

# *Quantum Factorization: Shor's algorithm*

*Quantum Capita Selecta*

Bernardo Villalba Frías, PhD

`b.r.villalba.frias@hva.nl`

- Rivest–Shamir–Adleman (RSA)
- Asymmetric–key system
  - Encryption key  $(e, N)$  is public
  - Decryption key  $(d)$  is private (and different)
- Based on the difficulty of the factorization of the product of two large prime numbers
- RSA algorithm involves four steps:
  - Key generation
  - Key distribution
  - Encryption, and
  - Decryption

- Alice encrypts a message  $M$  for Bob, which he has to decrypt
  - Encryption:
    - Obtain Bob's authentic public key  $(e, N)$
    - Pad  $M$  into  $m$
    - Encode the message:  $c = m^e \pmod{N}$
    - Send  $c$  to Bob
  - Decryption:
    - Use Bob's private key  $(d, N)$
    - Decode the message:  $m = c^d \pmod{N}$
    - De-pad  $m$  into  $M$

- Toy example:

Encryption key:  $(e, N)$

$(7, 15)$

Plain message:  $B \rightarrow 2$

Encoded message:  $2^7 \pmod{15}$

$128 \pmod{15}$

$8 \rightarrow H$

Decryption key:  $(d, N)$

$(23, 15)$

$8^{23} \pmod{15}$

$590295810358705651712 \pmod{15}$

$2 \rightarrow B$

- How do we generate the keys?
  1. Pick 2 prime numbers:  $p$  and  $q$

$$p = 3 \quad \wedge \quad q = 5$$

2. Obtain the modulus ( $N$ ):

$$\begin{aligned} N &= p \times q \\ &= 3 \times 5 \\ &= 15 \end{aligned}$$

3. Compute the function  $\Phi$ :

1	2	<del>3</del>	4	<del>5</del>
<del>6</del>	7	8	<del>9</del>	<del>10</del>
11	<del>12</del>	13	14	<del>15</del>

- Remove the numbers sharing a common factor with  $N$
- Keep only coprime with  $N$

$$C = \{1, 2, 4, 7, 8, 11, 13, 14\}$$

- How do we generate the keys?

3. Compute the function  $\Phi$ :

$$\begin{aligned}\Phi &= |C| \\ &= 8\end{aligned}$$

$$\begin{aligned}\Phi &= (p - 1) \times (q - 1) \\ &= (3 - 1) \times (5 - 1) \\ &= 2 \times 4 \\ &= 8\end{aligned}$$

4. Choose the encryption key ( $e$ ):

$$e = \begin{cases} 1 < e < \Phi \\ \text{coprime with } N \text{ and } \Phi \end{cases}$$

$$\text{coprime with } N \rightarrow e \in \{1, 2, 4, 7, 8, 11, 13, 14\}$$

$$1 < e < \Phi \rightarrow e \in \{2, 4, 7\}$$

$$\text{coprime with } \Phi \rightarrow e \in \{7\}$$

$$\Rightarrow (e, N) = (7, 15)$$

- How do we generate the keys?

5. Choose the decryption key ( $d$ ):

$$(d \times e) \pmod{\Phi} = 1$$

$$7d \pmod{8} = 1$$

$d$	$\rightarrow$	1	2	3	4	5	6	7	8	...
$7d$	$\rightarrow$	7	14	21	28	35	42	49	56	...
$7d \pmod{8}$	$\rightarrow$	7	6	5	4	3	2	1	0	...
								$\uparrow$		
								pick		

$$\rightarrow d \in \{7, 15, 23, 31, 39, \dots\}$$

$$\Rightarrow (d, N) = (23, 15)$$

- What about a more realistic example?
  - RSA-64:

