# 第九章 电子邮件安全学习笔记

## 一、 电子邮件安全概述

### 1. 为什么需要安全电子邮件？

* **E-mail是互联网上最广泛的应用**，但其基础协议（如SMTP）在设计之初并未过多考虑安全性。
* 默认的电子邮件通信是**明文**的，存在被窃听、篡改、伪造的风险。
* **核心解决的问题**：身份认证和保密性。

### 2. 安全电子邮件的目标 (Email安全增强)

* **保密性 (Confidentiality)**：防止邮件内容泄露，只有授权的接收者才能阅读。
* **身份认证 (Authentication)**：确认邮件确实是由声称的发送者发出的。
* **消息完整性 (Message Integrity)**：确保邮件在传输过程中没有被修改。
* **源的不可否-认性 (Non-repudiation of origin)**：使发送者无法否认自己发送过该邮件。

## 二、 PGP (Pretty Good Privacy)

### 1. PGP 简介

* 由 Phil Zimmermann 于1991年开发，提供电子邮件和文件存储的保密与认证服务。
* 已成为互联网标准（RFC2440, RFC3156）。
* **特点**：
  + 选用当时最优秀的、公认安全的加密算法（如RSA, SHA, 3DES等）。
  + 集成为一个通用应用程序，跨平台、跨系统。
  + 源码公开，有免费版本和商业版本。

### 2. PGP 符号约定

| **符号** | **意义** |
| --- | --- |
| Ks | Session Key (会话密钥) |
| KRa | 用户A的私钥 (Private Key of A) |
| KUa | 用户A的公钥 (Public Key of A) |
| EP | 公钥加密 (Public Key Encryption) |
| DP | 公钥解密 (Public Key Decryption) |
| EC | 对称加密 (Symmetric Encryption) |
| DC | 对称解密 (Symmetric Decryption) |
| H | Hash函数 |
| ‘||’ | 表示连接 |
| Z | 用ZIP算法压缩 |
| R64 | 转换为Base 64的ASCII格式 |

### 3. PGP 提供的服务

PGP通过组合使用多种密码学算法来实现全面的安全服务。

* **数字签名 (Digital Signature)**：
  + **目的**：提供身份认证、消息完整性和不可否认性。
  + **算法**：DSS/SHA 或 RSA/SHA。
  + **签名过程**：
    1. 使用 **SHA-1** 算法计算原始消息 M 的160位哈希值 H = H(M)。
    2. 使用**发送方的私钥 (KR\_a)** 对哈希值 H 进行加密（签名），生成签名 S = EP(KR\_a, H)。
    3. 将签名 S 附加到原始消息 M 后面形成 M || S 一起发送。
  + **验证过程**：
    1. 接收方使用**发送方的公钥 (KU\_a)** 解密签名 S，得到原始哈希值 H = DP(KU\_a, S)。
    2. 接收方对收到的消息 M 再次使用 SHA-1 计算新的哈希值 H' = H(M)。
    3. 比较 H 和 H'，如果两者匹配，则认证通过。
  + **流程图**：  
    ### 数字签名过程 (发件方) ###  
      
    [原始消息 M]  
     |  
     v  
    +-----------------+  
    | SHA-1 哈希算法 |  
    +-----------------+  
     |  
     v  
    [消息摘要 H = H(M)] <--+  
     | (加密)  
    +--------------------+  
    | 使用发件方私钥 KR\_a |  
    +--------------------+  
     |  
     v  
    [数字签名 S = EP(KR\_a, H)]  
     |  
     |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     | |  
     v v  
    [原始消息 M] [数字签名 S]  
     |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
     |  
     v  
     [发送 M || S]  
      
