Carlos Benitez - Luciano Bello

Ekoparty #17
2 de noviembre de 2021

The silent partners

...o de cómo Shor se asoció a Fourier para factorizar números grandes y romper el cifrado asimétrico en computadoras cuánticas...

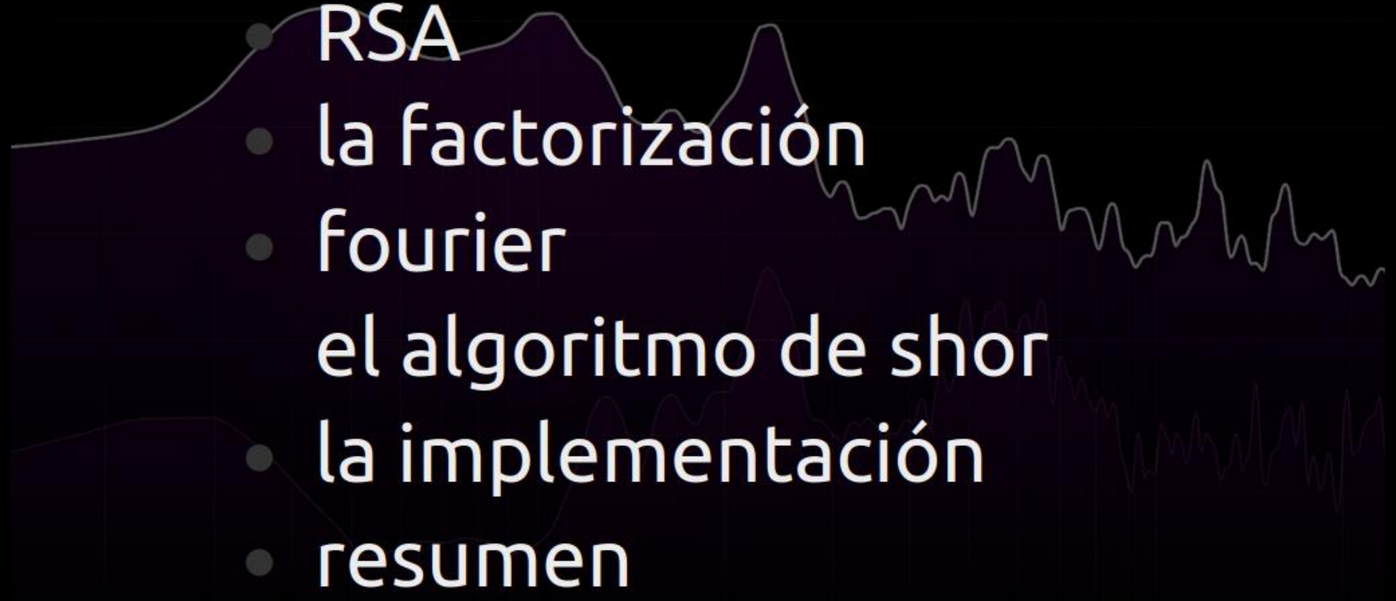
Carlos Benitez

Ing. y Mg. de la UTN FRBA
- Investigador en procesamiento de señales acústicas submarinas
- Primer Lab en Seguridad Informática (Si6) y primer SOC Defensa
- Asesor técnico de la Subsecretaría de Ciberdefensa
- Docente de posgrado y consultor en ciberseguridad
- Co-fundador de Platinumciber
- Formador y mentoring de equipos
- SOC, Ethical Hacking y Análisis y Gestión de Riesgos
- Quantum Computing enthusiast

Luciano Bello

Ing. de la UTN FRBA
PhD de Chalmers (Suecia)
- Desarrollador Senior en IBM Research
- Doctor en seguridad basada en lenguajes
- Python coder
- Desarrollador de Software Libre en Debian
- Diseñador de lenguajes formales
- Infosec enthusiast
- Geocacher

- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen

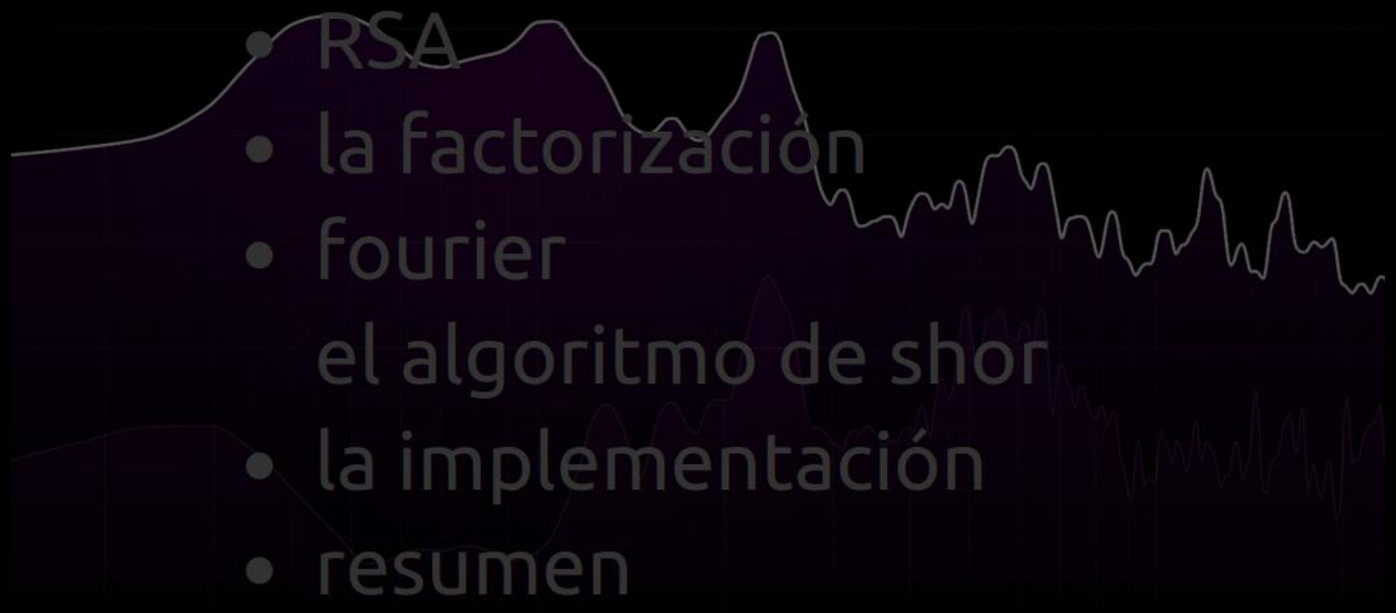


- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen

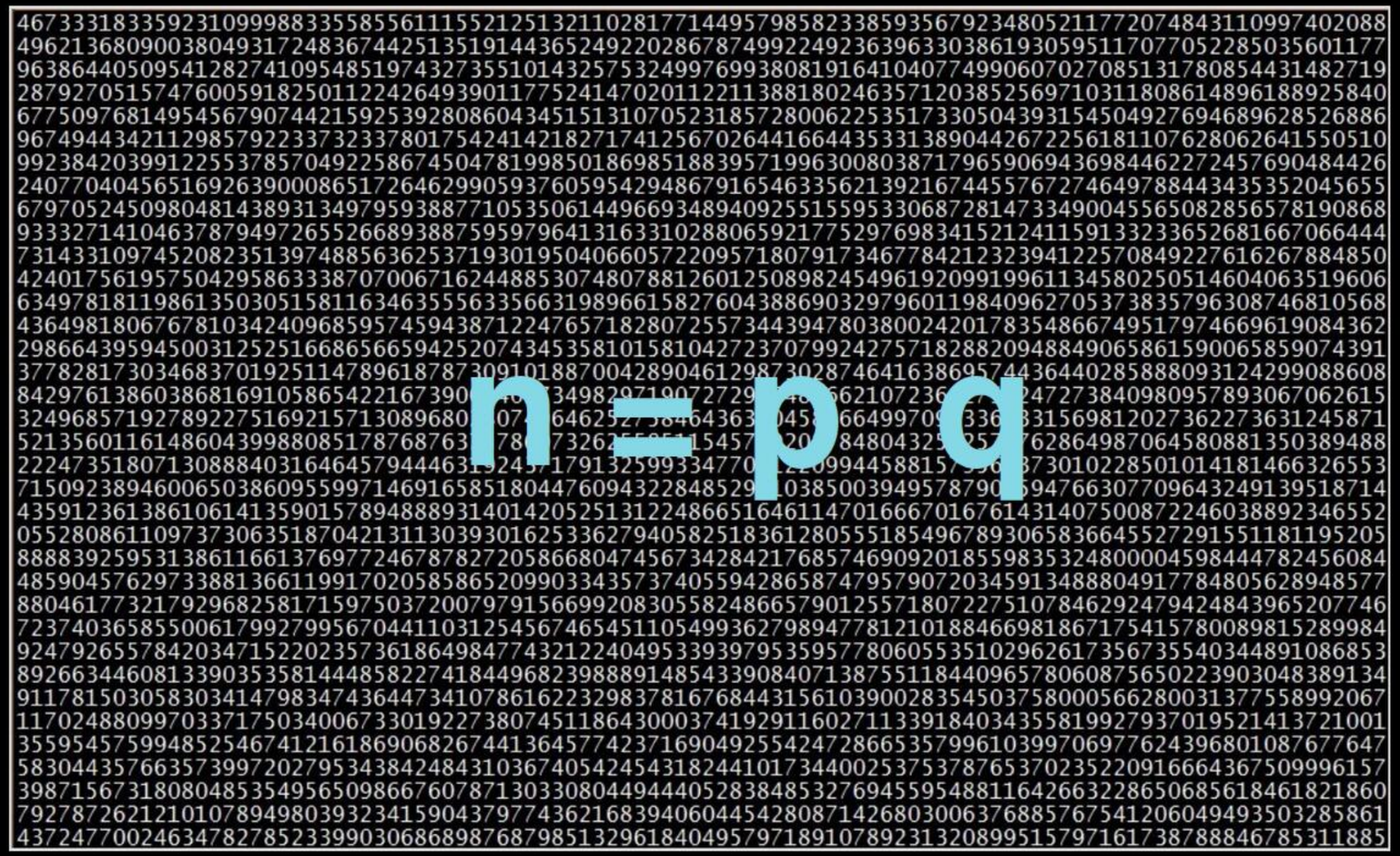
- Introducción a la QC
- Propiedades del qubit
- Hardware existente
- Algoritmo de Deutsch
- Los mitos de la QC
- Las QC online disponibles