$$p = 3970211251$$

$$q = 3873813143$$

$$N = 15379856524610271893$$

$$\Phi = 15379856516766247500$$

$$e = 15114048278816893619$$

$$d = 7635707568842743979$$



- Could you imagine RSA–4096?

```
6,992,279,463,099,306,513,415,800,460,317,519,890,444,896,931,045,397,932,378,348,225,853,676,792,140,441,678,927,628,875,456,695,024,506,006,332,626,365,194,962,240,453,639,352,597,632,032,589,188,462,141,
302,401,348,054,227,533,684,156,277,000,347,774,644,554,906,134,230,417,166,467,413,219,403,593,488,372,312,125,203,442,390,298,437,403,577,853,111,194,194,073,337,193,245,842,621,737,450,434,520,258,048,764,
745,324,287,720,818,622,371,924,503,705,133,543,897,936,223,512,988,969,472,832,152,278,521,237,265,459,888,516,313,452,283,340,922,585,409,923,962,202,001,816,710,554,500,547,973,295,462,450,900,608,383,198,
508,832,527,074,583,528,490,102,307,296,219,126,353,615,966,588,767,279,141,636,940,274,797,479,338,868,467,577,077,040,185,880,809,961,493,697,372,739,274,506,389,327,661,112,796,594,243,231,999,751,065,698,
221,122,093,945,938,153,794,335,420,948,898,380,618,543,875,275,305,915,903,137,507,781,571,030,725,956,338,741,630,099,402,346,557,511,395,955,348,213,164,544,819,539,861,611,260,066,779,617,072,171,274,236,
461,030,564,731,463,108,952,166,635,556,977,930,448,796,791,820,765,574,337,122,732,876,144,058,162,509,032,085,106,169,921,171,363,010,570,638,293,420,474,748,175,986,407,672,409,709,462,572,398,469,885,241,
136,827,783,903,553,607,615,860,304,004,678,260,488,181,419,668,941,548,126,659,869,282,357,195,261,075,765,993,158,569,755,459,695,855,779,838,519,150,400,678,997,539,754,753,620,115,891,706,483,333,102,727,
206,820,790,983,739,332,162,263,178,353,002,115,753,696,044,349,878,004,970,826,906,473,546,447,725,969,053,184,165,630,677,823,331,554,853,520,484,365,563,312,156,265,512,027,772,704,000,165,273,017,881,629,
322,645,084,015,737,503,938,308,637,219,196,946,991,281,480,219,697,353,770,968,409,150,636,207,505,499,687,872,610,706,551,662,688,369,435,010,005,223,929,553,909,894,961,694,936,984,813,150,984,853,928,733,
272,366,913,571,263,461,290,259,073,951,243,041,600,049,885,995,321,614,373,242,297,134,989,056,074,595,082,131,009,422,067,878,401,611,809,257,511,079,036,596,391,474,216,913,825,691,851,756,406,458,900,992,
452,193,614,942,226,229,267,834,529,562,766,859,797,289,560,557,008,367,906,697,561,658,204,923,257,957,542,893,608,902,316,867,574,460,647,152,207,143,506,972,269,723,597,269,684,792,602,420,424,12,803,728,
054,604,178,406,888,368,239,963,804,037,106,768,907,083,672,310,454,454,792,008,628,393,907,710,028,083,119,995,325,741,245,920,841,554,066,652,003,426,067,996,837,873,407,896,266,077,611,609,051,779,846,331,
132,310,665,942,838,672,142,892,387,046,969,680,276,296,369,719,330,271,890,336,299,545,000,804,876,159,738,728,851,140,778,102,381,072,526,544,481,501,722,189,148,758,458,147,824,387,259,572,079,408,550,505,
238,274,924,716,672,375,040,092,549,345,242,236,043,433,337,695,641,698,274,563,649,942,512,438,048,498,391,050,111,851,547,399,464,335,868,792,446,609,740,527,942,408,022,732,291,158,534,380,782,984,874,903,
734,594,682,640,370,253,644,738,490,174,114,868,044,841,363,039,584,963,346,431,569,117,223,233,992,891,032,375,459,679,726,017,306,363,948,847,311,296,864,664,829,582,413,242,829,254,966,415,059,452,814,265,
926,970,971,732,405,242,072,634,750,674,864,616,907,854,721,210,258,479,399,106,627,070,453,983,965,184,629,115,543,773,566,649,119,197,756,815,739,961,943,358,317,191,643,930,811,985,869,594,980,508,532,594,
770,602,813,598,592,010,937,241,400,263,041,502,271,041,567,641,785,394,554,558,934,958,600,811,691,583,044,870,056,816,646,027,246,480,517,336,467,156,136,550,764,185,309,202,071,460,225,586,921,971,205,726,
937,125,322,790,732,148,619,292,621,976,278,928,680,226,456,688,431,065,417,174,402,171,211,901,699,004,711,116,408,215,922,397,481,599,672,354,628,434,616,963,278,097,508,394,025,309,795,615,172,552,706,511,
335,157,038,435,346,663,736,621,325,121,361,377,897,308,179,215,608,218,895,702,881,920,839,329,792,878,228,433,988,077,114,328,278,108,615,652,093,531,528,483,542,465,164,914,231,061,515,474,340,685,234,123,
632,381,277,667,225,833,221,872,126,765,248,482,617,757,441,670,865,213,415,622,587,593,762,807,478,551,924,008,747,151,937,886,329,773,908,413,782,008,607,417,567,275,145,732,775,151,452,555,889,204,304,810,
037,160,023,568,879,388,647,754,364,578,201,087,523,198,146,728,324,917,724,707,627,335,814,044,944,509,225,580,348,611,677,539,057,072,724,455,505,877,156,848,165,927,030,952,494,705,055,413,441,621,866,484,
963,599,451,391,597,367,639,683,177,338,218,902,407,445,731,998,244,719,685,351,179,555,647,585,211,450,992,058,771,713,146,478,879,188,811,291,954,002,015,248,030,507,648,043,356,978,643,033,507,984,414,696,
868,346,901,806,615,025,923,026,574,584,086,649,333,668,657,071,384,157,937,052,336,638,621,185,925,655,174,227,548,342,944,705,332,502,578,208,062,958,135,570,156,698,003,520,427,680,524,985,360,686,955,163,
173,267,066,523,093,126,225,682,137,927,544,930,870,750,589,687,308,066,517,558,202,093,254,961,866,948,829,386,881,726,474,663,407,045,101,710,560,498,685,046,658,939,249,526,689,853,221,183,354,403,653,193,
170,915,208,560,649,167,848,695,333,852,263,797,814,344,951,098,273,351,068,894,433,130,126,650,222,421,368,818,855,623,456,656,390,563,845,764,254,708,465,562,369,840,220,402,169,722,197,810,178,223,218,318,
214,143,106,762,419,549,542,884,972,506,704,473,707,346,302,412,185,369,846,340,448,231,182,841,334,733,502,145,252,648,372,502,216,534,948,747,116,988,051,859,537,982,886,022,714,611,810,493,823,310,249,
245,322,558,003,709,734,613,513,935,693,443,262,381,525,289,616,054,454,502,761,397,654,830,200,303,186,350,576,861,535,587,404,211,485,045,595,793,533,650,726,110,470,607,721,865,591,987,814,970,049,649,092,
553,413,951,820,673,583,005,585,070,536,762,903,097,537,859,843,659,563,916,947,578,705,904,858,781,402,309,204,543,094,791,089,448,681,252,680,560,486,158,051,601,145,687,320,481,373,831,137,378,941,642,823,
711,763,179,577,067,929,878,071,284,740,815,934,735,499,133,059,762,858,467,879,340,629,735,457,005,678,508,090,770,674,644,191,495,915,246,475,471,440,300,118,509,153,379,359,242,362,827,810,750,626,082,095,
349,867,870,745,030,749,301,029,570,297,553,870,594,726,635,227,085,467,602,565,878,083,689,103,752,293,011,716,024,421,134,345,385,124,416,963,668,464,304,971,965,198,149,801,335,129,008,502,356,635,523,863,493,
658,251,934,079,933,824,322,890,962,991,371,786,998,564,825,839,127,353,147,971,735,467,071,505,777,855,868,246,336,800,060,399,102,768,528,066,286,092,157,524,569,115,231,568,564,381,056,950,317,023,486,997,
515,891,968,875,251,059,534,238,181,735,302,206,348,793,407,514,322,723,955,082,496,488,170,273,175,999,601,375,484,407,774,409,802,353,059,463,407,781,497,982,813,061,533,812,970,810,929,274,912,202,111,674,
316,293,029,967,329,068,961,306,345,906,022,664,579,612,785,311,159,652,676,988,504,553,505,440,457,105,930,889,166,477,535,457,928,862,396,584,339,446,112,789,912,758,223,093,529,667,543,811,785,024,352,893,
961,250,367,457,410,491,509,338,086,308,649,129,224,449,583,011,132,385,215,278,511,366,943,430,568,765,008,167,098,581,185,968,103,707,597,890,727,885,871,453,835,744,660,237,267,327,097,935,899,019,806,533,
818,129,327,914,792,498,822,326,975,253,591,462,237,090,828,287,009,240,557,237,363,791,683,668,363,966,877,574,904,063,387,995,975,732,537,357,096,369,911,842,539,638,399,261,035,320,244,345,856,130,779,838,
619,905,065,727,180,880,496,676,363,783,392,948,738,973,884,591,933,484,745,309,364,076,754,115,721,486,849,982,428,606,824,714,075,759,458,966,240,213,659,793,851,995,585,207,491,385,483,255,797,273,222,137,
639,144,361,762,475,341,638,568,451,888,429,589,374,090,169,186,703,984,643,881,768,293,276,404,095,379,227,279,226,030,277,174,319,142,041,219,771,046,905,788,307,970,708,796,593,995,048,386,983,290,229,705,
641,395,262,706,962,019,631,887,474,732,595,366,622,885,632,817,885,205,132,817,762,727,923,952,824,444,898,308,589,211,953,989,053,639,654,211,123,576,296,737,413,177,503,254,840,737,797,521,848,209,793,360,
672,507,232,884,644,695,259,293,484,882,351,701,627,718,030,488,647,224,729,013,259,069,207,173,772,293,011,716,024,421,134,345,385,124,416,963,668,464,304,971,965,198,149,801,335,129,008,502,356,635,523,863,493,
882,860,482,152,945,560,217,373,872,447,719,818,518,721,770,122,095,609,138,052,241,761,690,033,253,387,356,633,453,808,158,646,046,842,976,111,586,351,543,184,004,119,203,961,056,693,636,803,422,431,634,719,
323,849,217,256,452,070,685,222,774,892,968,778,322,217,503,458,935,065,651,357,990,526,391,486,530,700,806,761,492,249,258,587,206,721,271,650,490,019,875,722,738,723,797,039,694,533,243,263,289,988,756,241,
402,982,650,015,235,756,407,891,920,766,467,978,318,031,945,941,864,886,072,388,188,084,257,125,738,940,094,442,196,576,515,467,508,250,307,691,644,831,228,159,149,320,554,050,586,418,335,253,801,049,423,626,
046,725,086,043,977,854,358,317,739,055,430,625,311,862,224,252,362,460,177,015,895,680,900,269,523
```

