量子計算的數學基礎 MA5501*

Chapter 6. Shor's Factoring Algorithm

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

§6.1 RSA Encryption

RSA is an asymmetric encryption (非對稱式加密) technique that uses two different keys as public and private keys to perform the encryption and decryption. The public key is represented by the integers n and e, and the private key by the integer d. A basic principle behind RSA is to find three very large positive integers e, d, and n, such that with modular exponentiation all messages $m \in \mathbb{N}$ with $0 \leqslant m < n$ satisfies

$$(m^e)^d \equiv m \pmod{n}$$

and that knowing e and n, or even m, it can be extremely difficult to find d.

§6.1 RSA Encryption

RSA is an asymmetric encryption (非對稱式加密) technique that uses two different keys as public and private keys to perform the encryption and decryption. The public key is represented by the integers n and e, and the private key by the integer d. A basic principle behind RSA is to find three very large positive integers e, d, and n, such that with modular exponentiation all messages $m \in \mathbb{N}$ with $0 \leqslant m < n$ satisfies

$$(m^e)^d \equiv m \pmod{n}$$

and that knowing e and n, or even m, it can be extremely difficult to find d.

§6.1 RSA Encryption

§6.1.1 Mathematical foundation

Definition (Greatest common divisor)

Let a and b be non-zero integers. We say the integer d is the **greatest common divisor (gcd)** of a and b, and write $d = \gcd(a, b)$, if

1. d is a common divisor of a and b.
2. every common divisor c of a and b is not greater than d.

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. Suppose that $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ for $2 \leqslant j \leqslant k$, where $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ and $q_j \in \mathbb{N}$ for all $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If $\{s_j\}_{j=-1}^{k}$ and $\{t_j\}_{j=-1}^{k}$ are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1\,, \\ 0 & \text{if } j = 0\,, \\ s_{j-2} - q_{j-1}s_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1\,, \\ 1 & \text{if } j = 0\,, \\ t_{j-2} - q_{j-1}t_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k\,.$$

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. Suppose that $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ for $2 \leqslant j \leqslant k$, where $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ and $q_j \in \mathbb{N}$ for all $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If $\{s_j\}_{j=-1}^{k}$ and $\{t_j\}_{j=-1}^{k}$ are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1\,, \\ 0 & \text{if } j = 0\,, \\ s_{j-2} - q_{j-1} s_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1\,, \\ 1 & \text{if } j = 0\,, \\ t_{j-2} - q_{j-1} t_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k\,.$$

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. Suppose that $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ for $2 \leqslant j \leqslant k$, where $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ and $q_j \in \mathbb{N}$ for all $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If $\{s_j\}_{j=-1}^{k}$ and $\{t_j\}_{j=-1}^{k}$ are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1\,, \\ 0 & \text{if } j = 0\,, \\ s_{j-2} - q_{j-1} s_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1\,, \\ 1 & \text{if } j = 0\,, \\ t_{j-2} - q_{j-1} t_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k\,.$$

§6.1 RSA Encryption

Proof.

Let a and b be positive integers with $a \leqslant b$. By the Division Algorithm, there exists positive integer q_1 and non-negative integer r_1 such that $b = aq_0 + r_1$ and $0 \leqslant r_1 < a$. If $r_1 = 0$, the lists terminate; otherwise, for $0 < r_1 < a$, there exists positive integer q_1 and non-negative integer r_2 such that $a = r_1q_1 + r_2$ and $0 \leqslant r_2 < r_1$. If $r_2 = 0$, the lists terminate; otherwise, for $0 < r_2 < r_1$, there exists positive integer q_2 and non-negative integer r_3 such that $r_1 = r_2q_2 + r_3$ and $0 \leqslant r_3 < r_2$. Continuing in this fashion, we obtain a strictly decreasing sequence of non-negative integers $r_1, r_2, r_3, \cdots$. This lists must end, so there is an integer k such that $r_{k+1} = 0$.

Therefore, with r_{-1} and r_0 denoting b and a respectively, we have

$$r_{-1} \geqslant r_0 > r_1 > r_2 > \cdots > r_k > r_{k+1} = 0\,,$$

$$r_{j-1} = r_jq_j + r_{j+1} \quad \text{for all } 0 \leqslant j \leqslant k\,.$$

□

§6.1 RSA Encryption

Proof.

Let a and b be positive integers with $a \leqslant b$. By the Division Algorithm, there exists positive integer q_1 and non-negative integer r_1 such that $b = aq_0 + r_1$ and $0 \leqslant r_1 < a$. If $r_1 = 0$, the lists terminate; otherwise, for $0 < r_1 < a$, there exists positive integer q_1 and non-negative integer r_2 such that $a = r_1 q_1 + r_2$ and $0 \leqslant r_2 < r_1$. If $r_2 = 0$, the lists terminate; otherwise, for $0 < r_2 < r_1$, there exists positive integer q_2 and non-negative integer r_3 such that $r_1 = r_2 q_2 + r_3$ and $0 \leqslant r_3 < r_2$. Continuing in this fashion, we obtain a strictly decreasing sequence of non-negative integers $r_1, r_2, r_3, \cdots$. This lists must end, so there is an integer k such that $r_{k+1} = 0$.

Therefore, with r_{-1} and r_0 denoting b and a respectively, we have

$$r_{-1} \geqslant r_0 > r_1 > r_2 > \cdots > r_k > r_{k+1} = 0\,,$$
$$r_{j-1} = r_j q_j + r_{j+1} \quad \text{for all } 0 \leqslant j \leqslant k\,.$$

□

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.
   (a) First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.
   (b) On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

By (a), r_k is a common divisor of a and b. By (b), the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.

   (a) First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.

   (b) On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

   By (a), r_k is a common divisor of a and b. By (b), the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.
   - ⓐ First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.
   - ⓑ On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

   By ⓐ, r_k is a common divisor of a and b. By ⓑ, the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j \,, \qquad (\star)$$

we note that

ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1+q_0q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.

ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j\,, \qquad (\star)$$

we note that

ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1+q_0q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.

ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j \,, \qquad (\star)$$

we note that

   ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

   ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1+q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.

   ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1} \,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,
$$at_j + bs_j = r_j\,, \qquad (\star)$$
we note that
   ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.
   ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1 + q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.
   ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then
$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,
$$at_j + bs_j = r_j\,, \qquad (\star)$$
we note that
   - ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.
   - ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1 + q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.
   - ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then
$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Remark: Let $a, b \in \mathbb{N}$ with $a \leqslant b$. The algorithm to compute $\gcd(a, b)$ given in part 1 of the previous theorem is caleed **Euclid's Algorithm** (輾轉相除法), and the algorithm to compute $x, y \in \mathbb{Z}$ so that $ax + by = \gcd(a, b)$ given in part 2 of the previous theorem is called **Extended Euclid's Algorithm**.

§6.1 RSA Encryption

Example

We compute $\gcd(32, 12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12, 32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Example

We compute $\gcd(32, 12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12, 32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Example

We compute $\gcd(32,12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12,32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Theorem

Let a and b be non-zero integers. The gcd of a and b is the smallest positive linear combination of a and b; that is,

$$\gcd(a,b) = \min\{am+bn \,|\, am+bn>0\,, m,n\in\mathbb{Z}\}\,.$$

Proof.

