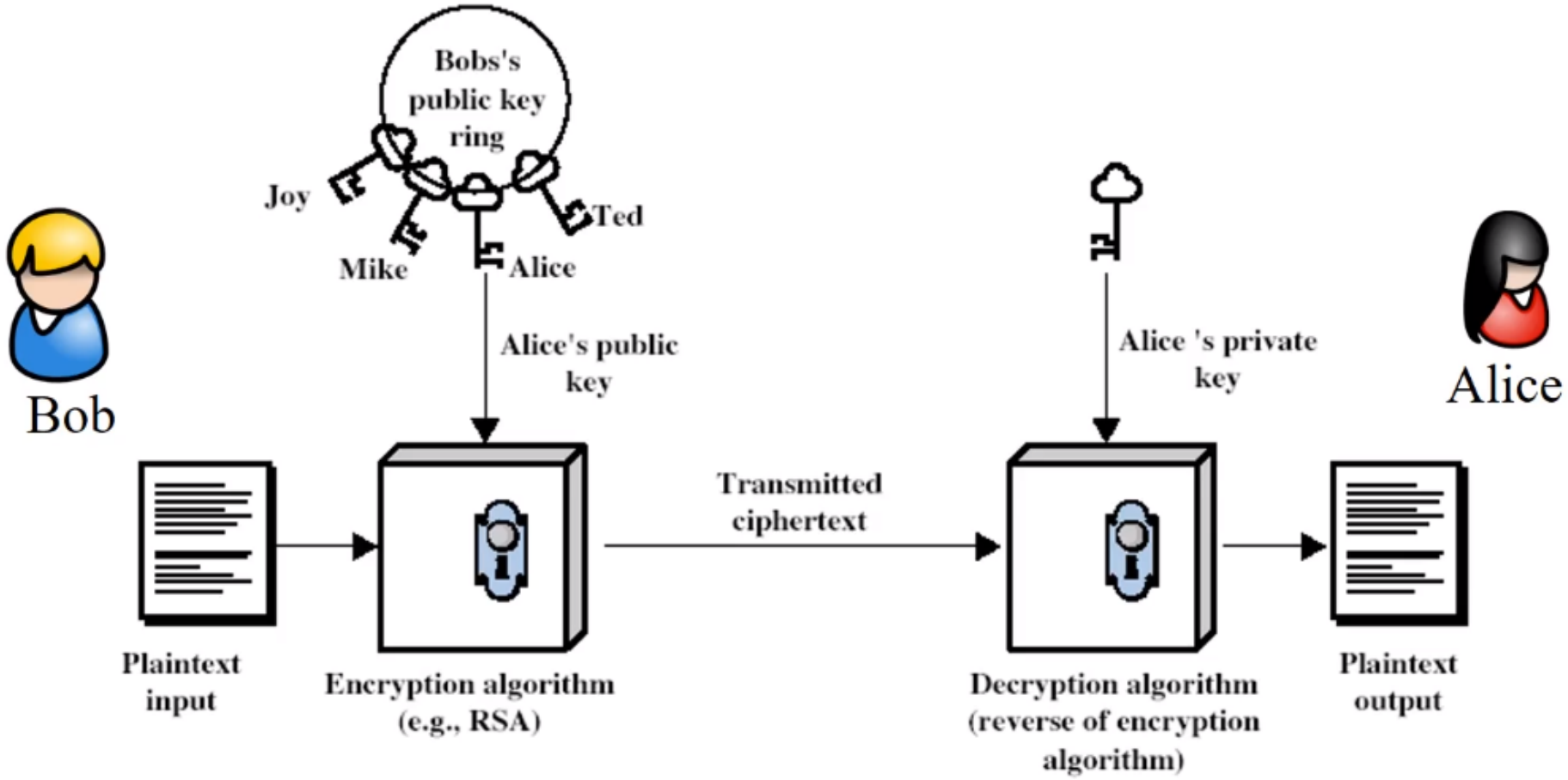
# 第五章 RSA加密算法

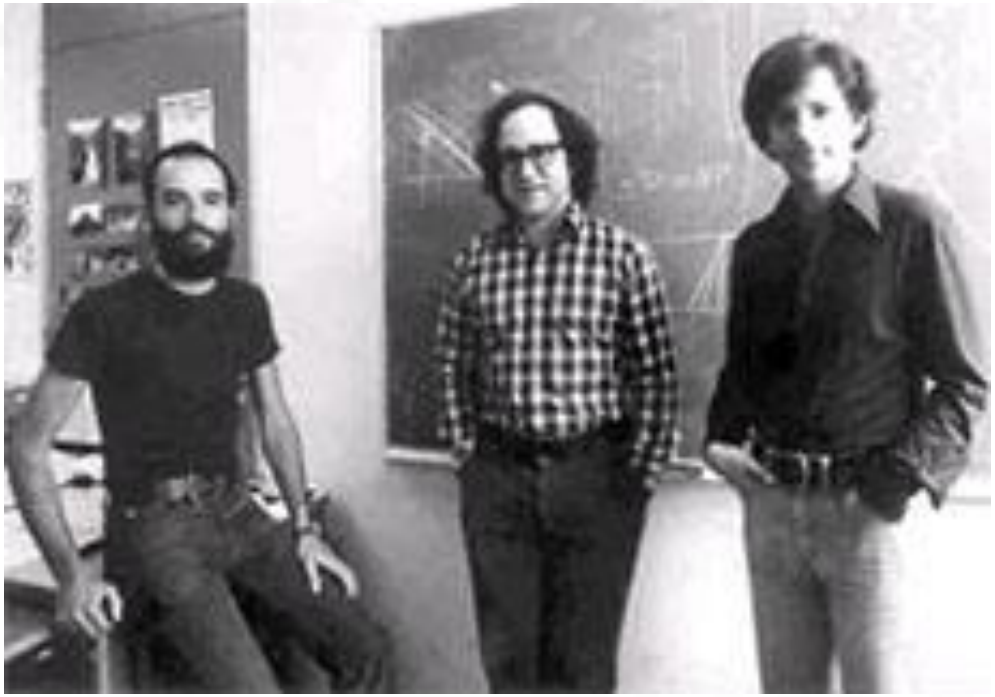
# 5.1 公钥加密体制

- 在公钥加密体制/非对称加密体制中，每个用户有一对密钥：公开的加密密钥（公钥）和保密的解密密钥（私钥），数据发送者使用公钥对数据进行加密，而接收者使用私钥解密密文获得数据。

- 公钥加密体制解决了对称加密体制中密钥管理问题，将密钥数量由$O(n^2)$减少为$O(n)$。


Bobs's public key ring
Joy
Ted
Mike
Alice
Alice's public key
Alice 's private key
Bob
Alice
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

# 5.2 RSA公钥加密体制历史



1977年，RSA加密算法由麻省理工学院的Ron Rivest, Adi Shamir, Leonard Adlema提出，并受到广泛关注。

- RSA加密算法是一种公钥加密算法，依赖于大整数分解问题。

- 1973年，在英国政府通讯总部工作的数学家Clifford Cocks在一份内部文件中提出了一个与RSA一致的算法，但该算法是机密文件，直到1997年才得以公开。

- RSA作为全世界第一个切实可行的公钥算法，目前被广泛应用及部署到不同的场景，比如上网时采用浏览器访问网页时使用的超文本传输安全协议（HTTPS, HyperText Transfer Protocol Secure)，它通过HTTP协议进行通信，但使用Transport Layer Security（TLS）或其前身Secure Socket Layer（SSL）加密数据包，因此也经常被称为TLS/SSL上的HTTP。

# 5.3 RSA公钥加密体制原理

# 5.3 RSA公钥加密体制原理

- 密钥生成（由Alice生成）
  （1）选取两个大素数p,q
  （2）计算n=pq， z=(p-1)(q-1)。（z为n的欧拉函数）
  （3）随机选取e（其中e<n），e与z互素
  （4）计算d，使得ed-1能够被z整除，即ed mod z = 1
  （5）公钥是(n,e)；私钥是(n,d)。