- Introducción a la QC
- Propiedades del qubit
- Hardware existente
- Algoritmo de Deutsch
- Los mitos de la QC
- Las QC online disponibles

- Algoritmo de Shor



- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen


n=pq

- previously...
- el problema
- RSA
- la factorización
- fourier

  el algoritmo de shor
- la implementación
- resumen

RSA

Se define un número arbitrariamente grande, producto de dos primos:

$$n = p \cdot q$$

Se calcula la función φ de Euler como:

$$\varphi(n) = (p-1)(q-1)$$

Se obtienen *e* (la clave para cifrar) y *d* (la clave para descifrar) y se generan las claves pública y privada:

$$d\ e \bmod \varphi(n) = 1$$
$$\text{pubkey} = (n,e) \qquad \text{privkey} = (n,d)$$

Teniendo un mensaje *m*:

$$\text{message} = m$$

Se genera un mensaje cifrado *c* como:

$$c = m^e \bmod n$$

Una vez enviado, se recupera el mensaje descifrado *m* como:

$$m = c^d \bmod n$$

RSA attack

Dado que en la siguiente ecuación se conocen:

$$d\ e \bmod \varphi(n) = 1$$

Entonces, para calcular d hace falta conocer $\varphi(n)$.

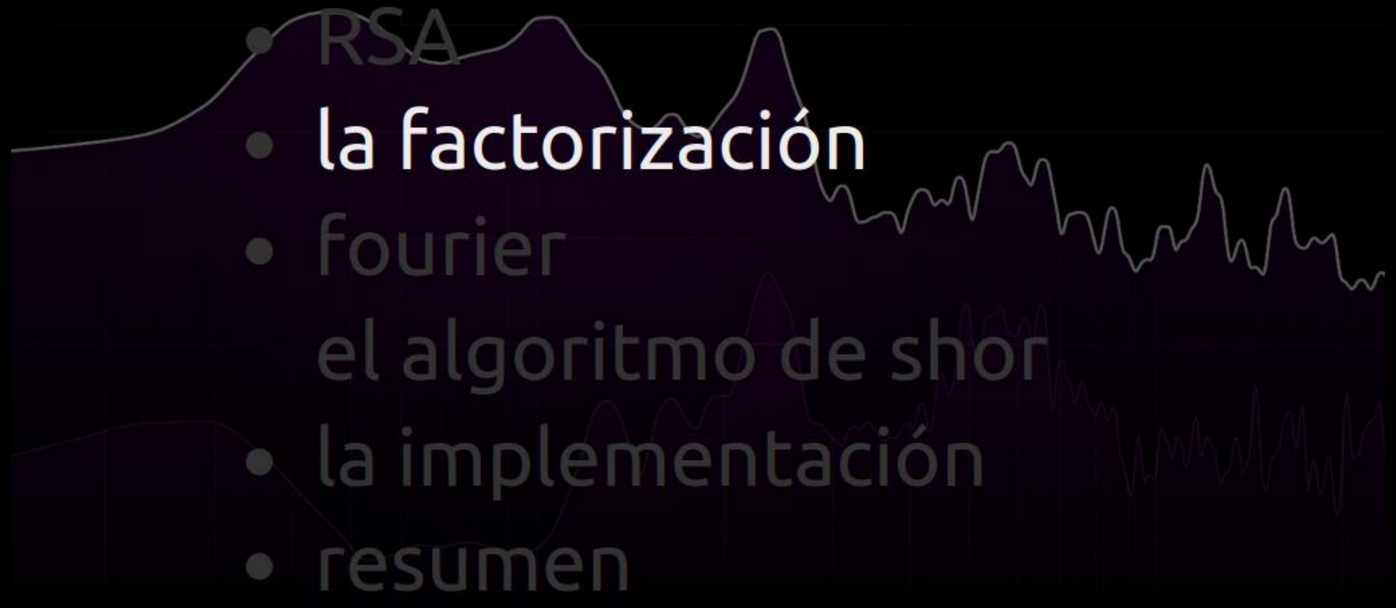
Pero $\varphi(n)$ es:

$$\varphi(n) = (p-1)(q-1)$$

Para obtener $\varphi(n)$ se necesitan p y q. Pero como:

$$n = p\ q$$

Entonces, factorizando n se obtienen p y q y, en consecuencia d.



- previously...
- el problema
- RSA
- la factorización
- fourier
  el algoritmo de shor
- la implementación
- resumen

Trial division

```python
# Python3 implementation of
# Trial Division Algorithm

# Function to check if a number is
# a prime number or not
def TrialDivision(N):

    # Initializing with the value 2
    # from where the number is checked
    i = 2

    # Computing the square root of
    # the number N
    k = int(N ** 0.5)

    # While loop till the
    # square root of N
    while(i<= k):

        # If any of the numbers between
        # [2, sqrt(N)] is a factor of N
        # Then the number is composite
        if(N % i == 0):
            return 0
        i += 1

    # If none of the numbers is a factor,
    # then it is a prime number
    return 1

# Driver code
if __name__ == "__main__":
    N = 49
    p = TrialDivision(N)

# To check if a number is a prime or not
    if(p):
        print("Prime")
    else:
        print("Composite")
```