Let $d = am + bn$ be the smallest positive linear combination of a and b.

1. By the Division Algorithm, there exist integers q and r such that $a = dq + r$, where $0 \leqslant r < d$. Then

$$r = a - dq = a - (am+bn)q = a(1-m) + b(-nq)\,;$$

thus r is a linear combination of a and b. Since $0 \leqslant r < d$, we must have $r = 0$. Therefore, $a = dq$; thus $d|a$. Similarly, $d|b$; thus d is a common divisor of a and b. □

§6.1 RSA Encryption

Theorem

Let a and b be non-zero integers. The gcd of a and b is the smallest positive linear combination of a and b; that is,

$$\gcd(a,b) = \min\{am+bn \mid am+bn>0\,, m,n\in\mathbb{Z}\}\,.$$

Proof.

Let $d = am + bn$ be the smallest positive linear combination of a and b.

1. By the Division Algorithm, there exist integers q and r such that $a = dq + r$, where $0 \leqslant r < d$. Then
$$r = a - dq = a - (am+bn)q = a(1-m) + b(-nq)\,;$$
thus r is a linear combination of a and b. Since $0 \leqslant r < d$, we must have $r = 0$. Therefore, $a = dq$; thus $d|a$. Similarly, $d|b$; thus d is a common divisor of a and b. □

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\{k \in \mathbb{N} \,|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\{k \in \mathbb{N} \,\big|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\{k \in \mathbb{N} \,|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p \mid n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, where $p_1, \cdots, p_r$ are prime numbers and $k_1, \cdots, k_r \in \mathbb{N}$, one has

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ be such that $\gcd(m, n) = 1$. Then $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, *where* $p_1, \cdots, p_r$ *are prime numbers and* $k_1, \cdots, k_r \in \mathbb{N}$, *one has*

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ *be such that* $\gcd(m, n) = 1$. *Then* $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, where $p_1, \cdots, p_r$ are prime numbers and $k_1, \cdots, k_r \in \mathbb{N}$, one has

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ be such that $\gcd(m,n) = 1$. Then $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n | (a - b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a+b) \bmod n \text{ outputs } c\,,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \bmod n \text{ outputs } c\,,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n|(a-b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a+b) \bmod n \text{ outputs } c,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \bmod n \text{ outputs } c,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n | (a - b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a + b) \text{ mod } n \text{ outputs } c,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \text{ mod } n \text{ outputs } c,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$;
2. *there exists an additive identity* 0 *(that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$*), and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse (that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$*).*

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod{n}$ *and* $b \equiv d \pmod{n}$*. Then* $a \cdot b \equiv c \cdot d \pmod{n}$*.*

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$*. If* $a \cdot b \equiv a \cdot c \pmod{n}$*, then* $b \equiv c \pmod{n}$*.*

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$;
2. *there exists an additive identity* 0 (*that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$), *and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse* (*that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$).

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod{n}$ *and* $b \equiv d \pmod{n}$. *Then* $a \cdot b \equiv c \cdot d \pmod{n}$.

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *If* $a \cdot b \equiv a \cdot c \pmod{n}$, *then* $b \equiv c \pmod{n}$.

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$;
2. *there exists an additive identity* 0 (*that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$), *and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse* (*that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$).

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod{n}$ *and* $b \equiv d \pmod{n}$. *Then* $a \cdot b \equiv c \cdot d \pmod{n}$.

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *If* $a \cdot b \equiv a \cdot c \pmod{n}$, *then* $b \equiv c \pmod{n}$.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$.

Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2m = p(q_1 - q_2m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$.

Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2m = p(q_1 - q_2m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Theorem

The integers coprime to n from the set $\{0, 1, \cdots, n-1\}$ *of n non-negative integers form a group under multiplication modulo n. In other words, let S be a subset of* $\mathbb{Z}_n$ *consisting of numbers coprime to n; that is,* $S = \{k \in \mathbb{N} \mid 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$. *Then* $(S, \odot)$ *is a group; that is,*

1. *S is closed under multiplication* $\odot$*;*
2. *there exists an multiplicative identity* 1 *(that is,* $a \odot 1 = a$ *for all* $a \in S$*), and*
3. *every element in S has an multiplicative inverse element (that is, for each* $a \in S$ *there exists* $b \in S$ *such that* $a \odot b = 1$*).*

§6.1 RSA Encryption

Theorem

The integers coprime to n from the set $\{0, 1, \cdots, n-1\}$ *of n non-negative integers form a group under multiplication modulo n. In other words, let S be a subset of* $\mathbb{Z}_n$ *consisting of numbers coprime to n; that is,* $S = \{k \in \mathbb{N} \mid 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$. *Then* $(S, \odot)$ *is a group; that is,*

1. *S is closed under multiplication* $\odot$*;*
2. *there exists an multiplicative identity* 1 *(that is,* $a \odot 1 = a$ *for all* $a \in S$*), and*
3. *every element in S has an multiplicative inverse element (that is, for each* $a \in S$ *there exists* $b \in S$ *such that* $a \odot b = 1$*).*

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b \bmod n$ is coprime to n as well. Therefore, $a \odot b \in S$.

3. Let $a \in S$. Then the set $a \odot S \equiv \left\{a \odot s \,\middle|\, s \in S\right\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$. This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b \bmod n$ is coprime to n as well. Therefore, $a \odot b \in S$.

3. Let $a \in S$. Then the set $a \odot S \equiv \{a \odot s \,|\, s \in S\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$.

   This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b \bmod n$ is coprime to n as well. Therefore, $a \odot b \in S$.
3. Let $a \in S$. Then the set $a \odot S \equiv \{a \odot s \,|\, s \in S\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$. This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Definition

The multiplicative group of integers modulo n (given in the previous theorem) is denoted by $(\mathbb{Z}_n^*, \odot)$.

Theorem

Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^*$. If $a \cdot x + n \cdot y = 1$ for some $x, y \in \mathbb{Z}$, then

$$a^{-1} \equiv x \pmod{n},$$

where a^{-1} denotes the unique number in $\mathbb{Z}_n^*$ satisfying

$$a \odot a^{-1} = a^{-1} \odot a = 1.$$

§6.1 RSA Encryption

Definition

The multiplicative group of integers modulo n (given in the previous theorem) is denoted by $(\mathbb{Z}_n^*, \odot)$.

Theorem

Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^*$. If $a \cdot x + n \cdot y = 1$ for some $x, y \in \mathbb{Z}$, then

$$a^{-1} \equiv x \ (mod \ n)\,,$$

where a^{-1} denotes the unique number in $\mathbb{Z}_n^*$ satisfying

$$a \odot a^{-1} = a^{-1} \odot a = 1\,.$$

§6.1 RSA Encryption

Theorem

Let $a, n \in \mathbb{N}$ be such that $\gcd(a, n) = 1$. Then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof.

