# 第四章 现代对称加密及其传输保密性 - 学习笔记

## 第一部分：现代对称加密算法

在DES的密钥长度（56位）逐渐无法满足安全需求的背景下，密码学界发展出了一系列强度更高、设计更现代的对称分组加密算法。

### 1.1 三重DES (Triple DES / 3DES)

3DES是为了替代DES而设计的方案，通过对DES进行多重加密来极大地增强其安全性。

#### **为什么不是双重DES？—— 中间相遇攻击 (Meet-in-the-Middle Attack)**

单纯地用两个不同密钥进行两次DES加密（双重DES）并不能将密钥长度有效地扩展到112位（56\*2）。攻击者可以采用**中间相遇攻击**来破解它：

1. **攻击原理**：
   * 对于加密过程 C = E\_K2(E\_K1(P))，存在一个中间值 X，使得 X = E\_K1(P) 且 X = D\_K2(C)。
   * 攻击者已知一对明文 P 和密文 C。
2. **攻击步骤**：
   * 用所有 2^56 个可能的密钥 K1 加密明文 P，将得到的结果 {X} 和对应的 {K1} 存储起来。
   * 用所有 2^56 个可能的密钥 K2 解密密文 C，将得到的结果 {X} 和对应的 {K2} 存储起来。
   * 通过比较两个存储集合，找到匹配的中间值 X。对应的 K1 和 K2 就是密钥对。
3. **结论**：这种攻击的计算复杂度约为 2^56 + 2^56 = 2^57，与暴力破解单DES的 2^55 复杂度在同一数量级，因此双重DES的安全性提升有限。

#### **3DES的工作原理**

为了有效抵抗中间相遇攻击，3DES采用了加密-解密-加密（EDE）的结构。

* **双密钥3DES (Two-Key 3DES)**
  + **过程**: C = E\_K1(D\_K2(E\_K1(P)))
  + **特点**:
    - 使用两个独立的密钥K1和K2。
    - 有效密钥长度为 56 \* 2 = 112 位。
    - 当 K1 = K2 时，该过程等价于单次DES加密，从而实现了对DES的向后兼容。这是采用EDE结构而非EEE（加密-加密-加密）的一个重要原因。
  + **防御中间相遇攻击**:
    - 之所以能防御该攻击，是因为攻击者无法将加密过程干净地分成两半（每一半只依赖一个密钥）。在 C = E\_K1(D\_K2(E\_K1(P))) 中，密钥 K1 同时出现在了运算的“首”和“尾”。
    - 攻击者如果想从两端向中间计算，例如计算 D\_K1(C) 和 D\_K2(E\_K1(P))，就必须在运算一端同时猜测 K1 和 K2 才能得到中间值，这使得攻击的复杂度回到了 2^112 级别，失去了中间相遇攻击的优势。
* **三密钥3DES (Three-Key 3DES)**
  + **过程**: C = E\_K3(D\_K2(E\_K1(P)))
  + **特点**:
    - 使用三个独立的密钥K1, K2, K3。
    - 有效密钥长度为 56 \* 3 = 168 位，是目前最安全的3DES变体。
    - 广泛应用于PGP、S/MIME等安全协议中。
  + **防御中间相遇攻击**:
    - 防御能力更强。攻击者若想在中间点例如： X = D\_K2(E\_K1(P)) 相遇，需要从明文端计算所有 (K1, K2) 组合（2^112 种可能性），从密文端计算所有 K3（2^56 种可能性）。
    - 攻击的整体复杂度由计算量较大的一方（即 2^112）决定，因此该攻击方法对于三密钥3DES是无效的。

### 1.2 Blowfish

* **设计者**: Bruce Schneier (1993/94年设计)
* **特点**:
  1. **快速**: 可在32位CPU上快速实现，每加密一个字节约需18个时钟周期。
  2. **紧凑**: 内存占用少，只需约5KB。
  3. **简单**: 算法结构简单，易于实现和分析其强度。
  4. **安全性可变**: 密钥长度是可变的，最长可达448位，提供了灵活的安全等级。