- How does it break?
  - Public key  $(e, N)$
  - Factor  $N$  into  $p$  and  $q$ :

$$N = p \times q$$

- Compute  $\Phi$ :

$$\Phi = (p - 1) \times (q - 1)$$

- Obtain  $d$

$$d = \text{Inverse of } e \pmod{\Phi}$$

- Quantum algorithm to find the prime factors of any given integer  $N$
- It takes  $\mathcal{O}((\log N)^3)$  time:
  - Solved in polynomial time by a quantum computer
  - Making RSA vulnerable to attacks
- It consists of two parts:
  - Reducing the factoring problem into an order-finding problem
  - Solve the order-finding problem (Quantum period finding)

- Preliminary Theorem:
  - Suppose that:
    - $N$  is a non–prime number
    - $x$  is a solution of  $x^2 \pmod{N} = 1$
    - $x \pmod{N} \neq 1$  and  $x \pmod{N} \neq -1$
  - Then:
    - $\gcd(x - 1, N)$  and  $\gcd(x + 1, N)$  are non–trivial factors of  $N$



1. Choose a random integer  $a < N$ , such that they are co-prime

2. Determine the unknown period  $r$  of the function:

$$f(x) = a^x \pmod{N}$$

3. If  $r$  is odd, then go to Step 1; otherwise go to Step 4

4. Since  $r$  is even:

$$((a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)) \pmod{N} = 0$$

If  $(a^{\frac{r}{2}} + 1) \pmod{N} = 0$ , then go to Step 1

If  $(a^{\frac{r}{2}} + 1) \pmod{N} \neq 0$ , then go to Step 5

5. Compute:

$$p = \gcd(a^{\frac{r}{2}} - 1, N)$$

$$q = \gcd(a^{\frac{r}{2}} + 1, N)$$

- Toy example:

