第七章学习笔记：消息认证与Hash算法

第一部分：消息认证导论 (Message Authentication)

1. 为什么需要消息认证？

信息安全旨在保护信息资源免受各种威胁，其核心需求包括：

- 保密性 **(Confidentiality)**: 防止信息被未经授权的个体获取。
- 完整性 **(Integrity)**: 确保数据在传输或存储过程中未被篡改或损坏。
- 可用性 **(Availability)**: 确保授权用户在需要时可以访问信息及相关资产。
- 认证 **(Authenticity)**: 验证通信实体（用户、系统、信息等）身份的真实性。
- 不可否认性 **(Non-repudiation)**: 防止通信的发送方或接收方否认其行为。

当你通过网络进行一次在线交易时，系统需要确保两件关键的事情：

- **"**这条交易指令真的是由合法的账户持有人发出的吗？**"——** 这就是认证 **(Authenticity)**，确认消息来源是真实的，而不是攻击者伪造的。
- **"**交易金额、收款账户等信息在传输过程中被修改过吗？**"——** 这就是完整性 **(Integrity)**，确保数据在从客户端到服务器的过程中没有被篡改。

消息认证就是为了解决这两个核心问题。在数字世界里，所有的数据都可能被拦截、伪造或修改，因此我们需要一种机制来验证信息的"真伪"和"完整"。

2. 常见的网络攻击与应对策略

攻击类型	技术应用实例	技术世界的对策
泄密（通信内容被窃取）	未加密的电子邮件或即时消息在公共Wi-Fi下被抓包窃听。	加密 **(Encryption)**
伪装 （攻击者伪装为合法用户发送消息）	收到一封伪装成银行官方的钓鱼邮件，要求你输入密码。	消息认证 **(Message Authentication)**
内容篡改（消息内容被修改）	在一次在线支付请求中， 攻击者通过中间人攻击将支付金额从10元修改为1000元。	消息认证 **(Message Authentication)**
否认（发送方/接收方否认发送/接收过消息）	用户完成了一次在线购买后，向平台声称 "我从未授权过这笔交易"。	数字签名 **(Digital Signature)**

3. 什么是消息认证？

消息认证是一个验证消息来源真实性和内容完整性的过程。其目标是确保：

- 接收者可以证实消息的合法性、真实性和完整性。
- 通信双方不能抵赖其行为。
- 除合法发送者外，任何第三方都无法伪造消息

第二部分：消息认证的实现方式

一、基于对称密钥的认证

对称密钥体系中，通信双方共享同一个密钥。这种体系下的认证机制无法提供不可否认性，因为任何一方都能生成有效的认证信息。

1.1 利用对称加密实现隐式认证

核心原理

- 通信双方共享密钥 K。
- 发送方用 K 加密消息 M 得到密文 C，接收方收到 C 后用 K 解密。
- 若解密成功且得到符合协议格式的明文，则隐式地验证了发送方身份——因为只有持有 K 的通信方才能生成可被正确解密的密文。

认证逻辑

这种认证是"隐式"的，依赖于以下推理链：

1. 只有 Alice 和 Bob 知道密钥 K
2. Bob 收到的密文能用 K 正确解密
3. 因此该密文必然来自 Alice

安全属性

属性	支持	说明
机密性	√	密文只有持有密钥的一方能解密
认证	√	能解密即证明对方持有密钥
完整性	√	篡改会导致解密失败或明文无效
不可否认性	✗	Alice 和 Bob 都能生成有效密文

应用实例

VPN 预共享密钥(PSK)认证——企业员工与 VPN 服务器预先配置相同的 PSK，客户端用 PSK 加密连接请求，服务器解密成功则确认为合法员工。

局限性分析

- 争议场景下的困境：若 Bob 声称收到 Alice 的某条消息，Alice 可以否认，因为 Bob 也能用共享密钥伪造该消息。第三方(如仲裁机构)无法判断消息的真实来源
- 密钥泄露的连锁反应：一旦密钥泄露，攻击者既能解密历史消息，也能伪装成任意一方
- 不适用场景：电子合同、数字证书、软件发布等需要法律效力的场景

1.2 消息认证码 (MAC - Message Authentication Code)

核心原理

MAC 是一种带密钥的哈希函数。将消息 M 和共享密钥 K 作为输入，输出固定长度的认证标签 T = MAC(K， M)。发送方将 (M， T) 一起发送，接收方重新计算 MAC(K， M) 并与收到的 T 比对。

认证逻辑

1. 只有持有密钥 K 的一方才能计算出正确的 MAC 值
2. 消息的任何修改都会导致 MAC 值改变
3. 攻击者即使截获 (M， T)，也无法在不知道 K 的情况下为篡改后的消息生成有效的 MAC

