中山大學
SUN YAT-SEN UNIVERSITY

Module I. Fundamentals of Information Security

Chapter 2
Cryptographic Techniques

Information Security: Theory & Applications

School of Data & Computer Science, Sun Yat-sen University

Outline

- **2.1 Cryptology Introduction**
  - Introduction
  - History
  - Concepts & Items
- **2.2 Symmetric Key Cryptographic Algorithms**
  - Introduction
  - Types & Modes
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)

Outline

- **2.3 Mathematical Foundations of Public-Key Cryptography**
  - Prime factorizations of integers
  - The *Euclidean* Algorithm
  - *Bézout*'s Theorem
  - Linear Congruence
  - The Extended_*Euclidean* Algorithm
  - The Chinese Remainder Theorem
  - *Euler*'s φ function
  - *Euler*'s Theorem
  - *Fermat*'s Little Theorem

Outline

- **2.4 Asymmetric Key Cryptographic Algorithms**
  - Introduction
  - The RSA Algorithm
  - Digital Signatures
- **2.5 Hashing Algorithms**
  - Introduction
  - Message-Digest Algorithm (MD5)
- **2.6 Typical Applications**
  - MD5 and Passwords
  - AES and WiFi Protected Access
  - RSA and e-Business

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Definition**
  - A cryptographic system that uses two keys -- a *public key* known to everyone and a *private* or *secret key* known only to the recipient of the message.
- **Scenery**
  - When *Bob* wants to send a secure message to *Alice*, he uses *Alice*'s public key to encrypt the message. *Alice* then uses her private key to decrypt it.

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Property**
  - It is important that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is virtually impossible to deduce the private key from the known public key.

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- 一个经典的非对称加密信息传输情景
  - *Bob* 需要通过不可靠的传信人 *Tracy* 送一个信物给 *Alice*。
    1. *Bob* 将物品放进一个结实的箱子里，加上一把锁 L_1，只有 *Bob* 有 L_1 的钥匙 K_1；
    2. *Tracy* 将用 L_1 锁好的箱子送给 *Alice*；
    3. *Alice* 给箱子再加上一把锁 L_2，只有 Alice 有 L_2 的钥匙 K_2；
    4. *Tracy* 将用 L_1 和 L_2 锁好的箱子送回给 *Bob*；
    5. *Bob* 用 K1 打开 L_1，箱子上留下了*Alice* 的锁 L_2；
    6. *Tracy* 将用 L_2 锁好的箱子送给 *Alice*；
    7. *Alice* 用 K_2 打开 L_2，取得物品。
  - 注意到：
    - ✧ (1) K_1 和 K_2 不同；(2) 没有发生钥匙的传递，也即 K_1 和 K_2 分别是 *Bob* 和 *Alice* 的私钥；(3) *Tracy* 走了3趟。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- 非对称加密算法/公钥密码学算法
  - 公钥密码学的起源
    - ✧ 1976年 *Whitfield Diffie* 和 *Matin Hellman* 发表了 “New Directions in Cryptography” (密码学的新方向)，奠定了公钥密码学的基础。
    - ✧ 公钥密码技术是二十世纪最伟大的思想之一，它改变了密钥分发的方式，并可以广泛用于数字签名和身份认证服务。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- 非对称加密算法/公钥密码学算法
  - 公钥密码学的基本思想
    - ✧ 非对称加密算法需要两个密钥：公开密钥 (public-key) 和私有密钥 (private-key)，因此也称公钥密码学算法。公开密钥与私有密钥是严格对应的，如果用一个公开密钥对数据进行加密，就只有用其对应的私有密钥才能解密；如果用一个私有密钥对数据进行加密，那么也只有用其对应的公开密钥才能解密。
    - ✧ 算法的加密过程和解密过程使用的是两个不同的密钥，所以称之为非对称加密算法。
    - ✧ 非对称加密算法具有比较好的保密性，它消除了最终用户交换密钥的需要。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- 非对称加密算法/公钥密码学算法
  - 公钥密码学算法的基本过程
    - ✧ 甲方生成一对密钥，将其中的一把作为私有密钥严密收藏，另一把作为公开密钥向其它方公开；
    - ✧ 乙方使用甲方的公开密钥对明文信息进行加密，将得到的密文(通过不安全渠道) 发送给甲方；
    - ✧ 甲方用自己的私有密钥对密文解密，得到原始明文信息。
    - ✧ 另一方面，甲方也可以使用自己的私有密钥对明文信息进行加密，将密文发送给乙方；乙方使用甲方发布的公开密钥对密文解密，得到原始明文信息。这个过程也称为数字签名。