Let $a\mathbb{Z}_n^*$ be the set $a\mathbb{Z}_n^* \equiv \left\{a \cdot s \,\middle|\, s \in \mathbb{Z}_n^*\right\}$. Then the set $a\mathbb{Z}_n^*$ mod $n \equiv \left\{(a \cdot s) \bmod n \,\middle|\, s \in \mathbb{Z}_n^*\right\}$ is identical to $\mathbb{Z}_n^*$. Therefore,

$$\prod_{k \in \mathbb{Z}_n^*} k \equiv \prod_{k \in a\mathbb{Z}_n^*} k \pmod{n}.$$

Since $\prod_{k \in a\mathbb{Z}_n^*} k = a^{\varphi(n)} \prod_{k \in \mathbb{Z}_n^*} k$ and $\prod_{k \in \mathbb{Z}_n^*} k$ is coprime to n, by the cancellation law for $\mathbb{Z}_n$ we find that $a^{\varphi(n)} \equiv 1 \pmod{n}$. □

§6.1 RSA Encryption

Theorem

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *Then* $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof.

Let $a\mathbb{Z}_n^*$ be the set $a\mathbb{Z}_n^* \equiv \{a \cdot s \,|\, s \in \mathbb{Z}_n^*\}$. Then the set $a\mathbb{Z}_n^*$ mod $n \equiv \{(a \cdot s) \bmod n \,|\, s \in \mathbb{Z}_n^*\}$ is identical to $\mathbb{Z}_n^*$. Therefore,

$$\prod_{k \in \mathbb{Z}_n^*} k \equiv \prod_{k \in a\mathbb{Z}_n^*} k \pmod{n}.$$

Since $\prod_{k \in a\mathbb{Z}_n^*} k = a^{\varphi(n)} \prod_{k \in \mathbb{Z}_n^*} k$ and $\prod_{k \in \mathbb{Z}_n^*} k$ is coprime to n, by the cancellation law for $\mathbb{Z}_n$ we find that $a^{\varphi(n)} \equiv 1 \pmod{n}$. □

§6.1 RSA Encryption

Corollary (Fermat little theorem)

Let p be a prime number, and $a \in \mathbb{N}$ satisfy $\gcd(a, p) = 1$. Then $a^{p-1} \equiv 1 \pmod{p}$.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   (b) p and q are kept secret.
2. Compute $n = pq$.
   (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   (b) n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   - (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   - (b) p and q are kept secret.
2. Compute $n = pq$.
   - (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   - (b) n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   - ⓐ For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   - ⓑ p and q are kept secret.
2. Compute $n = pq$.
   - ⓐ n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   - ⓑ n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   - ⓐ For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   - ⓑ p and q are kept secret.
2. Compute $n = pq$.
   - ⓐ n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   - ⓑ n is released as part of the public key.

§6.1 RSA Encryption

3. Compute $\varphi(n)$, where φ is the Euler function. By previous proposition, $\varphi(n) = (p-1)(q-1)$. $\varphi(n)$ is kept secret.
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; that is, e and $\varphi(n)$ are coprime.
   - (a) e having a short bit-length and small Hamming weight results in more efficient encryption - the most commonly chosen value for e is $2^{16} + 1 = 65537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
   - (b) e is released as part of the public key.

§6.1 RSA Encryption

3. Compute $\varphi(n)$, where φ is the Euler function. By previous proposition, $\varphi(n) = (p-1)(q-1)$. $\varphi(n)$ is kept secret.
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; that is, e and $\varphi(n)$ are coprime.
   - ⓐ e having a short bit-length and small Hamming weight results in more efficient encryption - the most commonly chosen value for e is $2^{16} + 1 = 65537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
   - ⓑ e is released as part of the public key.

§6.1 RSA Encryption

5. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; that is, d is the modular multiplicative inverse of e modulo $\varphi(n)$.
   - ⓐ This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\varphi(n)}$; d can be computed efficiently by using the extended Euclidean algorithm.
   - ⓑ d is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.

§6.1 RSA Encryption

5. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; that is, d is the modular multiplicative inverse of e modulo $\varphi(n)$.
   - ⓐ This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\varphi(n)}$; d can be computed efficiently by using the extended Euclidean algorithm.
   - ⓑ d is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.

§6.1 RSA Encryption

Remark:

1. In modern RSA implementation the use of Euler function φ is replaced by Carmichael's totient function λ defined by
$$\lambda(n) = \min\left\{k \in \mathbb{N} \,\middle|\, a^k \equiv 1 \pmod{n} \text{ for all } a \in \mathbb{Z}_n^*\right\}.$$
If $n = pq$ with prime numbers p and q, then $\lambda(n) = \text{rm lcm}(p-1, q-1)$, the least common multiple of $p-1$ and $q-1$.

2. If both n and $\varphi(n)$ are known, then two primes p and q satisfying
$$n = pq\,, \quad \varphi(n) = (p-1)(q-1)$$
can be solved easily since p and q are zeros of
$$x^2 + \big[\varphi(n) - (n+1)\big]x + n = 0\,.$$

§6.1 RSA Encryption

Remark:

1. In modern RSA implementation the use of Euler function φ is replaced by Carmichael's totient function λ defined by
$$\lambda(n) = \min\left\{k \in \mathbb{N} \,\middle|\, a^k \equiv 1 \pmod{n} \text{ for all } a \in \mathbb{Z}_n^*\right\}.$$
If $n = pq$ with prime numbers p and q, then $\lambda(n) = \text{rm lcm}(p-1, q-1)$, the least common multiple of $p-1$ and $q-1$.
2. If both n and $\varphi(n)$ are known, then two primes p and q satisfying
$$n = pq\,, \quad \varphi(n) = (p-1)(q-1)$$
can be solved easily since p and q are zeros of
$$x^2 + \big[\varphi(n) - (n+1)\big]x + n = 0\,.$$

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Decryption**: Alice can recover m from c by using her private key exponent d by computing

$$c^d \equiv (m^e)^d \equiv m \pmod{n}.$$

Given m, she can recover the original message M by reversing the padding scheme.

Example

Here is an toy example of RSA encryption and decryption.

1. Choose two prime numbers $p = 11$ and $q = 31$.
2. Compute $n = pq = 341$.
3. Compute $\varphi(n) = (p-1)(q-1) = 300$ / $(\lambda(n) = \mathrm{lcm}(10, 30) = 30)$.
4. Choose the encryption key $e = 17$ so that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$ / $(1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1)$.

§6.1 RSA Encryption

- **Decryption**: Alice can recover m from c by using her private key exponent d by computing

$$c^d \equiv (m^e)^d \equiv m \pmod{n}.$$

Given m, she can recover the original message M by reversing the padding scheme.

Example

Here is an toy example of RSA encryption and decryption.

1. Choose two prime numbers $p = 11$ and $q = 31$.
2. Compute $n = pq = 341$.
3. Compute $\varphi(n) = (p-1)(q-1) = 300$ / $(\lambda(n) = \mathrm{lcm}(10, 30) = 30)$.
4. Choose the encryption key $e = 17$ so that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$ / $(1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1)$.

