Quantum algorithms for factorization and other problems in cryptanalysis

Pierre-Alain Fouque

Centre Inria de l'Université de Rennes

Contents

Cryptanalysis and Security Levels

Cryptography

- Science of "secret": Confidentiality, Integrity, and Authentication
- Cryptosystem: encryption and signature schemes
- Public-Key vs. Secret-Key Cryptography

Cryptanalysis and Security Levels

Cryptography

- Science of "secret": Confidentiality, Integrity, and Authentication
- Cryptosystem: encryption and signature schemes
- Public-Key vs. Secret-Key Cryptography

Cryptanalysis

- Adversaries $\approx$ (classical or quantum) algorithms
- Complexity of the algorithms to evaluate the security parameters
- For Public-Key Cryptography: security is not perfect and use computational assumption: not possible to break the cryptosystem except if you break a mathematical hard problem

Cryptanalysis and Security Levels

Cryptography

- Science of "secret": Confidentiality, Integrity, and Authentication
- Cryptosystem: encryption and signature schemes
- Public-Key vs. Secret-Key Cryptography

Cryptanalysis

- Adversaries $\approx$ (classical or quantum) algorithms
- Complexity of the algorithms to evaluate the security parameters
- For Public-Key Cryptography: security is not perfect and use computational assumption: not possible to break the cryptosystem except if you break a mathematical hard problem

Security Levels

- If the number of steps is 2^{128}, the adversary requires too much time
- The logarithm is the security level and 128 is good, while 64 is low

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

- 8051 ?

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

- 8051 ?
- 91 ?

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

- 8051 ?
- 91 ?
- $91 = 100 - 9 = 10^2 - 3^2 = (10-3)(10+3) = 7 \times 13$

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

- 8051 ?
- 91 ?
- $91 = 100 - 9 = 10^2 - 3^2 = (10 - 3)(10 + 3) = 7 \times 13$
- $8051 = 8100 - 49 = 90^2 - 7^2 = (90 - 7)(90 + 7) = 83 \times 97$

Cryptography: Hard Computational problems

RSA (Rivest-Shamir-Adleman) describe a cryptosystem whose security is based on the untractability to solve the factorisation problem

Factorisation
Given an integer $N = pq$, where p and q are two primes. Recover p ?

- 8051 ?
- 91 ?
- $91 = 100 - 9 = 10^2 - 3^2 = (10-3)(10+3) = 7 \times 13$
- $8051 = 8100 - 49 = 90^2 - 7^2 = (90-7)(90+7) = 83 \times 97$

Classical algorithm:

- Number Field Sieve (NFS). Complexity: $2^{\tilde{O}(n^{1/3})}$ (constants matter...) where n is the size of N: $n = \log_2(N)$
- Record: 250-digits (830 bits): 2700 computer years
- $\approx 2^{128}$ for a 2048-bit modulus

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

- $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 5 \bmod 11, 2^5 = 10 \bmod 11, 2^6 = 9 \bmod 11, 2^7 = 7 \bmod 11, 2^8 = 3 \bmod 11, 2^9 = 6 \bmod 11, 2^{10} = 1 \bmod 11...$

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

- $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 5 \bmod 11, 2^5 = 10 \bmod 11, 2^6 = 9 \bmod 11, 2^7 = 7 \bmod 11, 2^8 = 3 \bmod 11, 2^9 = 6 \bmod 11, 2^{10} = 1 \bmod 11$...
- What is the subgroup generated by 4 ? generated by 10 ?