$$N = 15$$

$$a = 2$$

$$a^r \pmod{N} = 1$$

$$2^0 \pmod{15} = 1$$

$$2^1 \pmod{15} = 2$$

$$2^2 \pmod{15} = 4$$

$$2^3 \pmod{15} = 8$$

$$2^4 \pmod{15} = 1$$

$$\Rightarrow r = 4 \text{ (even)}$$

$$p = \gcd(a^{\frac{r}{2}} - 1, N)$$

$$= \gcd(2^2 - 1, 15)$$

$$= \gcd(3, 15)$$

$$= 3$$

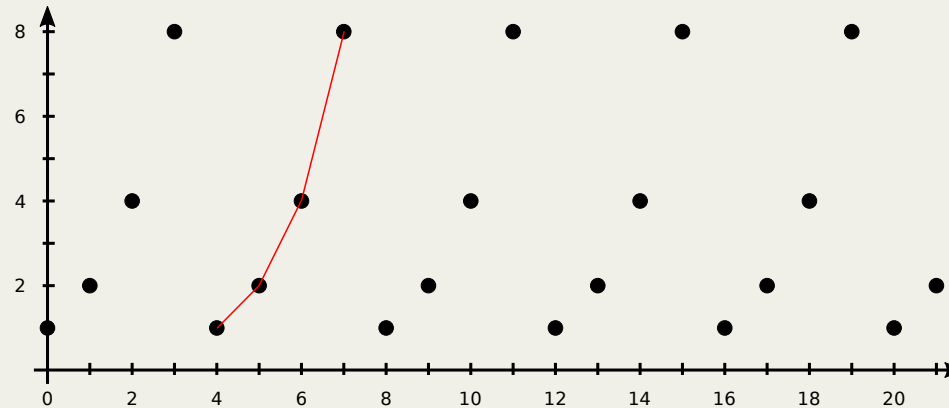
$$q = \gcd(a^{\frac{r}{2}} + 1, N)$$

$$= \gcd(2^2 + 1, 15)$$

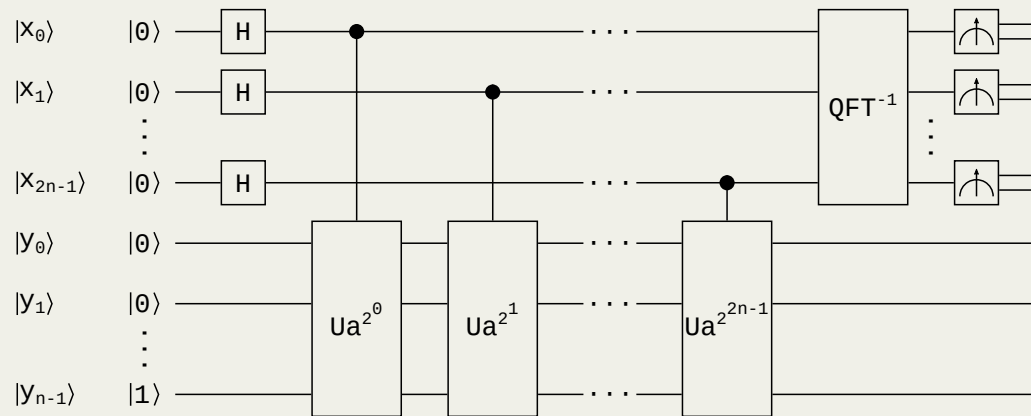
$$= \gcd(5, 15)$$

$$= 5$$

$$(2^2 + 1) \pmod{15} \neq 0$$

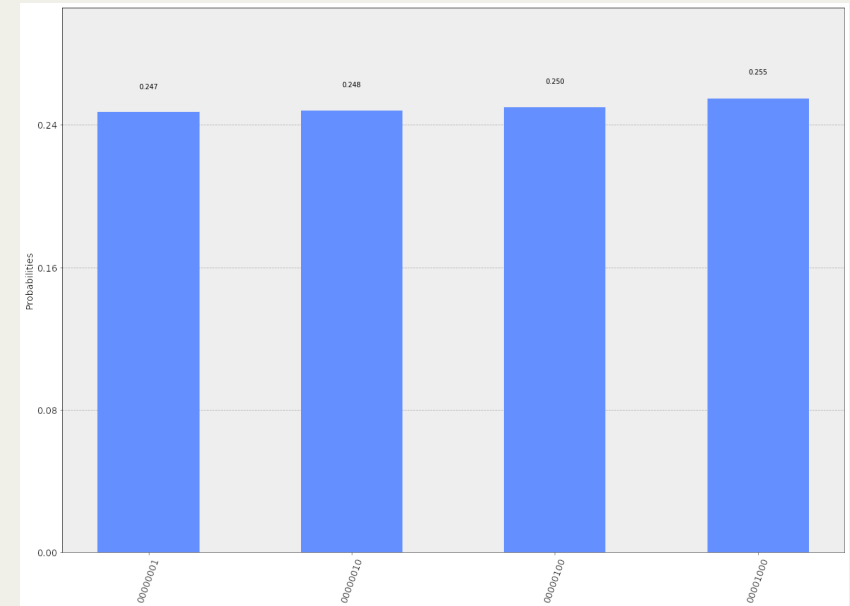
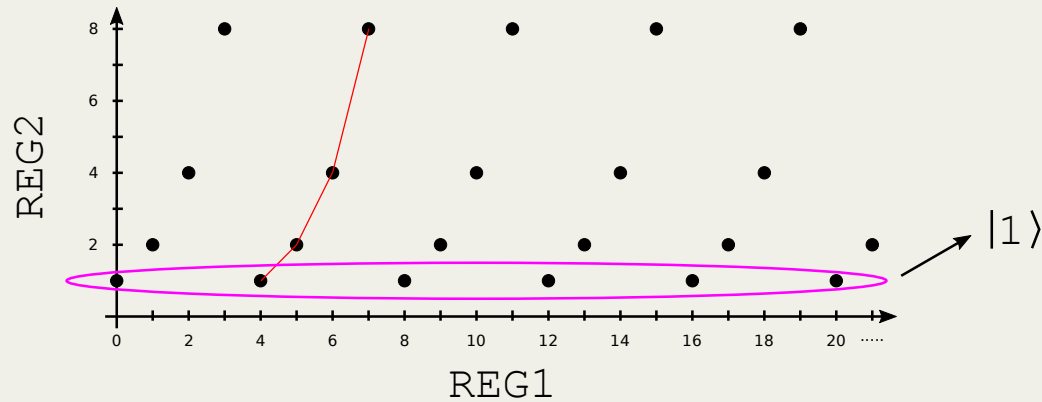


- Period finding:
  - Main building block of the Shor's algorithm
  - $f(x) = a^x \pmod{N}$  is periodic
