CAN304 W4

The public-key revolution

对称加密 (symmetric cryptograph) 带来了一系列安全问题：如何安全地共享密钥？多个人如何共享密钥 (每两人一个？)？

非对称加密 (asymmetric cryptography)：一方生成一对密钥 - 公钥pk 和私钥 sk，公钥被广泛传播，私钥是保密的。

一些问题表现出非对称性 - 易于计算，但难以反转 (invert)。我们使用这些问题构建非对称加密。

- Factoring problem：计算两个数字的乘积很容易；但根据乘积分解数字很难

Public-key encryption

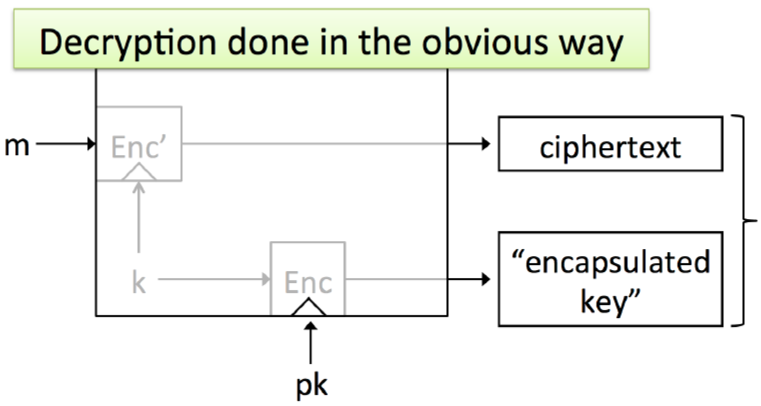
Public-key encryption (PKE)

public-key encryption scheme 由三种算法组成：

- Gen：key-generation algorithm，生成公钥 pk 和私钥 sk
- Enc：根据输入 pk 和 message m，输出密文 c 的加密算法
- Dec：根据输入 sk 和密文 c，输出 message m 或 ⊥ (error) 的解密算法

For all m and pk, sk output by Gen,
$Dec_{sk}(Enc_{pk}(m)) = m$

Hybrid encryption



使用对称加密对 m 进行加密，然后用非对称加密对密钥 k 进行加密。解密时先用私钥解密 k，再用 k 解密密文。(由于 m 会很长，直接用非对称加密计算量会很大)

Dlog-based PKE: ElGamal encryption

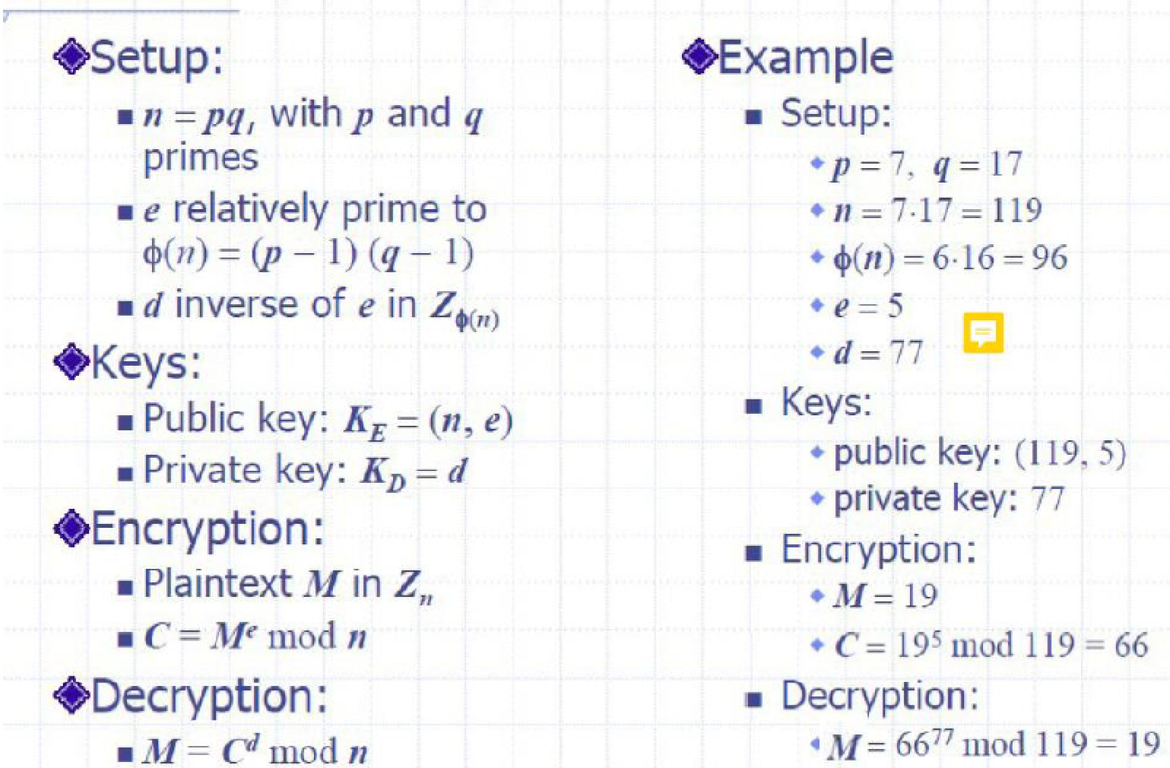
Dlog problem：给定 g 和 group G 中的一个元素 h，找到 x 使得 $g^x = h$。