§6.1 RSA Encryption

Example (cont'd)

5. Compute the decryption key d by Extended Euclid's algorithm:

j	r_j	q_j	s_j	t_j
-1	300		1	0
0	17	17	0	1
1	11	1	1	-17
2	6	1	-1	18
3	5	1	2	-35
4	1	5	-3	53

j	r_j	q_j	s_j	t_j
-1	30		1	0
0	17	1	0	1
1	13	1	1	-1
2	4	3	-1	2
3	1	4	4	-7

which implies that $300\times(-3)+17\times 53 = 1$ $(30\times 4+17\times(-7) = 1)$; thus $d = 53$ ($d \equiv -7 \pmod{30}$ or $d = 23$).

§6.1 RSA Encryption

Example (cont'd)

Therefore, to encrypt $m = 30$, we raise to the power of 17 and obtain the encrypted message:

$$30^{17} \equiv 123 \pmod{341}.$$

To decrypt the encrypted message, we raise it to the power of 53 (23) and obtain that

$$123^{53} \equiv (123^3)^{17} \cdot 123^2 \equiv 30^{17} \cdot 125 \equiv 123 \cdot 125 \equiv 30 \pmod{341}$$

$$(123^{23} \equiv (123^3)^7 \cdot 123^2 \equiv 30^7 \cdot 125 \equiv 123 \cdot 125 \equiv 30 \pmod{341}).$$

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. This r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). Assuming that N is odd and not a prime power (those cases are easy to factor anyway), it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N. In that case we have:

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. This r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). Assuming that N is odd and not a prime power (those cases are easy to factor anyway), it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N. In that case we have:

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. This r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). Assuming that N is odd and not a prime power (those cases are easy to factor anyway), it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N. In that case we have:

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

§6.2 Reduction from Factoring to Period-finding

Note that $k > 0$ because both $x^{r/2} + 1 > 0$ and $x^{r/2} - 1 > 0$ since $x > 1$. Hence $x^{r/2} + 1$ or $x^{r/2} - 1$ will share a factor with N. Because $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N this factor will be less than N, and in fact both these numbers will share a non-trivial factor with N. Accordingly, if we have r then we can compute the greatest common divisors $\gcd(x^{r/2} + 1, N)$ and $\gcd(x^{r/2} - 1, N)$, and both of these two numbers will be non-trivial factors of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Note that $k > 0$ because both $x^{r/2} + 1 > 0$ and $x^{r/2} - 1 > 0$ since $x > 1$. Hence $x^{r/2}+1$ or $x^{r/2}-1$ will share a factor with N. Because $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N this factor will be less than N, and in fact both these numbers will share a non-trivial factor with N. Accordingly, if we have r then we can compute the greatest common divisors $\gcd(x^{r/2}+1, N)$ and $\gcd(x^{r/2}-1, N)$, and both of these two numbers will be non-trivial factors of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Thus the problem of factoring reduces to finding the period r of the function given by modular exponentiation $f(a) = x^a \bmod N$. In general, the period-finding problem can be stated as follows:

The period-finding problem: We are given some function $f : \mathbb{N} \to \{0, 1, \cdots, N-1\}$ with the property that there is some unknown $r \in \{0, 1, \cdots, N-1\}$ such that $f(a) = f(b)$ if and only if $a \equiv b \bmod r$. The goal is to find r.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log \log N)$ evaluations of f (query) and $\mathcal{O}(\log \log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding efficiently. We will show below how we can solve this problem efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

How many steps (elementary gates) does Shor's algorithm take? For $a = N^{\mathcal{O}(1)}$, we can compute $f(a) = x^a \bmod N$ in

$$\mathcal{O}((\log N)^2 \log\log N \log\log\log N)$$

steps by the "square-and-multiply" method, using known algorithms for fast integer multiplication mod N.

Moreover, as explained in the previous chapter, the quantum Fourier transform can be implemented using $\mathcal{O}((\log N)^2)$ steps. Accordingly, Shor's algorithm finds a factor of N using an expected number of $\mathcal{O}((\log N)^2(\log\log N)^2 \log\log\log N)$ gates, which is only slightly worse than quadratic in the input length.

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) U|x^k \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) U|x^k \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi isk}{r}\Big)|x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi isk}{r}\Big)|x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s(k+1)}{r}\Big) |x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s(k+1)}{r}\Big) |x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \text{ mod } N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \text{ mod } N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \text{ mod } N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big)|x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big)\frac{1}{\sqrt{r}}\sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big)|x^\ell \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big)|x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big)\frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big)|x^\ell \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big)|\psi_s\rangle\,.$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big)|x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big)|\psi_s\rangle\,.$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-finding Algorithm

Now we will show how Shor's algorithm finds the period r of the function f, given a "black-box" that maps $|a\rangle|0^n\rangle \mapsto |a\rangle|f(a)\rangle$. We can always efficiently pick some $q = 2^\ell$ such that $N^2 < q \leqslant 2N^2$. Then we can implement the Fourier transform QFT using $\mathcal{O}((\log N)^2)$ gates. Let O_f denote the unitary that maps $|a\rangle|0^n\rangle \mapsto |a\rangle|f(a)\rangle$, where the first register consists of ℓ qubits, and the second of $n = \lceil \log N \rceil$ qubits.

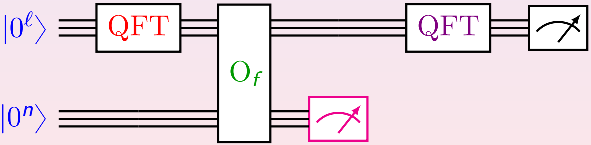

Figure 1: Shor's period-finding algorithm

§6.3 Shor's Period-finding Algorithm

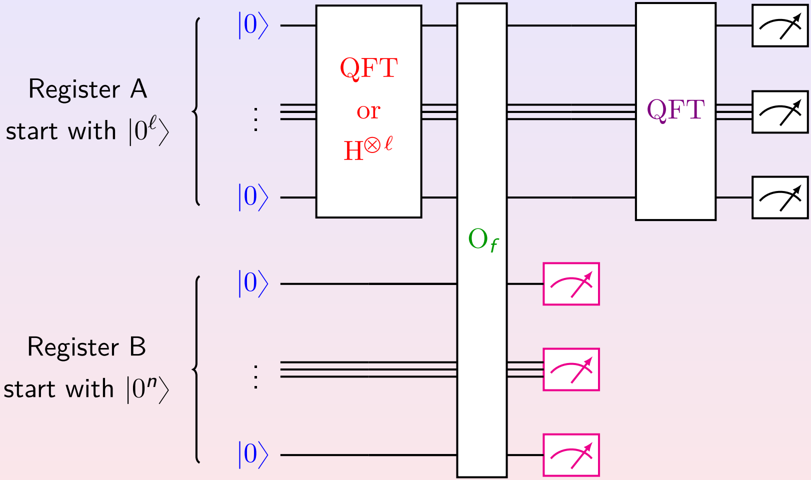

Figure 2: Shor's period-finding algorithm

§6.3 Shor's Period-finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|0^\ell\rangle|0^n\rangle$. Apply the QFT (or just ℓ Hadamard gates) to the first register to build the uniform superposition

