eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

Decoding RSA and Caesar Code

By Asier Las Hayas, Lucia Del Rio, Shaman Alonso and Laura Calvo

--Discrete Mathematics--

THE CODE

299,109,179 179,275,275,179,102,197,179 324,127

The formula used to code the message is:

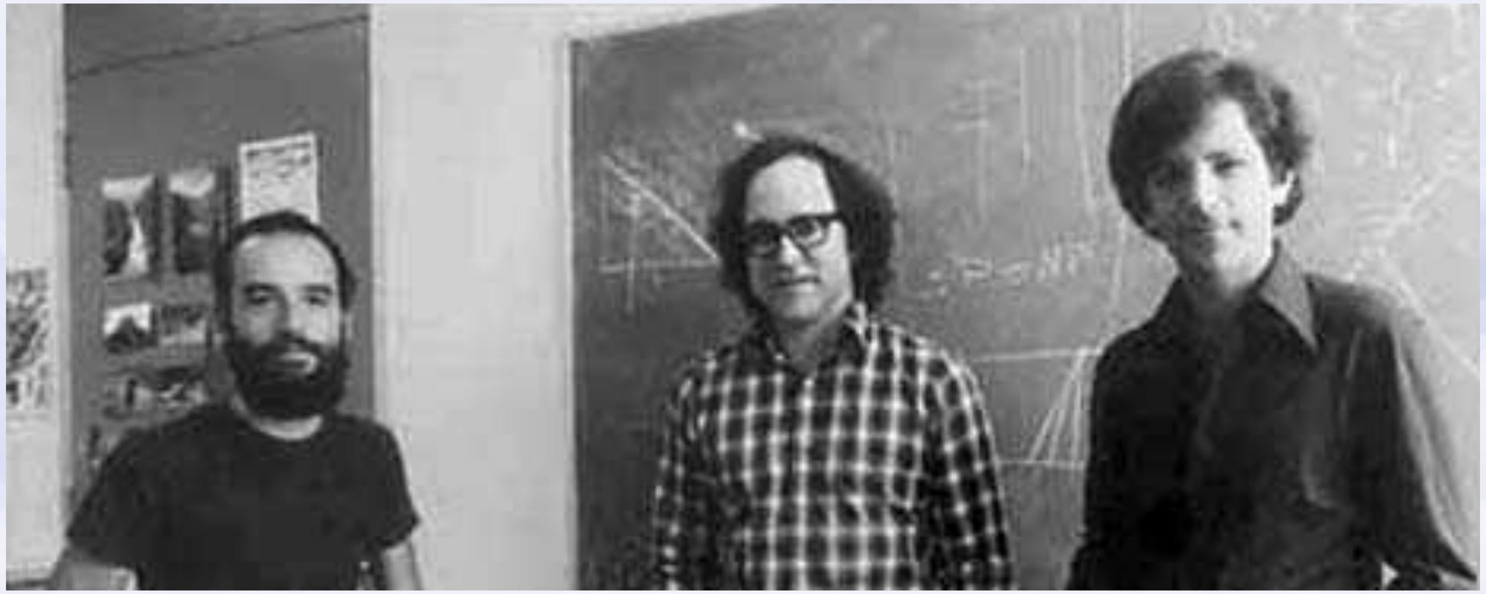
$$C \equiv M^e \bmod (pq)$$

- e is the key used to code the message and equal to 13
- p is equal to 17
- q is equal to 23

CONCEPTS TO BE STUDIED

1. RSA encryption
2. Congruence modulo m
3. Inverse of a modulo n and Fermat's little theorem
4. Caesar cipher
5. The exercise

Introduction to RSA



- Developed by Ron Rivest, Adi Shamir, Leonard Adleman in 1979

- Used for secure data transmission, and it's based on the difficulty of factoring the product of two large prime numbers.

- One of the most widely used public key encryption algorithms

RSA Decryption Process

1

Step 1: Select p,q

First, we need to select two large prime numbers p,q.

2

Step 2: Find n = p * q

Next, we need to find the value of n.

3

Step 3: Eulers totient function Ø(n)=(p-1)*(q-1)

Then, we need to find the value of Ø(n).

4

Step 4: Select 'e' such that gcd (e, Ø(n)) = 1

Later, we need to select e.

5

Step 5: Find $d \equiv e^{-1} \bmod \phi(n)$

Then, we will find the value of d.

6

Step 6: Cipher text $C \equiv M^e \bmod n$

By this process, we will encrypt the message.

7

Step 7: Plain text $M \equiv C^d \bmod n$

By this process, we will decrypt the message.