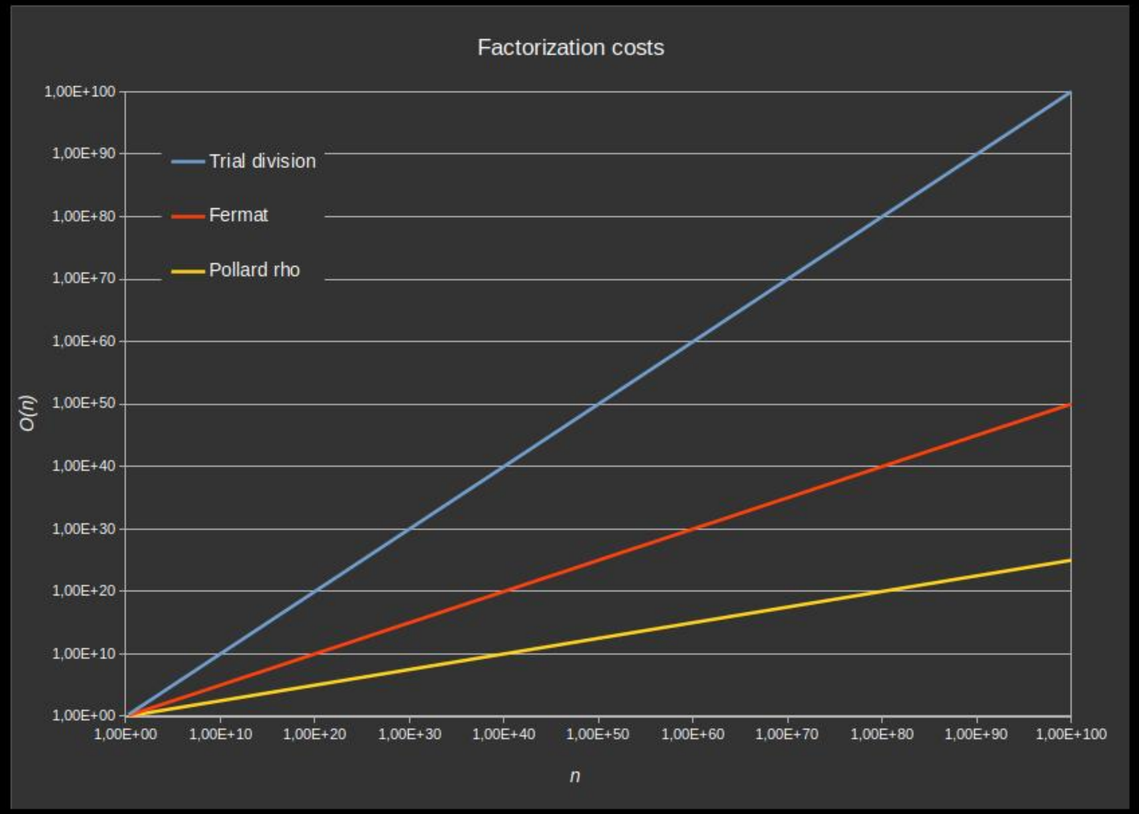
Fermat

```python
# Python 3 implementation of fermat's factorization

from math import ceil, sqrt

#This function finds the value of a and b
#and  returns a+b and a-b
def FermatFactors(n):

   # since fermat's factorization applicable
   # for odd positive integers only
    if(n<= 0):
        return [n]

    # check if n is a even number
    if(n & 1) == 0:
        return [n / 2, 2]

    a = ceil(sqrt(n))

    #if n is a perfect root,
    #then both its square roots are its factors
    if(a * a == n):
        return [a, a]

    while(True):
        b1 = a * a - n
        b = int(sqrt(b1))
        if(b * b == b1):
            break
        else:
            a += 1
    return [a-b, a + b]

# Driver Code
print(FermatFactors(6557))
```

Pollard rho

```python
# Python 3 program to find a prime factor of composite using
# Pollard's Rho algorithm
import random
import math

# Function to calculate (base^exponent)%modulus
def modular_pow(base, exponent,modulus):

    # initialize result
    result = 1

    while (exponent > 0):

        # if y is odd, multiply base with result
        if (exponent & 1):
            result = (result * base) % modulus

        # exponent = exponent/2
        exponent = exponent >> 1

        # base = base * base
        base = (base * base) % modulus

    return result

# method to return prime divisor for n
def PollardRho( n):

    # no prime divisor for 1
    if (n == 1):
        return n

    # even number means one of the divisors is 2
    if (n % 2 == 0):
        return 2

    # we will pick from the range [2, N)
    x = (random.randint(0, 2) % (n - 2))
    y = x
```

```python
    # the constant in f(x).
    # Algorithm can be re-run with a different c
    # if it throws failure for a composite.
    c = (random.randint(0, 1) % (n - 1))

    # Initialize candidate divisor (or result)
    d = 1

    # until the prime factor isn't obtained.
    # If n is prime, return n
    while (d == 1):

        # Tortoise Move: x(i+1) = f(x(i))
        x = (modular_pow(x, 2, n) + c + n)%n

        # Hare Move: y(i+1) = f(f(y(i)))
        y = (modular_pow(y, 2, n) + c + n)%n
        y = (modular_pow(y, 2, n) + c + n)%n

        # check gcd of |x-y| and n
        d = math.gcd(abs(x - y), n)

        # retry if the algorithm fails to find prime factor
        # with chosen x and c
        if (d == n):
            return PollardRho(n)

    return d

# Driver function
if __name__ == "__main__":

    n = 10967535067
    print("One of the divisors for", n , "is ",PollardRho(n))

# This code is contributed by chitranayal
```

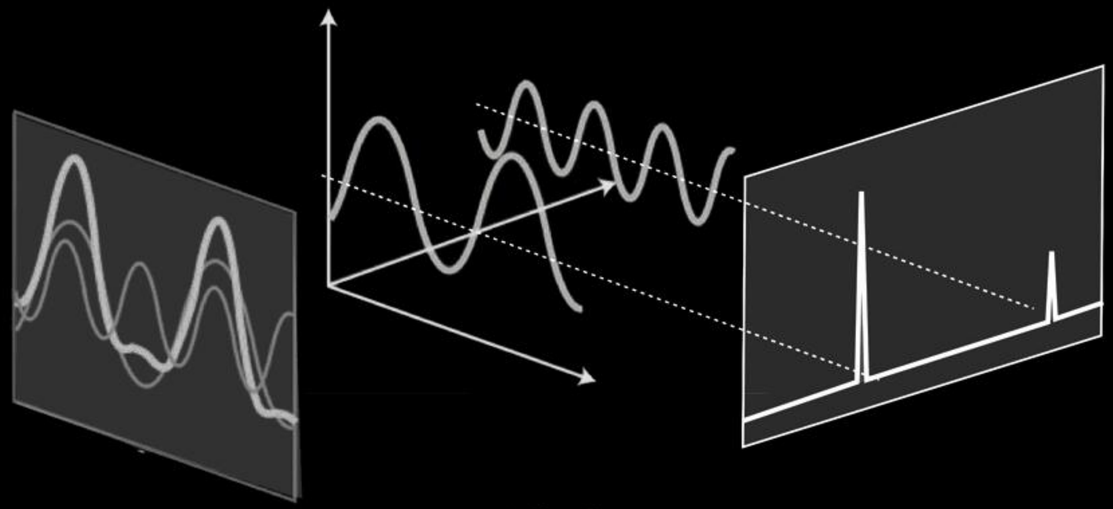

Factorization costs
1,00E+100
1,00E+90
1,00E+80
1,00E+70
1,00E+60
1,00E+50
1,00E+40
1,00E+30
1,00E+20
1,00E+10
1,00E+00
O(n)
Trial division
Fermat
Pollard rho
1,00E+00
1,00E+10
1,00E+20
1,00E+30
1,00E+40
1,00E+50
1,00E+60
1,00E+70
1,00E+80
1,00E+90
1,00E+100
n

- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen

Joseph Fourier 1768-1830

Transformada de Fourier

$$\mathcal{F}(\omega) = \int_{-\infty}^{\infty} f(\ x)\ \cdot e^{j\omega x} \cdot dx$$



Time Domain
f(t)

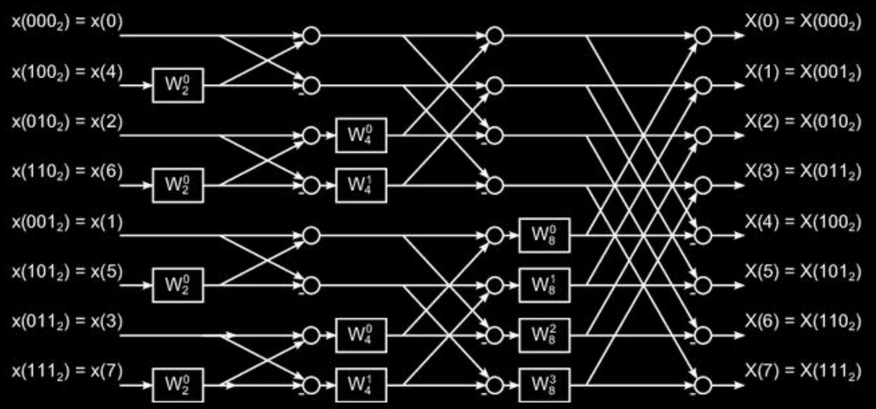
Frequency Domain
F(ω)