      
    ### 验证过程 (收件方) ###  
      
     [收到 M || S]  
     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     | |  
     v v  
    [原始消息 M] [数字签名 S]  
     | |  
     v | (解密)  
    +-----------------+ |  
    | SHA-1 哈希算法 | |  
    +-----------------+ v  
     | +-------------------+  
     v | 使用发件方公钥 KU\_a |  
    [新的消息摘要 H' = H(M)] +-------------------+  
     | |  
     | v  
     | [原始消息摘要 H = DP(KU\_a, S)]  
     |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
     |  
     v  
     +--------------+  
     | 比较 H' 和 H |  
     +--------------+  
     |  
     v  
     <是否相等?>  
     / \  
     是 否  
     / \  
     v v  
     [验证成功] [验证失败]
* **消息加密 (Message Encryption)**：
  + **目的**：提供保密性。
  + **方式**：采用**对称加密**和**公钥加密**结合的混合加密模式。
  + **算法**：对称加密用 CAST-128、IDEA 或 3DES；公钥加密用 Diffie-Hellman 或 RSA。
  + **加密过程**：
    1. 发送方生成一个**一次性**的随机**会话密钥 (Ks)**。
    2. 使用这个**会話密钥 (Ks)** 对消息 M 进行对称加密。
    3. 使用**接收方的公钥 (KU\_b)** 对**会话密钥 (Ks)** 进行加密。
    4. 将加密后的消息和加密后的会话密钥一起发送。
  + **解密过程**：
    1. 接收方使用自己的**私钥 (KR\_b)** 解密被加密的会话密钥，得到 Ks。
    2. 使用 Ks 对加密的消息进行对称解密，恢复出原始消息 M。
  + **流程图**：  
    ### 加密过程 (发件方) ###  
      
    [原始消息 M]--+  
     | (对称加密)  
    [会话密钥 Ks] <-+  
     |  
     | (公钥加密)  
     v  
    +---------------------+  
    | 使用收件方公钥 KU\_b |  
    +---------------------+  
     |  
     v  
    [加密的会话密钥 K\_enc = EP(KU\_b, Ks)]----+  
     |  
    [加密的消息 C = EC(Ks, M)]----------+-----> [通过网络发送]  
      
      
    ### 解密过程 (收件方) ###  
      
     [收到数据]  
     |  
     \_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     | |  
     v v  
    [加密的会话密钥 K\_enc] [加密的消息 C]  
     | |  
     v (公钥解密) |  
    +---------------------+ |  
    | 使用收件方私钥 KR\_b | |  
    +---------------------+ |  
     | |  
     v |  
    [恢复的会话密钥 Ks = DP(KR\_b, K\_enc)]--+  
     | (对称解密)  
     v  
     [原始消息 M = DC(Ks, C)]
* **保密与认证并用**：
  + **原则**：先签名，后加密。
  + **过程**：发送方首先用自己的私钥对消息进行签名，然后将“消息+签名”的组合体用一次性的会话密钥进行加密，最后再用接收方的公钥加密该会话密钥。
* **数据压缩 (Compression)**：
  + **算法**：使用 ZIP 算法。
  + **时机**：在**签名之后、加密之前**进行压缩。
  + **好处**：
    - 节省存储和传输开销。
    - 压缩后的报文更难进行密码分析。
* **邮件兼容 (Base64 转换)**：
  + **问题**：电子邮件系统设计为传输文本（ASCII字符），而加密后的数据是二进制格式。
  + **解决方案**：使用 **Base64 (Radix-64)** 编码，将二进制数据流转换为可打印的ASCII字符序列。
  + **转换规则**：将每 24 bits (3字节) 的二进制数据转换为 32 bits (4个ASCII字符)，数据量会增大约33%。
* **数据分段与重组 (Segmentation and Reassembly)**：
  + **问题**: 邮件系统通常对单封邮件的大小有限制，例如，最大长度常被限制在**50000字节**左右。
  + **解决方案**:
    - 当消息超过这个限制时，PGP会自动将其**分段**（分割成多个小片段）。
    - 每一个片段会作为一封独立的邮件发送。
    - 接收方的PGP客户端会自动识别这些片段，并将它们**重组**恢复成原始的完整消息。
  + **签名效率**: 整个消息的数字签名只需要生成一次，并包含在**第一个分段**中即可，无需对每个片段都签名。

