第五章 公钥密码学与RSA学习笔记

引言：从对称加密到公钥加密

1. 为什么需要公钥加密?

对称加密（如AES, DES）速度快、效率高，但在应用中面临两个无法解决的关键问题，这促使了公钥加密的诞生：

- **问题一：密钥分配** **(Key Distribution)**
    - **描述**：对称加密依赖于通信双方必须共享同一个秘密密钥。在不安全的网络环境中（如互联网），如何安全地将这个密钥从一方交给另一方，成为了一个“鸡生蛋还是蛋生鸡”的难题。
    - **扩展性问题**：在没有可信的密钥分发中心（KDC）时，如果 n 个用户希望两两之间都能安全通信，就需要 $C(n,2)=n(n-1)/2$ 个密钥。当用户量增大时，密钥的数量会呈指数级增长，导致密钥管理变得极其复杂。

- **问题二：数字签名** **(Digital Signature)**
    - **描述**：对称加密无法实现 **抗抵赖性 (Non-repudiation)** 。
    - **原因**：因为密钥是共享的，A 用密钥加密的消息，B 也可以声称是自己加密的；B 也可以伪造一个消息声称是 A 发送的。双方无法向第三方证明某个消息到底是谁发出的。

2. 公钥加密（Public-Key Cryptography）

为了解决上述问题，公钥加密（也称为双钥加密或非对称加密）被提了出来。

- **历史地位**：这可能是3000年加密历史中最重大的成果。
- **核心特征**：
    - **使用两个密钥**：一个**公钥** (Public Key) 和一个 **私钥** (Private Key) 。
    - **不对称** **(Asymmetric)**：公钥公开，私钥保密，通信双方的地位是不平等的。

- **数学基础**：巧妙地使用了数论中的“单向陷门函数”概念。
- **关系**：公钥加密**补充**了私钥加密，而不是**替代**它。在实际应用中，两者通常结合使用（例如，用公钥加密来安全地交换对称密钥）。

第一讲：公钥密码体系

1. 核心思想

为了解决传统对称密码体系中的两大难题：**密钥分发**和 **数字签名** 。其核心思想在于，将加密和解密的密钥进行分离，使用两个数学上相关联但又不同的密钥。

2. 主要构成

- **公钥 (Public Key)**
    - 可以被公之于众，任何人都可以获取。
    - 用于**加密**数据和 **验证签名(不能说解密签名)** 。
    - 好比一个可以接收信件的保险箱投递口，谁都可以往里投信。
- **私钥 (Private Key)**
    - 由用户自己严格保管，绝不外泄。
    - 用于**解密**数据和 **生成签名** 。
    - 好比打开保险箱的唯一钥匙，只有持有者才能取出信件。
- **密钥对 (Key Pair)**
    - 公钥和私钥成对出现，在数学上具有紧密的关联性。
    - 通过一个密钥（公钥）加密的信息，只能通过与之配对的另一个密钥（私钥）才能解密。
    - 从公钥很难（在计算上几乎不可能）推导出私钥。

3. 主要工作模式

模式一：加密通信

这是公钥密码体系最主要的应用场景，用于保证数据的机密性。

- **流程**：
    a. **接收方B** 生成一对密钥（公钥B，私钥B）。
    b. B 将自己的**公钥B**公开，任何人都可以获取。
    c. **发送方A** 想要给B发送机密信息，他会使用B的**公钥B**对信息进行加密。
    d. A 将加密后的密文发送给B。
    e. B 收到密文后，使用自己持有的**私钥B**进行解密，得到原始信息。
- **安全性**：即使中间人截获了密文，并且也知道B的公钥，无法解密信息。

模式二：数字签名 （完整版）

核心思想：“我用我的私钥做个标记，你们用我的公钥来验证这个标记对不对”，其中：

- 数字签名 **验证信息的来源（身份认证）、（提供不可否认性）**
- 哈希函数 **防篡改（完整性）** 。

流程图概览

- **签名方 (A)** : `文件原文` -> `[哈希函数]` -> `摘要(指纹)` -> `[使用A的私钥加密]` -> `数字签名`
- **验证方 (B)** :
    a. `数字签名` -> `[使用A的公钥解密]` -> `解密出的摘要（指纹A)`
    b. `文件原文` -> `[哈希函数]` -> `新计算的摘要（指纹B)`
    c. **核心步骤：比较** `指纹A` **和** `指纹B` **是否完全一样？**

详细步骤分解

第1部分：发送方 A 的签名过程

1. **生成摘要** ： A 准备发送一份文件。他首先用 **哈希函数** （例如 SHA-256）对**文件原文**进行运算，生成一个固定长度的、独一无二的字符串。这个字符串称为**摘要 (Digest)** 或 **数字指纹** 。
2. **加密摘要** ： A 使用自己的 **私钥** ， 对上一步生成的 ***Digest*** 进行加密。
3. **形成签名** ： 这个 **加密后的Digest** ， 就是本次操作的 **数字签名** 。
4. **发送** ： A 将**文件原文**和这个**数字签名**打包， 一同发送给接收方 B。