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

- $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 5 \bmod 11, 2^5 = 10 \bmod 11, 2^6 = 9 \bmod 11, 2^7 = 7 \bmod 11, 2^8 = 3 \bmod 11, 2^9 = 6 \bmod 11, 2^{10} = 1 \bmod 11$...
- What is the subgroup generated by 4 ? generated by 10 ?
- As $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic, for all $d | p-1$, there is a subgroup of order d

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

- $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 5 \bmod 11, 2^5 = 10 \bmod 11, 2^6 = 9 \bmod 11, 2^7 = 7 \bmod 11, 2^8 = 3 \bmod 11, 2^9 = 6 \bmod 11, 2^{10} = 1 \bmod 11$...
- What is the subgroup generated by 4 ? generated by 10 ?
- As $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic, for all $d|p-1$, there is a subgroup of order d

Complexity and Security level

- Classical algorithms: Pollard $\sqrt{q}$ and NFS: $2^{\tilde{O}((\log_2 p)^{1/3})}$

Cryptography: Hard Computational problems (II)

Discrete Logarithm
Let p a prime and q a prime divisor of $p-1$, and g a generator of the q-order subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$. Given $y = g^x \bmod p$, recover x ?

Example: $g = 2$ **in** $(\mathbb{Z}/11\mathbb{Z})^*$

- $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 5 \bmod 11, 2^5 = 10 \bmod 11, 2^6 = 9 \bmod 11, 2^7 = 7 \bmod 11, 2^8 = 3 \bmod 11, 2^9 = 6 \bmod 11, 2^{10} = 1 \bmod 11$...
- What is the subgroup generated by 4 ? generated by 10 ?
- As $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic, for all $d | p-1$, there is a subgroup of order d

Complexity and Security level

- Classical algorithms: Pollard $\sqrt{q}$ and NFS: $2^{\tilde{O}((\log_2 p)^{1/3})}$
- p a 2048-bit prime and q a 256-bit prime

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Breakthrough

- Polynomial-time algorithm $O(n^2)$ and $O(n)$ qubits

Shor's quantum factorisation algorithm

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Breakthrough

- Polynomial-time algorithm $O(n^2)$ and $O(n)$ qubits
- If we were able to built a noise-free quantum algorithm, we will be able to break all communications...

Shor's quantum factorisation algorithm

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Breakthrough

- Polynomial-time algorithm $O(n^2)$ and $O(n)$ qubits
- If we were able to built a noise-free quantum algorithm, we will be able to break all communications...
- Post-Quantum Cryptography: classical algorithms where hard problems are conjectured to resist quantum computers ...

Shor's quantum factorisation algorithm

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Breakthrough

- Polynomial-time algorithm $O(n^2)$ and $O(n)$ qubits
- If we were able to built a noise-free quantum algorithm, we will be able to break all communications...
- Post-Quantum Cryptography: classical algorithms where hard problems are conjectured to resist quantum computers ...
- E.g.: hard lattice problems, coding problems, ...

Shor's quantum factorisation algorithm

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

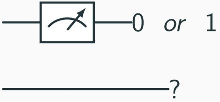
Peter W. Shor†

Breakthrough

- Polynomial-time algorithm $O(n^2)$ and $O(n)$ qubits
- If we were able to built a noise-free quantum algorithm, we will be able to break all communications...
- Post-Quantum Cryptography: classical algorithms where hard problems are conjectured to resist quantum computers ...
- E.g.: hard lattice problems, coding problems, ...
- Standards are available since 2024 and the transition to PQC begins

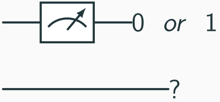
Basic Circuits: Deutsch-Jozsa and Simon algorithms

Partial Measurement of a 2-qubit

- $|\psi\rangle = \alpha\,|0.0\rangle + \beta\,|0.1\rangle + \gamma\,|1.0\rangle + \delta\,|1.1\rangle$, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- $|\psi\rangle$ 

- Let $|\psi\rangle = \frac{\sqrt{2}}{2}\,|0.0\rangle + \frac{1}{2}\,|0.1\rangle + \frac{1}{2}\,|1.1\rangle$. If one measures the first qubit as 1, what is the second qubit ?

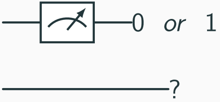
Partial Measurement of a 2-qubit

- $|\psi\rangle = \alpha\,|0.0\rangle + \beta\,|0.1\rangle + \gamma\,|1.0\rangle + \delta\,|1.1\rangle$, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- $|\psi\rangle$ 

- Let $|\psi\rangle = \frac{\sqrt{2}}{2}\,|0.0\rangle + \frac{1}{2}\,|0.1\rangle + \frac{1}{2}\,|1.1\rangle$. If one measures the first qubit as 1, what is the second qubit ?
- the second is $|1\rangle$, but what if we observe $|0\rangle$?