### 1.3 RC5

* **设计者**: Ron Rivest (RSA公司，1994年设计)
* **特点**:
  1. **高度灵活/参数化**:
     + **可变的字长 (w)**: 可适应不同位数的CPU（如16, 32, 64位）。
     + **可变的加密轮数 (r)**: 轮数越多，安全性越高，但速度越慢。
     + **可变的密钥长度 (b)**: 密钥长度可灵活调整。
  2. **核心操作**: 其安全性强依赖于**数据相关的循环移位**，这一设计能有效增强其抵抗差分和线性密码分析的能力。
  3. **设计简洁**: 易于在不同的CPU上实现，且对内存要求低。

### 1.4 常用分组算法比较

| **算法名称** | **分组长度 (bits)** | **密钥长度 (bits)** | **加密速度 (cycle/byte)** |
| --- | --- | --- | --- |
| DES | 64 | 56 | 45 |
| **3DES** | **64** | **168** | **108** |
| **Blowfish** | **64** | **128 ~ 448** | **18** |
| **RC5** | **64 ~ 256** | **64 ~ 256** | **23** |
| AES | 128 | 128 ~ 256 | 20-27 |

*注：加密速度为奔腾CPU上的参考值，数值越小表示速度越快。*

### 1.5 流密码 (Stream Ciphers)

与分组密码一次处理一个数据块不同，流密码一次处理一位或一个字节的数据。

* **工作原理**:
  + 核心思想是生成一个与明文长度相同的、看似随机的密钥流（Keystream）。
  + 然后使用简单的**异或 (XOR)** 操作将明文流与密钥流结合，生成密文流。
  + **加密**: C\_i = M\_i ⊕ StreamKey\_i
  + **解密**: M\_i = C\_i ⊕ StreamKey\_i （因为 M ⊕ K ⊕ K = M）
  + 密钥流本身由一个**伪随机数生成器 (PRNG)** 根据一个较短的初始密钥生成。
* **设计原则**:
  1. **绝对不能重用密钥流**: 这是流密码的“黄金法则”。如果用同一个密钥流加密两个不同的明文（P1, P2），攻击者可以截获两个密文（C1, C2）并进行异或：C1 ⊕ C2 = (P1 ⊕ K) ⊕ (P2 ⊕ K) = P1 ⊕ P2。这将消除密钥，直接暴露两个明文的异或结果，攻击者可利用语言的统计特性从中破解出明文。
  2. **密钥流周期要长**: 密钥流最终会重复，这个重复的长度称为周期。周期必须足够长，以避免在加密长消息时重复使用密钥流。
  3. **统计上随机**: 密钥流应通过所有标准的随机性测试，不能有任何可被利用的统计偏差。
  4. **依赖于足够大的密钥**: 伪随机数生成器的初始密钥必须足够长，以抵抗暴力破解攻击。

## 第二部分：传输安全性

保证数据在网络中传输时的机密性，需要考虑加密的设置、密钥的分配和随机数的生成。

### 2.1 加密设置 (两种模式)

#### **1. 链路加密 (Link-to-Link Encryption)**

* **工作方式**: 数据在从一个节点到下一个节点的**每一段物理链路**上都被独立加密和解密。在每个中间节点（如路由器），数据包需要被解密以读取路由信息，然后再用下一段链路的密钥重新加密。
* **优点**: 对用户透明，由网络服务提供商实现，可以硬件实现，速度快。
* **缺点**: 数据在中间节点以**明文形式**短暂存在，如果中间节点被攻破，数据就会泄露。无法防范节点内部的攻击。
* **实际应用举例**:
  + **公司VPN**: 当你使用公司的VPN（虚拟专用网络）访问内部资源时，你的数据在离开你的电脑到公司网络入口的这段公共互联网链路上是加密的。但一旦进入公司内网，数据通常会解密成明文进行传输。
  + **Wi-Fi加密 (WPA2/3)**: 你连接Wi-Fi时，你的设备和无线路由器之间的数据传输是加密的。但这只保护了“空中”这段链路，路由器本身以及数据离开路由器之后的部分是不受此加密保护的。

