The RSA system

The RSA system (a public key cryptography system) was developed by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977. To understand it, you only need to know the elementary number theory. The public key is a set of two integers n and e, and the private key is a third integer d. In practice, these integers need to be very large; typically n will have at least 300 digits.

Of course, n, e, and d need to be chosen in a special way to make the system work. In particular, n will be a product of two distinct primes p and q, and d and e will be integers less than n chosen so that $de \equiv 1 \pmod{(p-1)(q-1)}$. The secrecy of the private key relies on the fact that given e and n, it is computationally impossible to compute d because doing so would be the same as factoring n, which is computationally impossible for n very large. In the RSA system, each user sets up his or her own public and private keys. The public keys are "really" public – they are published in some location so that everyone who wants to can find them. The private keys need to be kept "very" private. If anyone (besides the one user who owns them) has access to them, then that person can read any secret messages sent to the rightful owner of the key.

Description

Here is a description of how a user, Alice, would go about setting up her keys. We'll give an example at each step.

1. **Pick primes:** The first step for Alice is to pick two prime numbers p and q. In practice, these primes need to be very large – about 200 digits each (not in this example).

Pick primes
Take $p = 281$ and $q = 167$.

2. **Calculate n and $\phi(n) := (p-1)(q-1)$:** Next, Alice simply sets $n = pq$. Since Alice knows the factorization of n, she can compute $\phi(n) = (p-1)(q-1)$. Notice that in practice, n will have about 300 digits. This means that computing $\phi(n)$ would be very difficult without knowing the factorization of n, and factoring n would also be very difficult.

Calculate n and $\phi(n)$
We have $n = (281)(167) = 46\,927$ and $\phi(n) = (280)(166) = 46\,480$.

3. **Choose e – the encoding exponent:** The next step is to pick a value of e at random, making sure that $\gcd(e, \phi(n)) = 1$. Alice does this by first selecting a value for e and then performing the Euclidean Algorithm to calculate $\gcd(e, \phi(n))$. If this gcd is 1, great. Otherwise, Alice simply chooses a new value of e.

Choose e
Take $e = 39\,423$. You can check that $\gcd(39\,423, 46\,480) = 1$ using the Euclidean Algorithm.

4. **Find d – the decoding exponent:** Now, Alice needs to find a value of d so that $de \equiv 1 \pmod{\varphi(n)}$. She can do this by using the Extended Euclidean Algorithm to find x and y so that $ex + \phi(n)y = 1$, and then setting $d = x$.

Find d
One finds that $d = 26\,767$.

Now Alice does the following:

5. **Publish** the public keys: $n = 46\,927$ and $e = 39\,423$.

6. **Keep secret** the private key: $d = 26\,767$.

Maths behind the RSA system

Let M stand for the plaintext message (converted to a numeric value), C stand for the secret message (converted to a numeric value), f stand for the encoding function and f^{-1} the corresponding decoding function.

A. Encode: To encode the message M for Alice, Bob simply computes

$$C = f(M) \quad \text{where } f(M) := M^e \bmod n \tag{1}$$

In fact, anyone in the world can do this, since everyone knows Alice's public keys.

B. Decode: To decode an encoded message C, Alice needs to compute

$$M = f^{-1}(C) \quad \text{where } f^{-1}(C) := C^d \bmod n \tag{2}$$

Only she can do this, because only she knows the value of d. Incidentally, Alice should also destroy her records of p, q, and $\phi(n)$. These bits of information are no longer needed by her, and if anyone else finds them, they will be able to figure out her (private) decoding exponent d.