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|0^n\rangle\,.$$

The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|f(a)\rangle\,.$$

Measuring the second register gives some value $f(s)$, with $0 \leqslant s < r$. Let m be the number of elements of $\{0,1,\cdots,q-1\}$ that map to the observed value $f(s)$; that is,

$$m = \#\big\{a \in \{0,1,\cdots,q-1\}\,\big|\, f(a) = f(s)\big\}\,.$$

§6.3 Shor's Period-finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|0^\ell\rangle|0^n\rangle$. Apply the QFT (or just ℓ Hadamard gates) to the first register to build the uniform superposition

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|0^n\rangle\,.$$

The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|f(a)\rangle\,.$$

Measuring the second register gives some value $f(s)$, with $0 \leqslant s < r$. Let m be the number of elements of $\{0,1,\cdots,q-1\}$ that map to the observed value $f(s)$; that is,

$$m = \#\big\{a \in \{0,1,\cdots,q-1\}\,\big|\,f(a)=f(s)\big\}\,.$$

§6.3 Shor's Period-finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|0^\ell\rangle|0^n\rangle$. Apply the QFT (or just ℓ Hadamard gates) to the first register to build the uniform superposition

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|0^n\rangle\,.$$

The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|f(a)\rangle\,.$$

Measuring the second register gives some value $f(s)$, with $0 \leqslant s < r$. Let m be the number of elements of $\{0,1,\cdots,q-1\}$ that map to the observed value $f(s)$; that is,

$$m = \#\big\{a \in \{0,1,\cdots,q-1\}\,\big|\,f(a)=f(s)\big\}\,.$$

§6.3 Shor's Period-finding Algorithm

Because $f(a) = f(s)$ if and only if $a \equiv s \pmod{r}$, the a of the form $a = jr + s$ $(0 \leqslant j < m)$ are exactly the a for which $f(a) = f(s)$. Thus the first register collapses to a superposition of $|s\rangle$, $|r+s\rangle$, $|2r+s\rangle$, $|3r+s\rangle$ $\cdots$; this superposition runs until the last number of the form $jr+s$ that is less than q. Since m is the number of elements in this superposition; that is, the number of integers j such that $jr+s \in \{0, 1, \cdots, q-1\}$, (depending on s) we must have $m = [q/r]$ or $m = [q/r]+1$. The second register collapses to the classical state $|f(s)\rangle$. We can now ignore the second register, and have in the first register:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr+s\rangle .$$

§6.3 Shor's Period-finding Algorithm

Because $f(a) = f(s)$ if and only if $a \equiv s \pmod{r}$, the a of the form $a = jr + s$ $(0 \leqslant j < m)$ are exactly the a for which $f(a) = f(s)$. Thus the first register collapses to a superposition of $|s\rangle$, $|r+s\rangle$, $|2r+s\rangle$, $|3r+s\rangle$ $\cdots$; this superposition runs until the last number of the form $jr + s$ that is less than q. Since m is the number of elements in this superposition; that is, the number of integers j such that $jr + s \in \{0, 1, \cdots, q-1\}$, (depending on s) we must have $m = [q/r]$ or $m = [q/r] + 1$. The second register collapses to the classical state $|f(s)\rangle$. We can now ignore the second register, and have in the first register:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + s\rangle .$$

§6.3 Shor's Period-finding Algorithm

Because $f(a) = f(s)$ if and only if $a \equiv s \pmod{r}$, the a of the form $a = jr + s$ $(0 \leqslant j < m)$ are exactly the a for which $f(a) = f(s)$. Thus the first register collapses to a superposition of $|s\rangle$, $|r+s\rangle$, $|2r+s\rangle$, $|3r+s\rangle$ $\cdots$; this superposition runs until the last number of the form $jr + s$ that is less than q. Since m is the number of elements in this superposition; that is, the number of integers j such that $jr + s \in \{0, 1, \cdots, q-1\}$, (depending on s) we must have $m = [q/r]$ or $m = [q/r] + 1$. The second register collapses to the classical state $|f(s)\rangle$. We can now ignore the second register, and have in the first register:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + s\rangle .$$

§6.3 Shor's Period-finding Algorithm

Because $f(a) = f(s)$ if and only if $a \equiv s \pmod{r}$, the a of the form $a = jr + s$ $(0 \leqslant j < m)$ are exactly the a for which $f(a) = f(s)$. Thus the first register collapses to a superposition of $|s\rangle$, $|r+s\rangle$, $|2r+s\rangle$, $|3r+s\rangle$ $\cdots$; this superposition runs until the last number of the form $jr + s$ that is less than q. Since m is the number of elements in this superposition; that is, the number of integers j such that $jr + s \in \{0, 1, \cdots, q-1\}$, (depending on s) we must have $m = [q/r]$ or $m = [q/r] + 1$. The second register collapses to the classical state $|f(s)\rangle$. We can now ignore the second register, and have in the first register:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + s\rangle .$$

§6.3 Shor's Period-finding Algorithm

Applying the QFT again gives

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}\frac{1}{\sqrt{q}}\sum_{b=0}^{q-1}e^{2\pi i\frac{(jr+s)b}{q}}|b\rangle = \frac{1}{\sqrt{mq}}\sum_{b=0}^{q-1}e^{2\pi i\frac{sb}{q}}\left(\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}}\right)|b\rangle\,.$$

We want to see which $|b\rangle$ have amplitudes with large squared absolute value - those are the b we are likely to see if we now measure. Using that

$$\sum_{j=0}^{m-1}z^j = \frac{1-z^m}{1-z} \qquad \forall\, z\neq 1\,,$$

we have

$$\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}} = \sum_{j=0}^{m-1}\left(e^{2\pi i\frac{rb}{q}}\right)^j = \begin{cases} m & \text{if } e^{2\pi i\frac{rb}{q}}=1\,,\\ \dfrac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}} & \text{if } e^{2\pi i\frac{rb}{q}}\neq 1\,. \end{cases} \tag{1}$$

§6.3 Shor's Period-finding Algorithm

Applying the QFT again gives

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}\frac{1}{\sqrt{q}}\sum_{b=0}^{q-1}e^{2\pi i\frac{(jr+s)b}{q}}|b\rangle=\frac{1}{\sqrt{mq}}\sum_{b=0}^{q-1}e^{2\pi i\frac{sb}{q}}\left(\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}}\right)|b\rangle\,.$$

We want to see which $|b\rangle$ have amplitudes with large squared absolute value - those are the b we are likely to see if we now measure.