CONGRUENCE MODULO M

Definition:

a, b ∈ Z are congruent modulo m if a - b is a multiple of m and k ∈ Z such that: $a-b=k \cdot m$ or $a=b+k \cdot m$

Example: $a \equiv b \mod m$

a=23, b=17
$23 - 17 = 6 = 2 \cdot 3 \rightarrow 23 \equiv 17 \mod 3$ (23 and 17 are congruent modulo 3)

FERMAT'S LITTLE THEORY

p is a prime number and a is any integer coprime with p

$$a^p \equiv a \pmod{p} \Leftrightarrow a^{p-1} \equiv 1 \pmod{p}$$

FERMAT'S LITTLE THEORY & ITS RELATION WITH EULER'S THEOREM

Euler's theorem is a generalisation of Fermat's theory:

- a coprime to n

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

- Euler's totient function: $\varphi(n) \in Z$ from 1 to n and coprime with n

If n is prime $\rightarrow$ $\phi(n)=n-1$

This allow us to reduce modular exponentiation with large exponents to exponents smaller than n

Corollary of Euler's theorem:

$$M \equiv C \ (\mathrm{mod}\varphi(n)) \Rightarrow a^M \equiv a^C \pmod{n}$$

$$M \equiv C \ (\mathrm{mod}\varphi(n)) \rightarrow M=C+k\varphi(n)$$
$$a^M \equiv a^{C+\varphi(n)k} \equiv [a^{\varphi(n)}]^k \cdot a^C \equiv 1^k \cdot a^C \equiv a^C (\mathrm{mod}\ n)$$

EULER'S THEOREM USED IN CRYPTOGRAPHY

$C = M^e \pmod{n}$

Bézout's identity

Modular inverse of e modulo $\varphi(n) = d$

$ed \equiv 1 \pmod{\varphi(n)}$

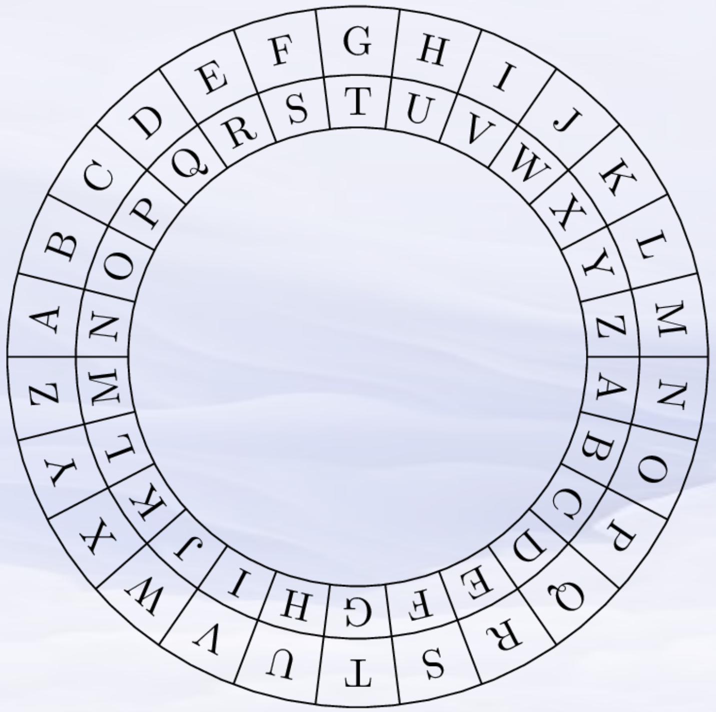
$C^d \equiv M^{ed} \pmod{\varphi(n)}$

$M^{ed} \equiv M^1 \equiv M$

$M \equiv M^{ed} \equiv (M^e)^d \equiv C^d \pmod{\varphi(n)}$

$M \equiv C^d \pmod{\varphi(n)}$

INTRODUCTION TO CAESAR CIPHER



- Named after Julius Caesar, who used this code with a shift of 3.
- One of the simplest and most known encryption techniques.
- Often incorporated as part of a more complex schemes.

CAESAR DECRYPTION PROCESS

1

Step 1: **Identify the Shift Value**

First, we need to identify the number of places each letter of the message was shifted.

2

Step 2: **Decoding the Caesar Encrypted Message**

Next, we replace each letter with the letter that comes a certain number of places before it in the alphabet.

3

Step 3: **Decrypting the Message**

Finally, we have to determine the correct shift value and decrypt the message completely.