Now let's see why the RSA system works – i.e., why, for any message M, we have

$$C^d \equiv M \pmod n \quad \text{or, equivalently,} \quad C^d \bmod n = M \bmod n.$$

Remember that d and e were chosen so that $de \equiv 1 \pmod{\phi(n)}$. This means that

$$de = \phi(n)k + 1, \quad \text{for some integer } k.$$

Also, from (1), $C \equiv M^e \pmod n$ Therefore,

$$\begin{aligned} f^{-1}(C) = f^{-1}(M^e \bmod n) = (M^e)^d \bmod n = M^{\phi(n)k+1} \bmod n = M^{\phi(n)k} M \bmod n \\ = (M^{\phi(n)} \bmod n)^k M \bmod n = 1^k M \bmod n = M \bmod n. \end{aligned}$$

Encoding–Decoding example

Bob wants to send Alice a message in which the characters can be any letter of the alphabet (upper case) or the space. Specifically, Bob would like to tell Alice "NO WAY" to her invitation to go sky-diving. Bob needs to follow these steps:

i. Look up Alice's public keys.

Public keys
Bob finds $n = 46\,927$, $e = 39\,423$.

ii. Break up his message into blocks: Bob wants to break up "NO WAY" into blocks that are as large as possible so that frequency analysis cannot be used to decipher his message. Since the encryption function encrypts numbers, and each block will be assigned a number, Bob needs to know how many different "words"(using the characters A–Z and the space) could be formed using a block of a certain length. There are a total of 27 possible different characters in a message from Bob, and so there are $27 \cdot 27 = 729$ 2–letter "words", $27 \cdot 27 \cdot 27 = 19\,683$ 3-letter "words", $27 \cdot 27 \cdot 27 \cdot 27 = 531\,441$ 4-letter "words", etc.

Since $27^3 = 19\,683$ is the largest power of 27 less than the public modulus $n = 46\,927$ while $27^4 = 531\,441$ is greater, the plaintext should be broken into trigraphs (blocks of size 3) for enciphering. If Bob used blocks of size larger than 3, he would run into the problem of having 2 different plaintext blocks correspond to the same enciphered block since the encoding function works modulo n.

Break up the message
Break up "NO WAY" into the two blocks "NO " and "WAY". Convert the alphabetic blocks to numbers: Letting A be 0, B be 1, …, Z be 25, and the space be 26. He thinks of each block as a number in base 27 and converts it to base 10. "NO ": $13 \cdot 27^2 + 14 \cdot 27^1 + 26 \cdot 27^0 = 9\,881$ "WAY": $22 \cdot 27^2 + 0 \cdot 27^1 + 24 \cdot 27^0 = 16\,062$.

iii. Encode the numbers: Now, Bob simply encodes each block separately using fast exponentiation to do the computations.

Encode process

To encode "NO " compute $9\,881^{39\,423} \bmod 46\,927 = 9\,388$.

To encode "WAY" compute $16\,062^{39\,423} \bmod 46\,927 = 21\,358$.

iv. Convert these encrypted numbers to strings: Since the encryption step can produce numbers between 0 and 46 926, each plaintext block of 3 characters will have to correspond to a ciphertext block of 4 characters to accommodate enciphered numbers in excess of $27^3 = 19\,683$ and less than $27^4 = 46\,927$. This means that Bob should convert the enciphered numbers to base 27, being careful to always include 4 digits.

Convert 9 388 and 21 358

$9\,388 \div 27 = 347$ R 19

$347 \div 27 = 12$ R 23

$12 \div 27 = 0$ R 12

Therefore

$$9\,388 = 0 \cdot 27^3 + 12 \cdot 27^2 + 23 \cdot 27^1 + 19 \cdot 27^0,$$

which corresponds to the string "AMXT".

$21\,358 \div 27 = 791$ R 1

$791 \div 27 = 29$ R 8

$29 \div 27 = 0$ R 1

Therefore

$$21\,358 = 1 \cdot 27^3 + 2 \cdot 27^2 + 8 \cdot 27^1 + 1 \cdot 27^0,$$

which corresponds to the string "BCIB".

v. Send the encrypted message.

Convert 9 388 and 21 358

Bob sends "AMXTBCIB" to Alice.

Now let's look at the decoding step. Alice receives the message "AMXTBCIB"from Bob. In order to read the message, she performs the following steps:

i. Break the message up into blocks of size 4.

Break the message

Alice breaks "AMXTBCIB" up into "AMXT" and "BCIB"

ii. Convert the alphabetic blocks to numbers: Thinking of the blocks as being base 27 representations, she converts them to base 10.