2.4 Asymmetric Key Crypt. Algorithms

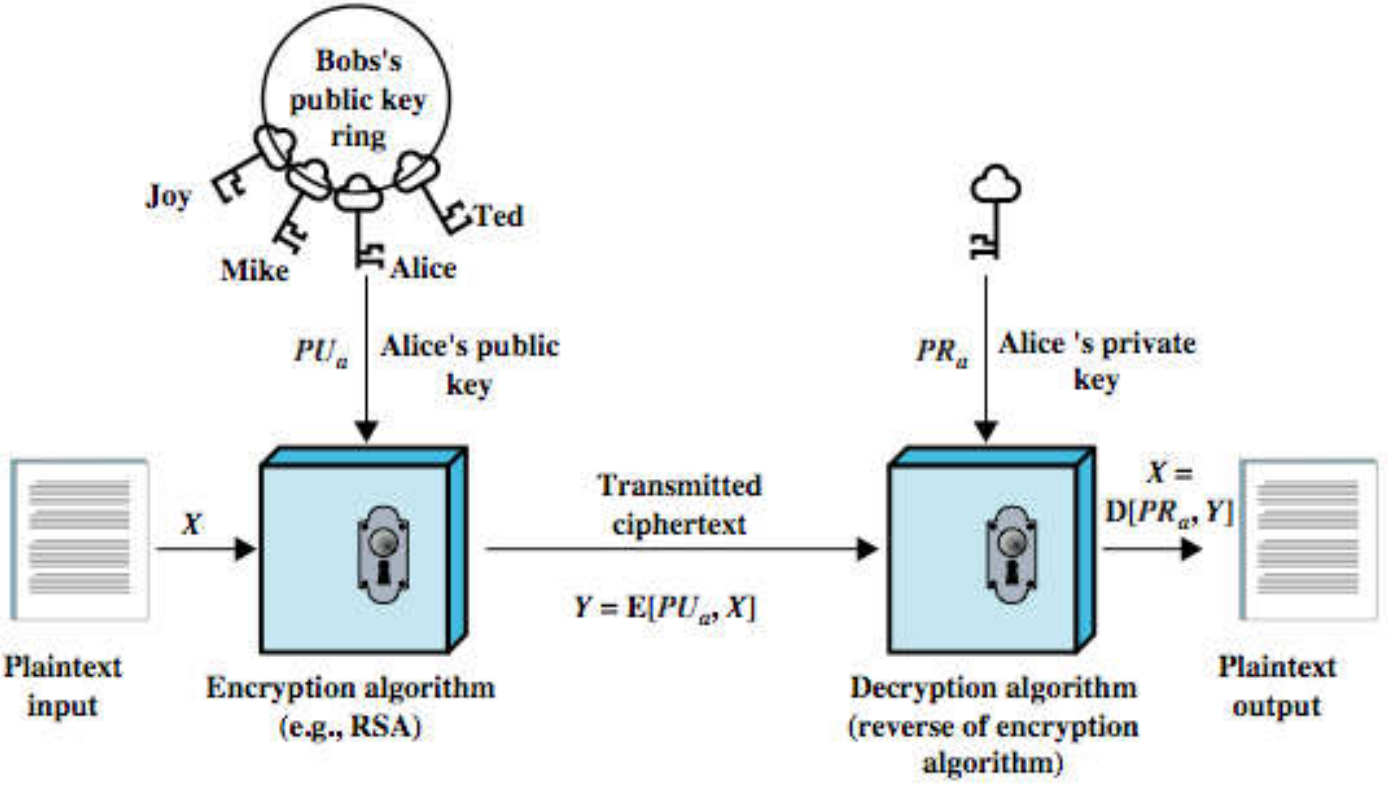
2.4.1 Introduction

- 非对称加密算法/公钥密码学算法
  - 公钥密码学算法的安全性
    - ✧ 非对称密码体制的安全性依赖于其算法与密钥。算法的复杂程度以及密钥长度的增加使得非对称密码体制的加密解密速度相对于对称密码体制要慢得多。
    - ✧ 对称密码体制中只有一种密钥，并且是非公开的，如果要解密就得让对方共享密钥，共享方的可靠性和密钥传递的安全性对加密安全性有重大影响。而非对称密钥体制是双钥制，不需要像对称密码那样传输对方的私有密钥，从而大大增强了安全性。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