Using that

$$\sum_{j=0}^{m-1}z^j=\frac{1-z^m}{1-z}\qquad\forall\, z\neq 1\,,$$

we have

$$\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}}=\sum_{j=0}^{m-1}\left(e^{2\pi i\frac{rb}{q}}\right)^j=\begin{cases} m & \text{if } e^{2\pi i\frac{rb}{q}}=1\,,\\ \dfrac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}} & \text{if } e^{2\pi i\frac{rb}{q}}\neq 1\,.\end{cases}\tag{1}$$

§6.3 Shor's Period-finding Algorithm

Applying the QFT again gives

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}\frac{1}{\sqrt{q}}\sum_{b=0}^{q-1}e^{2\pi i\frac{(jr+s)b}{q}}|b\rangle=\frac{1}{\sqrt{mq}}\sum_{b=0}^{q-1}e^{2\pi i\frac{sb}{q}}\left(\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}}\right)|b\rangle\,.$$

We want to see which $|b\rangle$ have amplitudes with large squared absolute value - those are the b we are likely to see if we now measure. Using that

$$\sum_{j=0}^{m-1}z^j=\frac{1-z^m}{1-z}\qquad\forall\, z\neq 1\,,$$

we have

$$\sum_{j=0}^{m-1}e^{2\pi i\frac{jrb}{q}}=\sum_{j=0}^{m-1}\left(e^{2\pi i\frac{rb}{q}}\right)^j=\begin{cases} m & \text{if } e^{2\pi i\frac{rb}{q}}=1\,,\\ \dfrac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}} & \text{if } e^{2\pi i\frac{rb}{q}}\neq 1\,.\end{cases}\qquad(1)$$

§6.3 Shor's Period-finding Algorithm

Applying the QFT again gives

$$\frac{m}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i \frac{rb}{q}}\neq 1} e^{2\pi i \frac{sb}{q}} \frac{1-e^{2\pi i \frac{mrb}{q}}}{1-e^{2\pi i \frac{rb}{q}}} |b\rangle .$$

We want to see which $|b\rangle$ have amplitudes with large squared absolute value - those are the b we are likely to see if we now measure. Using that

$$\sum_{j=0}^{m-1} z^j = \frac{1-z^m}{1-z} \qquad \forall\, z \neq 1 ,$$

we have

$$\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} = \sum_{j=0}^{m-1} \left(e^{2\pi i \frac{rb}{q}}\right)^j = \begin{cases} m & \text{if } e^{2\pi i \frac{rb}{q}} = 1 , \\ \dfrac{1-e^{2\pi i \frac{mrb}{q}}}{1-e^{2\pi i \frac{rb}{q}}} & \text{if } e^{2\pi i \frac{rb}{q}} \neq 1 . \end{cases} \qquad (1)$$

§6.3 Shor's Period-finding Algorithm

- **The case r divides q:**

Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0, 1, \cdots, q-1\}$ of f, and $m = q/r$. Looking at the quantum state before applying the final measurement:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}} \neq 1} e^{2\pi i \frac{sb}{q}} \frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} |b\rangle .$$

For the first sum, note that $e^{2\pi i \frac{rb}{q}} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such basis states b, together they have all the amplitude: the sum of squares of those amplitudes is 1, so the amplitudes of b that are not integer multiples of q/r must all be 0.

§6.3 Shor's Period-finding Algorithm

- **The case r divides q:**

Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0, 1, \cdots, q-1\}$ of f, and $m = q/r$. Looking at the quantum state before applying the final measurement:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1,\, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1,\, e^{2\pi i \frac{rb}{q}} \neq 1} e^{2\pi i \frac{sb}{q}} \frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} |b\rangle .$$

For the first sum, note that $e^{2\pi i \frac{rb}{q}} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such basis states b, together they have all the amplitude: the sum of squares of those amplitudes is 1, so the amplitudes of b that are not integer multiples of q/r must all be 0.

§6.3 Shor's Period-finding Algorithm

- **The case r divides q:**

Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0, 1, \cdots, q-1\}$ of f, and $m = q/r$. Looking at the quantum state before applying the final measurement:

$$\frac{m}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}=1} e^{2\pi i\frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}\neq 1} e^{2\pi i\frac{sb}{q}} \frac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}} |b\rangle .$$

For the first sum, note that $e^{2\pi i\frac{rb}{q}} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such basis states b, together they have all the amplitude: the sum of squares of those amplitudes is 1, so the amplitudes of b that are not integer multiples of q/r must all be 0.

§6.3 Shor's Period-finding Algorithm

- **The case r divides q:**

Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0, 1, \cdots, q-1\}$ of f, and $m = q/r$. Looking at the quantum state before applying the final measurement:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}} \neq 1} e^{2\pi i \frac{sb}{q}} \frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} |b\rangle .$$

For the first sum, note that $e^{2\pi i \frac{rb}{q}} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such basis states b, together they have all the amplitude: the sum of squares of those amplitudes is 1, so the amplitudes of b that are not integer multiples of q/r must all be 0.

§6.3 Shor's Period-finding Algorithm

- **The case r divides q:**

Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0, 1, \cdots, q-1\}$ of f, and $m = q/r$. Looking at the quantum state before applying the final measurement:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}} \neq 1} e^{2\pi i \frac{sb}{q}} \frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} |b\rangle .$$

For the first sum, note that $e^{2\pi i \frac{rb}{q}} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such basis states b, together they have all the amplitude: the sum of squares of those amplitudes is 1, so the amplitudes of b that are not integer multiples of q/r must all be 0.

§6.3 Shor's Period-finding Algorithm

Thus we are left with a superposition:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle = \frac{1}{\sqrt{r}} \sum_{c=0}^{r-1} e^{2\pi i \frac{sc}{r}} \left| c\frac{q}{r} \right\rangle .$$

Measuring this final superposition gives some random multiple $b = cq/r$, with c a uniformly random number $0 \leqslant c < r$; thus

$$\frac{b}{q} = \frac{c}{r},$$

where b and q are known to the algorithm, and c and r are not. There are $\varphi(r) \in \Omega(r/\log\log r)$ numbers smaller than r that are coprime to r, where φ is the Euler (phi) function, so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $\mathcal{O}(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r. Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

§6.3 Shor's Period-finding Algorithm

Thus we are left with a superposition:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle = \frac{1}{\sqrt{r}} \sum_{c=0}^{r-1} e^{2\pi i \frac{sc}{r}} \left| c\frac{q}{r} \right\rangle .$$

Measuring this final superposition gives some random multiple $b = cq/r$, with c a uniformly random number $0 \leqslant c < r$; thus

$$\frac{b}{q} = \frac{c}{r} ,$$

where b and q are known to the algorithm, and c and r are not. There are $\varphi(r) \in \Omega(r/\log\log r)$ numbers smaller than r that are coprime to r, where φ is the Euler (phi) function, so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $\mathcal{O}(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r. Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

§6.3 Shor's Period-finding Algorithm

Thus we are left with a superposition:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle = \frac{1}{\sqrt{r}} \sum_{c=0}^{r-1} e^{2\pi i \frac{sc}{r}} \left| c\frac{q}{r} \right\rangle .$$

Measuring this final superposition gives some random multiple $b = cq/r$, with c a uniformly random number $0 \leqslant c < r$; thus

$$\frac{b}{q} = \frac{c}{r} ,$$

where b and q are known to the algorithm, and c and r are not. There are $\varphi(r) \in \Omega(r/\log\log r)$ numbers smaller than r that are coprime to r, where φ is the Euler (phi) function, so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $\mathcal{O}(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r. Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