The exercise

Algorithm
1.Select large prime numbers p, q
2. Find $n = p \cdot q$
3. Eulers totient function $\Phi(n) = (p-1) \cdot (q-1)$
4. Select e such that $\gcd(e, \Phi(n)) = 1$
5. Find $d \equiv e^{-1} \bmod \Phi(n)$
6. Decipher $M \equiv C^{d} \bmod n$
7. Caesar decipher

The exercise

Algorithm	Problem
1. Select p, q	$p = 17, q = 23$
2. Find $n = p \cdot q$	$n = 17 \cdot 23 = 391$
3. $\Phi(n) = (p-1) \cdot (q-1)$	$\Phi(n) = 16 \cdot 22 = 352$
4. Select e such that $\gcd(e, \Phi(n)) = 1$	$e = 13$ and $\gcd(13, 352) = 1$
5. Find $d \equiv e^{-1} \bmod \Phi(n)$	$d \equiv 13^{-1} \bmod 352$

How do we find d?

<u>Bézout theorem</u>

$a,b \in Z^+$ $p,q \in Z$ $ap + bq = 1$

$$\begin{array}{l|l} \mathbf{352} & \mathbf{13} \\ \hline 1 & 27 \end{array}$$

$352 = 13 \cdot 27 + 1$
$352 - 13 \cdot 27 = 1$
$352 \cdot 1 + 13 \cdot (-27) = 1$

$\gcd(13, 352) = 1$ $d \cdot e \equiv 1 \pmod{\Phi(n)}$

$352 \equiv 0 \pmod{352}$
$13 \cdot (-27) \equiv 13 \cdot (-27) \pmod{352}$

$352 + 13 \cdot (-27) \pmod{352} \equiv 13 \cdot (-27) \pmod{352} \equiv 1$

$13 \cdot (-27) \equiv 1 \pmod{352}$
$e \cdot d \equiv 1$
$d = -27 \equiv 325 \pmod{352}$
$d = 325 \pmod{352}$

	Example
6. Decipher text $M \equiv C^d \bmod n$	C = 299 $M \equiv 299^{325} \bmod 391$
7. Caesar decipher	For the text to make sense, we must substract 3 from the result $M\equiv(p+3) \bmod 26$

How do we obtain M?

Using *Fast Modular Exponentiation*

We can create an algorithm in python or make use of mathematica to speed up the process

Fast Modular Exponentiation

$299^{325} \bmod 391$

325_{10}: 101000101_2

$299^{325} = 299^{256+128+4+1} = \boxed{299^{256} \cdot 299^{128} \cdot 299^{4} \cdot 299^{1}}$ *

$\boxed{} \bmod 391 = 23$

*

299^1	299	$299 \bmod 391 = 299$
299^2	299^2	$299^2 \bmod 391 = 253$
299^4	$(299^2)^2$	$253^2 \bmod 391 = 276$
299^8	$(299^4)^2$	$276^2 \bmod 391 = 322$
299^{16}	$(299^8)^2$	$322^2 \bmod 391 = 69$
299^{32}	$(299^{16})^2$	$69^2 \bmod 391 = 69$
299^{64}	$(299^{32})^2$	...
299^{128}	$(299^{64})^2$	...
299^{256}	$(299^{128})^2$	...

```python
def code():
    p = 17
    q = 23
    e = 13

    n = p * q

    phi = (p - 1) * (q - 1)

    s=str(input("insert the code given between spaces, to separate words write . between spaces: "))
    num_str=s.split()
    d=inverse(e,phi)
    for num in num_str:
        if num == ".":
            print(" ", end="")
        else:
            numin=int(num)
            an=desencription(numin, d, n)
            v="abcdefghijklmnopqrstuvwxyz"
            s=an-4
            index=ind(s)
            print(v[index], end="")

code()
```

Variables

Decodification process

Subprograms

```python
def inverse(e,m):
    d = pow(e, -1, m)
    return d
```

```python
def desencription(c,d,n):
    m = pow(c,d,n)
    return m
```

```python
def ind(num):
    if num < 0:
        return num + 26
    else:
        return num
```

MATHEMATICA

inverse

```
In[14]:= d = PowerMod[13, -1, 352]
         potencia modular
Out[14]= 325
```

```
In[15]:= PowerMod[299, d, 391]
         potencia modular
Out[15]= 23

In[16]:= PowerMod[109, d, 391]
         potencia modular
Out[16]= 11

In[17]:= PowerMod[179, d, 391]
         potencia modular
Out[17]= 8

In[18]:= PowerMod[275, d, 391]
         potencia modular
Out[18]= 22
```

```
In[19]:= PowerMod[102, d, 391]
         potencia modular
Out[19]= 17

In[20]:= PowerMod[197, d, 391]
         potencia modular
Out[20]= 6

In[21]:= PowerMod[324, d, 391]
         potencia modular
Out[21]= 18

In[22]:= PowerMod[127, d, 391]
         potencia modular
Out[22]= 9
```

THE MESSAGE

"THE ESSENCE OF"