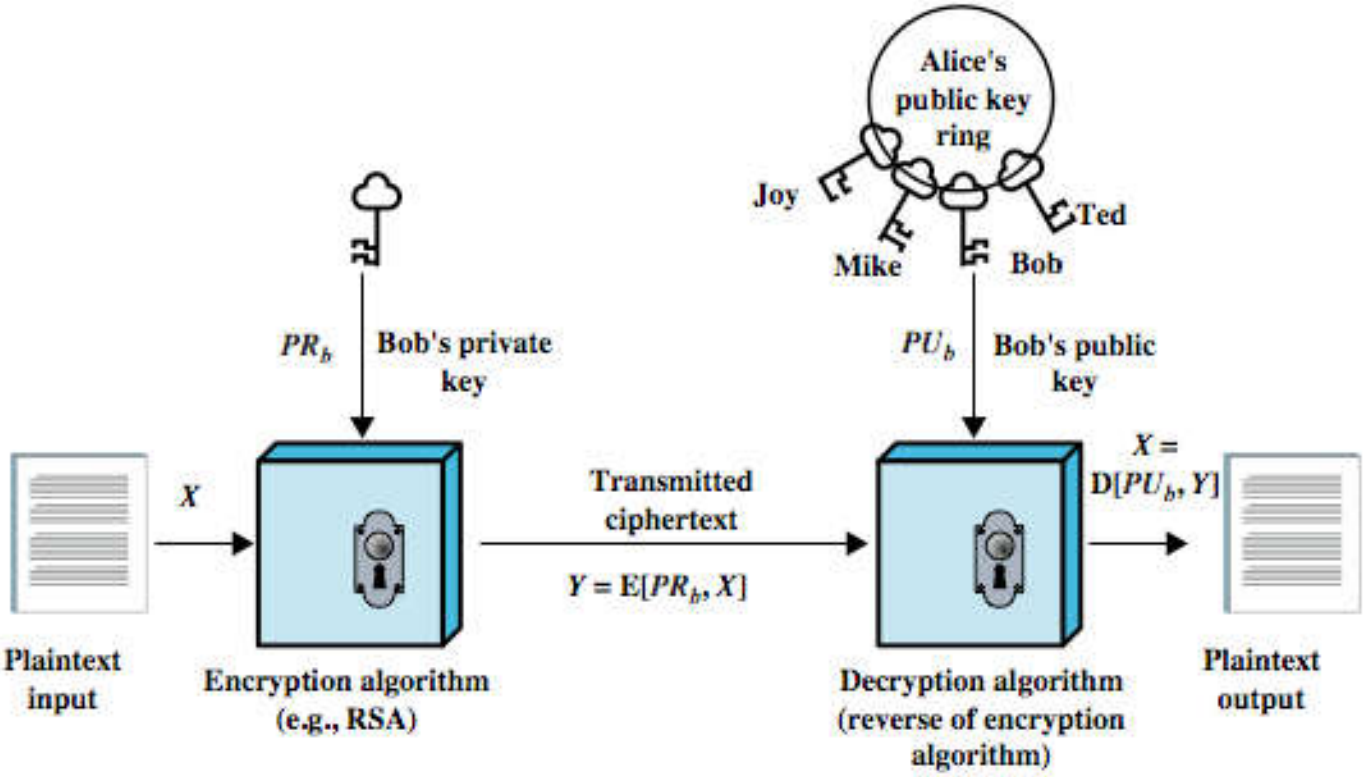
- **Asymmetric Key Cryptographic Encryption**
  - For confidentiality 用于加密



2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Authentication**
  - For Authentication / data integrity 用于认证或完整性验证



2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Requirements**
  - computationally easy to create key pairs
  - computationally easy for sender knowing public key to encrypt messages
  - computationally easy for receiver knowing private key to decrypt ciphertext
  - computationally infeasible for opponent to determine private key from public key
  - computationally infeasible for opponent to otherwise recover original message
  - useful if either key can be used for each role

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Requirements**
  - 构造一个公钥密码系统的要求
    - ✧ 产生一对密钥是计算可行的
    - ✧ 发送方利用公钥和明文，产生密文是计算可行的
    - ✧ 接收方利用私钥和密文，产生明文是计算可行的
    - ✧ 对于攻击者，利用公钥来推断私钥是计算不可行的
    - ✧ 已知公钥和密文，恢复明文是计算不可行的
    - ✧ (可选) 加密和解密的顺序可交换