§6.3 Shor's Period-finding Algorithm

Thus we are left with a superposition:

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle = \frac{1}{\sqrt{r}} \sum_{c=0}^{r-1} e^{2\pi i \frac{sc}{r}} \left| c\frac{q}{r} \right\rangle.$$

Measuring this final superposition gives some random multiple $b = cq/r$, with c a uniformly random number $0 \leqslant c < r$; thus

$$\frac{b}{q} = \frac{c}{r},$$

where b and q are known to the algorithm, and c and r are not. There are $\varphi(r) \in \Omega(r/\log\log r)$ numbers smaller than r that are coprime to r, where φ is the Euler (phi) function, so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $\mathcal{O}(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r. Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

§6.3 Shor's Period-finding Algorithm

- **The case r does not divide q:**

Because our q is a power of 2, it is actually quite likely that r does not divide q. However, the same algorithm will still yield with high probability a b which is close to a multiple of q/r. Note that q/r is no longer an integer, so $m = \lfloor q/r \rfloor$ or $m = \lfloor q/r \rfloor + 1$. Using $|1 - e^{i\theta}| = 2\left|\sin\frac{\theta}{2}\right|$, we can rewrite the absolute value of the second case of equation (1) to

$$\left|\frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}}\right| = \left|\frac{\sin(\pi m r \frac{b}{q})}{\sin(\pi r \frac{b}{q})}\right|.$$

The right-hand side is the ratio of two sine-functions of b, where the numerator oscillates much faster than the denominator because of the additional factor of m.

§6.3 Shor's Period-finding Algorithm

- **The case r does not divide q:**

Because our q is a power of 2, it is actually quite likely that r does not divide q. However, the same algorithm will still yield with high probability a b which is close to a multiple of q/r. Note that q/r is no longer an integer, so $m = [q/r]$ or $m = [q/r] + 1$. Using $|1 - e^{i\theta}| = 2\left|\sin\frac{\theta}{2}\right|$, we can rewrite the absolute value of the second case of equation (1) to

$$\left|\frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}}\right| = \left|\frac{\sin(\pi m r \frac{b}{q})}{\sin(\pi r \frac{b}{q})}\right|.$$

The right-hand side is the ratio of two sine-functions of b, where the numerator oscillates much faster than the denominator because of the additional factor of m.

§6.3 Shor's Period-finding Algorithm

- **The case r does not divide q:**

Because our q is a power of 2, it is actually quite likely that r does not divide q. However, the same algorithm will still yield with high probability a b which is close to a multiple of q/r. Note that q/r is no longer an integer, so $m = [q/r]$ or $m = [q/r] + 1$. Using $|1 - e^{i\theta}| = 2\left|\sin\frac{\theta}{2}\right|$, we can rewrite the absolute value of the second case of equation (1) to

$$\left|\frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}}\right| = \left|\frac{\sin(\pi m r \frac{b}{q})}{\sin(\pi r \frac{b}{q})}\right| .$$

The right-hand side is the ratio of two sine-functions of b, where the numerator oscillates much faster than the denominator because of the additional factor of m.

§6.3 Shor's Period-finding Algorithm

Now we apply the final measurement to the quantum state

$$\frac{m}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}=1} e^{2\pi i\frac{sb}{q}}|b\rangle + \frac{1}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}\neq 1} e^{2\pi i\frac{sb}{q}} \frac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}}|b\rangle$$

and obtain $|b\rangle$. Treating $\left|\frac{\sin(m\pi x)}{\sin(\pi x)}\right| = m$ if $x \in \mathbb{N}$,

the probability of obtaining $|b\rangle$ is $\frac{1}{mq}\left|\frac{\sin(\pi mr\frac{b}{q})}{\sin(\pi r\frac{b}{q})}\right|^2$.

The $|b\rangle$ that we will obtain is **most likely** one of those b's satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q} \quad \text{for some } c \in \mathbb{N}.$$

It can be shown that with high probability the final measurement yields a b satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q}.$$

§6.3 Shor's Period-finding Algorithm

Now we apply the final measurement to the quantum state

$$\frac{m}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0\leqslant b\leqslant q-1, e^{2\pi i \frac{rb}{q}}\neq 1} e^{2\pi i \frac{sb}{q}} \frac{1-e^{2\pi i \frac{mrb}{q}}}{1-e^{2\pi i \frac{rb}{q}}} |b\rangle$$

and obtain $|b\rangle$. Treating $\left|\frac{\sin(m\pi x)}{\sin(\pi x)}\right| = m$ if $x \in \mathbb{N}$,

the probability of obtaining $|b\rangle$ is $\frac{1}{mq}\left|\frac{\sin(\pi m r \frac{b}{q})}{\sin(\pi r \frac{b}{q})}\right|^2$.

The $|b\rangle$ that we will obtain is **most likely** one of those b's satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q} \quad \text{for some } c \in \mathbb{N}.$$

It can be shown that with high probability the final measurement yields a b satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q}.$$

§6.3 Shor's Period-finding Algorithm

Now we apply the final measurement to the quantum state

$$\frac{m}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}}=1} e^{2\pi i \frac{sb}{q}} |b\rangle + \frac{1}{\sqrt{mq}} \sum_{0 \leqslant b \leqslant q-1, e^{2\pi i \frac{rb}{q}} \neq 1} e^{2\pi i \frac{sb}{q}} \frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} |b\rangle$$

and obtain $|b\rangle$. Treating $\left|\frac{\sin(m\pi x)}{\sin(\pi x)}\right| = m$ if $x \in \mathbb{N}$,

the probability of obtaining $|b\rangle$ is $\frac{1}{mq}\left|\frac{\sin(\pi mr\frac{b}{q})}{\sin(\pi r\frac{b}{q})}\right|^2$.

The $|b\rangle$ that we will obtain is **most likely** one of those b's satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q} \quad \text{for some } c \in \mathbb{N}.$$

It can be shown that with high probability the final measurement yields a b satisfying

$$\left|\frac{rb}{q} - c\right| \leqslant \frac{r}{2q}.$$

§6.3 Shor's Period-finding Algorithm

Now we apply the final measurement to the quantum state

$$\frac{m}{\sqrt{mq}}\sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}=1} e^{2\pi i\frac{sb}{q}}|b\rangle + \frac{1}{\sqrt{mq}}\sum_{0\leqslant b\leqslant q-1, e^{2\pi i\frac{rb}{q}}\neq 1} e^{2\pi i\frac{sb}{q}}\frac{1-e^{2\pi i\frac{mrb}{q}}}{1-e^{2\pi i\frac{rb}{q}}}|b\rangle$$

and obtain $|b\rangle$. Treating $\left|\frac{\sin(m\pi x)}{\sin(\pi x)}\right| = m$ if $x\in\mathbb{N}$,

the probability of obtaining $|b\rangle$ is $\frac{1}{mq}\left|\frac{\sin(\pi mr\frac{b}{q})}{\sin(\pi r\frac{b}{q})}\right|^2$.

