



# Theory of algorithms (12th lecture)

Pál Pusztai  
[pusztai@sze.hu](mailto:pusztai@sze.hu)

## Outline

- Number-theoretic algorithms
  - Powers of an element
  - Powers of an element modulo  $n$
- The RSA cryptosystem
  - Public-key cryptosystems
  - Encryption
  - Digital signature
  - A simple example of RSA
- Exercises



## Powers of an element

### ■ Exponentiation

**Problem:** Calculate  $a^b$  as less multiplication as possible where  $a$  and  $b$  are nonnegative integer.

**Solution:** A recursive algorithm that decreases the number of multiplication with repeated squaring.

For example:  $2^{10} = 2^5 \cdot 2^5$ ,  $2^5 = 2 \cdot 2^4$ ,  $2^4 = 2^2 \cdot 2^2$ ,  $2^2 = 2 \cdot 2$ .

EXPONENTIATION( $a, b$ )

```
1  if  $b == 0$ 
2      return 1
3  if  $b \bmod 2 == 0$ 
4       $c = \text{EXPONENTIATION}(a, b / 2)$ 
5      return  $c \cdot c$ 
6  return  $a \cdot \text{EXPONENTIATION}(a, b-1)$ 
```

**Efficiency:** The number of recursive calls thus the number of multiplication is  $O(\lg b)$ .



## Powers of an element modulo $n$

### ■ Modular exponentiation

The sequence of powers of an  $a \in \mathbf{Z}_n$  element  $a^0, a^1, a^2, a^3, \dots$  can be calculated modulo  $n$ , where  $a^0 \bmod n = 1$ , and the  $i$ th. value is  $a^i \bmod n$ .

**Remark:** Note that  $a^i \equiv a(a^{i-1} \bmod n) \pmod{n}$ .

$i$	0	1	2	3	4	5	6	7	
$2^i \bmod 5$	1	2	4	3	1	2	4	3	...
$3^i \bmod 5$	1	3	4	2	1	3	4	2	...

$i$	0	1	2	3	4	5	6	7	8	9	10	11	
$2^i \bmod 7$	1	2	4	1	2	4	1	2	4	1	2	4	...
$3^i \bmod 7$	1	3	2	6	4	5	1	3	2	6	4	5	...

### The powers of 2 and 3 modulo 5 and modulo 7

**Euler's theorem:** For any  $n > 1$ ,  $a^{\phi(n)} \equiv 1 \pmod{n}$  for all  $a \in \mathbf{Z}_n^*$ .

**Fermat's theorem:** If  $p$  is prime, then  $a^{p-1} \equiv 1 \pmod{p}$  for all  $a \in \mathbf{Z}_n^*$ .



## Powers of an element modulo $n$

### ■ Modular exponentiation

**Problem:** Calculate  $a^b \bmod n$ , where  $a$  and  $b$  are nonnegative integers and  $n$  is a positive integer.

**Solution:**

- Let  $\langle b_k, b_{k-1}, \dots, b_0 \rangle$  be the binary representation of  $b$  (that is, the binary representation is  $k+1$  bits long,  $b_k$  is the most significant bit, and  $b_0$  is the least significant bit).

- Using

$$a^{2^c} \bmod n = (a^{2^{c-1}})^2 \bmod n$$

$$a^{2^{c+1}} \bmod n = a (a^{2^c})^2 \bmod n$$

and

$$a^i \equiv a (a^{i-1} \bmod n) \pmod{n}$$

equations  $a^c \bmod n$  can be calculated with an iteration.



## Powers of an element modulo $n$

MODULAR-EXPONENTIATION( $a, b, n$ )

```
1   $c = 0$ 
2   $d = 1$ 
3  Let  $\langle b_k, b_{k-1}, \dots, b_0 \rangle$  be the binary representation of  $b$ 
4  for  $i = k$  downto 0
5       $c = 2c$ 
6       $d = (d \cdot d) \bmod n$ 
7      if  $b_i == 1$ 
8           $c = c + 1$ 
9           $d = (d \cdot a) \bmod n$ 
10 return  $d$ 
```

**Efficiency:** The total number of arithmetic operations is  $O(\lg b)$ .

**Remark:** The variable  $c$  is not really needed by the algorithm it may help the understanding.



## Powers of an element modulo $n$

$i$	5	4	3	2	1	0
$b_i$	1	1	0	0	1	0
$c$	1	3	6	12	25	50
$d$	2	3	4	1	2	4

### Computing $2^{50} \bmod 5$

$i$	9	8	7	6	5	4	3	2	1	0
$b_i$	1	0	0	0	1	1	0	0	0	0
$c$	1	2	4	8	17	35	70	140	280	560
$d$	7	49	157	526	160	241	298	166	67	1