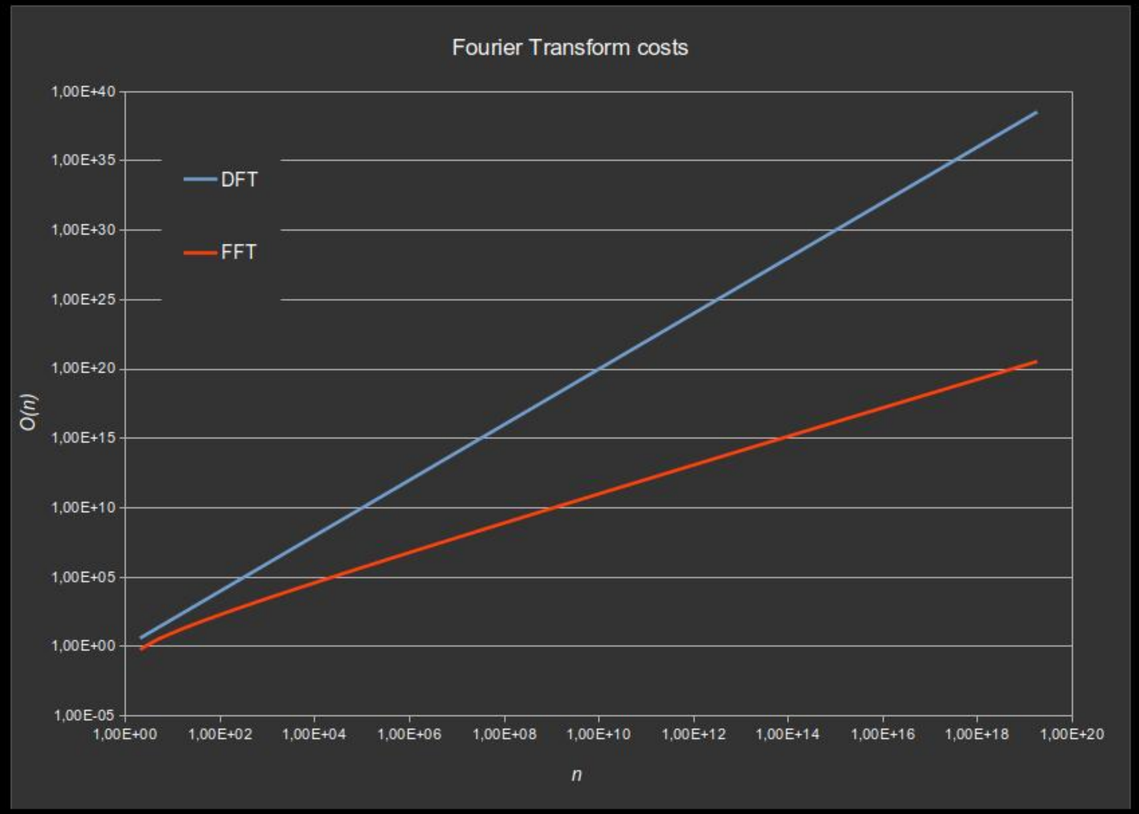
Transformada Discreta de Fourier

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi}{N} kn}$$

$$= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - j \cdot \sin\left(\frac{2\pi}{N} kn\right) \right]$$

Transformada Rápida de Fourier (FFT)




Fourier Transform costs
1,00E+40
1,00E+35
1,00E+30
1,00E+25
1,00E+20
1,00E+15
1,00E+10
1,00E+05
1,00E+00
1,00E-05
O(n)
DFT
FFT
1,00E+00
1,00E+02
1,00E+04
1,00E+06
1,00E+08
1,00E+10
1,00E+12
1,00E+14
1,00E+16
1,00E+18
1,00E+20
n

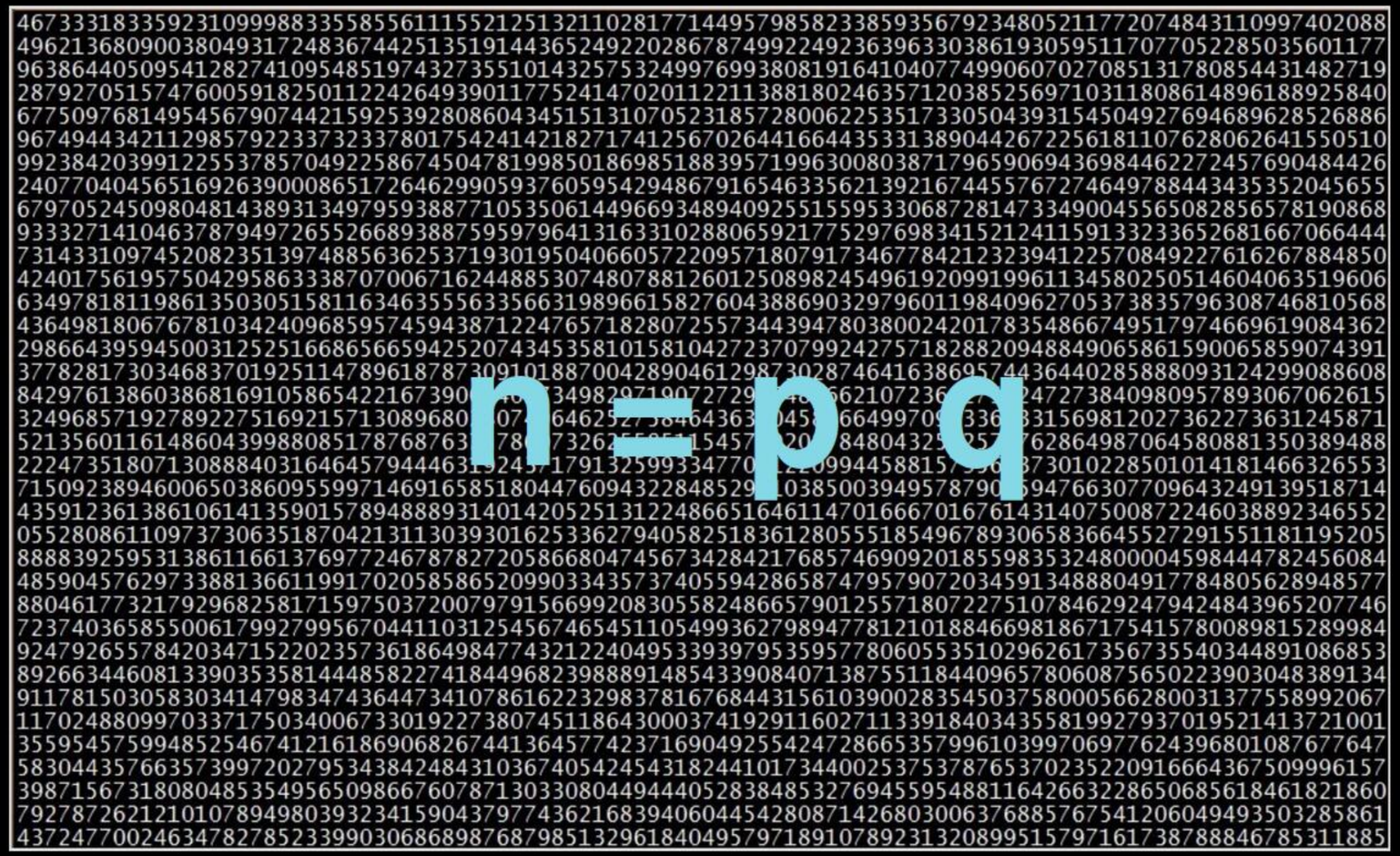
FFT

SOSUS/Caesar

Garmonya


DSP
TMS320AV7100PGW
CH 05A06HW
ATTACK
SUSTAIN
ON
LINK
spl
Transient Designer
BOSS Chorus Ensemble
GAIN
THRESHOLD
COMPRESSION
OUTPUT GAIN
dbx 160
COMPRESSOR/LIMITER
GLOBAL
OUTPUT
brainworx

- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen


n=pq

El algoritmo de Shor (I)

$$N = g \ . \ h$$

$$\hookrightarrow 2,3,5,7,11,13,17,...$$

$$35 = ? \ . \ ?$$

$$\hookrightarrow 2 \rightarrow \{ \text{SHOR} \} \rightarrow 5$$

Dados dos coprimos g y N existen p y m tales que

$$g^p = m \ . \ N + 1$$

El algoritmo de Shor (II)

$$g^p = m \,.\, N + 1$$

$$g^p - 1 = m \,.\, N$$

$$\left(g^{\frac{p}{2}} - 1\right)\left(g^{\frac{p}{2}} + 1\right) = m \,.\, N$$

$$p = ?$$

$$g^p = m \,.\, N + 1$$

$$g^p \bmod N = 1$$

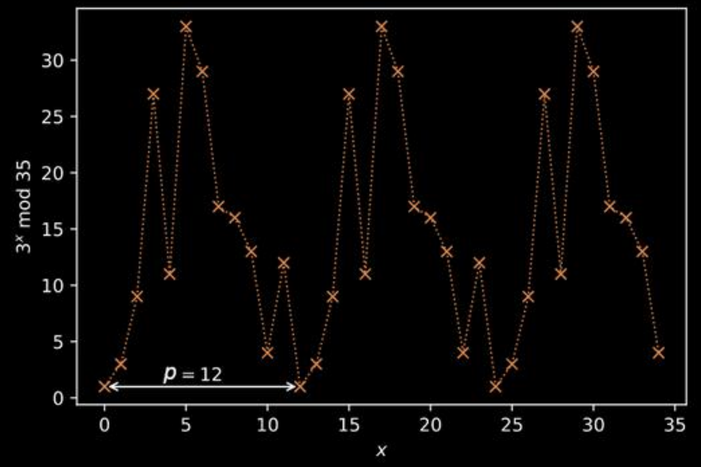
El algoritmo de Shor (III)