Convert blocks to numbers

"AMXT": $0 \cdot 27^3 + 12 \cdot 27^2 + 23 \cdot 27^1 + 19 \cdot 27^0 = 9\,388$

"BCIB": $1 \cdot 27^3 + 2 \cdot 27^2 + 8 \cdot 27^1 + 1 \cdot 27^0 = 21\,358$.

iii. Decode the encrypted numbers: Alice must decode each block separately, by raising it to the dth power and reducing modulo n. She'll use fast exponentiation to do the computations.

Encode process

To decode "AMXT" compute $9\,388^{26\,767} \bmod 46\,927 = 9\,881$.

To decode "BCIB" compute $21\,358^{26\,767} \bmod 46\,927 = 16\,062$.

iv. Convert the decoded numbers to strings: We first find the base 27 representations of our decoded numbers.

Convert 9 388 and 21 358
$9\,881 = 13 \cdot 27^2 + 14 \cdot 27^1 + 26 \cdot 27^0$, corresponds to "NO "
$16\,062 = 22 \cdot 27^2 + 0 \cdot 27^1 + 24 \cdot 27^0$ corresponds to "WAY"

The message that Bob sent Alice was "NO WAY". Too bad for Alice; she'll have to skydive alone.

Exercises

Exercise 1.- Given positive integers x, n, e, define a function `expmod.m` that implements the modular exponentiation algorithm to get

$$y = x^e \bmod n$$

REFERENCE: [R] – PAGE 254 AND AULA GLOBAL.

```
function y=expmod(x,e,n)

% INPUT: x,e,n positive integers x basis, e exponent, n modulus
% OUTPUT: y integer such that  y=x^e mod (n)
```

Exercise 2.- Given positive integers a and b, define a function `extendeuclides.m` that computes the greatest common divisor of a and b and the numbers s and t such that

$$\gcd(a, b) = sa + tb$$

REFERENCE: [R] - PAGE 273 – EXERCISE 45

```
function [s,t,gcd]=extendeuclides(a,b)

% INPUT: a,b positive integers
% OUTPUT:   gcd greatest common divisor of a and b;
%           s, t integers such that   gcd(a,b)=s*a+t*b
```

Exercise 3.- Define a function `rsa_encrypt.m` that encrypts any message (check it with "FUNDAMENTALS OF ALGEBRA") by using the RSA system with public key $(n, e) = (78\,817, 39\,423)$. To do so, follow (and adapt, if necessary) the example described above and use your auxiliary function `exmod.m`.

```
 function [C,CC]=rsa_encrypt(M,n,e)

% INPUT:
%       M: string of characters: Capital letters A-Z and the space
%       n: positive integer, defined as the product of p and q
%       e: positive integer, such that gcd(e, (p-1)(q-1))=1
%
% OUTPUT:
%      C: vector of nonnegative integers that encode the message M
%      CC: string of characters, conversion of C
%
```

Exercise 4.- Define a function `rsa_decrypt.m` that decodes a message encrypted with the public key $(n, e) = (78\,817, 39\,423)$. To do so, use the private key $(p \cdot q, d) = (293 \cdot 269, d)$ and d is such that $de \equiv 1 \pmod{(p-1)(q-1)}$. Follow (and adapt, if necessary) the example described above and use your auxiliary functions `exmod.m` and `extendeuclides.m`.

```
 function [M,d]=rsa_decrypt(CC,p,q,e)

% INPUT:
%       CC: string of characters
%       p,q: prime numbers
%       e: positive integer, such that gcd(e, (p-1)(q-1))=1
%
% OUTPUT:
%      M: string of characters: the secret message that corresponds to CC.
%      d: private key exponent, such that  0<d<(p-1)*(q-1)
%
```

Rules

1. This task is optional.
2. You have to upload the files
   (a) `expmod.m`
   (b) `extendeuclides.m`
   (c) `rsa_encrypt.m`
   (d) `rsa_decrypt.m`

   to Aula Global by Dec. 10 at 14:00.
3. Those of you that solve the task correctly will get an extra 0.4 points on their final mark.