与对称加密认证的区别

维度	对称加密认证	MAC
主要目的	机密性(认证是副产品)	认证和完整性
计算开销	高(需加解密全部数据)	低(仅计算哈希)
消息可见性	密文，不可读	明文可读
典型场景	需要保密的通信	仅需验证真实性的场景

常用 MAC 算法

1. **HMAC (Hash-based MAC)**
   - 构造：HMAC(K， M) = H((K ⊕ opad) || H((K ⊕ ipad) || M))
   - 其中 H 是哈希函数(如 SHA-256)，opad 和 ipad 是固定填充值
   - 优势：基于哈希函数，安全性依赖于哈希函数的抗碰撞性
   - 常见变体：HMAC-SHA256、HMAC-SHA3
2. **CMAC (Cipher-based MAC)**
   - 构造：基于分组密码(如 AES)的 CBC 模式
   - 优势：在已部署 AES 硬件的环境中效率高
   - 典型实现：AES-CMAC
3. **GMAC (Galois MAC)**
   - 构造：AES-GCM 模式的认证部分
   - 特点：可并行计算，性能优异
   - 应用：常与 GCM 模式结合提供认证加密

应用实例

API 请求签名——Web 应用调用第三方支付 API 时，客户端用共享密钥对请求内容计算 HMAC 签名，服务器用相同密钥验证签名以确认请求来源和完整性。

防重放攻击增强

- 加入时间戳：拒绝超过时间窗口的请求
- 加入 nonce：服务器维护已使用的 nonce 列表，拒绝重复请求

局限性

- 无不可否认性：与对称加密相同，双方都能生成有效 MAC
- 密钥分发问题：如何安全地在通信双方之间共享密钥
- 密钥管理复杂度：n 个用户两两通信需要 n(n-1)/2 个密钥

二、基于非对称密钥的认证

非对称密钥体系中，每个实体拥有一对密钥：公钥(可公开)和私钥(必须保密)。这种体系能提供不可否认性，因为只有私钥持有者能生成有效的签名。

2.1 数字签名

核心原理

发送方 Alice 用自己的私钥 SK_A 对消息 M(或其哈希值)进行签名运算，得到签名 σ = Sign(SK_A， M)。
接收方 Bob 用 Alice 的公钥 PK_A 验证签名，Verify(PK_A， M， σ) 返回 true 或 false。

签名流程

- 签名生成**(Alice)**：$h = H(M)$，$\sigma = Sign(SK_A, h)$，发送 (M, σ)
- 签名验证**(Bob)**：收到 (M, σ)，计算 $h' = H(M)$，验证 $Verify(PK_A, h', \sigma) = true$

为什么对哈希值签名而非直接签名消息

1. 性能考虑：签名运算(如 RSA)的计算开销远大于哈希运算。对大文件直接签名不可行
2. 固定长度：哈希值长度固定(如 SHA-256 输出 256 位)，签名算法对输入长度有限制
3. 安全性：现代签名方案(如 RSA-PSS)的安全性证明基于对哈希值签名的模型

安全属性

属性	支持	说明
机密性	✗	签名不加密消息，任何人都能读取
认证性	√	验证通过即证明消息来自私钥持有者
完整性	√	消息的任何修改都会导致验证失败
不可否认性	√	只有私钥持有者能生成有效签名，无法抵赖

常用签名算法

1. **RSA** 签名
    - 原理：基于大整数分解困难问题
    - 签名：σ = m^d mod n (d 是私钥)
    - 验证：m = σ^e mod n (e 是公钥)
    - 现代标准：RSA-PSS(Probabilistic Signature Scheme)，使用随机填充增强安全性
    - 密钥长度：至少 2048 位，推荐 3072 位
2. **ECDSA (Elliptic Curve DSA)**
    - 原理：基于椭圆曲线离散对数问题
    - 优势：相同安全强度下密钥更短(256 位 ECDSA ≈ 3072 位 RSA)
    - 应用：TLS 证书、比特币交易签名
    - 注意：需要高质量随机数生成器，否则可能泄露私钥
3. **EdDSA (Edwards-curve DSA)**
    - 原理：基于 Twisted Edwards 曲线
    - 优势：性能优于 ECDSA，实现更不容易出错
    - 典型实现：Ed25519(使用 Curve25519)
    - 应用：SSH 密钥、Signal 消息协议

应用实例

软件包签名验证——Linux 发行版发布软件包时，发行商用私钥对软件包哈希值签名；用户下载后用发行商公钥验证签名，确保软件来源可信且未被篡改。

与 MAC 的对比

维度	MAC	数字签名
密钥类型	对称(共享密钥)	非对称(公私钥对)
验证方	只有持有密钥的通信方	任何持有公钥的人
不可否认性	无	有
计算开销	低	高
典型场景	两方通信	一对多发布、法律文书