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Requirements**
  - 一些讨论
    - 公钥和私钥必须相关，而且从公钥到私钥不可推断
      - 必须要找到一个难题，从一个方向走是容易的，从另一个方向走是困难的；然后把这个困难问题跟加解密结合起来
      - 找到一个困难问题不一定能产生一个保密性很好的密码系统吗，还需要有适当的构造方法。
    - 计算复杂性理论的指导
      - 计算可行和计算不可行的界限。
      - 计算复杂性通常针对一个孤立的问题进行研究，并主要考虑最坏的情形。
      - 公钥密码学往往需要考虑一些相关的问题，比如密码分析还需要考虑已知明文、选择明文等相关的情形。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 背包问题
    - ✧ 背包问题：有一个容积为 S 的背包和体积分别为 $a_1, \ldots, a_n$ 的 n 个物品，已知从这些物品中选出若干个刚好能够装满这个背包，求一个选择的方案。
    - ✧ 背包问题：有一个容积为 S 的空水桶和容积分别为 $a_1, \ldots, a_n$ 的 n 个装满水的杯子，已知从这些杯子中选出若干个，它们装的水刚好能够倒满水桶，求一个选择的方案。
    - ✧ 一般的背包问题是一个 NP 完全问题，解决问题所需要的时间与 n 呈指数增长。
      - ◌ 相当于含有 n 个原子的逻辑表达式的解释。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 背包问题
    - 0-1 背包问题：给定一个正整数 S 和一个 $A = (a_1, \ldots, a_n)$，其中 a_i 是正整数，求满足方程

      $$S = \sum a_i x_i \ (i = 1 .. n).$$

      的二进制向量 $X = (x_1, \ldots, x_n)$。
      - $A = (a_1, \ldots, a_n)$ 称为背包向量。
      - $x_1 \in \{0, 1\}$.
      - 问题有可能无解。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 背包问题用于公钥密码学
    - 背包向量的结构特征决定了两类背包问题：
      - 简单背包：可以在线性时间内求解，也称易解背包；
      - 难解背包：不一定能在线性时间内求解的非简单背包，即一般背包。
    - 基于背包问题的公钥密码系统的设计
      - 以 X 为明文、S 为密文，设计一个易解的背包向量作为私钥，再把易解的背包问题转换成难解的背包问题，将构造得到的难解背包问题的背包向量作为公开密钥。
    - 背包问题用于公钥密码学，构成了第一个公钥密码系统。但在实践过程中，大多数的背包方案已被破解，或者被证明存在缺陷。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 易解的背包问题—超递增背包
    - ✧ 设有背包 $A = (a_1, \ldots, a_n)$。如果对所有的 i, i = 1, …, n 都有

      $a_i > \sum a_j$ (j = 1, …, i-1).

      这类背包称为超递增背包，是一类简单背包。
    - ✧ 超递增背包的求解
      - ◌ 已知密文正整数 S 和作为解密密钥的超递增背包 $A = (a_1, \ldots, a_n)$，求明文 $X = (x_1, \ldots, x_n)$。

        (1) i = n.

        (2) If i = 0 failed.

        (3) If $S > a_i$ then $x_i = 1$, $S = S - a_i$; else $x_i = 0$. i = i - 1.

        (4) If $S = 0$ return ; else goto (2).

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 易解的背包问题—超递增背包
    - ✧ 超递增背包的求解
      - ◌ 例：简单背包 {2, 3, 6, 13, 27, 52}，求解 S=70 的背包问题。

i	a	S	x
6	52	70	1
5	27	18	0
4	13	18	1
3	6	5	0
2	3	5	1
1	2	2	1
		0	

      - ◌ 结果为 $X = (x_1, x_2, x_3, x_4, x_5, x_6) = (1, 1, 0, 1, 0, 1)$

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 基于背包问题的公钥密码系统—MH 公钥密码算法
    - *Ralph Merkle, Martin Hellman*, 1978
      - 1982年 *Adi Shamir* 破解了使用一次循环的 MH 背包密码。
      - MH 背包算法始终未能付诸实用。
    - 私钥：选择一个超递增背包 $A = (a_1, \ldots, a_n)$ 做为私钥。
    - 公钥：基于私钥背包 $A = (a_1, \ldots, a_n)$ 产生相应的公钥。
      - 选择一个整数 $m > \sum a_i \, (i = 1, \ldots, n)$；
      - 选择一个与 m 互素的整数 w，构造