$$g^{p} \bmod N = 1$$
$$g^{x} \bmod N = r$$

$$3^{x} \bmod 35 = r$$

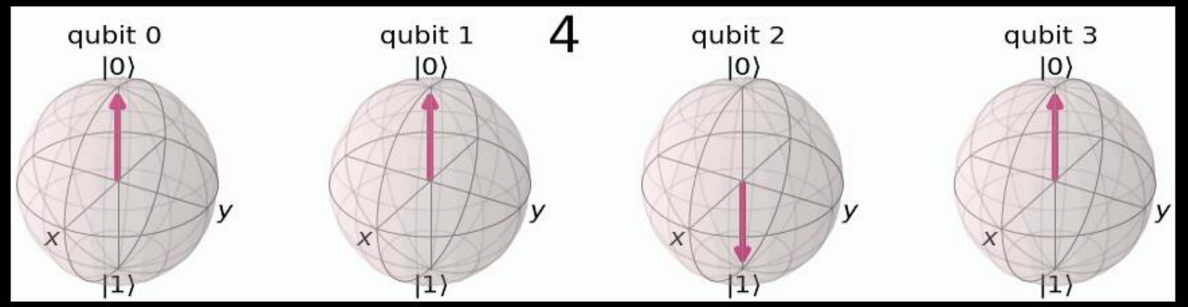
$$x_{k} = k.p$$
$$g^{x_{k}} \bmod N = r_{x}$$
$$g^{kp} \bmod N = r_{x}$$

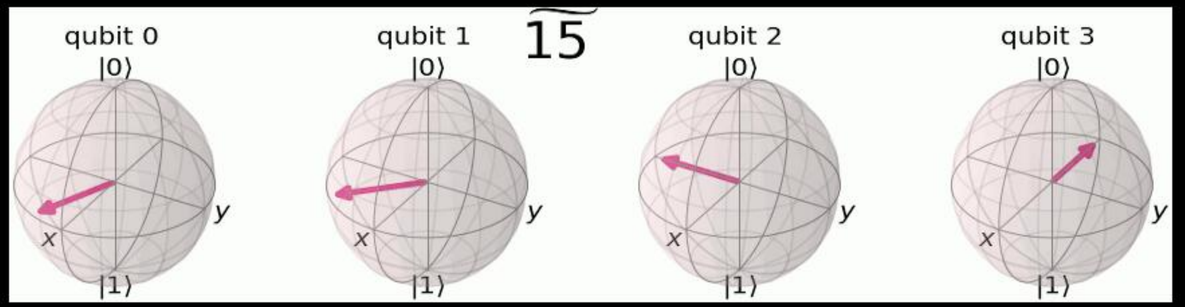
$$3^{k.\,12} \bmod 35 = r_{x}$$
$$3^{0} \bmod 35 = 3^{12} \bmod 35 = 3^{24} \bmod 35 = 1$$



Transformada Cuántica de Fourier (QFT)

$$\mathrm{QFT}_N |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{xk} |k\rangle$$



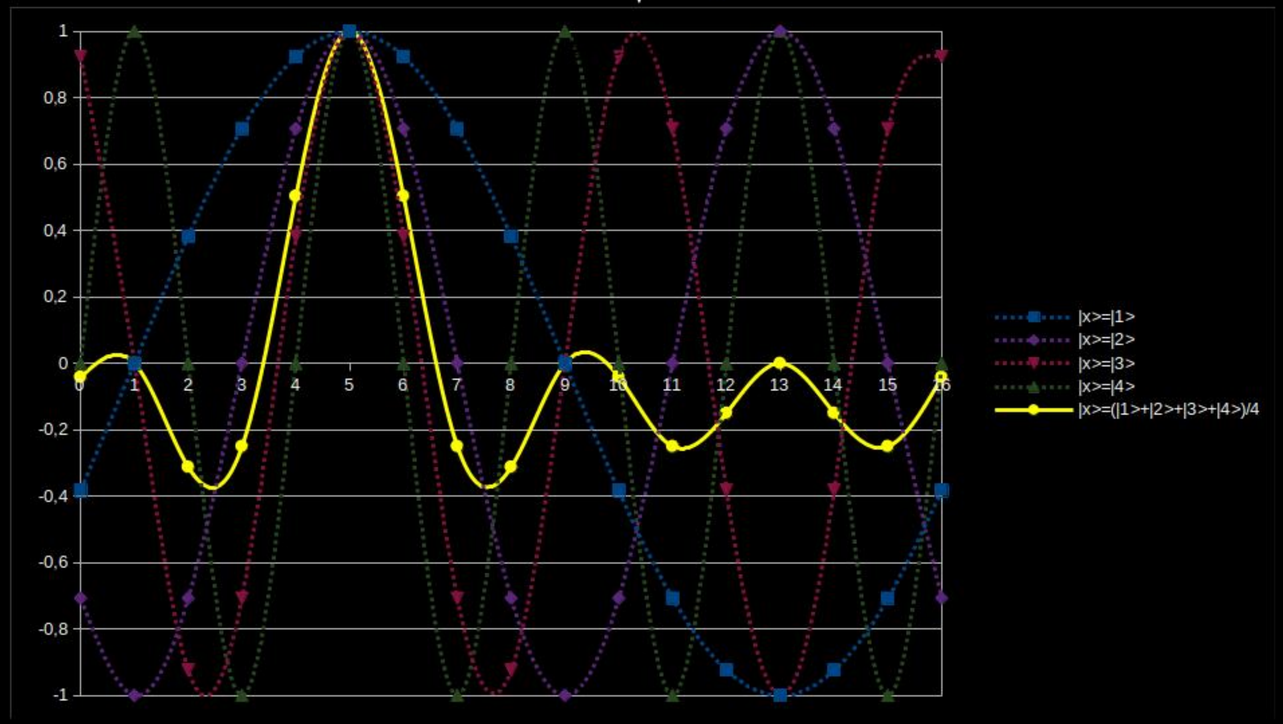


Transformada Cuántica de Fourier (QFT)

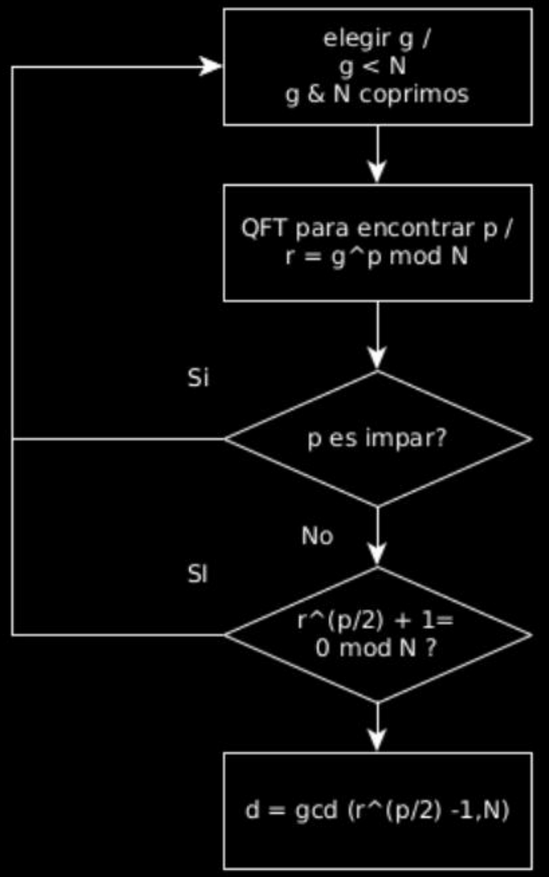
$$|x\rangle + |x+p\rangle + |x+2p\rangle ... + ... |x+np\rangle \mapsto \{\mathrm{QFT}\} \mapsto |\frac{1}{p}\rangle$$