- Gen
  - 初始化 group 参数 G，q，g；选择 uniform $x \in Z_q$ (Z_q = {1, 2, ..., q-1})，计算 h = g^x
  - Public key is h, private key is x
- Enc$_{pk}$(m), where $m \in G$
  - 选择 uniform $y \in Z_q$
  - The ciphertext is $(c_1, \ c_2) = (g^y, \ h^y \cdot m)$
- Dec$_{sk}$(c_1, c_2)
  - 解密输出 $\frac{c_2}{c_1^x}$

RSA encryption

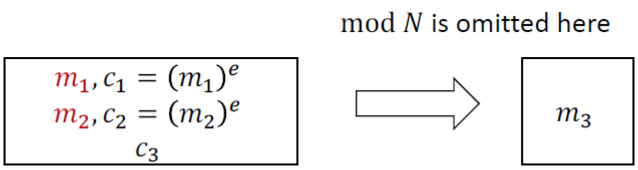
We then choose two numbers *e* and *d* such that

1. *e* and $\varphi(n)$ are relatively prime, i.e. $\gcd(e, \varphi(n)) = 1$
2. $ed \equiv 1 \pmod{\varphi(n)}$ (by Extended Euclidean algorithm)



Chosen-plaintext attack (CPA)

攻击类型：Ciphertext-only attack -> Known-plaintext attack -> Chosen-plaintext attack -> Chosen-ciphertext attack，强度依次增加。



假如攻击者知道一些 plaintext-ciphertext pairs，例如 (m_1, c_1) 和 (m_2, c_2)。现在有一个密文 c_3，如果 $c_1 \cdot c_2 = c_3$，那么 $m_1 \cdot m_2 = m_3$。

因此，Plain RSA is not CPA-secure，不过这个问题可以通过 PKCS 解决。

PKCS: Public-Key Cryptography Standard (PKCS)

- idea：add random padding
  - 要加密 m，随机选择一个 r，把 r 添加到 m 里
  - $c = [(r|m)^e modN]$

Digital signature

Digital signature 和 MACs 的区别：

- Public verifiability
  - “任何人”都可以验证 signature，而只有密钥持有者才能验证 MAC 的 tag
- Transferability
  - 可以将 signature 转发给其他人
- Non-repudiation
  - 签名者不能（轻易地）否认签发的签名 (可以使用 pk 的公共副本验证签名)
  - 而 MAC 无法提供此功能 (无法访问密钥，无法验证 tag)，而且无法确定是谁发出的签名 (有密钥的都可以发)

Signature schemes

签名方案由三种 PPT 算法（Gen、Sign、Vrfy）定义：

- Gen：输入 1^n (指定密钥长度)，输出 pk 和 sk
- Sign：将私钥 sk 和 message $m \in \{0,1\}^*$，输出 signature σ

$$\sigma \leftarrow Sign_{sk}(m)$$

- Vrfy：输入公钥 pk，message m 和 signature σ，输出 0 或 1 (拒绝和接受)

For all m and all pk, sk output by Gen,
$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$