        $$a_i' = wa_i \pmod{m}, \; i = 1, \ldots, n.$$

        得到一个非超递增背包 $A' = (a_1', \ldots, a_n')$；这里的 a_i' 是伪随机分布的；
      - 把 $A' = (a_1', \ldots, a_n')$ 作为公钥发布。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Knapsack Problem and MH Algorithm**
  - 基于背包问题的公钥密码系统—MH 公钥密码算法
    - ✧ 加密
      - ◌ 将明文分组，设为长度为 n 的二进制块 $X = (x_1, \ldots, x_n)$。
      - ◌ 利用公钥 $A' = (a_1', \ldots, a_n')$ 将明文变为密文 S

$$S = E(X) = \sum a_i' x_i \ (i = 1 \ldots n)$$

    - ✧ 解密
      - ◌ 先计算 $S' = w^{-1} S \bmod m$，其中 w^{-1} 是 w 的模 m 逆元。
      - ◌ 再利用私钥 $A = (a_1, \ldots, a_n)$ 求解简单背包问题

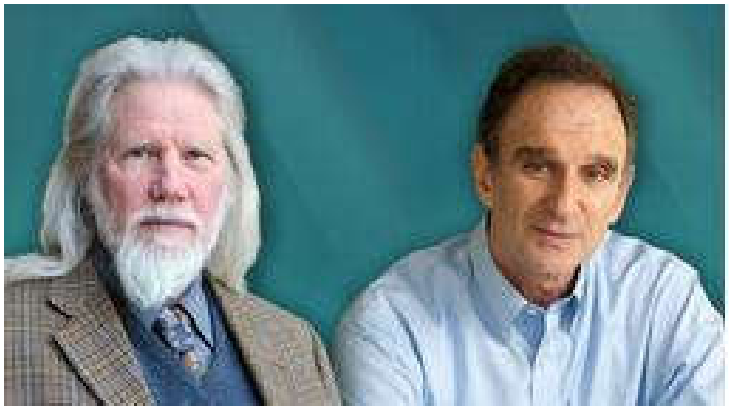
$$S' = \sum a_i x_i \ (i = 1 \ldots n)$$

      得到的解 $X = (x_1, \ldots, x_n)$ 就是恢复的明文。
    - ✧ 公钥和私钥配对才能进行加解密，否则构造的背包问题无解。

2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

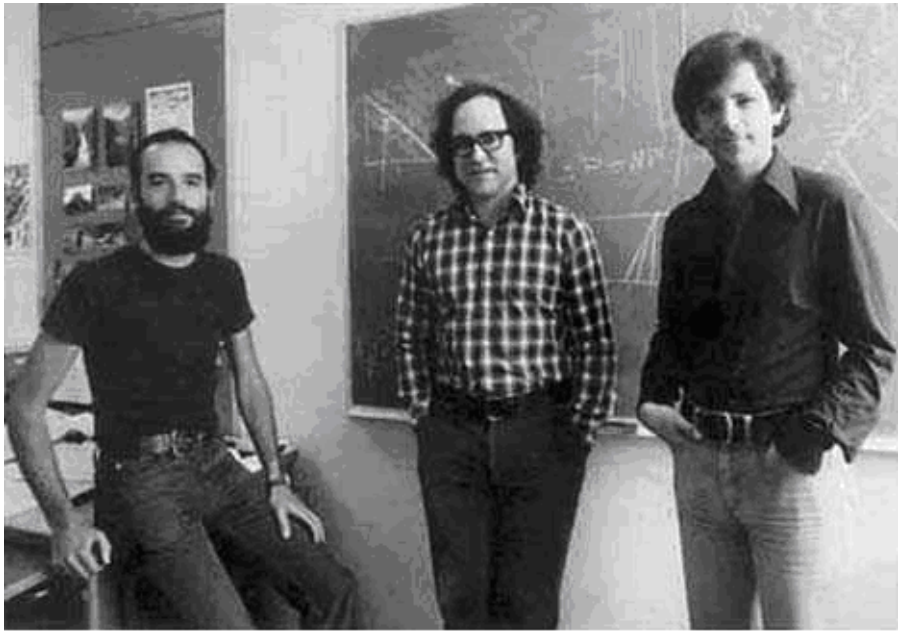
- **Asymmetric Key Cryptographic Algorithms**
  - *Diffie-Hellman* Key Exchange Algorithm
    - ✧ Developed in 1976 by *Whitfield Diffie* and *Martin Hellman*
    - ✧ Only allows exchange of a secret key
    - ✧ *Diffie-Hellman* 密钥交换算法：加密和解密可以使用不同的规则，只要这两种规则之间存在某种对应关系，即可在不直接传递密钥的情况下，完成解密，回避了直接传递密钥过程的风险。