# 5.3 RSA公钥加密体制原理

- 加密/解密算法

  如上所述给出的(n, e)和(n, d)。

  （1）加密：由$c = m^e \bmod n$将明文m转变为密文c（即：$m^e$除以n所得的余数）。 其中$m<n$

  （2）解密：$m = c^d \bmod n$（即：$c^d$除以n所得的余数）

核心思想：$m = (m^e \bmod n)^d \bmod n$

# 5.3 RSA公钥加密体制原理

- 由欧拉定理得出：

  当(a, N)=1时，$a^{\varphi(N)} \equiv 1 \pmod{m}$

  在RSA中有：

  1. N = pq
  2. $\varphi(N) = (p-1)(q-1)$
  3. 选择整数e和d，d为e关于模$\varphi(n)$ 的模反元素
  4. $ed \equiv 1 \pmod{\varphi(N)}$

  于是有：$C^d \equiv (M^e)^d \equiv M^{1+k\cdot\varphi(N)} \equiv M^1 \cdot (M^{\varphi(N)})^k$

  $\equiv M^1 \cdot (1)^k \equiv M^1 \equiv M \pmod{N}$

# 5.4 模重复平方方法

# 5.4 模重复平方方法

- 在模算术计算中，常常要对大整数m和n，计算$b^n \pmod m$.如果用递归，则可以得到：

$$b^n \pmod m = b^n \% m = b*b^{(n-1)} \% m .$$

根据模的运算规则，可以进一步得到：

$$b \times b^{n-1} \equiv b \times (b^{n-1} \pmod m) \pmod m$$

所以有：$$b^n \equiv b \times (b^{n-1} \pmod m) \pmod m$$

得到$b^n \pmod m$的递归公式为：

func(b, n, m) = b* func(b, n-1, m) % m.

# 5.4 模重复平方法

- 模重复计算法的递归实现，思路简单清晰，但是因为递归层次过深，同时需要执行n-1次乘法，所以很难用于实际应用之中。
- 一般我们会使用模重复平方法来实现（时间复杂度0(logn)）。

# 5.4 模重复平方法

现在，将n写成二进制: $n = n_0 + n_1 2 + \cdots n_{k-1} 2^{k-1}$

其中，$n_i \in \{0, 1\}$，$i=0, 1, \ldots, k-1$

则 $b^n \pmod m$ 的计算可归纳为:

$$bn \equiv b^{n_0} (b^2)^{n_1} \cdots (b^{2^{k-1}})^{n_{k-1}} \pmod m$$

# 5.4 模重复平方方法

具体步骤

(0) 令a=1, 并将n写成二进制 $n = n_0 + n_1 2 + \cdots n_{k-1} 2^{k-1}$

其中, $n_i \in \{0, 1\}$, i=0, 1, . . . , k−1

(1) 如果$n_0$=1, 则计算 $a_0 \equiv a \cdot b \pmod m$ , 否则取$a_0$=a, 即计算 $a_0 \equiv a \cdot b^{n_0} \pmod m$ , 再计算 $b_1 \equiv b^2 \pmod m$

(2) 如果$n_1$=1, 则计算 $a_1 \equiv a_0 \cdot b_1 \pmod m$ , 否则取$a_1$=$a_0$, 即计算 $a_1 \equiv a_0 \cdot b_1^{n_1} \pmod m$ , 再计算 $b_2 \equiv b_1^2 \pmod m$

. . . . . .

(k−1) 如果$n_{k-2}$=1, 则计算 $a_{k-2} \equiv a_{k-3} \cdot b_{k-2} \pmod m$ , 否则取$a_{k-2}$=$a_{k-3}$, 即计算 $a_{k-2} \equiv a_{k-3} \cdot b_{k-2}^{n_{k-2}} \pmod m$, 再计算 $b_{k-1} \equiv b_{k-2}^2 \pmod m$

# 5.4 模重复平方方法

具体步骤

（k）如果$n_{k-1}=1$, 则计算$a_{k-1} \equiv a_{k-2} \cdot b_{k-1} \pmod m$ ，否则取$a_{k-1}=a_{k-2}$, 即计算 $a_{k-1} \equiv a_{k-2} \cdot b_{k-1}^{n_{k-1}} \pmod m$

最后$a_{k-1}$就是$b^n \pmod m$

问题：用RSA算法对明文“math”加解密。