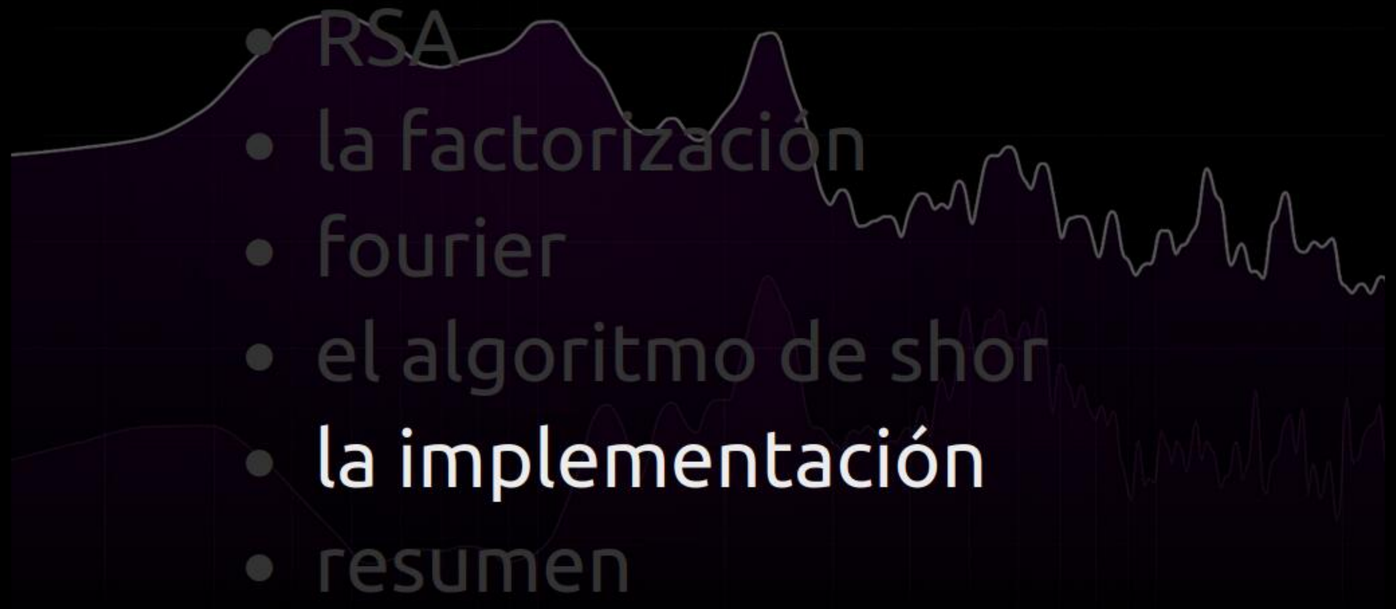
Transformada Cuántica de Fourier (QFT)

$$\mathrm{QFT}_N |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{xk} |k\rangle$$



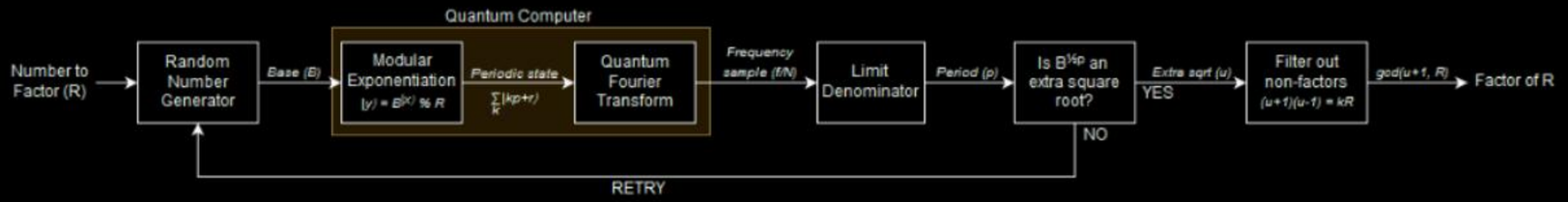
El Algoritmo de Shor (IV)



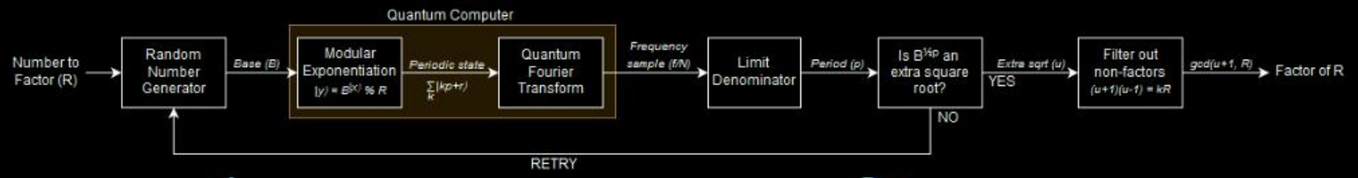


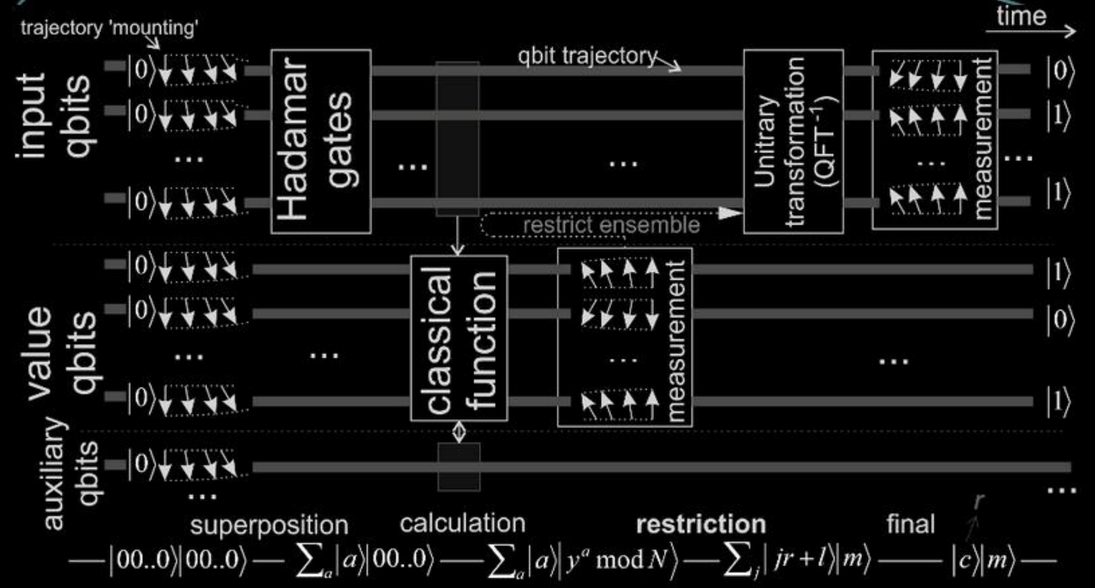
- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen

El algoritmo de Shor



El algoritmo de Shor





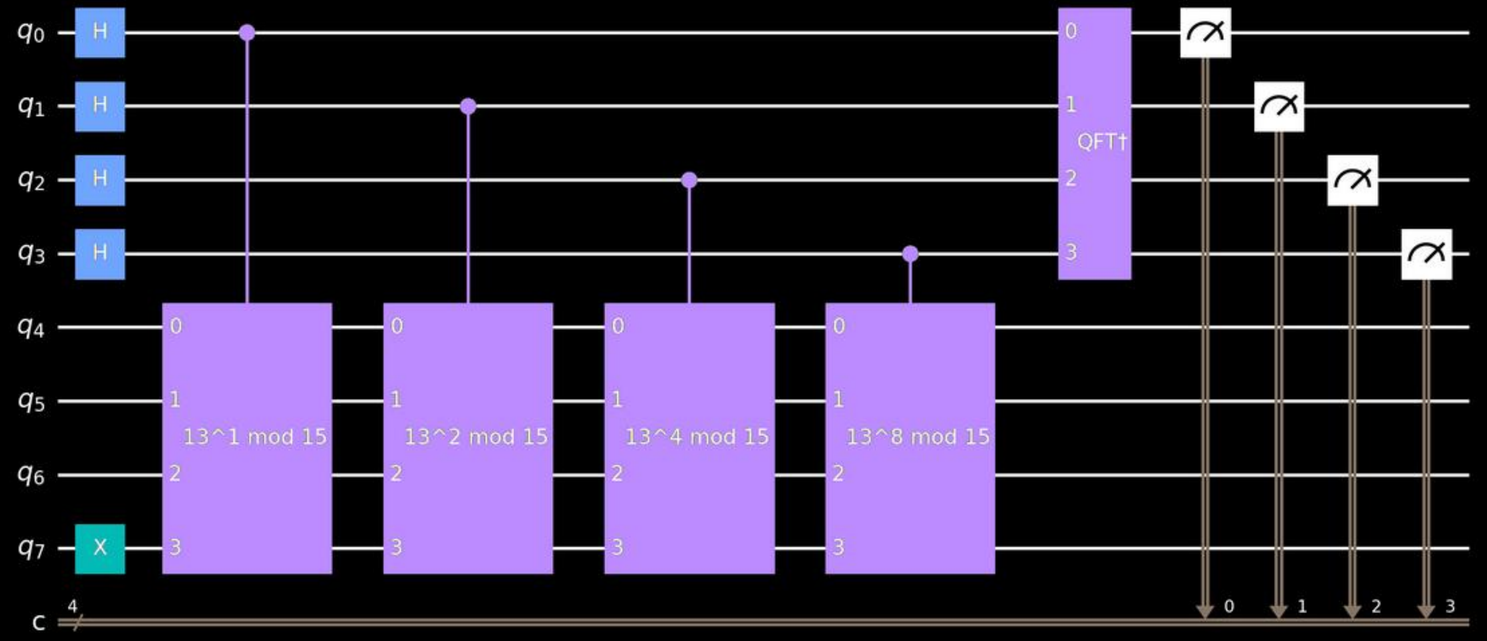
Algoritmo de Shor: Implementación

$$\mathrm{QFT}_2 \left| x \right\rangle \mapsto \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