2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Algorithms**
  - RSA
    - ✧ Developed in 1977 by *Rivest, Shamir* and *Adleman*
    - ✧ Only widely accepted public-key encryption algorithm
    - ✧ Given tech advances need 1024+ bit keys



2.4 Asymmetric Key Crypt. Algorithms

2.4.1 Introduction

- **Asymmetric Key Cryptographic Algorithms**
  - Digital Signature Standard (DSS)
    - ✧ Provides only a digital signature function with SHA-1
  - Elliptic Curve Cryptography (ECC)
    - ✧ New, security like RSA, but with much smaller keys

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- **RSA 算法**
  - RSA 算法的起源
    - ✧ RSA 算法在1977年由 MIT 的 *Ron Rivest*、*Adi Shamir* 和 *Leonard Adleman* 一起提出，并以他们三人姓氏开头字母命名，是一种获得广泛使用的非对称加密算法。
    - ✧ 1983年麻省理工学院在美国为 RSA 算法申请了专利。这个专利2000年9月21日失效。由于该算法在申请专利前就已经发表，在世界上大多数其它地区这个专利权不被承认。

2.4.2 RSA 算法

- **RSA 算法**
  - RSA 算法的安全性概述
    - ✧ 对极大整数进行因数分解的难度 (*The Factoring Problem*) 决定了 RSA 算法的可靠性。换言之，对一个极大整数做因数分解愈困难，RSA 算法就愈可靠。假如有人找到一种快速因数分解的算法的话，那么用 RSA 加密的信息的可靠性就肯定会极度下降。目前看来找到这样的算法的可能性非常小。
    - ✧ 目前还没有可靠的攻击 RSA 算法的方式。短的 RSA 钥匙可能被强力方式解破 (比如 768-bit，即232个十进制位以下的整数)。只要其钥匙的长度足够长 (比如1024-15360-bit)，用 RSA 加密的信息看来很难被破解。
    - ✧ 在分布式计算技术和量子计算理论日趋成熟的今天，RSA 加密的安全性受到了挑战。

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- **How RSA Works**
  - Generating of Public key and Private key
    - Select two different, big primes p and q. Let $N=pq$.
    - Compute $\varphi(N)$ By *Euler*'s φ function:

      $$\varphi(N) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1).$$

    - Select an integer e which is less than and prim to $\varphi(N)$
    - Compute d by： $de \equiv 1 \pmod{\varphi(N)}$.
      - d is an inverse of e modulo $\varphi(N)$.
    - Destroy p and q (when N is big enough, it will be extremely difficult to find p and q from N).
    - (e, N) is used as the public key, and (d, N) the private key. *Alice* will send her (e, N) to *Bob*, but keep her (d, N) in secret.

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- **How RSA Works**
  - Encryption
    - ✧ Suppose that *Bob* want to send *Alice* a message M. *Bob* has *Alice*'s public key (e, N) in hand. Firstly he transforms M to an integer n less than N in a presumed method. By applying the key (e, N), n is encrypted to c:

      $$c = n^e \bmod N.$$

      Then *Bob* can send c to Alice through an unsafe channel.
  - Decryption
    - ✧ By applying her private (d, N), *Alice* can decode n' from c by

      $$n' = c^d \bmod N.$$

      and recover the M from n'.

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- **How RSA Works**
  - Principle of RSA
    - ✧ By *Euler's Theorem* on φ function, given two primes p and q satisfied $N=pq$ and one integer n satisfied $0<n<N$, we have

      $n^{k\varphi(N)+1} \equiv n \pmod{N}$.

      In another hand, d was generated from

      $de \equiv 1 \pmod{\varphi(N)}$, or $de = k\varphi(N)+1$,

      so $n^{de} \equiv n \pmod{N}$.

      Now we have $c = n^e \bmod N$, $n' = c^d \bmod N$.

      $n' = c^d \bmod N \equiv (n^e)^d \bmod N$

      $= n^{ed} \bmod N \equiv n \bmod N$.
    - ✧ In practice, RSA encryption is combined with zero padding, salt, and message hash functions to securely transmit messages.

