公钥密码

NIST后量子密码算法征集概述

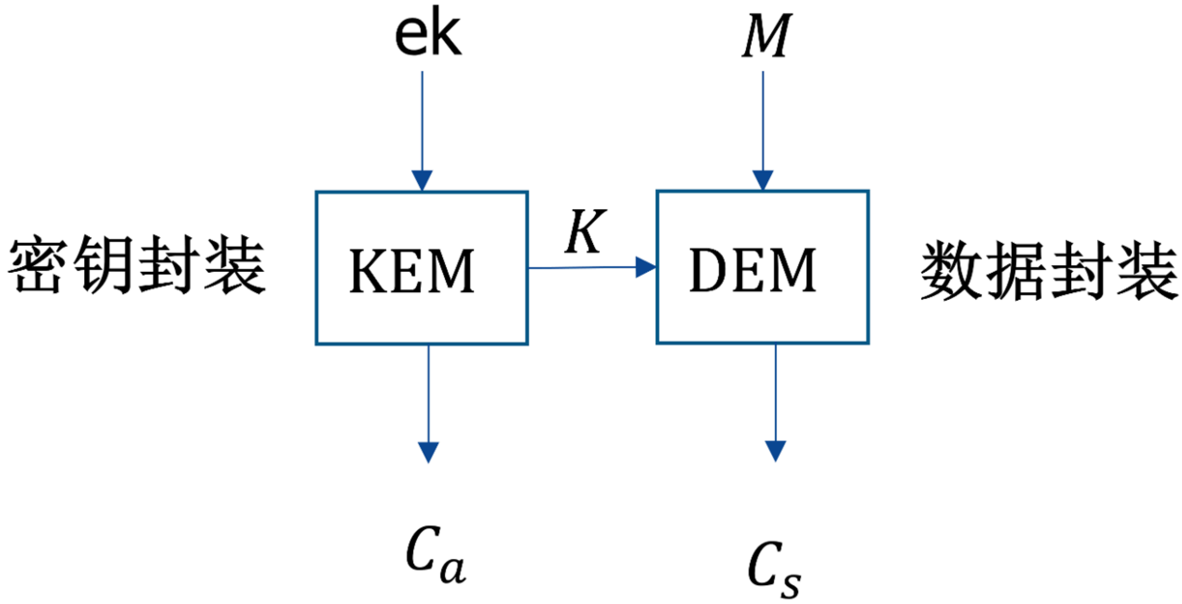
密码算法种类:

- 公钥加密与密钥封装机制（KEMs）：用于在不安全信道中安全建立共享密钥。
- 数字签名：用于身份认证和数据完整性验证。

数学难题种类:

- 基于格 (Lattice-based)：CRYSTALS-KYBER（用于加密/KEM）和CRYSTALS-DILITHIUM（用于签名）
- 基于编码 (Code-based)
- 基于多变量 (Multivariate-based)
- 基于哈希 (Hash-based)
- 其他

公钥加密和密钥封装的区别



将传统的公钥加密拆分为密钥封装+对称加密(KEM-DEM)两个部分

优点:

- 效率优化：专为密钥传输设计的KEM在效率和密文大小上通常比直接使用通用的公钥加密算法更优；后续对数据进行对称加密通常比直接使用公钥加密更快。
- 简化安全性证明：在可证明安全领域，为KEM设计安全证明比为一个能加密任意消息的全功能公钥加密方案更简单。

密钥分发与密钥交换

- 密钥分发：由通信中的一方生成一个对称会话密钥，然后通过某种安全方式将这个密钥"分发"或"传递"给另一方的过程。

- 密钥交换：指的是通信双方通过交换一些公开参数，各自独立地计算出一个相同的共享密钥。这个共享密钥不是由一方生成并传给另一方的，而是双方共同贡献随机性并通过数学计算衍生出来的。

现代密码学（尤其后量子密码）中，密钥分发和密钥交换逐渐融合成密钥封装(KEM)：分发简单（密钥分发形式）；内部数学原理复杂，安全性高（密钥交换内核）。

极简CRYSTALS-KYBER

CRYSTALS-KYBER：$R_q = \mathbb{Z}_q[X]/(X^n+1)$其中$q = 3329, n = 256$。

符号说明

- A：一个矩阵，其元素来自多项式环 R_q。它是公共参数，通常由随机数种子生成。
- s, e, e_1, e_2, r：小的误差多项式，其系数很小（来自错误分布）。安全性就依赖于这些误差的存在。
- t：公钥的一部分。
- (pk, sk)：公钥和私钥对。
- m：要加密的消息（明文），被编码为一个环元素。
- (u, v)：密文，由两个组件构成。