  - Challenge: Find the period!... but, how?
    - Modular exponentiation
      - Resource expensive
    - Inverse Quantum Fourier Transform



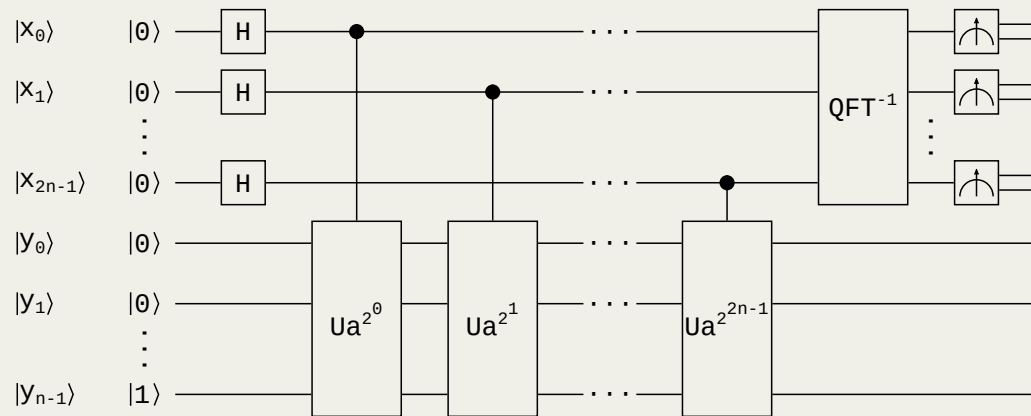
- Creates a quantum register with 2 parts:
  1. REG1: Superposition of  $x$  in  $a^x \pmod{N}$ 
    - $x \in [0, b - 1]$ , with  $N^2 < 2^b < 2N^2 \rightarrow 2^b = 256$
  2. REG2: Possible outcomes of  $a^x \pmod{N}$
- Measuring the second register (REG2) collapses:
  - REG2 into some observed value
  - REG1 into a state consistent with REG2





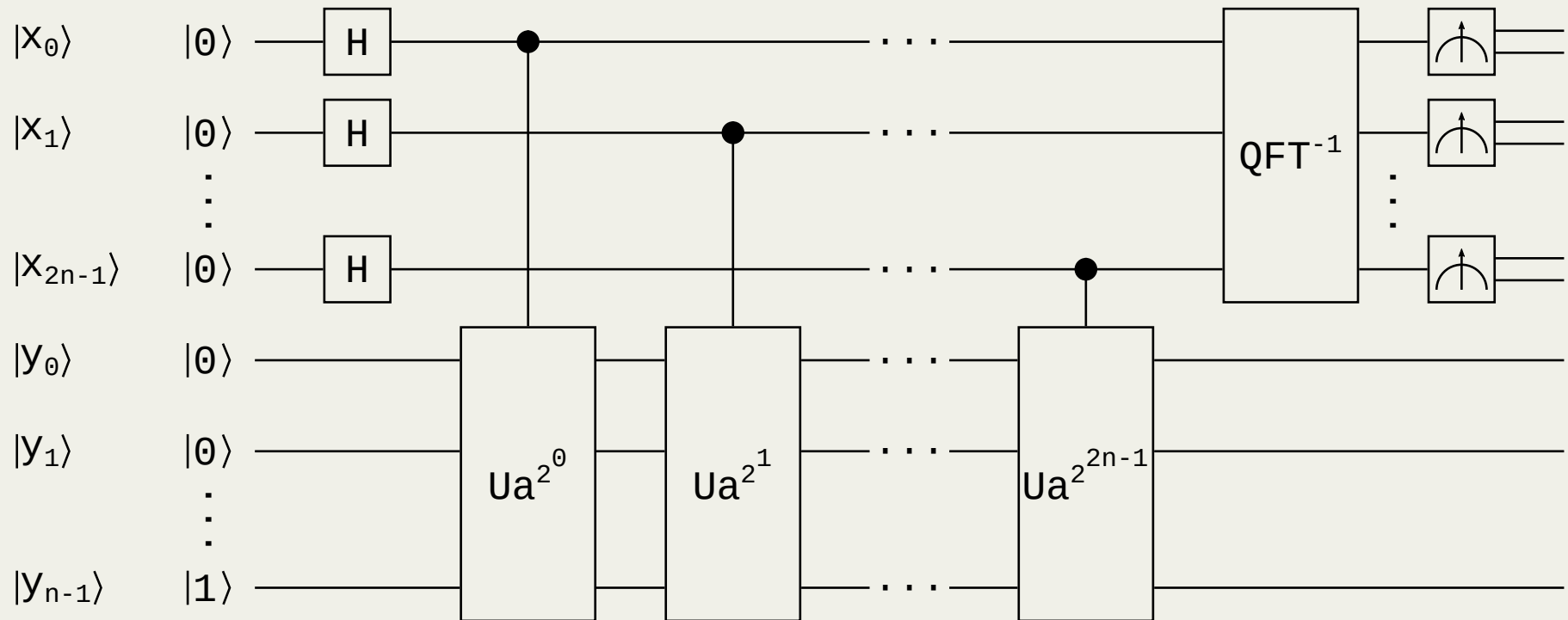
- Assume REG2 collapses to  $|1\rangle$ 
  - REG1 will collapse to