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- RSA 算法原理
  - 公钥和私钥的产生
    - ✧ 选择两个不同的大素数 p 和 q，计算 $N=pq$。
    - ✧ 引用欧拉 φ 函数，小于 N 且与 N 互素的正整数个数为

      $$\varphi(N) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1).$$

    - ✧ 选择一个整数 e 与 $\varphi(N)$ 互素，并且 e 小于 $\varphi(N)$。
    - ✧ 求一个满足下列同余式的 d：

      $$ed \equiv 1 \pmod{\varphi(N)}.$$

      - ◌ d 就是 e 的模 $\varphi(N)$ 逆元，可以应用扩展欧几里德算法得到。
    - ✧ 将 p 和 q 的记录销毁 (当 N 足够大时，寻找 p, q 将极其困难)
    - ✧ 以 (e, N) 作为公钥，(d, N) 作为私钥。假设 *Alice* 想通过一个不可靠的媒体接收 *Bob* 的一条私人讯息，*Alice* 将她的公钥 (e, N) 公开给 *Bob*，而将她的私钥 (d, N) 严密收藏起来。

2.4.2 RSA 算法

- RSA 算法原理
  - 加密消息
    - ✧ 现在 *Bob* 想给 *Alice* 送一个消息 M，他知道 *Alice* 产生的公钥 (e, N)。
    - ✧ *Bob* 使用事先与 *Alice* 约好的格式将 M 转换为一个小于 N 的整数 n，比如他可以将每一个字符转换为其数字形式的 Unicode 码，然后将这些数字连在一起组成整数 n (假如他的信息非常长的话，他可以将这个信息分为几段，逐段加密)。
    - ✧ *Bob* 引用公钥 (e, N)，利用下面的同余式，将 n 加密为 c：

      $$n^e = c \pmod N, \text{ 即 } c = n^e \bmod N$$

    - ✧ *Bob* 算出 c 后可以将它经公开媒体传递给 *Alice*。

2.4.2 RSA 算法

- RSA 算法原理
  - 解密消息
    - ✧ *Alice* 得到 *Bob* 的消息 c 后利用她的私钥 (d, N) 解码。
    - ✧ *Alice* 利用以下的同余式将 c 转换为 n'：

      $$c^d = n' \pmod{N}, \text{ or } n' = c^d \bmod N$$

    - ✧ *Alice* 得到的 n' 就是 *Bob* 的 n，因此可以将原来的信息 M 准确复原。

2.4 Asymmetric Key Crypt. Algorithms

2.4.2 RSA 算法

- RSA 算法原理
  - 解码原理
    - ✧ 欧拉定理
      - ◌ 对互素的 a 和 m，有 $a^{\varphi(m)} \equiv 1 \pmod{m}$
      - ◌ m 是素数 p 时，$\varphi(p)=p-1$，即为费马小定理 $a^{p-1} \equiv 1 \pmod{p}$
    - ✧ 推论 (证略)
      - ◌ 给定满足 $N = pq$ 的两个不同素数 p 和 q 以及满足 $0<n<N$ 的整数 n，有 $n^{k\varphi(N)+1} \equiv n \pmod{N}$
    - ✧ 由：$ed \equiv 1 \pmod{\varphi(N)}$, 即 $ed = k\varphi(N)+1$, 故

      $n^{ed} \equiv n \pmod{N}$

      现在有：$c = n^e \bmod N$, $n' = c^d \bmod N$

      故：$n' = c^d \bmod N \equiv (n^e)^d \bmod N$

      $= n^{ed} \bmod N \equiv n \bmod N.$

2.4.2 RSA 算法

- RSA 算法的可靠性
  - 算法的敏感数据分析
    - $N = pq$
    - $\varphi(N) = (p-1)(q-1)$
    - p, q 被销毁
    - (e, N) 作为公钥被公开
    - d 是 e 的模 $\varphi(N)$ 逆元：$de \equiv 1 \pmod{\varphi(N)}$
  - 结论
    - 私钥被破解即 (d, N) 被推算出来的必要条件是 N 被因数分解。
    - 对极大整数做因数分解的难度决定了 RSA 算法的可靠性。对一极大整数做因数分解愈困难，RSA 算法愈可靠。
    - 极大整数的因数分解非常困难。目前除了暴力破解，还没有发现其它有效方法。

2.4.2 RSA 算法

- RSA 算法的可靠性
  - 针对算法的攻击
    - ✧ 基于数学的攻击
      - ◌ 分解 $N = pq \Rightarrow \varphi(N) = (p-1)(q-1) \Rightarrow d = e^{-1} \pmod{\varphi(N)}$。
      - ◌ 不求出 p, q，直接求 $\varphi(N) \Rightarrow d = e^{-1} \pmod{\varphi(N)}$。
      - ◌ 不求出 $\varphi(N)$，直接计算 d。
    - ✧ 同模攻击 Common Modulus Attack
      - ◌ 两对密钥的模 N 相同。
    - ✧ 计时攻击 Timing Attack
      - ◌ 利用 CPU 处理某些特殊的模乘比较慢的规律来确定每一位指数。

2.4 Asymmetric Key Crypt. Algorithms

2.4.3 Example of RSA

- *Example.*
  - For continent, we select small p, q and e.
  - Suppose *Alice* want to sent a string "key" to *Bob* by RSA encryption.