密钥生成

- 生成一个随机矩阵 A（在实践中，由一个随机种子扩展而来，但这里我们假设它是直接随机生成的）。
- 随机生成一个私钥向量 s，其元素是小的误差多项式。
- 随机生成一个错误向量 e，其元素也是小的误差多项式。
- 计算公钥向量：$t = A \cdot s + e$
- 公钥：$pk = (A, t)$，私钥 $sk = s$

密钥生成例

假设环 $R_{17} = \mathbb{Z}_{17}[X]/(X^4+1)$。使用向量表示多项式，例如 $f = 3X^3 + 0X^2 + 1X + 4$ 表示为 $[4, 1, 0, 3]$。

- 生成随机公共矩阵 A (为了简化，我们使用 1x1 矩阵，即单个多项式):
  $A = [2X^3 + 3X^2 + X + 1] = [1, 1, 3, 2]$
- 生成小的私钥多项式 s 和错误多项式 e:
  - $s = [-1X^2 + 1] = [1, 0, -1, 0]$
  - $e = [X - 1] = [-1, 1, 0, 0]$
- 计算 $t = A \cdot s + e$
  - 先计算 $A \cdot s$：
    $[1, 1, 3, 2] \cdot [1, 0, -1, 0]$ (注意：这是多项式乘法，然后模 $X^4 + 1$)
    $(1 * 1) + (1 * 0)X + (3 * -1)X^2 + (2 * 0)X^3 = 1 - 3X^2$
    结果为 $[1, 0, -3, 0]$。模 $q = 17$ 后，$-3 \equiv 14$ 所以是 $[1, 0, 14, 0]$。
  - 然后加错误 e：$[1, 0, 14, 0] + [-1, 1, 0, 0] = [0, 1, 14, 0]$
  - 所以 $t = [0, 1, 14, 0]$ (即 $X + 14X^2$)
- 公钥：$pk = (A, t) = ([1, 1, 3, 2], [0, 1, 14, 0])$，私钥：$sk = s = [1, 0, -1, 0]$

加密

- 将消息 m 编码到环 R_q 中的一个多项式。常用方法：将比特 0 映射为 0，比特 1 映射为 $\lfloor q/2 \rceil$（例如，在真实Kyber中，0 映射为 0，1 映射为 1664）。
- 随机生成一个小的向量 r（用于"随机重线性化"）。
- 随机生成两个小的错误多项式 e_1 和 e_2。
- 计算密文的两个组件：
  - $u = A^T \cdot r + e_1$
  - $v = t^T \cdot r + e_2 + m$
- 输出密文 (u, v)。

加密例

- 假设我们要加密的消息 m 的编码是 $[0, 0, 8, 0]$（这里 $q/2 \approx 8$，代表一个"1"比特被编码在了 X^2 的位置上）。
- 生成随机向量 r 和错误 e_1, e_2：
  - $r = [X] = [0, 1, 0, 0]$
  - $e_1 = [1] = [1, 0, 0, 0]$
  - $e_2 = [-1] = [-1, 0, 0, 0]$
- 计算 $u = A^T \cdot r + e_1$
  - $A^T \cdot r = [1, 1, 3, 2] \cdot [0, 1, 0, 0] = 1 * X = X = [0, 1, 0, 0]$
  - $u = [0, 1, 0, 0] + [1, 0, 0, 0] = [1, 1, 0, 0]$
- 计算 $v = t^T \cdot r + e_2 + m$
  - $t^T \cdot r = [0, 1, 14, 0] \cdot [0, 1, 0, 0] = 1 * X = X = [0, 1, 0, 0]$
  - $v = [0, 1, 0, 0] + [-1, 0, 0, 0] + [0, 0, 8, 0] = [-1, 1, 8, 0]$。模 $q = 17$ 后，$-1 \equiv 16$，所以 $v = [16, 1, 8, 0]$
- 密文 $(u, v) = ([1, 1, 0, 0], [16, 1, 8, 0])$

解密

- 计算近似值：$v - s^T \cdot u$
- 这个结果会近似等于 $m + \text{small error}$。
- 对结果的每个系数进行"舍入"，恢复出编码后的消息 m'。
  - 如果系数接近 0，解码为 0。
  - 如果系数接近 $\lfloor q/2 \rceil$，解码为 1。

解密例

- 我们有：
  - 私钥 $s = [1, 0, -1, 0]$
  - 密文 $u = [1, 1, 0, 0], v = [16, 1, 8, 0]$
- 计算 $s^T \cdot u$: $s \cdot u = [1, 1, -1, 0]$。
- 计算 $v - s \cdot u = [16, 1, 8, 0] - [1, 1, -1, 0] = [15, 0, 9, 0]$

- 解码：检查向量 $[15, 0, 9, 0]$，$m' = [0, 0, 1, 0]$,