# 5. Application

## Public-Key Cryptosystems

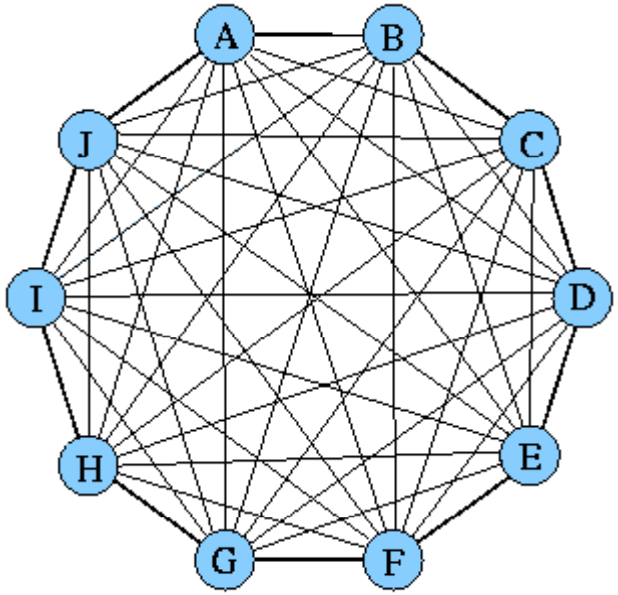
# Secret key cryptosystems

Parties want to exchange secure messages. Encryption is done with a key.

## Traditional encryption of messages

### Disadvantages:

1. Prior to transmission of the message, the key *k* has to be exchanged between the parties A und B.
2. For encryption of messages between *n* parties, $n(n-1)/2$ keys are required.




$\Theta(n^2)$

Large number of keys need exchange via a safe channel.

# Secret key encryption systems

**Advantage:**

Encryption and decryption are fast.

# Electronic security services

**Guarantees:**

- Confidentiality of the transmission
- Integrity of the data
- Authenticity of the sender
- Liability of the transmission

# Public-key cryptosystems

Framework

Diffie and Hellman (1976)

**Idea:** Each participant *A* holds two keys:

1. A public key $P_A$, accessible to all other participants.
2. A private key $S_A$ that is kept secret.

# Public-key cryptosystems

$D$ = Set of all valid messages,
e.g. set of all bitstrings of finite length

$D = \{0,1\}^*$

$$P_A(\ ), S_A(\ ): D \xrightarrow{1\text{–}1} D$$

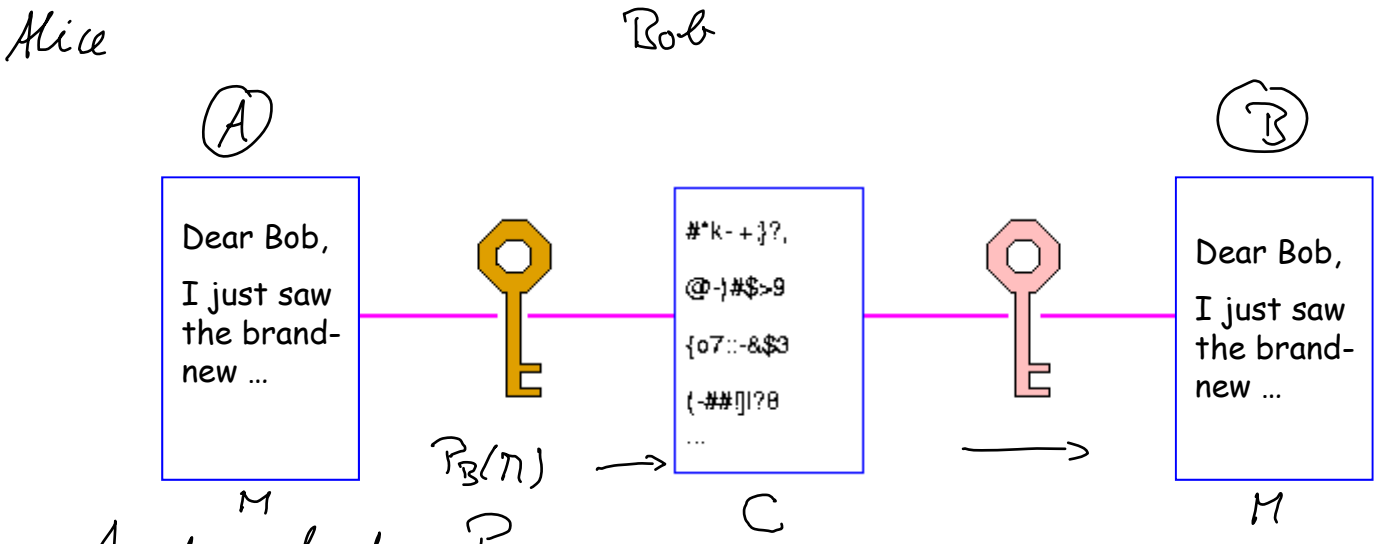
**Three constraints:**

1. $P_A()$, $S_A()$ efficiently computable — polynomial time algorithms
2. $S_A(P_A(M)) = M$ and $P_A(S_A(M)) = M$ — correctness
3. $S_A()$ is not computable from $P_A()$ (with realistic effort) — security

RSA

# Encryption in a public-key system

*A* sends a message *M* to *B*:




A downloads $P_B$

A applies $P_B(M) = C$

A sends C to B via insecure channel

B applies $S_B(C) = S_B(P_B(M)) = M$

# Encryption in a public key system

1. *A* receives *B`s* public key $P_B$ from a public directory or directly from *B*.