#### **2. 端到端加密 (End-to-End Encryption)**

* **工作方式**: 数据在**发送端**被加密，直到**最终接收端**才被解密。整个传输过程中，无论经过多少中间节点，数据内容始终是加密的。
* **优点**: 提供了更高的安全性，即使中间节点不可信，数据内容也是安全的。用户可以自己选择加密算法和实现方式（通常是软件实现）。
* **缺点**: 网络头部信息（如IP地址）不能被加密，否则中间节点无法路由。因此，无法防止**流量分析**攻击（即攻击者虽然不知道通信内容，但知道谁在和谁通信）。
* **实际应用举例**:
  + **安全聊天软件 (WhatsApp, Signal)**: 你发送的消息在你的手机上就被加密了，只有聊天对象的手机才能解密。服务商（例如Meta公司）的服务器也无法看到你们的聊天内容。
  + **HTTPS/TLS**: 当你访问一个 https:// 开头的网站（例如网上银行）时，你的浏览器和该网站的服务器之间建立了一条端到端加密的通道。中间的任何网络设备都无法窃听你的密码或浏览内容。

#### **对比总结**

| **特点** | **链路加密** | **端到端加密** |
| --- | --- | --- |
| **数据状态** | 在中间节点为**明文** | 在中间节点为**密文** |
| **用户参与** | 用户无感知，对用户透明 | 用户需参与，自己选择算法 |
| **实现方式** | 通常为硬件实现 | 通常为软件实现 |
| **安全性** | 保护数据在链路上的安全 | 保护数据从源到最终目的地的安全 |
| **头部信息** | 整个数据包（包括头）都加密 | 只有应用层数据被加密，网络头不加密 |

### 2.2 密钥分配 (Key Distribution)

对称加密的前提是通信双方必须共享一个秘密密钥。如何安全地让双方获得这个密钥是关键问题。

* **密钥层次**:
  + **主密钥 (Master Key)**: 长期使用、手动或通过安全渠道分发的密钥，用于加密会话密钥。
  + **会话密钥 (Session Key)**: 临时密钥，仅用于一次通信会话，由主密钥保护进行分发，用完即弃。这样可以限制单次会话被破解后造成的影响。
* **集中式密钥分配（具体见后文）**:
  + 使用一个可信的第三方——**密钥分发中心 (KDC)**。
  + 每个用户都与KDC共享一个唯一的主密钥。
  + 当用户A想和B通信时，A向KDC请求一个会话密钥；
  + KDC生成一个随机的会话密钥：然后用A的主密钥加密一份发给A，用B的主密钥加密另一份（也发给A，由A转发给B）。
  + 通信双方使用这个会话密钥进行加密通信。
  + **临时交互号 (Nonce)**: 在请求和响应中加入一个随机数（Nonce），可以防止**重放攻击**。

### 2.3 随机数的产生 (Random Number Generation)

随机数在密码学中至关重要，用于生成会话密钥、Nonce、初始化向量等。

* **关键特性**:
  1. **随机性**: 序列没有统计规律（均匀分布、独立性）。
  2. **不可预测性**: 不能从序列的一部分推断出后续的部分。
* **伪随机数生成器 (PRNG)**:
  + 使用一个确定性算法，从一个短的随机“种子”（seed）开始，生成一个长的、看起来随机的比特序列。
  + **线性同余法**: 是一个简单但**不安全**的PRNG。
    - **公式**: X\_n+1 = (a \* X\_n + c) mod m
    - **缺点**: 生成的序列周期有限，且如果知道序列中的几个数，就可以预测后续所有的数，不具备密码学要求的不可预测性。**绝对不能用于生成密钥等敏感数据！**