The $|b\rangle$ that we will obtain is **most likely** one of those b's satisfying

$$\left|\frac{rb}{q}-c\right| \leqslant \frac{r}{2q} \quad \text{for some } c\in\mathbb{N}.$$

It can be shown that with high probability the final measurement yields a b satisfying

$$\left|\frac{rb}{q}-c\right| \leqslant \frac{r}{2q}.$$

§6.3 Shor's Period-finding Algorithm

Therefore, with high probability the final measurement yields a b satisfying

$$\left|\frac{b}{q}-\frac{c}{r}\right| \leqslant \frac{1}{2q}\,.$$

As in the previous case, b and q are known to us while c and r are unknown. Two distinct fractions, each with denominator not greater than N, must be at least $1/N^2$ apart. Since $1/N^2 > 1/q$, c/r is the only fraction with denominator not greater than N at distance not greater than $1/(2q)$ from b/q. Applying a classical method called "continued-fraction expansion" to b/q efficiently gives us the fraction with denominator not greater than N that is closest to b/q (see the next section). This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives us r.

§6.3 Shor's Period-finding Algorithm

Therefore, with high probability the final measurement yields a b satisfying

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leqslant \frac{1}{2q}.$$

As in the previous case, b and q are known to us while c and r are unknown. Two distinct fractions, each with denominator not greater than N, must be at least $1/N^2$ apart. Since $1/N^2 > 1/q$, c/r is the only fraction with denominator not greater than N at distance not greater than $1/(2q)$ from b/q. Applying a classical method called "continued-fraction expansion" to b/q efficiently gives us the fraction with denominator not greater than N that is closest to b/q (see the next section). This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives us r.

§6.3 Shor's Period-finding Algorithm

Therefore, with high probability the final measurement yields a b satisfying

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leqslant \frac{1}{2q}\,.$$

As in the previous case, b and q are known to us while c and r are unknown. Two distinct fractions, each with denominator not greater than N, must be at least $1/N^2$ apart. Since $1/N^2 > 1/q$, c/r is the only fraction with denominator not greater than N at distance not greater than $1/(2q)$ from b/q. Applying a classical method called "continued-fraction expansion" to b/q efficiently gives us the fraction with denominator not greater than N that is closest to b/q (see the next section). This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives us r.

§6.3 Shor's Period-finding Algorithm

Therefore, with high probability the final measurement yields a b satisfying

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leqslant \frac{1}{2q}\,.$$

As in the previous case, b and q are known to us while c and r are unknown. Two distinct fractions, each with denominator not greater than N, must be at least $1/N^2$ apart. Since $1/N^2 > 1/q$, c/r is the only fraction with denominator not greater than N at distance not greater than $1/(2q)$ from b/q. Applying a classical method called "continued-fraction expansion" to b/q efficiently gives us the fraction with denominator not greater than N that is closest to b/q (see the next section). This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives us r.

§6.4 Continued fractions

A continued fraction (連分數), or simply CF, is a real number of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}\,.$$

The continued fraction above is denote by $[a_0, a_1, a_2, \cdots]$ (here the number of a_i's can be finite or infinite), and the a_i's are called the partial quotients. We assume these to be positive natural numbers. $[a_0, \cdots, a_n]$ is called the n-th convergent of the continued fraction $[a_0, a_1, a_2, \cdots]$, and can be simply computed by the following iterative scheme: $[a_0, \cdots, a_n]$, in its lowest terms, is p_n/q_n, where

$$p_0 = a_0\,, \quad p_1 = a_1 a_0 + 1\,, \quad p_n = a_n p_{n-1} + p_{n-2}\,,$$
$$q_0 = 1\,, \quad q_1 = a_1\,, \quad q_n = a_n q_{n-1} + q_{n-2}\,.$$

§6.4 Continued fractions

A continued fraction (連分數), or simply CF, is a real number of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}.$$

The continued fraction above is denote by $[a_0, a_1, a_2, \cdots]$ (here the number of a_i's can be finite or infinite), and the a_i's are called the partial quotients. We assume these to be positive natural numbers. $[a_0, \cdots, a_n]$ is called the n-th convergent of the continued fraction $[a_0, a_1, a_2, \cdots]$, and can be simply computed by the following iterative scheme: $[a_0, \cdots, a_n]$, in its lowest terms, is p_n/q_n, where

$$p_0 = a_0\,, \quad p_1 = a_1 a_0 + 1\,, \quad p_n = a_n p_{n-1} + p_{n-2}\,,$$
$$q_0 = 1\,, \quad q_1 = a_1\,, \quad q_n = a_n q_{n-1} + q_{n-2}\,.$$

§6.4 Continued fractions

Note that q_n increases at least exponentially with n since $q_n \geqslant 2q_{n-2}$. Given a real number x, the following "algorithm" gives a continued fraction expansion of x:

$$\begin{aligned} a_0 &\equiv [x]\,, & x_1 &\equiv 1/(x - a_0)\,, \\ a_1 &\equiv [x_1]\,, & x_2 &\equiv 1/(x_1 - a_1)\,, \\ a_2 &\equiv [x_2]\,, & x_3 &\equiv 1/(x_2 - a_2)\,, \\ & & &\vdots \end{aligned}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number.

§6.4 Continued fractions

Note that q_n increases at least exponentially with n since $q_n \geqslant 2q_{n-2}$. Given a real number x, the following "algorithm" gives a continued fraction expansion of x:

$$\begin{aligned} a_0 &\equiv [x]\,, & x_1 &\equiv 1/(x - a_0)\,, \\ a_1 &\equiv [x_1]\,, & x_2 &\equiv 1/(x_1 - a_1)\,, \\ a_2 &\equiv [x_2]\,, & x_3 &\equiv 1/(x_2 - a_2)\,, \\ &\vdots \end{aligned}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number.

§6.4 Continued fractions

The convergents of the CF approximate x follows from the fact that

$$\text{if } x = [a_0, a_1, \cdots], \text{ then } \left| x - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n^2} .$$

Recall that q_n increases exponentially with n, so this convergence is quite fast. Moreover, p_n/q_n provides the best approximation of x among all fractions with denominator not greater than q_n:

$$\text{if } n \geqslant 1,\ q \leqslant q_n,\ \frac{p}{q} \neq \frac{p_n}{q_n}, \text{ then } \left| x - \frac{p_n}{q_n} \right| < \left| x - \frac{p}{q} \right| .$$

§6.4 Continued fractions

The convergents of the CF approximate x follows from the fact that

$$\text{if } x = [a_0, a_1, \cdots], \text{ then } \left| x - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n^2}.$$

Recall that q_n increases exponentially with n, so this convergence is quite fast. Moreover, p_n/q_n provides the best approximation of x among all fractions with denominator not greater than q_n:

$$\text{if } n \geqslant 1,\ q \leqslant q_n,\ \frac{p}{q} \neq \frac{p_n}{q_n}, \text{ then } \left| x - \frac{p_n}{q_n} \right| < \left| x - \frac{p}{q} \right|.$$