Hash-and-sign paradigm

给定：

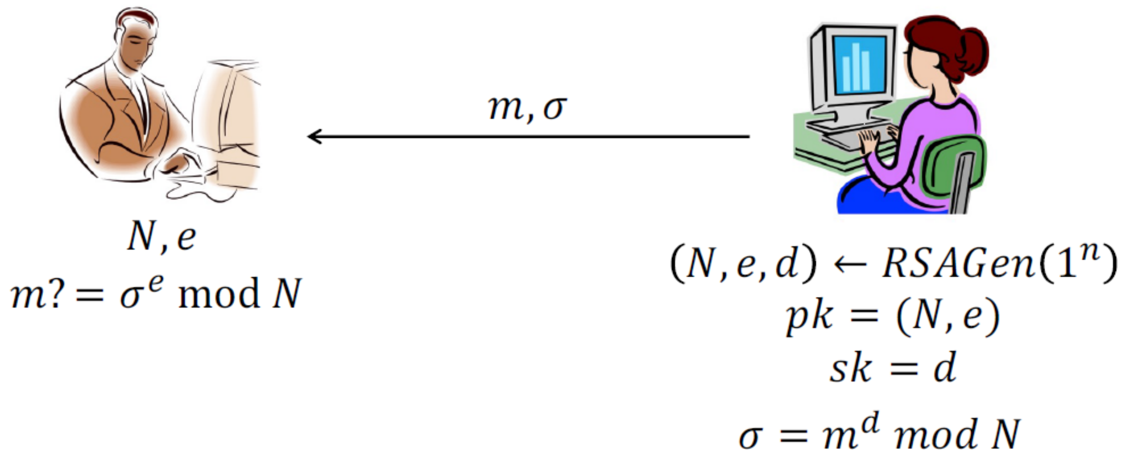
- 一个 signature scheme Π=(Gen, Sign, Vrfy) 来签名长度为 n 的短 message
- Hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$

构建一个可以适用于任意长度 message 的 signature scheme Π‘ =(Gen’, Sign‘, Vrfy’)：

- $Sign'_{sk}(m) = Sign_{sk}(H(m))$
- $Vrfy'_{pk}(m, \sigma) = Vrfy_{pk}(H(m), \sigma)$

RSA-based signatures

“Plain” RSA signatures



Attacks

- sign specific messages
  - 假如给定 m=1，可以得到 $\sigma = 1$，因为 $\sigma = [1^d mod e N] = 1$
- sign “random” messages
  - 选择随机的 σ，设置 $m = [\sigma^e mod e N]$
- combine two signatures to obtain a third
  - 签名 σ_1，σ_2 是合法的签名，它们分别对应于 m_1，m_2
  - 那么 $\sigma' = \sigma_1\sigma_2 \; mod \; N$ 是合法的签名，它对应于 $m' = m_1m_2 \; mod \; N$。因为 $(\sigma_1\sigma_2)^e = \sigma_1^e\sigma_2^e = m_1m_2 \; mod \; N$

RSA-FDH

RSA-FDH: RSA full-domain hash，也是一个 Hash-and-sign paradigm。

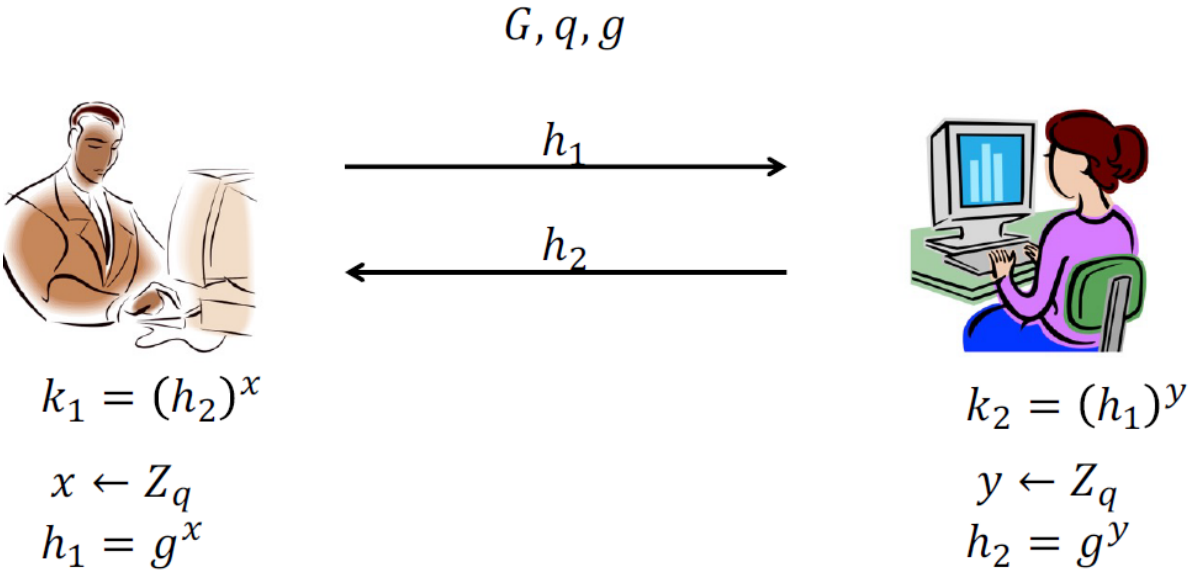
- Public key: (N, e); private key: d
- $\text{Sign}_{\text{sk}}(m) = H(m)^d \bmod N$
- $\text{Vrfy}_{\text{pk}}(m, \sigma)$: output 1 iff