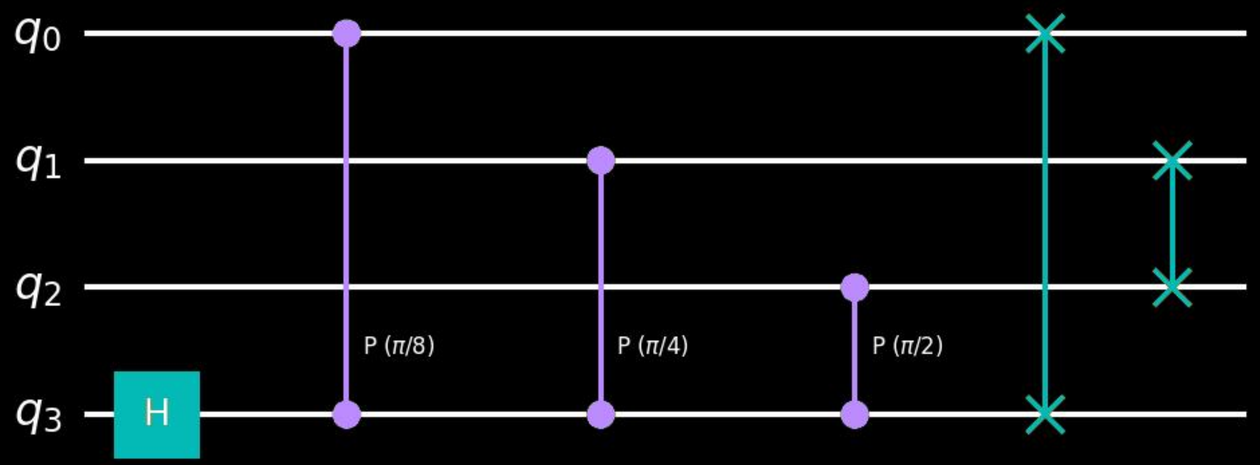
$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathrm{U\ ROT}_k = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & e^{j\frac{2\pi}{2^k}} \end{bmatrix}$$

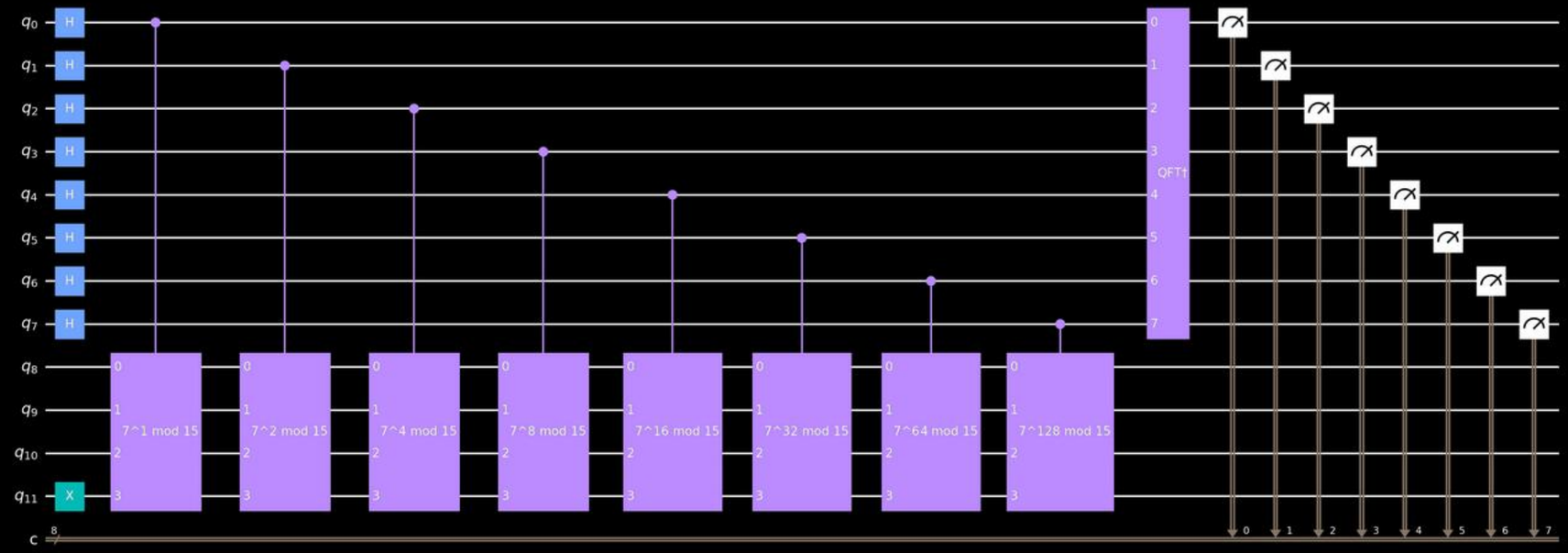
El algoritmo de Shor
circuito cuántico (4 qubits)



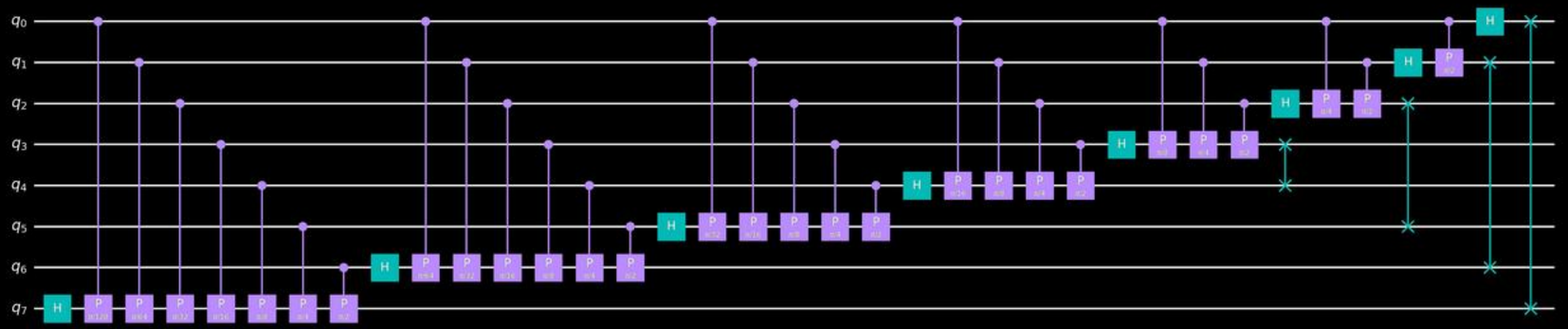
El algoritmo de Shor
circuito cuántico de QFT (4 qubits)



El algoritmo de Shor
circuito cuántico (8 qubits)