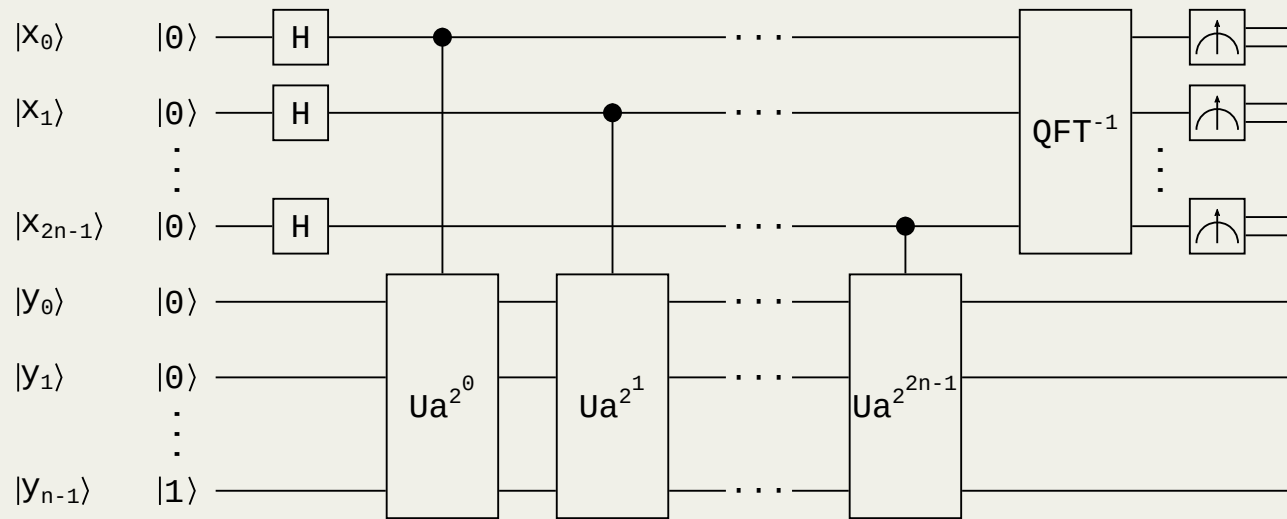
$$\frac{1}{\sqrt{64}} (|0\rangle + |4\rangle + |8\rangle + \dots)$$



- Inverse Quantum Fourier Transform on REG1
  - Peak the probabilities amplitudes at integer multiples of the  $\frac{2^b}{r}$
  - Measuring REG1 will collapse, with high probability, to a multiple of the inverse period
  - Additional processing required