$$\sigma^e = H(m) \bmod N$$

Diffie-Hellman key agreement

Decisional Diffie-Hellman (DDH) problem：给定 g^x 和 g^y，要从 g^{xy} 中找到它们很困难。

G：cyclic group；q: prime, order of G； g: generator of G。

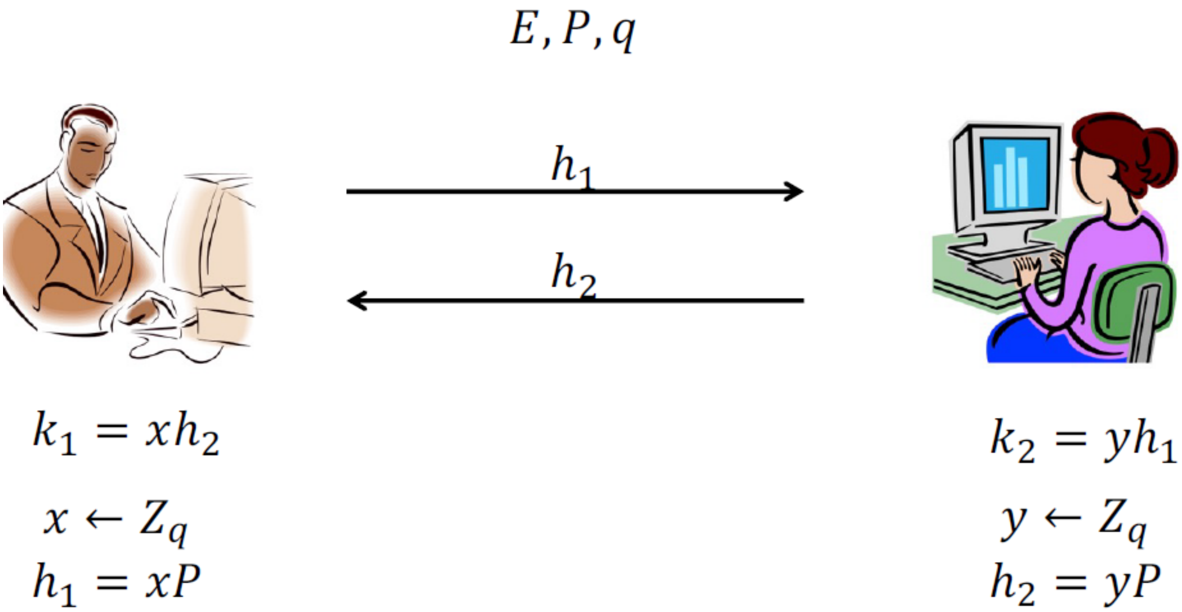


两人分别从 Z_q 中生成 x 和 y，然后计算出 h_1 和 h_2 并交换，最后生成 k_1 和 k_2，k_1 和 k_2 是相等的 (因为 $(g^x)^y = (g^y)^x$)。

Elliptic curve Diffie-Hellman key agreement

ECDDH problem：给定 yP 和 xP，要从 xyP 中区分它们很困难。

E: elliptic curve group； q: prime, order of E； P: generator of E。



k_1 和 k_2 也是相等的 (因为 $xyP = yxP$)。

Man-in-the-middle to EC(DH)

$$E, P, q$$

$$k_1 = xh \qquad k_3 = ch_1, k_4 = ch_2 \qquad k_2 = yh$$

$$x \leftarrow Z_q \qquad c \leftarrow Z_q \qquad y \leftarrow Z_q$$

$$h_1 = xP \qquad h = cP \qquad h_2 = yP$$

攻击者可以替换双方的密钥，然后 k_1 和 k_3 相等，k_2 和 k_4 相等。

可以通过在协议中引入身份验证 (authentication) 来解决。