2.2 混合认证方案

设计动机

- 对称加密：速度快，但密钥分发困难，无不可否认性
- 非对称加密：密钥管理简单，有不可否认性，但速度慢(比对称加密慢 100-1000 倍)
- 混合方案：结合两者优势——用对称加密保护数据，用非对称密码保护密钥和提供认证

典型流程

- 发送方**(Alice)**：
  - 生成随机会话密钥 $K_{session}$
  - 对称加密：$C = Encrypt_{sym}(K_{session}, M)$
  - 签名：$\sigma = Sign(SK_A, H(M))$
  - 加密会话密钥：$K_{encrypted} = Encrypt_{asym}(PK_B, K_{session})$
  - 发送：$(C, K_{encrypted}, \sigma)$
- 接收方**(Bob)**：
  - 解密会话密钥：$K_{session} = Decrypt_{asym}(SK_B, K_{encrypted})$
  - 解密消息：$M = Decrypt_{sym}(K_{session}, C)$
  - 验证签名：$Verify(PK_A, H(M), \sigma) = true$

安全属性

属性	支持	说明
机密性	√	只有 Bob 能解密会话密钥，进而解密消息
认证性	√	数字签名验证消息来源
完整性	√	签名验证保证消息未被篡改
不可否认性	√	只有 Alice 能用私钥生成签名

实际应用

1. **TLS/SSL (Transport Layer Security)**
   - 握手阶段：服务器用证书证明身份(包含公钥)，客户端验证证书
   - 密钥协商：通过 ECDHE 等协议协商会话密钥
   - 数据传输：用协商的对称密钥加密 HTTP 流量
   - 认证加密：使用 AES-GCM 等模式同时提供机密性和认证性
2. **PGP/GPG (Pretty Good Privacy)**
   - 邮件加密：用接收方公钥加密对称密钥，用对称密钥加密邮件正文

- 邮件签名：用发送方私钥对邮件签名
- 信任网络：通过密钥签名建立信任链

3. **S/MIME (Secure/Multipurpose Internet Mail Extensions)**
   - 基于 X.509 证书体系
   - 支持邮件加密和签名
   - 广泛集成于企业邮件系统(Outlook、Thunderbird)

现代发展：认证加密(Authenticated Encryption)

传统混合方案分别处理"加密"和"认证"，容易出现实现漏洞(如先加密后认证 vs 先认证后加密的争议)。现代密码学发展出一体化方案：

- **AES-GCM (Galois/Counter Mode)**：同时提供加密和认证，单次计算完成
- **ChaCha20-Poly1305**：性能优异的流密码 + MAC 组合
- 优势：避免组合使用时的安全陷阱，性能更优

三、基础密码学原语

密码学原语是构建上层安全协议的基础构件。它们本身不能直接提供完整的安全保证，但是实现认证机制不可或缺的组件。

3.1 哈希函数

定义与特性

哈希函数 H 将任意长度输入映射为固定长度输出：H: {0，1}* → {0，1}^n

核心安全特性

1. 单向性 **(Preimage Resistance)**
   - 给定哈希值 h，计算上不可行找到消息 m 使得 H(m) = h
   - 实际含义：无法通过哈希值反推原始数据
2. 弱抗碰撞性 **(Second Preimage Resistance)**
   - 给定消息 m1，计算上不可行找到 m2 ≠ m1 使得 H(m1) = H(m2)
   - 实际含义：攻击者无法为指定消息找到替代消息
3. 强抗碰撞性 **(Collision Resistance)**
   - 计算上不可行找到任意两个不同消息 m1 ≠ m2 使得 H(m1) = H(m2)
   - 实际含义：攻击者无法找到任何哈希碰撞对
   - 注意：生日攻击表明，对于 n 位哈希，碰撞复杂度约为 O(2^(n/2))

在认证中的作用

1. **MAC** 的基础：HMAC 直接基于哈希函数构造
2. 数字签名的优化：对消息哈希签名而非直接签名消息
3. 完整性校验：文件下载时用哈希值验证完整性

常用哈希算法

算法	输出长度	安全状态	备注
MD5	128 位	已破解	存在实用碰撞攻击，不应用于安全场景
SHA-1	160 位	已弱化	2017 年 Google 演示碰撞攻击，逐步淘汰
SHA-256	256 位	安全	SHA-2 家族，广泛应用
SHA-3	可变	安全	基于 Keccak，设计思路与 SHA-2 不同
BLAKE3	256 位	安全	高性能，支持并行计算

