_b\}_{PK(A)}$
- Message 3. $A \rightarrow B : A.B.\{N_b\}_{PK(B)}.$

消息刻画

$$\begin{aligned}MSG1 \triangleq & \{Msg1.a.b.Encrypt.k.n_a.a' | \\& a \in Initiator, a' \in Initiator, b \in Responder, \\& k \in Key, n_a \in Nonce\},\end{aligned}$$

$$\begin{aligned}MSG2 \triangleq & \{Msg2.b.a.Encrypt.k.n_a.n_b | \\& a \in Initiator, b \in Responder, \\& k \in Key, n_a \in Nonce, n_b \in Nonce\},\end{aligned}$$

$$\begin{aligned}MSG3 \triangleq & \{Msg3.a.b.Encrypt.k.n_b | \\& a \in Initiator, b \in Responder, k \in Key, n_b \in Nonce\}, \\MSG \triangleq & MSG1 \cup MSG2 \cup MSG3.\end{aligned}$$

- $Msg1.a.b$: A发给B的消息 $Msg1$

使用CSP进行安全协议建模

- 系统的刻画：消息，信道，agent，攻击者
- 规范的刻画：响应者，发起者
- 运行检查：trace