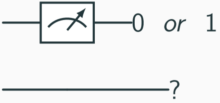
Partial Measurement of a 2-qubit

- $|\psi\rangle = \alpha\,|0.0\rangle + \beta\,|0.1\rangle + \gamma\,|1.0\rangle + \delta\,|1.1\rangle$, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- $|\psi\rangle$



- Let $|\psi\rangle = \frac{\sqrt{2}}{2}\,|0.0\rangle + \frac{1}{2}\,|0.1\rangle + \frac{1}{2}\,|1.1\rangle$. If one measures the first qubit as 1, what is the second qubit ?
- the second is $|1\rangle$, but what if we observe $|0\rangle$?
- $|\psi\rangle = \frac{|0\rangle}{2} \cdot (\sqrt{2}\,|0\rangle + |1\rangle) + \frac{1}{2}\,|1\rangle\,|1\rangle$, the 2^{nd} is $\sqrt{\frac{2}{3}}\,|0\rangle + \frac{1}{\sqrt{3}}\,|1\rangle$

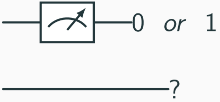
Partial Measurement of a 2-qubit

- $|\psi\rangle = \alpha\,|0.0\rangle + \beta\,|0.1\rangle + \gamma\,|1.0\rangle + \delta\,|1.1\rangle$, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- $|\psi\rangle$ 

- Let $|\psi\rangle = \frac{\sqrt{2}}{2}\,|0.0\rangle + \frac{1}{2}\,|0.1\rangle + \frac{1}{2}\,|1.1\rangle$. If one measures the first qubit as 1, what is the second qubit ?
- the second is $|1\rangle$, but what if we observe $|0\rangle$?
- $|\psi\rangle = \frac{|0\rangle}{2} \cdot (\sqrt{2}\,|0\rangle + |1\rangle) + \frac{1}{2}\,|1\rangle\,|1\rangle$, the 2^{nd} is $\sqrt{\frac{2}{3}}\,|0\rangle + \frac{1}{\sqrt{3}}\,|1\rangle$
- More generally, $|\psi\rangle = |0\rangle \cdot (\alpha\,|0\rangle + \beta\,|1\rangle) + |1\rangle \cdot (\gamma\,|0\rangle + \delta\,|1\rangle)$, and if one measures $|0\rangle$ for the first qubit, the second $\frac{\alpha}{\sqrt{|\alpha|^2+|\beta|^2}}\,|0\rangle + \frac{\beta}{\sqrt{|\alpha|^2+|\beta|^2}}\,|1\rangle$

Partial Measurement of a 2-qubit

- $|\psi\rangle = \alpha\,|0.0\rangle + \beta\,|0.1\rangle + \gamma\,|1.0\rangle + \delta\,|1.1\rangle$, $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$
- $|\psi\rangle$



- Let $|\psi\rangle = \frac{\sqrt{2}}{2}\,|0.0\rangle + \frac{1}{2}\,|0.1\rangle + \frac{1}{2}\,|1.1\rangle$. If one measures the first qubit as 1, what is the second qubit ?
- the second is $|1\rangle$, but what if we observe $|0\rangle$?
- $|\psi\rangle = \frac{|0\rangle}{2} \cdot (\sqrt{2}\,|0\rangle + |1\rangle) + \frac{1}{2}\,|1\rangle\,|1\rangle$, the 2^{nd} is $\sqrt{\frac{2}{3}}\,|0\rangle + \frac{1}{\sqrt{3}}\,|1\rangle$
- Exo: If $|\psi\rangle = \frac{1}{5}(2\,|0.0.0\rangle - |0.0.1\rangle + 3\,|0.1.0\rangle + |0.1.1\rangle - 2\,|1.0.0\rangle + 2\,|1.0.1\rangle + \sqrt{2}\,|1.1.1\rangle)$, and we measure 0.0, what is the last qubit ?

