

## 6.9 *The RSA cryptosystem*

In the *simple cryptosystem* Alice and Bob need to meet either in person or communicate by a perfectly secure channel in order to agree on the value of encryption and decryption keys.

It is not always convenient to arrange such a hand-off of the keys.

Another issue: private keys need to be renewed periodically, since they eventually become compromised after multiple uses.

## 6.9 The RSA cryptosystem

### *Public key cryptography*

Public key cryptosystems were first realized in the 1970s

Bob has an *encryption key* (*public key*) that he provides publicly so that anyone can use it to send him an encrypted message.  
Bob holds a matching *decryption key* (*private key*) that he keeps privately to decrypt messages.

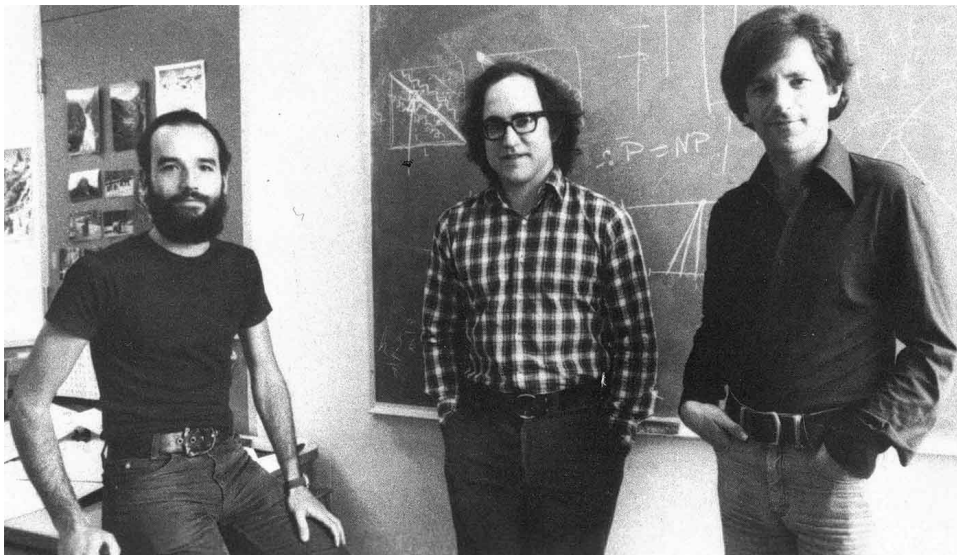
While anyone can use the public key to encrypt a message, the security of the scheme depends on the fact that it is difficult to decrypt the message without having the matching private decryption key.

## 6.9 The RSA cryptosystem

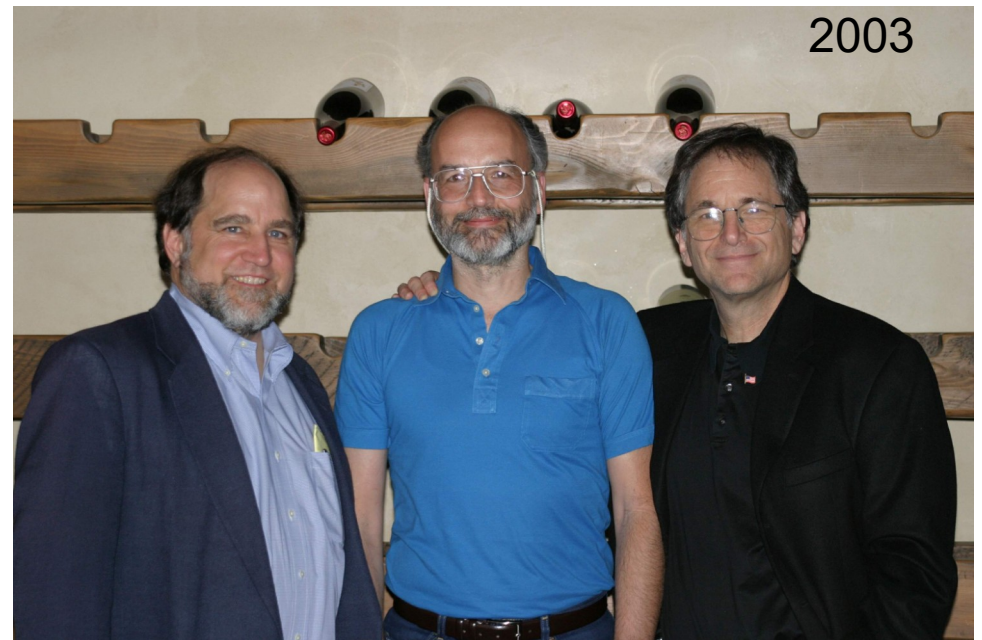
The most widely used public key cryptosystem, developed by Rivest, Adelman, and Shamir in 1978, called the *RSA cryptosystem*.