### Computing $7^{560} \bmod 561$



## Exercises

- Give the result of the below given modular exponentiation.
  - $2^{15} \bmod 5$
  - $7^{35} \bmod 11$
- What values of  $d$  are calculated by the MODULAR-EXPONENTIATION with below given input data?
  - $a=2, b=15, n=5$
  - $a=7, b=35, n=11$





# The RSA cryptosystem

## ■ A public-key cryptosystem

It can be used for:

- sending encrypt messages,
- unforged digital signature.

### Properties:

Each participant has both a **public key** and a **secret key**. For example, in the RSA cryptosystem, each key consists of a pair of integers.

The participants Alice and Bob are traditionally used in cryptography examples; we denote their public and secret keys as  $P_A, S_A$  for Alice and  $P_B, S_B$  for Bob.

Let  $\mathcal{D}$  denote the set of permissible messages (for example the set of all finite-length bit sequences), and  $M \in \mathcal{D}$  an arbitrary message.

The public and secret keys for any participant are a „matched pair” in that they specify functions that are inverses of each other. That is,

$$M = S_A (P_A (M))$$

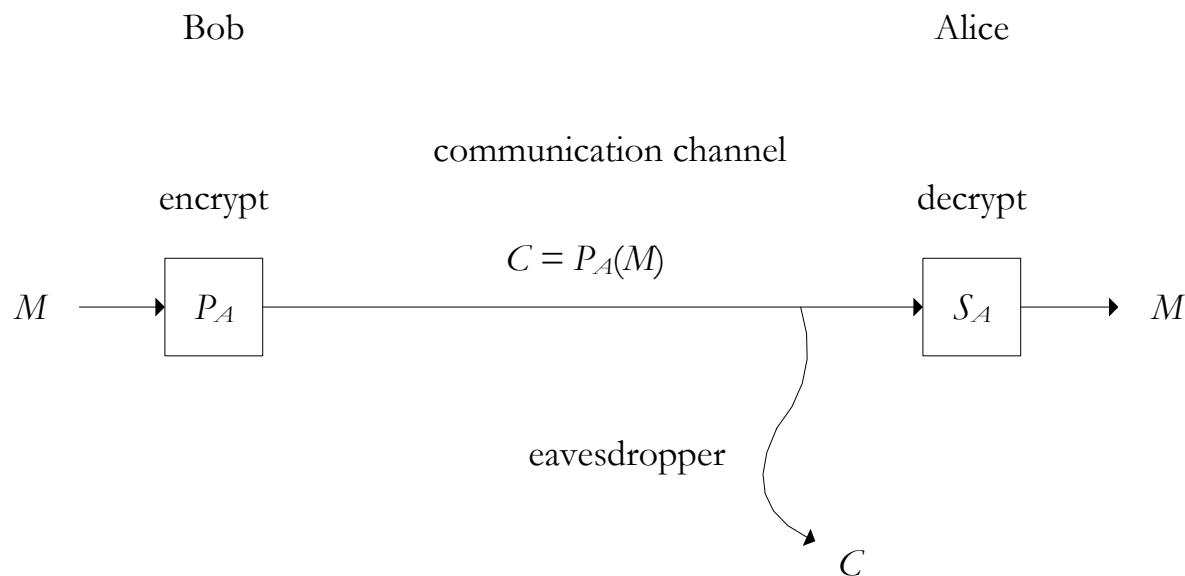
$$M = P_A (S_A (M))$$

for any message  $M \in \mathcal{D}$ .

Only Alice can be able to compute the function  $S_A$  in any practical amount of time (and Bob the function  $S_B$ ) even though everyone knows  $P_A$  and can compute  $P_A$  the inverse function to  $S_A$  efficiently.