Quantum oracle gate

Oracle

- Let $f : E \longrightarrow \mathbb{Z}/2\mathbb{Z}$ be a function
- $(\mathbb{Z}/2\mathbb{Z}, +) = (\{0, 1\}, \oplus)$
- $F : E \times \mathbb{Z}/2\mathbb{Z} \longrightarrow E \times \mathbb{Z}/2\mathbb{Z}, \quad (x, y) \longmapsto (x, y \oplus f(x))$, is a bijection

Quantum oracle gate

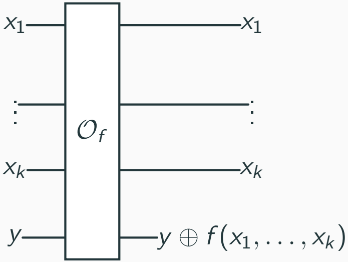
Oracle

- Let $f : E \longrightarrow \mathbb{Z}/2\mathbb{Z}$ be a function
- $(\mathbb{Z}/2\mathbb{Z}, +) = (\{0,1\}, \oplus)$
- $F : E \times \mathbb{Z}/2\mathbb{Z} \longrightarrow E \times \mathbb{Z}/2\mathbb{Z}, \ \ (x,y) \longmapsto (x, y \oplus f(x))$, is a bijection
- Proof: $F^{-1} = F$, $F(F(x,y)) = F(x, y \oplus f(x)) = (x,y)$

Quantum oracle gate

Oracle

- Let $f : E \longrightarrow \mathbb{Z}/2\mathbb{Z}$ be a function
- $(\mathbb{Z}/2\mathbb{Z}, +) = (\{0, 1\}, \oplus)$
- $F : E \times \mathbb{Z}/2\mathbb{Z} \longrightarrow E \times \mathbb{Z}/2\mathbb{Z}, \quad (x, y) \longmapsto (x, y \oplus f(x))$, is a bijection
- Proof: $F^{-1} = F$, $F(F(x, y)) = F(x, y \oplus f(x)) = (x, y)$
- Deutsch-Jozsa Oracle $f : (\mathbb{Z}/2\mathbb{Z})^k \longrightarrow \mathbb{Z}/2\mathbb{Z}$:



Deutsch-Jozsa problem

Goal

- Let $f : \{0,1\} \longrightarrow \{0,1\}$.
- There are 4 such functions: two are constant and two are balanced (0 and 1 are taken the same number of times)
  $f_0 = \left\{ \begin{array}{l} 0 \mapsto 0 \\ 1 \mapsto 0 \end{array} \right. \quad f_1 = \left\{ \begin{array}{l} 0 \mapsto 1 \\ 1 \mapsto 1 \end{array} \right. \quad f_2 = \left\{ \begin{array}{l} 0 \mapsto 0 \\ 1 \mapsto 1 \end{array} \right. \quad f_3 = \left\{ \begin{array}{l} 0 \mapsto 1 \\ 1 \mapsto 0 \end{array} \right.$
- **Decide** if f is constant or balanced ?

Deutsch-Jozsa problem

Goal

- Let $f : \{0,1\} \longrightarrow \{0,1\}$.
- There are 4 such functions: two are constant and two are balanced (0 and 1 are taken the same number of times)
$$f_0 = \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 0 \end{cases} \quad f_1 = \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 1 \end{cases} \quad f_2 = \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 1 \end{cases} \quad f_3 = \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases}$$
- **Decide** if f is constant or balanced ?
- Classically, ask 2 queries ($f(0)$ and $f(1)$), quantumly 1 query !

Deutsch-Jozsa problem

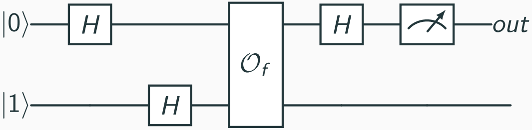
Goal

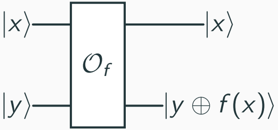
- Let $f : \{0,1\} \longrightarrow \{0,1\}$.
- There are 4 such functions: two are constant and two are balanced (0 and 1 are taken the same number of times)
$$f_0 = \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 0 \end{cases} \quad f_1 = \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 1 \end{cases} \quad f_2 = \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 1 \end{cases} \quad f_3 = \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases}$$
- **Decide** if f is constant or balanced ?
- Classically, ask 2 queries ($f(0)$ and $f(1)$), quantumly 1 query !