The security of the RSA cryptosystem rests on the assumption that it is difficult to factor large numbers.

If there were an efficient algorithm to factor numbers, then messages encrypted through RSA could be decrypted by someone who does not hold the matching decryption key.



*at the time of creation*



Ron Rivest    Adi Sharir    Leonard Adelman

## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

#### Example:

1. Let  $p = 17$  and  $q = 23$

## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

#### Example:

1. Let  $p = 17$  and  $q = 23$
2.  $N = pq = 17 * 23 = 391$        $j = (p-1)(q-1) = 16 * 22 = 352$

## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

#### Example:

1. Let  $p = 17$  and  $q = 23$
2.  $N = pq = 17 * 23 = 391$        $j = (p-1)(q-1) = 16 * 22 = 352$
3.  $352 = 2^5 * 11$ , so let's pick  $e = 197$  (*prime number*)

## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

#### Example:

1. Let  $p = 17$  and  $q = 23$
2.  $N = pq = 17 * 23 = 391$        $j = (p-1)(q-1) = 16 * 22 = 352$
3.  $352 = 2^5 * 11$ , so let's pick  $e = 197$  (*prime number*)
4. using *Extended Euclid's Algorithm* for finding  $\gcd(197, 352)$ ,  
we will find  $s$  and  $t$ , such that  
 $1 = s * 197 + t * 352$  ... *see the process in a separate file...*     $s = 109, t = -61$   
 $d = s \bmod j = 109 \bmod 352 = 109$



## 6.9 The RSA cryptosystem

### Preparation of public and private keys in RSA

1. Bob selects two large prime numbers,  $p$  and  $q$  (*several hundred digits*)
2. Bob computes  $N = pq$  and  $j = (p-1)(q-1)$   
(*! assumption: given  $pq$ , it would be infeasible to discover  $p$  or  $q$* )
3. Bob finds an integer  $e$  such that  $\gcd(e, j) = 1$  (i.e. *relatively prime*)  
( *$e$  is often chosen to be a prime number*)
4. Bob computes the multiplicative inverse of  $e \bmod j$ :  
an integer  $d$  such that  $(ed \bmod j) = 1$
5. Public (encryption) key:  $N$  and  $e$
6. Private (decryption) key:  $d$

#### Example:

1. Let  $p = 17$  and  $q = 23$
2.  $N = pq = 17 * 23 = 391$        $j = (p-1)(q-1) = 16 * 22 = 352$
3.  $352 = 2^5 * 11$ , so let's pick  $e = 197$  (*prime number*)
4. using *Extended Euclid's Algorithm* for finding  $\gcd(197, 352)$ ,  
we will find  $s$  and  $t$ , such that  
 $1 = s * 197 + t * 352$  ... *see the process in a separate file...*       $s = 109, t = -61$   
 $d = s \bmod j = 109 \bmod 352 = 109$

5. public key:  $N = 391, e = 197$
6. private key:  $d = 109$

## 6.9 The RSA cryptosystem

When Alice wants send a *plaintext* message  $m$  to Bob, the RSA scheme requires that:

- $m \in \mathbf{Z}_N$ , and
- $m$  is not a multiple of  $p$  or  $q$ .

$p$  and  $q$  are primes with hundreds of digits, hence it is extremely unlikely that  $m$  is a multiple of primes  $p$  or  $q$ .