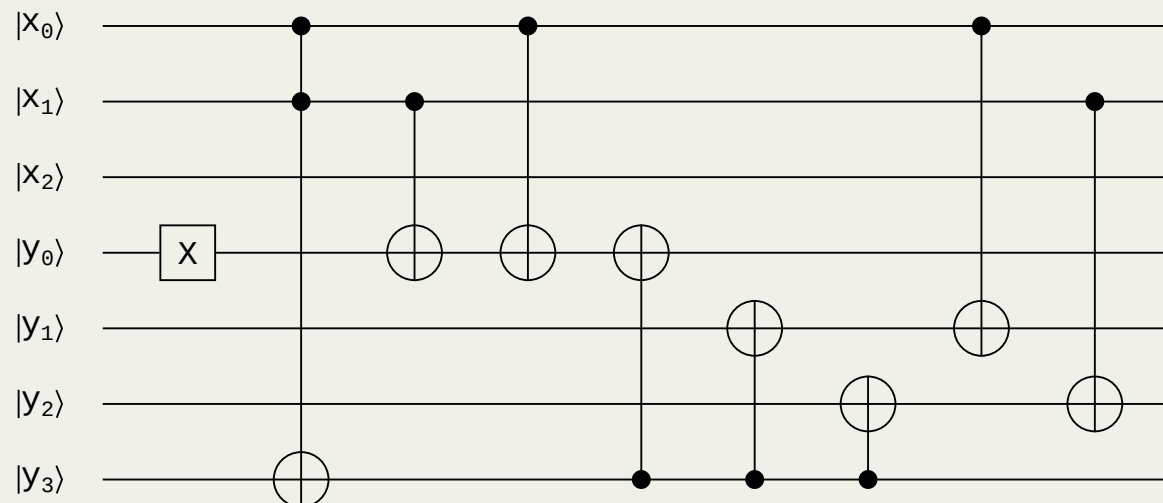
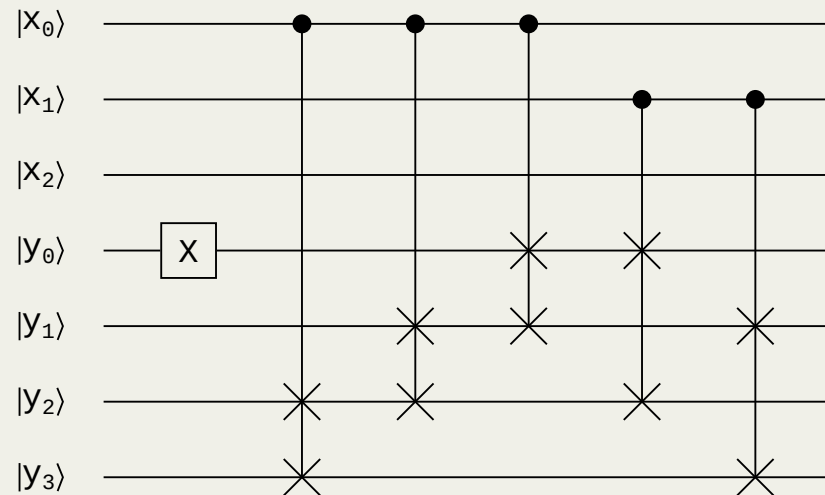
# Shor's algorithm



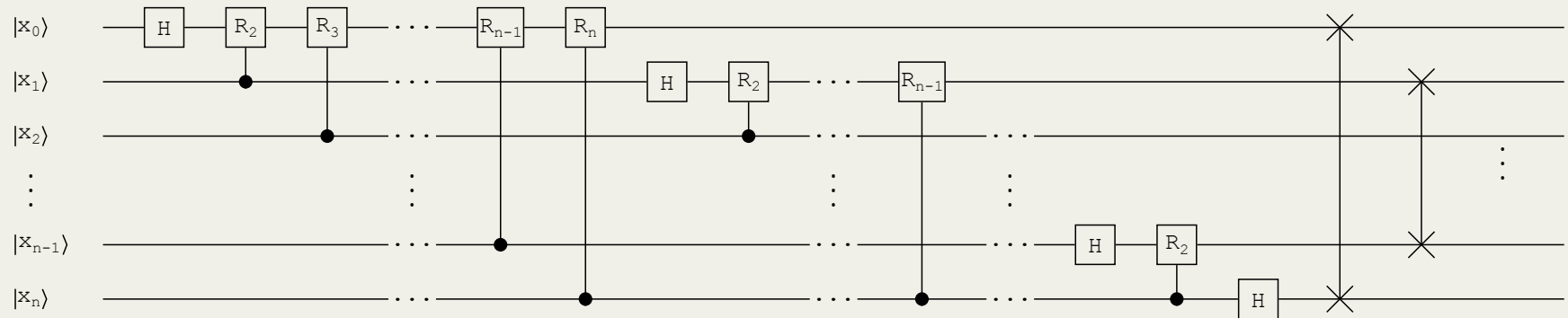


- Modular exponentiation:
  - $U_a : |x\rangle |0\rangle \rightarrow |x\rangle |a^x \pmod{N}\rangle$
  - Repeated squaring:  $x^n = (x^{\frac{n}{2}})^2$
  - Implementation depends on  $a$
  - How do we build a generic circuit for a random  $a$ ?
    - “Compiled” versions

- Compiled circuits for  $N = 15$  and  $a = 2$



- Quantum Fourier Transform:

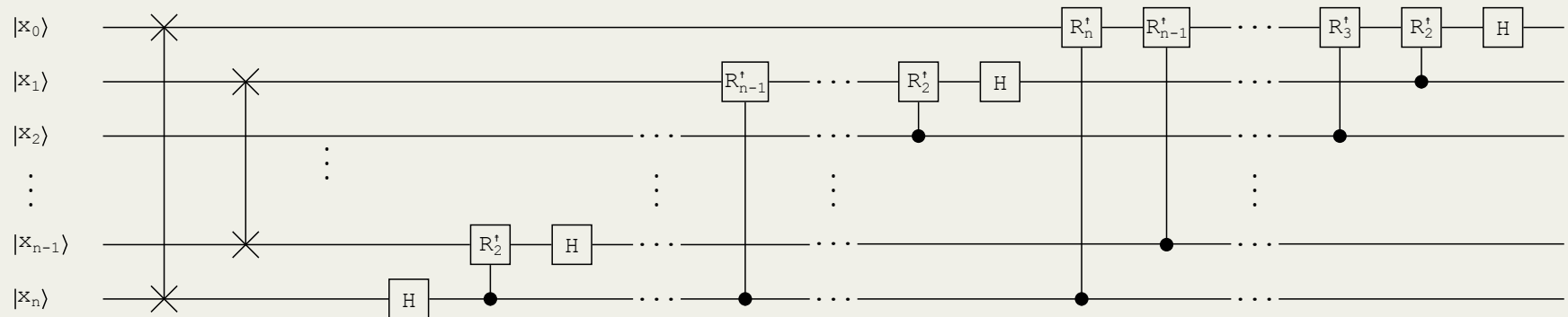


- Several quantum algorithms use the QFT
- Implemented by: Hadamard,  $CR_m$  and Swap gates

$$CR_m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\lambda i} \end{bmatrix}, \text{ where } \lambda = \frac{2\pi}{2^m}$$

- But, we need the inverse!