Exponential gap: Let $f : \{0,1\}^n \longrightarrow \{0,1\}$ and we have the promise f is either balanced or constant.

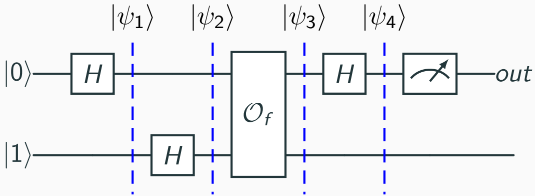
Classically, one need at most $2^{n-1} + 1$ queries, while only 1 quantumly !

Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

- with 

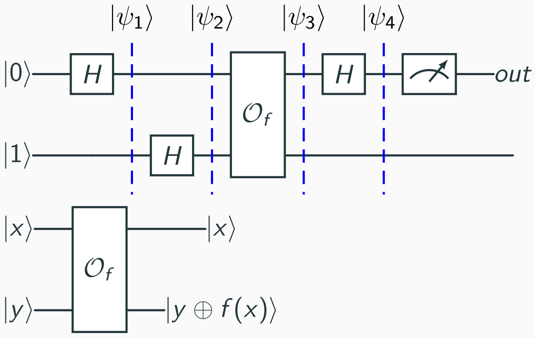
Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

$|\psi_1\rangle = (|0\rangle + |1\rangle).\,|1\rangle,$

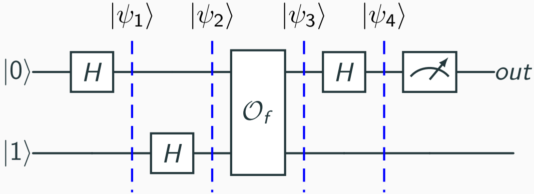
$|\psi_2\rangle = (|0\rangle + |1\rangle).(|0\rangle - |1\rangle) = 0.0 - 0.1 + 1.0 - 1.1$

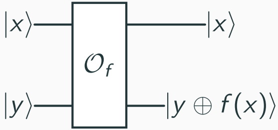
Deutsch-Jozsa Quantum Circuit ($n = 1$)



- $|\psi_2\rangle = 0.0 - 0.1 + 1.0 - 1.1,$

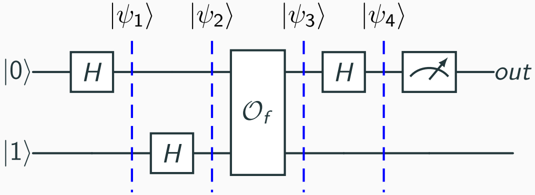
Deutsch-Jozsa Quantum Circuit ($n = 1$)

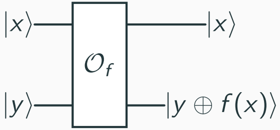




- $|\psi_2\rangle = 0.0 - 0.1 + 1.0 - 1.1,$
- $|\psi_3\rangle = \underbrace{0.(0 \oplus f(0)) - 0.(1 \oplus f(0))}_{A} + \underbrace{1.(0 \oplus f(1)) - 1.(1 \oplus f(1))}_{B}$
- $A = \begin{cases} 0.0 - 0.1 \text{ if } f(0) = 0 \\ -(0.0 - 0.1) \text{ if } f(0) = 1 \end{cases}$ so $A = (-1)^{f(0)}(0.0 - 0.1)$

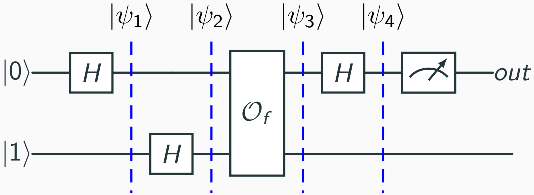
Deutsch-Jozsa Quantum Circuit ($n = 1$)