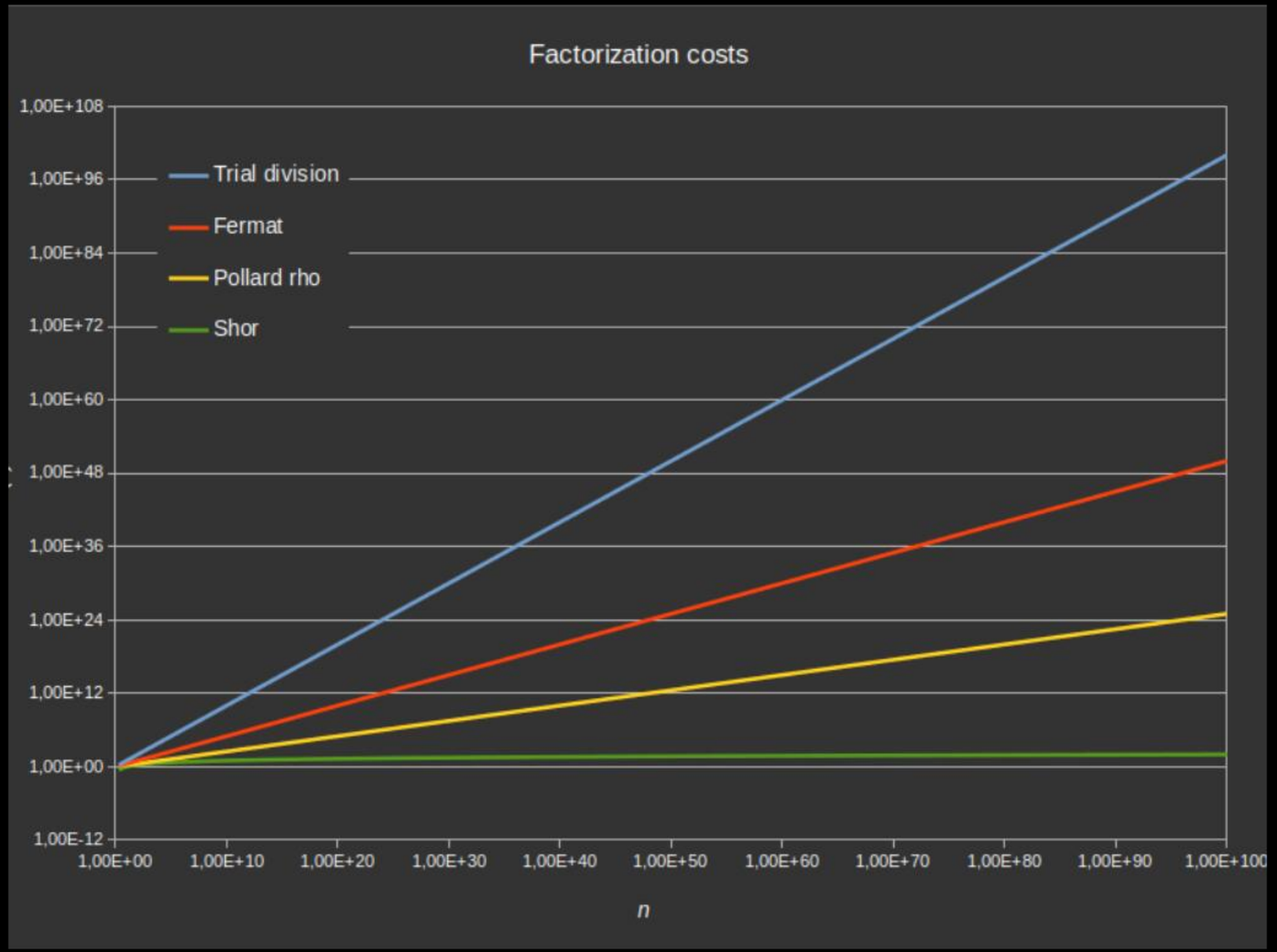


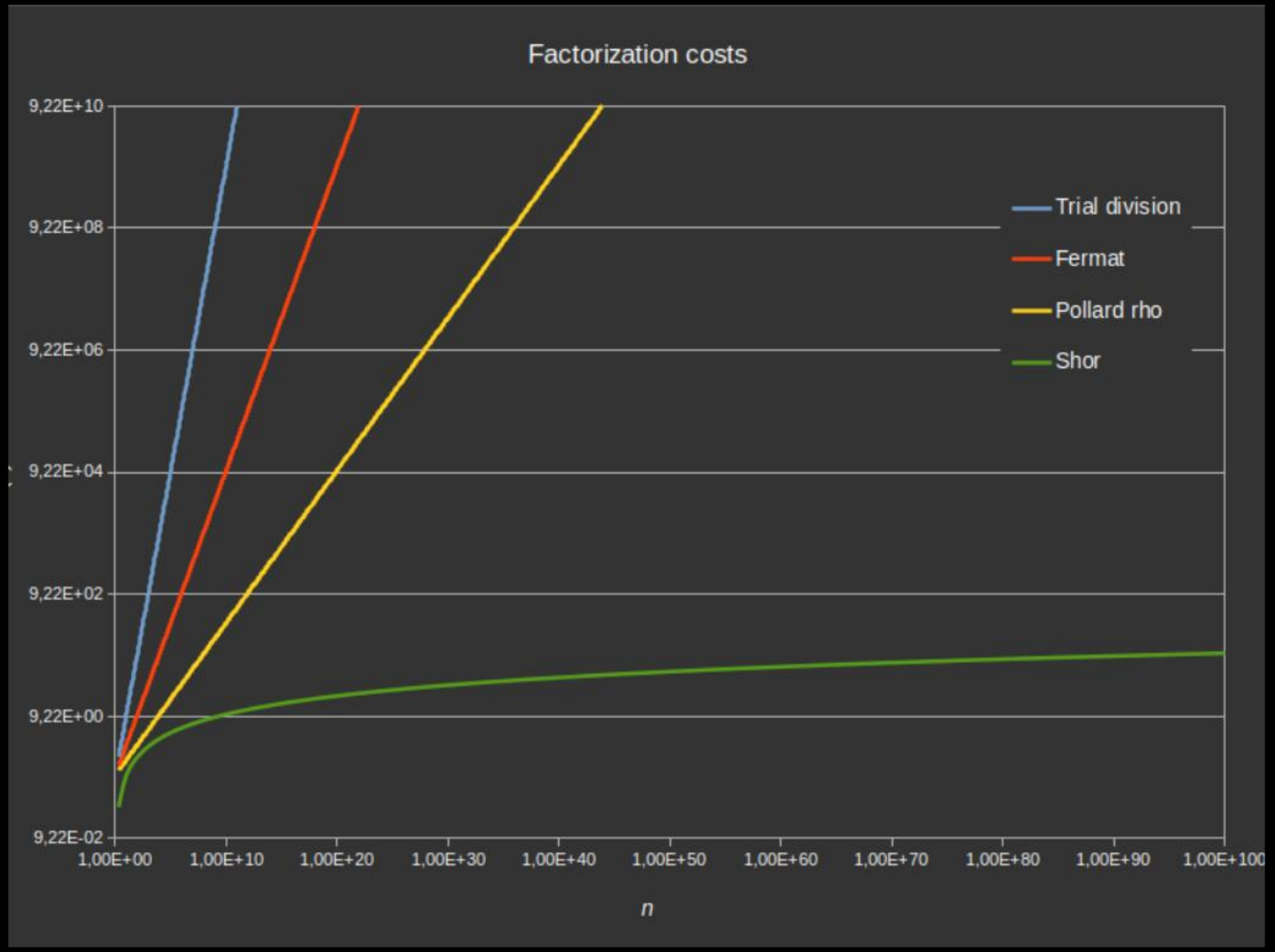
El algoritmo de Shor
circuito cuántico de QFT (8 qubits)



- previously...
- el problema
- RSA
- la factorización
- fourier
- el algoritmo de shor
- la implementación
- resumen


We are talking
about speed...


Factorization costs
1,00E+108
1,00E+96
1,00E+84
1,00E+72
1,00E+60
1,00E+48
1,00E+36
1,00E+24
1,00E+12
1,00E+00
1,00E-12
Trial division
Fermat
Pollard rho
Shor
1,00E+00
1,00E+10
1,00E+20
1,00E+30
1,00E+40
1,00E+50
1,00E+60
1,00E+70
1,00E+80
1,00E+90
1,00E+100
n


Factorization costs
9,22E+10
9,22E+08
9,22E+06
9,22E+04
9,22E+02
9,22E+00
9,22E-02
1,00E+00
1,00E+10
1,00E+20
1,00E+30
1,00E+40
1,00E+50
1,00E+60
1,00E+70
1,00E+80
1,00E+90
1,00E+100
n
Trial division
Fermat
Pollard rho
Shor

de qué hablamos...?

- previously...
- factorización de grandes números...
- Fourier su serie y su transformada
- reescritura de la ecuación de factorización
- periodicidad en la función
- Shor 'crea' la QFT para encontrar el período
- del período sale 'p'... y de p 'q'
- implementación de Shor en una QC
- no se puede factorizar N > 56153=233·241
- ...por ahora...

Referencias

How does RSA works
https://hackernoon.com/how-does-rsa-work-f44918df914b

Trial Division Algorithm
https://www.geeksforgeeks.org/trial-division-algorithm-for-prime-factorization/amp/

Fermat's Factorization Method
https://www.geeksforgeeks.org/fermats-factorization-method/

Pollard's Factorization Method
https://www.geeksforgeeks.org/pollards-rho-algorithm-prime-factorization/amp/

Fourier Transform in qiskit
https://leftasexercise.com/2019/02/25/implementing-the-quantum-fourier-transform-with-qiskit/

Quantum Factorization
https://towardsdatascience.com/quantum-factorization-b3f44be9d738

How Quantum Computers Break Encryption | Shor's Algorithm Explained
https://www.youtube.com/watch?v=FRZQ-efABeQ

How Shor's Algorithm Factors 314191
https://www.youtube.com/watch?v=lvTqbM5Dq4Q

New largest number factored on a quantum device is 56,153
https://phys.org/news/2014-11-largest-factored-quantum-device.html

Self
https://github.com/ch4r1i3b/Eko2021_The_Silent_Partners

¡muchas gracias!

Luciano Bello
1ucian0<at>ieee.org
@microluciano

Carlos Benitez
carlos<at>platinumciber.com
@ch4r1i3b

Ekoparty #17
2 de noviembre de 2021