- Inverse Quantum Fourier Transform:



- Flip the gates' order: from right to left
- Also the gates should be inverted
  - $H = H^\dagger \rightarrow$  No problem!
  - $R_m \neq R_m^\dagger \rightarrow$  Obtain  $R_m^\dagger$ 
    - $\lambda$  has negative sign ( $-\lambda$ )
    - In Qiskit: `qc.cu1(-lambda, ctrl, tgt)`

- Continued fractions

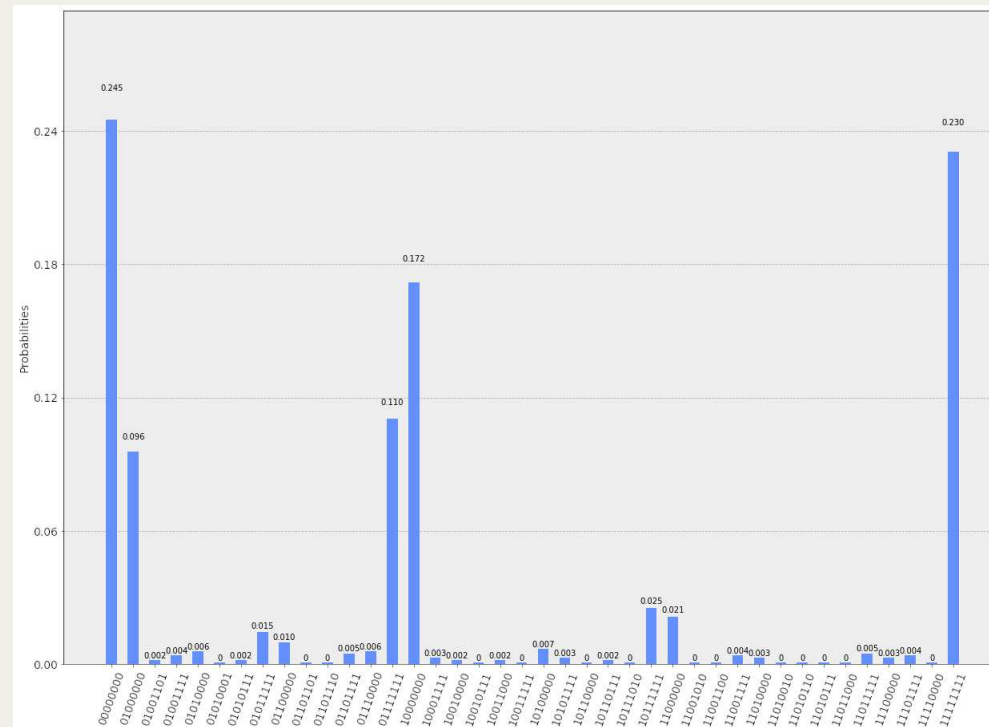
$$[a_0, \dots, a_M] = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ddots + \frac{1}{a_M}}}}$$

- Any rational number can be represented as a continued fraction:

$$\begin{aligned}\frac{13}{5} &= 2 + \frac{3}{5} \\ &= 2 + \frac{1}{\frac{5}{3}} \\ &= 2 + \frac{1}{1 + \frac{2}{3}} \\ &= 2 + \frac{1}{1 + \frac{1}{\frac{3}{2}}} \\ &= 2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}\end{aligned}$$



- Continuous fractions for post-processing



- Probable measurements:  $\{0, 64, 127, 128, 191, 255\}$
- Let's assume that we measured  $s = 191$

- Continuous fractions for post-processing
- Assume  $s = 191$

$$\begin{aligned}\frac{256}{191} &= 1 + \frac{1}{\frac{191}{65}} \\ &= 1 + \frac{1}{2 + \frac{1}{\frac{65}{61}}} \\ &= 1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{\frac{61}{4}}}} \\ &= 1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{15 + \frac{1}{4}}}}\end{aligned}$$

- Continuous fractions for post-processing
- Assume  $s = 191$ 
  - Coefficients:  $[1, 2, 1, 15, 4]$
  - Compute the  $m$ -th convergent:

$$\begin{aligned} 3\text{-rd} &= [1, 2, 1] \\ &= 1 + \frac{1}{2 + 1} \\ &= 1 + \frac{1}{3} \\ &= \frac{4}{3} \end{aligned}$$

- Continuous fractions for post-processing
- Assume  $s = 191$ 
  - Coefficients:  $[1, 2, 1, 15, 4]$
  - $m$ -th convergent:  $\left[1, \frac{3}{2}, \frac{4}{3}, \frac{63}{47}, \frac{256}{191}\right]$
- We are interested in the numerators:
  - Even
  - $a^x \pmod{N} = 1$
- Possible periods:  $[4, 256]$
- Compute  $\gcd(a^{\frac{r}{2}} \pm 1, N)$ 
  - $r = 4 \rightarrow p = 3$  and  $q = 5$