第2部分：接收方 B 的验证过程

1. **分离文件和签名** ： B 收到数据后，将其分为**文件原文**和**数字签名**两部分。
    a. **解密签名，得到指纹A** ： B 使用发送方 A 的 **公钥** （这是公开的）对**数字签名**进行解密。如果能成功解密，就会得到一个摘要。我们称之为“ **指纹A** ”。
    b. **计算原文，得到指纹B** ： B 对接收到的 **文件原文** ， 使用与 A 相同的 **哈希函数** （例如 SHA-256）重新计算一遍，也会得到一个摘要。我们称之为“ **指纹B** ”。
2. **对比指纹** ： B 将“ **指纹A** ”和“ **指纹B** ”进行精确对比。
3. **验证结论** ： **如果 指纹A === 指纹B** ： 验证成功！这同时说明了两点：
    - **身份可信** ： 只有 A 的公钥才能正确解密其私钥加密的签名。
    - **完整性** ： 文件原文计算出的“指纹B”与解密出的“指纹A”完全一致，证明文件在传输过程中未被篡改过。
    - **不可否认性** ： 只有私钥持有者才能签名

第二讲：数学原理

公钥密码体系的安全性，建立在几个关键的数学“难题”之上。这些难题保证了加密过程（正向计算）很容易，而破解过程（逆向推导）在计算上几乎不可能。

1. 模运算 (Modular Arithmetic)

模运算是数论的基础，也是公钥密码学的核心运算。

- **定义与符号**：

  $a \equiv b \pmod n$
  表示 a 和 b 除以 n 的余数相同。可以理解为 $(a-b)$ 是 n 的整数倍。

- **常见性质**：

  如果 $a \equiv b \pmod n$ 且 $c \equiv d \pmod n$，那么：

1. **加法** : $a + c \equiv b + d \pmod n$
2. **减法** : $a - c \equiv b - d \pmod n$
3. **乘法** : $a \times c \equiv b \times d \pmod n$
4. **幂运算** : $a^k \equiv b^k \pmod n$ (k为正整数)

2. 扩展欧几里得算法与模逆元

这是计算RSA私钥指数 d 的关键工具。

- **模逆元 (Modular Inverse)**：

  如果两个整数 a 和 n 互质，那么一定存在一个整数 b，使得：

$$a \times b \equiv 1 \pmod n$$

我们称 b 是 a 关于模 n 的 **模逆元** 。在RSA中，私钥指数 d 就是公钥指数 e 关于模 $\phi(n)$ 的模逆元。

- **扩展欧几里得算法**：
- **基础**：普通的欧几里得算法基于原理 $\gcd(a, b) = \gcd(b, a \pmod b)$，通过连续取余，直到余数为0，此时最后一个非零余数就是最大公约数。
- **扩展逻辑**：扩展欧几里得算法在执行普通欧几里得算法的同时，利用递归（或迭代）的性质，从后向前反向推导出**裴蜀定理** $ax + by = \gcd(a, b)$ 中的系数 x 和 y。
- **推导过程**：算法在递归到最后一层时（例如 $gcd(g, 0) = g$），可以轻易得到一组解 $g \cdot 1 + 0 \cdot 0 = g$，即 $x = 1, y = 0$。然后，算法在每一层递归返

回时，利用下一层的计算结果，迭代更新当前层的 x 和 y 值，直到返回到最顶层，最终得到我们需要的解。

- **如何用它求解模逆元？**：
我们要求解的是 $e \times d \equiv 1 \pmod{\phi(n)}$。
根据模运算的定义，这个式子等价于：
$e \times d = k \times \phi(n) + 1$，其中 k 是某个整数。
移项后得到：
$e \times d - k \times \phi(n) = 1$

  因为我们选择的 e 与 $\phi(n)$ 互质，所以 $\gcd(e, \phi(n)) = 1$。这与裴蜀等式 $ax + by = \gcd(a, b)$ 的形式完全一样！

  因此，我们就是要求解方程 $e \cdot d + k' \cdot \phi(n) = 1$ 中的整数解 (d, k')。通过**扩展欧几里得算法**，输入 e 和 $\phi(n)$，就可以求出对应的系数 d 和 k'，其中 d 就是我们想要的私钥指数。