单独使用的局限性

场景：从网站下载文件

不安全的做法：

网站提供：

- file.zip (下载文件)
- file.zip.sha256 (哈希值)

用户流程：

1. 下载 file.zip
2. 下载 file.zip.sha256
3. 本地计算哈希并比对

问题：中间人攻击

- 攻击者截获下载请求
- 替换 file.zip 为恶意文件 malicious.zip
- 计算 malicious.zip 的哈希值
- 同时替换 file.zip.sha256
- 用户验证通过，但实际安装了恶意软件

安全增强

正确做法：对哈希值进行数字签名
网站提供：

- file.zip (下载文件)
- file.zip.sha256 (哈希值)
- file.zip.sha256.sig (用网站私钥对哈希值的签名)

用户流程：

1. 下载三个文件
2. 用网站公钥验证签名(公钥预装在系统或浏览器中)
3. 若签名有效，再比对哈希值

安全性：攻击者无法伪造签名，即使替换文件和哈希值

3.2 加密算法

加密算法是实现机密性的基础，同时也是对称加密认证和混合方案的核心组件。

对称加密算法

1. **AES (Advanced Encryption Standard)**

- 分组长度：128 位
- 密钥长度：128、192、256 位
- 模式：ECB、CBC、CTR、GCM 等
- 地位：NIST 标准，硬件加速支持广泛

2. **ChaCha20**
   - 类型：流密码
   - 优势：软件实现性能优异，抗时序攻击
   - 应用：TLS 1.3(ChaCha20-Poly1305)、WireGuard VPN

非对称加密算法

1. **RSA (Rivest-Shamir-Adleman)**
   - 原理：大整数分解困难性
   - 用途：密钥交换、数字签名
   - 密钥长度：2048 位(当前最低安全要求)、3072 位、4096 位
2. **ECC (Elliptic Curve Cryptography)**
   - 原理：椭圆曲线离散对数问题
   - 优势：更短的密钥实现相同安全强度
   - 常用曲线：P-256、Curve25519
   - 应用：ECDH(密钥交换)、ECDSA(签名)

加密算法在认证中的角色

- 直接认证：对称加密的隐式认证
- 密钥保护：混合方案中保护会话密钥
- 与认证机制结合：认证加密模式(GCM、CCM)

认证方式综合对比

方式	机密性	认证性	完整性	不可否认性	计算开销	密钥管理	典型应用
对称加密认证	√	√	√	✗	低	复杂	VPN、内部通信
MAC	✗	√	√	✗	低	复杂	API 签名、消息验证
数字签名	✗	√	√	√	高	简单	软件发布、证书
混合方案	√	√	√	√	中	简单	TLS、PGP、S/MIME
认证加密(AEAD)	√	√	√	✗	低	复杂	TLS 1.3、QUIC

说明：

- 机密性：防止未授权方读取消息内容
- 认证性：验证消息来源
- 完整性：检测消息是否被篡改
- 不可否认性：发送方无法否认曾发送该消息
- 密钥管理复杂度：对称体系需要 $O(n^2)$ 个密钥，非对称体系仅需 $O(n)$ 个密钥对

第三部分：Hash算法详解 MD5

MD5 基本概念

功能：将任意长度的数据转换为 128 位（16 字节）的固定长度哈希值。

设计目标：提供数据完整性校验，确保数据未被篡改。

MD5 工作流程（简化）

MD5 算法包含以下核心步骤：

1. 填充 **(Padding)**：在消息末尾添加填充位，使消息长度 448 (mod 512)
2. 附加长度：添加 64 位表示原始消息长度
3. 初始化缓冲区：设置四个 32 位寄存器初始值：A=0x67452301, B=0xEFCDAB89, C=0x98BADCFE, D=0x10325476
4. 压缩函数：将消息分为 512 位块，进行 4 轮共 64 步的非线性变换
5. 输出：拼接四个寄存器的最终值得到 128 位哈希值

核心非线性函数（增强混淆性）：

- F(B, C, D) = (B & C) | (~B & D)
- G(B, C, D) = (B & D) | (C & ~D)
- H(B, C, D) = B C D
- I(B, C, D) = C (B | ~D)

MD5 为什么不安全？

核心问题：抗碰撞性被攻破

- 碰撞攻击：2004 年王小云教授团队发现高效方法，能在实际可接受时间内构造 MD5 碰撞
- 安全隐患：攻击者可构造两个内容不同但 MD5 值相同的文件，从而伪造签名或绕过完整性检查
- 实际影响：恶意软件可伪装成合法软件通过 MD5 校验

安全建议：

- 禁止用途：密码存储、数字签名、代码完整性校验等任何安全相关场景
- 替代方案：SHA-256、SHA-3、BLAKE3 等抗碰撞能力更强的哈希算法