2. *A* computes the ciphertext $C = P_B(M)$ and sends it to *B.*

3. After receiving message *C*, *B* decrypts the message using his private key $S_B$: $M = S_B(C)$

# Generating a digital signature

With a digital signature we want to be able to prove that we are indeed the person we claim to be.

*A* sends a digitally signed message *M´* to *B*:

1. *A* computes the digital signature $\sigma$ for *M´* using his private key:

   $\sigma = S_A(M')$

   $M'$ = "I am Alice"

2. *A* sends the pair ($M'$, $\sigma$) to *B.*

3. After receiving ($M'$, $\sigma$), *B* checks the digital signature:

   $P_A(\sigma) = M'$ $\qquad P_A(\sigma) = P_A(S_A(M')) = M'$

Anybody is able to check $\sigma$ using $P_A$ (e.g. for bank checks).

# RSA cryptosystem

Received Turing Award

R. Rivest, A. Shamir, L. Adleman

Generating the public and private keys:

1. Select at random two large primes *p* and *q* of *l+1* bits ($l \geq 500$).

2. Compute $n = pq$.

3. Select a natural number *e* is that is relatively prime to $(p-1)(q-1)$.
   $\gcd(e, (p-1)\cdot(q-1)) = 1$

4. Compute $d = e^{-1}$ multiplicative inverse
   $d*e \equiv 1 \pmod{(p-1)(q-1)}$

# RSA cryptosystem

5. Publish $P = (e, n)$ as public key.

6. Keep $S = (d, n)$ as private key.

Split the (binary coded) message into blocks of length $2l$.
Interpret each block $M$ as a binary number: $0 \le M < 2^{2l}$ $M < n$

public key private key

$P(M) = M^e \bmod n$ $S(C) = C^d \bmod n$

Correctness: $P(S(M)) = M$ and $S(P(M)) = M$

# Correctness of RSA

$P(S(M)) = M$ to show

$P(S(M)) = (M^d \bmod n)^e \bmod n = M^{ed} \bmod n$

$d \cdot e = 1 \bmod (p-1)(q-1) \Leftarrow$

$M^{ed} \bmod n = M^{1 + k \cdot (p-1)(q-1)} \bmod n$

$= \underbrace{(M \bmod n)}_{M} \cdot \underbrace{(M^{k \cdot (p-1)(q-1)} \bmod n)}_{\equiv 1}$

$= M$

$S(P(M)) = M$ analogously

(1) $M^{p-1} \bmod p = 1$ (Fermat) $\Rightarrow$ $M^{(p-1) \cdot k \cdot (q-1)} \bmod n = \prod_{i=1}^{k \cdot (q-1)} M^{(p-1)} \bmod n$

(2) $M^{q-1} \bmod q = 1$ $\Rightarrow$ $M^{(q-1) \cdot k \cdot (q-1)} \bmod n = 1^{k \cdot (q-1)} = 1$

$M^{k \cdot (p-1)(q-1)}$

$= 1 + \underbrace{l_1 \cdot p}_{q \text{ is a factor in } l_1} = 1 + \underbrace{l_2 \cdot q}_{p \text{ is a factor in } l_2} = 1 + l_3 \cdot p \cdot q = 1 \bmod pq$

$= 1 \bmod n$

# Multiplicative inverse

$d : \quad d \cdot e = 1 \mod (p-1)(q-1)$

**Theorem:** (GCD recursion theorem)

For any numbers *a* and *b* with $b>0$:

$$\mathrm{GCD}(a,b) = \mathrm{GCD}(b, a \bmod b).$$

$a = \lfloor \frac{a}{b} \rfloor \cdot b + a \bmod b \quad \Rightarrow \quad a \bmod b = a - \lfloor \frac{a}{b} \rfloor \cdot b$

$x \mid a \text{ and } x \mid b \quad \Leftrightarrow \quad x \mid b \text{ and } x \mid a \bmod b \quad x \mid (a - \lfloor \frac{a}{b} \rfloor \cdot b)$

**Algorithm:** Euclid

**Input:** Two integers *a* and *b* with $b \geq 0$

**Output:** GCD(*a*,*b*)

**if** $b = 0$

**then return** *a*

**else return** Euclid(*b*, *a* mod *b*)

# Multiplicative inverse

**Algorithm:** extended-Euclid

**Input:** Two integers $a$ and $b$ with $b \geq 0$

**Output:** GCD($a$,$b$) and two integers $x$ and $y$ with

$$xa + yb = \text{GCD}(a,b)$$

**if** $b = 0$ **then return** $(a, 1, 0)$;

$(d, x', y') := \text{extended-Euclid}(b, a \bmod b)$;

$x := y'$; $y := x' - \lfloor a/b \rfloor y'$;

**return** $(d, x, y)$;

**Application:** $a = (p-1)(q-1)$, $b = e$

The algorithm returns numbers $x$ and $y$ with

$$x(p-1)(q-1) + ye = \text{GCD}((p-1)(q-1),e) = 1$$

$y \cdot e = 1 + k \cdot x(p-1)(q-1) = 1 \bmod (p-1)\cdot(q-1)$

$y = d$ multiplicative inverse