### 4. 综合实例：Alice 向 Bob 发送一封安全邮件

假设 Alice 想给 Bob 发送一封同时**加密**和**签名**的邮件，内容为 "Hi Bob, let's meet at 10am."。

**发送方 (Alice) の操作流程：**

1. **签名 (Authentication & Integrity)**
   * **流程**: M -> H(M) -> S = EP(KR\_Alice, H(M))
   * 首先计算消息 M 的哈希值 H，然后用 Alice 的私钥 KR\_Alice 对 H 签名，得到签名 S。
2. **压缩 (Compression)**
   * **流程**: M || S -> Z(M || S)
   * 将原始消息 M 和签名 S 连接起来，然后用 ZIP 算法进行压缩。
3. **加密 (Confidentiality)**
   * **流程**:
     + Ks -> K\_enc = EP(KU\_Bob, Ks)
     + Z(M || S) -> C = EC(Ks, Z(M || S))
   * 生成一次性会话密钥 Ks。用 Bob 的公钥 KU\_Bob 加密 Ks。用 Ks 对压缩数据进行对称加密。
4. **编码 (Email Compatibility)**
   * **流程**: C || K\_enc -> R64(C || K\_enc)
   * 将加密后的消息 C 和加密后的密钥 K\_enc 连接，然后用 Base64 编码成ASCII文本。
5. **发送**
   * 将最终的 Base64 文本作为邮件正文发送。

**接收方 (Bob) の操作流程：**

1. **解码 (Decoding)**
   * **流程**: R64(...) -> C || K\_enc
   * 将收到的 Base64 文本解码回二进制数据，分离出加密的消息 C 和加密的密钥 K\_enc。
2. **解密 (Decryption)**
   * **流程**:
     + K\_enc -> Ks = DP(KR\_Bob, K\_enc)
     + C -> Z(M || S) = DC(Ks, C)
   * 用 Bob 自己的私钥 KR\_Bob 解密 K\_enc 得到会话密钥 Ks。然后用 Ks 解密消息 C 得到压缩数据Z(M || S) 。
3. **解压缩 (Decompression)**
   * **流程**: Z(M || S) -> M || S
   * 将压缩数据解压，分离出原始消息 M 和签名 S。
4. **验证 (Verification)**
   * **流程**:
     + M -> H'(M)
     + S -> H = DP(KU\_Alice, S)
     + 比较 H' 与 H
   * 对收到的消息 M 计算新的哈希 H'。用 Alice 的公钥 KU\_Alice 解密签名 S 得到原始哈希 H。比较 H' 和 H 是否相等来验证邮件。

### 5. PGP 安全性讨论

* **没有绝对的安全**：正如PGP的作者所说，“没有哪个数据安全系统是牢不可破的”。
* **软件自身漏洞**：PGP作为一种软件，可能存在自身的编程漏洞，历史上也出现过需要修复补丁的情况。
* **依赖算法强度**：PGP的安全性建立在其使用的加密算法（如3DES, RSA, SHA-1）之上。这些算法都是密码学研究和攻击的目标，一旦其中任何一个算法被攻破，PGP的安全性也将受到严重威胁。

## 三、 S/MIME (Secure/Multipurpose Internet Mail Extensions)

### 1. S/MIME 简介

* 是对标准MIME电子邮件格式的安全扩展。
* 技术主要源于 RSA Data Security 公司。

### 2. S/MIME 与 PGP 的核心区别：信任的建立方式

S/MIME 和 PGP 的根本区别在于它们如何解决密码学中的核心问题：“我如何确定这个公钥真的属于 Bob？” 它们采用了两种截然不同的信任模型。

* **S/MIME：中心化的“权威认证”模式 (PKI)**
  + **核心思想**：信任由一个集中的、权威的第三方机构来背书。
  + **工作机制**：S/MIME 依赖于一个严格的层级结构，即**公钥基础设施 (Public Key Infrastructure, PKI)**。在这个体系中，有广受信任的**证书颁发机构 (Certificate Authority, CA)**，如 Verisign, DigiCert 等。
  + **数字证书 (Digital Certificate)**:
    - **定义**: 是一种标准化的数字文档 (X.509标准)，它将一个实体（如个人、公司）的**身份信息** (User\_Info) 与其**公钥** (KU\_user) 绑定在一起，并由CA进行数字签名以保证其权威性。它就像一张权威的“数字身份证”。
    - **证书的生成**:
      1. **申请**: 用户向CA提交自己的身份信息和公钥。
      2. **验证**: CA通过严格的流程核实用户的身份。
      3. **签名**: CA将用户信息和公钥打包，计算其哈希值，然后用**CA自己的私钥 (KR\_ca)** 对该哈希值进行加密（签名）。这个签名是证书的核心部分，证明了证书内容的真实性。

Certificate\_Signature = EP(KR\_ca, H(User\_Info || KU\_user))