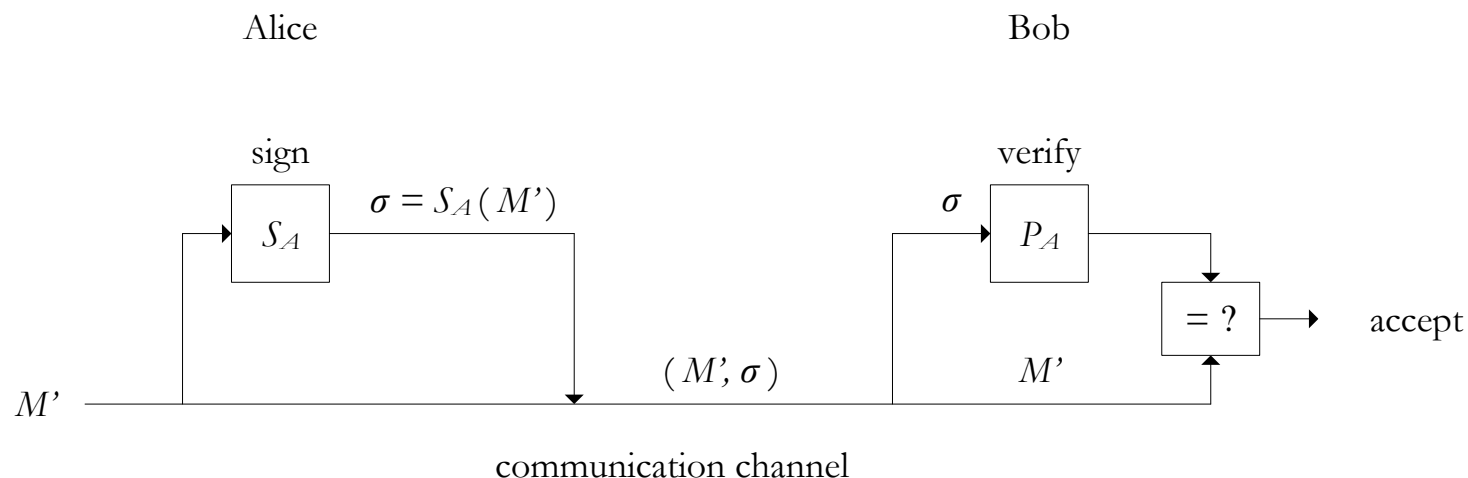
# The RSA cryptosystem



## Encryption in a public key system.

- Bob obtains Alice's public key  $P_A$  (for example from a public directory or directly from Alice).
- Bob encrypts the message  $M$  using Alice's public key  $P_A$  and transmits the resulting **ciphertext**  $C = P_A(M)$  over a communication channel to Alice. (An eavesdropper who captures the transmitted ciphertext gains no information about  $M$ .)
- Alice receives  $C$  and decrypts it applying her secret key  $S_A$  to retrieve the original message:  $S_A(C) = S_A(P_A(M)) = M$ .

# The RSA cryptosystem



## Digital signatures in a public-key system:

- Alice computes her **digital signature**  $\sigma$  for the message  $M'$  using her secret key  $S_A$  and the equation  $\sigma = S_A(M')$ .
- Alice sends the message/signature pair  $(M', \sigma)$  to Bob.
- Bob receives the  $(M', \sigma)$  pair and verifies it by using Alice's public key  $P_A$ . He checks the equation  $M' = P_A(\sigma)$ . If the equation holds, he accepts  $(M', \sigma)$  as a message that Alice has signed.

**Remarks:** A digital signature must be verifiable by anyone who has access to the signer's public key. A signed message is not necessarily encrypted; the message can be “in the clear” and not protected from disclosure. (Presumably,  $M'$  contains Alice's name, so Bob knows whose public key to use.)

# The RSA cryptosystem

## ■ Creating the public keys and the secret keys

In the **RSA public-key cryptosystem**, a participant creates his or her public and secret keys with the following procedure:

1. Select at random two large prime numbers  $p$  and  $q$  such that  $p \neq q$ . The primes  $p$  and  $q$  might be, say, 1024 bits each.
2. Compute  $n=pq$ .
3. Select a small odd integer  $e$  that is relatively prime to  $\Phi(n)$ , where  $\Phi(n) = (p-1)(q-1)$ .
4. Compute  $d$  as the multiplicative inverse of  $e$ , modulo  $\Phi(n)$ .
5. Publish the pair  $P = (e, n)$  as the participant's **RSA public key**.
6. Keep secret the pair  $S = (d, n)$  as the participant's **RSA secret key**.

### Remarks:

- Factoring large integers is easy  $\Rightarrow$  breaking the RSA cryptosystem is easy.
- Factoring large integers is hard  $\Rightarrow$  ? breaking the RSA cryptosystem is hard.



## The RSA cryptosystem

- Using the public and the secret keys

In the RSA public-key cryptosystem the domain  $\mathcal{D} = \mathbf{Z}_n = \{[a]_n : 0 \leq a \leq n-1\}$ .