    ① Compute (e, N) and (d, N)

    ✧ Let p=3, q=11, $N = pq = 3\times11 = 33$

    $f(N) = (p-1)(q-1) = 2\times10 = 20$

    ✧ We take e=3 (3 is prime to 20) and try to get d such that

    $ed \equiv 1 \bmod f(n)$, that is $3d \equiv 1 \bmod 20$

    ✧ We get d by trial: (or by the *Extended Euclidean Algorithm*)

2.4 Asymmetric Key Crypt. Algorithms

2.4.3 Example of RSA

- *Example.*

① Compute (e, N) and (d, N)

✧ $f(N) = (p-1)(q-1) = 2 \times 10 = 20$. Take $e=3$ and try d.

✧ From the table, we find that when $d=\mathbf{7}$, $ed \equiv 1 \bmod f(n)$ holds.

✧ So the public key and the private key are as

$K_U = (e, N) = (3, 33)$

and $K_R = (d, N) = (7, 33)$

d	$e \times d$	$(e \times d) \bmod f(N)$
1	3	3
2	6	6
3	9	9
4	12	12
5	15	15
6	18	18
7	**21**	**1**
8	24	4
9	27	7

2.4 Asymmetric Key Crypt. Algorithms

2.4.3 Example of RSA

- *Example.*

② Digitization

a	b	c	d	e	f	g	h	i	j	k	l	m
01	02	03	04	05	06	07	08	09	10	11	12	13
n	o	p	q	r	d	t	u	v	e	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25	26

Code Table for message transcoding

✧ By the Code Table, the code values of the string "key" are

M_1=11, M_2=05 and M_3=25

✧ Note that the max value of any code is 26 (it can not over N=33).

2.4 Asymmetric Key Crypt. Algorithms

2.4.3 Example of RSA

- *Example.*

③ Encryption

✧ By $M^e = C \pmod N$ we have $C = M^e \pmod N$

That is $C_1 = 11^3 \bmod 33 = 11$

$C_2 = 5^3 \bmod 33 = 26$

$C_3 = 25^3 \bmod 33 = 16$

④ Decryption

✧ By $C^d = M \pmod N$ we have $M = C^d \pmod N$

That is $M_1 = 11^7 \bmod 33 = 11$

$M_2 = 26^7 \bmod 33 = 5$

$M_3 = 16^7 \bmod 33 = 25$

⑤ Message recovery

✧ By checking the Code Table, the string "key" is recovered.

2.4 Asymmetric Key Crypt. Algorithms

2.4.4 Applications of RSA

- **Digital Signatures**
  - The RSA algorithm can be repurposed for digitally signing a message *m*
    - ✧ Public key $K_U = (e, N)$, private key $K_R = (d, N)$
    - ✧ Signing: Compute
      $$s = \text{Sign}(K_R, m) \equiv m^d \pmod{N}$$
    - ✧ Verification: Compute $m' = s^e \bmod N$.
    - ✧ If $m' \equiv m \pmod{N}$, the signature is valid.

2.4 Asymmetric Key Crypt. Algorithms

2.4.4 Applications of RSA

- **Digital Signatures**
  - RSA 签名原理
    - ✧ RSA 可以用来为一个消息署名。假如 *Alice* 想给 *Bob* 传递一个署名的消息 *M*，她可以为 *M* 计算一个散列值 *h* (消息摘要 Message Digest 算法，比如 MD5)，然后用她的私钥 K_{AR} 加密 *h* 并将得到的 *H* 作为“署名”加在 *M* 的后面。*H* 只有用 *Alice* 的公钥 K_{AU} 才能被正确还原成 *h*。*Bob* 获得 *M* 及 *H* 后用 K_{AU} 解密 *H* 得到 *h'*，并采用相同的 MD 算法从 *M* 得到 *h*，如果 *h* = *h'*，那么他就可以认为发信人持有 *Alice* 的私钥 (据此认为发信人就是 *Alice*)，以及这个消息在传播路径上没有被篡改，从而得到完整性和不可抵赖性校验。

End of Chapter 2.4