* + - **证书代替了什么？** 它代替了用户之间需要**手动、相互验证公钥的繁琐过程**。
  + **信任链 (Chain of Trust)**:
    - 用户的邮件客户端或操作系统预置了对顶级“根CA”的信任。当收到一份证书时，客户端会使用相应的CA公钥来验证证书上的签名。只要签名有效，并且该CA是受信任的，那么证书内的公钥就会被自动信任。
  + **X.509 认证过程 (Authentication Processes)**
    - 一旦各方通过证书获得了对方可信的公钥，X.509标准定义了三种可选的认证过程：
    - **单向认证 (One-way authentication)**:
      * **流程**: A向B出示自己的证书。B验证该证书即可确认A的身份。
      * **应用**: 这是最常见的模式，例如浏览器访问HTTPS网站时，网站服务器向浏览器证明自己的身份。
    - **双向认证 (Two-way authentication)**:
      * **流程**: A向B出示证书，B也向A出示证书。双方互相验证对方的身份。
      * **应用**: 用于高安全级别的场景，如企业VPN或网上银行，客户端也需要持有证书来向服务器证明自己的身份。
    - **三向认证 (Three-way authentication)**:
      * **流程**: 在双向认证的基础上，增加第三次通信，通常包含由接收方生成的随机数（Nonce），以防止恶意的重放攻击。
      * **应用**: 用于对实时性要求极高的安全通信，确保交换的信息是“新鲜”的。
  + **简要流程**：
    - **阶段一：证书颁发 (一次性)**
      * Alice -> CA: 申请 {ID\_Alice, KU\_Alice}
      * CA -> Alice: 证书\_A = {ID\_Alice, KU\_Alice} || EP(KR\_CA, H(ID\_Alice || KU\_Alice))
    - **阶段二：邮件签名与验证 (重复使用)**
      * Alice -> Bob: {邮件M, 签名S=EP(KR\_Alice, H(M)), 证书\_A}
      * Bob验证: (使用KU\_CA验证证书\_A -> 信任KU\_Alice) -> (使用KU\_Alice验证S -> 信任邮件M)

**总结**

| **特性** | **S/MIME (权威模式)** | **PGP (社交模式)** |
| --- | --- | --- |
| **信任模型** | 等级森严的 **PKI** | 分布式的**信任网** (Web of Trust) |
| **信任来源** | 权威的 **CA 机构** | 用户间的**相互签名** |
| **凭证** | **X.509 数字证书** (数字身份证) | **公钥本身** + 他人的签名 |
| **应用场景** | **企业**、**政府**、商业环境 (需要统一管理和明确责任) | **个人**、技术社区、记者、活动家 (强调去中心化和个人隐私) |

* **现状**：由于其集中管理和与操作系统/主流软件的深度集成，S/MIME 在商业环境中更为普遍。许多主流邮件客户端（如 Microsoft Outlook）都内置了对 S/MIME 的支持。

## 四、 安全电子邮件的配置步骤 (以S/MIME为例)

这是一个端到端的安全邮件配置流程：

1. **获取数字标识 (Digital ID)**
   * 数字标识本质上是一个由 **证书颁-发机构 (CA)** 签发的 **X.509数字证书**，它包含了你的公钥和身份信息。
   * 可以向商业CA（如Verisign）购买或从一些提供免费试用服务的CA获取（如讲义中提到的广州市数字证书管理中心）。
2. **将数字标识添加到邮件账号中**
   * 将获取到的数字证书（通常是一个文件）导入到你的邮件客户端（如Outlook Express）的账户安全设置中。
3. **交换公钥**
   * **导出你的数字标识**：将你的证书（只包含公 "钥部分）导出一个文件，或通过发送一封**已签名的邮件**给你的联系人。对方收到签名邮件后，其邮件客户端可以自动提取你的公钥和证书。
   * **将联系人的数字标识添加到通讯簿**：收到对方的签名邮件后，将其证书（公钥）添加到你的地址簿或联系人条目中。
4. **发送安全邮件**
   * **发送签名邮件**：写完邮件后，点击“签名”按钮。这会用你的私钥对邮件内容进行签名。
   * **发送加密邮件**：写完邮件后，点击“加密”按钮。这需要你已经拥有对方的公钥（证书），邮件客户端会用对方的公钥加密邮件。
   * **同时签名和加密**：可以同时选择签名和加密，提供最全面的安全保障。