Alice encrypts her plaintext using  $e$  and  $N$  to produce ciphertext  $c$  as follows:

$$c = m^e \bmod N \text{ (encryption)}$$

Alice transmits  $c$  to Bob. Bob decrypts the cyphertext using  $d$  to recover  $m$  from  $c$ :

$$m = c^d \bmod N \text{ (decryption)}$$

## 6.9 The RSA cryptosystem

When Alice wants send a *plaintext* message  $m$  to Bob, the RSA scheme requires that:

- $m \in \mathbf{Z}_N$ , and
- $m$  is not a multiple of  $p$  or  $q$ .

$p$  and  $q$  are primes with hundreds of digits, hence it is extremely unlikely that  $m$  is a multiple of primes  $p$  or  $q$ .


Alice encrypts her plaintext using  $e$  and  $N$  to produce ciphertext  $c$  as follows:

$$c = m^e \bmod N \text{ (encryption)}$$

Alice transmits  $c$  to Bob. Bob decrypts the cyphertext using  $d$  to recover  $m$  from  $c$ :

$$m = c^d \bmod N \text{ (decryption)}$$

**Example** (continues): public key:  $N = 391$ ,  $e = 197$       private key:  $d = 109$

Alice will encode message 223.  *not a multiple of 17 or 23*

## 6.9 The RSA cryptosystem

When Alice wants send a *plaintext* message  $m$  to Bob, the RSA scheme requires that:

- $m \in \mathbf{Z}_N$ , and
- $m$  is not a multiple of  $p$  or  $q$ .

$p$  and  $q$  are primes with hundreds of digits, hence it is extremely unlikely that  $m$  is a multiple of primes  $p$  or  $q$ .

Alice encrypts her plaintext using  $e$  and  $N$  to produce ciphertext  $c$  as follows:

$$c = m^e \bmod N \text{ (encryption)}$$

Alice transmits  $c$  to Bob. Bob decrypts the cyphertext using  $d$  to recover  $m$  from  $c$ :

$$m = c^d \bmod N \text{ (decryption)}$$

**Example** (continues): public key:  $N = 391$ ,  $e = 197$       private key:  $d = 109$

Alice will encode message 223.  *not a multiple of 17 or 23*

$c = 223^{197} \bmod 391$  - here we will recall our modular exponentiation

$C = 151$  (Section 6.7)

## 6.9 The RSA cryptosystem

When Alice wants send a *plaintext* message  $m$  to Bob, the RSA scheme requires that:

- $m \in \mathbf{Z}_N$ , and
- $m$  is not a multiple of  $p$  or  $q$ .

$p$  and  $q$  are primes with hundreds of digits, hence it is extremely unlikely that  $m$  is a multiple of primes  $p$  or  $q$ .

Alice encrypts her plaintext using  $e$  and  $N$  to produce ciphertext  $c$  as follows:

$$c = m^e \bmod N \text{ (encryption)}$$

Alice transmits  $c$  to Bob. Bob decrypts the cyphertext using  $d$  to recover  $m$  from  $c$ :

$$m = c^d \bmod N \text{ (decryption)}$$

**Example** (continues): public key:  $N = 391$ ,  $e = 197$       private key:  $d = 109$

When Bob get message from Alice ( $151$ ), he will decrypt it:

## 6.9 The RSA cryptosystem

When Alice wants send a *plaintext* message  $m$  to Bob, the RSA scheme requires that:

- $m \in \mathbf{Z}_N$ , and
- $m$  is not a multiple of  $p$  or  $q$ .

$p$  and  $q$  are primes with hundreds of digits, hence it is extremely unlikely that  $m$  is a multiple of primes  $p$  or  $q$ .

Alice encrypts her plaintext using  $e$  and  $N$  to produce ciphertext  $c$  as follows:

$$c = m^e \bmod N \text{ (encryption)}$$

Alice transmits  $c$  to Bob. Bob decrypts the cyphertext using  $d$  to recover  $m$  from  $c$ :

$$m = c^d \bmod N \text{ (decryption)}$$

**Example** (continues): public key:  $N = 391$ ,  $e = 197$       private key:  $d = 109$

When Bob get message from Alice ( $151$ ), he will decrypt it:

$$m = 151^{109} \bmod 391 \text{ - here we will again recall our modular exponentiation}$$
$$m = 223 \text{ (Section 6.7)}$$