To transform a message  $M$  associated with a public key  $P(e, n)$ , compute

$$P(M) = M^e \pmod{n}.$$

To transform a ciphertext  $C$  associated with a secret key  $S(d, n)$ , compute

$$S(C) = C^d \pmod{n}.$$

**Remark:** These equations apply to both encryption and signatures.



## A simple example of RSA

### ■ Creating the public keys and the secret keys

1. Let  $p = 2$  and  $q = 5$ .
2.  $n = pq = 10$ .
3. Let  $e = 3$ , as  $\Phi(n) = (p-1)(q-1) = 4$ , and  $\gcd(\Phi(n), e) = \gcd(4, 3) = 1$ .
4. Solving equation  $ed \equiv 1 \pmod{\Phi(n)}$  we get  $d = 3$ , as the multiplicative inverse of  $e$ , modulo  $\Phi(n)$ .
5. The public key is  $P = (e, n) = (3, 10)$  pair.
6. The secret key is  $S = (d, n) = (3, 10)$  pair.

**Remark:** The EXTENDED-EUCLID results  $\gcd(a, b) = \gcd(e, \Phi(n)) = \gcd(3, 4) = 1$ ,  $x = -1$ ,  $y = 1$  for the input data  $a = e$  and  $b = \Phi(n)$ . As  $\gcd(e, \Phi(n)) = ex + \Phi(n)y$ ,  $1 = 3 \cdot (-1) + 4 \cdot 1$ , thus  $ex \equiv 1 \pmod{\Phi(n)}$ . Since  $x = -1$  a negative number, thus we shift it with  $\Phi(n)$  to be positive, we get  $d = 3$ , as the multiplicative inverse of  $e$ .



## A simple example of RSA

- A possible way to implement encrypting

Let the message be as follow:

Text	'a'	'p'	'p'	'l'	'e'
Binary	01100001	01110000	01110000	01101100	01100101
Decimal	97	112	112	108	101

### A message to be encrypted

As  $n = 10$ , and  $2^3 < 10 < 2^4$ , thus encrypting is a conversion from 3 to 4 bits, and decrypting is a conversion from 4 to 3 bits.

Binary	011	000	010	111	000	001	110	000	011	011	000	110	010	1
Decimal	3	0	2	7	0	1	6	0	3	3	0	6	2	1

### A message to be encrypted separated groups of 3 bits



## A simple example of RSA

- A possible way to implement encrypting

Encrypting with equation  $P(M) = M^e \pmod{n}$ , where  $e = 3$ ,  $n = 10$ .

With MODULAR-EXPONENTIATION the values of  $P(M)$  can be computed for all possible message  $0 \leq M \leq n-1$  (though the values  $0 \leq M \leq 7$  will be used, since the 3 bits separation).

$M$	0	1	2	3	4	5	6	7
$P(M)$	0	1	8	7	4	5	6	3

### Encrypting

**Remark:** The last „3 bits” may be not 3 bits long, thus the message can be appended with the length of the last „3 bits” (in this example it is 1), so the correct length of the original message can be set after decrypting.

Message	3	0	2	7	0		6	2	1	1
Encrypted	7	0	8	3	0	...	6	8	1	1
Binary	0111	0000	1000	0011	0000	...	0110	1000	0001	0001

**The encrypted message (in 4 bits groups) with length completion**





## A simple example of RSA

- A possible way to implement decrypting

Decrypting with equation  $S(C) = C^d \pmod{n}$ , where  $d = 3$ ,  $n = 10$ .

$C$	0	1	2	3	4	5	6	7	8	9
$S(C)$	0	1	8	7	4	5	6	3	2	9

### Decrypting

Encrypted	7	0	8	3	0	1	6	0	7	7	0	6	8	1	1
Decrypted	3	0	2	7	0	1	6	0	3	3	0	6	2	1	1
Binary	011	000	010	111	000	001	110	000	011	011	000	110	010	1	

### After decrypting considering the length completion

**Remark:** If  $p$  and  $q$  primes have 100 decimal digits, then  $n=pq$  has 200 decimal digits, that is  $200 \lg 10 \approx 665$  binary digits, thus the message can be separated into such long (for example 512 bits = 64 bytes) groups.

## Exercises

- Suppose that we broke the public RSA key  $(3, 319)$ , so we know that  $p=11$  and  $q=29$ .
  - What is the encryption of the message  $M=100$ ?
  - What is the secret RSA key?