- **举例说明**：

  在后续RSA的例子中，我们需要求解 $e = 17$ 关于 $\phi(n) = 3120$ 的模逆元 d。即求解：

$$17d \equiv 1 \pmod{3120}$$

  这等价于找到整数 d 和 k，使得 $17d + 3120k = 1$。使用扩展欧几里得算法过程如下（反向代入法）：

1. **正向计算最大公约数**：

   $3120 = 183 \times 17 + 9$

   $17 = 1 \times 9 + 8$

   $9 = 1 \times 8 + 1$

   $8 = 8 \times 1 + 0 \implies \gcd(3120, 17) = 1$

2. **反向代入求解**：
   从余数为1的式子开始：

   $1 = 9 - 1 \times 8$

   将上一行的 $8 = 17 - 1 \times 9$ 代入：

   $1 = 9 - 1 \times (17 - 1 \times 9) = 9 - 17 + 9 = 2 \times 9 - 1 \times 17$

   再将更上一行的 $9 = 3120 - 183 \times 17$ 代入：

   $1 = 2 \times (3120 - 183 \times 17) - 1 \times 17$

   $1 = 2 \times 3120 - 366 \times 17 - 1 \times 17$

$1 = 2 \times 3120 - 367 \times 17$

3. **整理结果**：
   我们得到了 $17 \times (-367) + 3120 \times 2 = 1$。

   此时的系数 -367 就是 d 的一个解。由于我们通常需要一个正整数解，可以加上模数进行转换：

   $d = -367 + 3120 = 2753$

   因此，17关于模3120的逆元就是2753。

3. 费马小定理 (Fermat's Little Theorem)

这是数论中的一个重要定理，为RSA算法的正确性提供了理论依据。

- **内容**：如果 p 是一个 **素数** （质数），而整数 a 不是 p 的倍数（即 a 与 p 互质），那么有：

$$a^{p-1} \equiv 1 \pmod{p}$$

4. 欧拉定理 (Euler's Theorem)

欧拉定理是费马小定理的推广，是RSA算法的直接理论基础。

- **欧拉函数** $\phi(n)$：
  - 定义：对于一个正整数 n，欧拉函数 $\phi(n)$ 表示小于或等于 n 的正整数中与 n **互质**的数的数目。
  - **重要性质**：如果 $n = p \times q$（p, q为不同素数），那么 $\phi(n) = (p-1)(q-1)$。
- **内容**：如果整数 a 和正整数 n **互质**，那么有：

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

5. 离散对数问题 (Discrete Logarithm Problem)

这是另一个支撑其他公钥密码体系（如DH密钥交换）的"困难问题"。

- **定义**：给定素数 p、原根 g 和整数 y，其中 $y \equiv g^x \pmod{p}$。在已知 p, g, y 时，求解 x 是非常困难的。

第三讲：RSA算法

1. 算法原理

RSA巧妙地利用了欧拉定理，并基于 **大整数分解的困难性** ，构建了一套安全的公钥密码体系。

2. 算法实现步骤

第一步：生成密钥对

1. **选择两个大素数** p 和 q。
2. **计算模数** $n = p \times q$。
3. **计算欧拉函数** $\phi(n) = (p-1) \times (q-1)$。
4. **选择公钥指数** e，要求 $1 < e < \phi(n)$ 且 e 与 $\phi(n)$ 互质。
5. **计算私钥指数** d，使得 $(e \times d) \pmod{\phi(n)} = 1$。这一步通过**扩展欧几里得算法**完成。
6. **封装密钥** ：公钥为 $\{e, n\}$，私钥为 $\{d, n\}$。

第二步：加密

用公钥 $\{e, n\}$ 加密明文 m：

$$c = m^e \pmod{n}$$

第三步：解密

用私钥 $\{d, n\}$ 解密密文 c：

$$m' = c^d \pmod{n}$$

3. RSA加解密正确性证明

我们需要证明，经过加密再解密后，能够还原出原始明文，即 $m' = m$。

$$m' = c^d \pmod{n} = (m^e)^d \pmod{n} = m^{ed} \pmod{n}$$

根据密钥生成步骤5，我们知道 $(e \times d) \pmod{\phi(n)} = 1$。所以，$ed$ 可以写成 $k \cdot \phi(n) + 1$ 的形式，其中 k 为某个正整数。
代入上式：

$$m' \equiv m^{k \cdot \phi(n)+1} \pmod{n} \equiv (m^{\phi(n)})^k \cdot m^1 \pmod{n}$$

接下来分两种情况讨论：

1. m **与** n **互质**：
   根据 **欧拉定理** ，$m^{\phi(n)} \equiv 1 \pmod{n}$。

所以，$m' \equiv 1^k \cdot m \pmod n \equiv m \pmod n$。证明成立。

2. m **与** n **不互质**：
   因为 $n = pq$，$m < n$，所以 m 必然是 p 或 q 的倍数。假设 $m = z \cdot p$ (z为整数)。
   在这种情况下，虽然不能直接用欧拉定理，但可以通过**中国剩余定理**证明 $m^{ed} \equiv m \pmod n$ 依然成立。

综上所述，无论何种情况，RSA的解密过程总能正确地还原出原始明文。

4. 安全性分析

RSA的安全性依赖于 **大整数分解的困难性** 。攻击者知道公钥 $\{e, n\}$，想要推导出私钥 d，就需要知道 $\phi(n)$。而要知道 $\phi(n) = (p-1)(q-1)$，就必须先对 n 进行因数分解得到 p 和 q。当 n 非常大时（如2048位），这是目前经典计算机无法在有效时间内完成的。