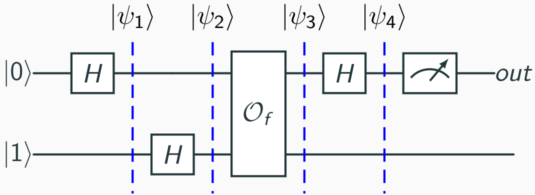
- $|\psi_2\rangle = 0.0 - 0.1 + 1.0 - 1.1,$
- $|\psi_3\rangle = \underbrace{0.(0 \oplus f(0)) - 0.(1 \oplus f(0))}_{A} + \underbrace{1.(0 \oplus f(1)) - 1.(1 \oplus f(1))}_{B}$
- $A = (-1)^{f(0)}(0.0 - 0.1)$ and $B = (-1)^{f(1)}(1.0 - 1.1)$
- $|\psi_3\rangle = (-1)^{f(0)}(0.0 - 0.1) + (-1)^{f(1)}(1.0 - 1.1)$

Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

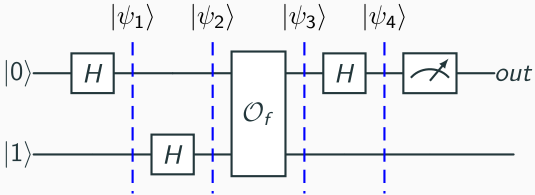
- $|\psi_3\rangle = (-1)^{f(0)}(0.0 - 0.1) + (-1)^{f(1)}(1.0 - 1.1)$
- $|\psi_4\rangle = (-1)^{f(0)}((0+1).0-(0+1).1)+(-1)^{f(1)}((0-1).0-(0-1).1)$
- $|\psi_4\rangle = (-1)^{f(0)}(0.0-0.1+1.0-1.1)+(-1)^{f(1)}(0.0-0.1-1.0+1.1)$

Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

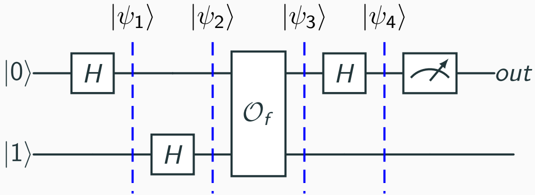
- $|\psi_4\rangle = (-1)^{f(0)}(0.0 - 0.1 + 1.0 - 1.1) + (-1)^{f(1)}(0.0 - 0.1 - 1.0 + 1.1)$

Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

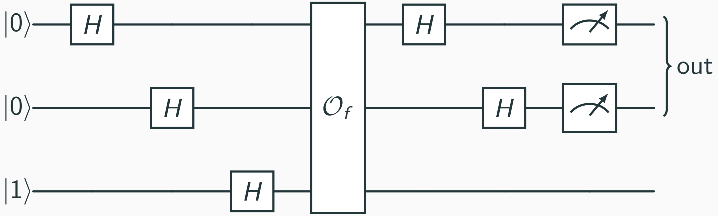
- $|\psi_4\rangle = (-1)^{f(0)}(0.0 - 0.1 + 1.0 - 1.1) + (-1)^{f(1)}(0.0 - 0.1 - 1.0 + 1.1)$
- $|\psi_4\rangle = ((-1)^{f(0)} + (-1)^{f(1)})0.0 + (-(-1)^{f(0)} - (-1)^{f(1)})0.1 + ((-1)^{f(0)} - (-1)^{f(1)})1.0 + (-(-1)^{f(0)} + (-1)^{f(1)})1.1$

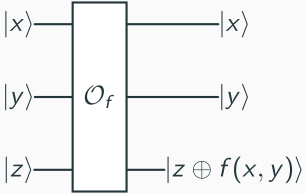
Deutsch-Jozsa Quantum Circuit ($n = 1$)

- 

- $|\psi_4\rangle = (-1)^{f(0)}(0.0 - 0.1 + 1.0 - 1.1) + (-1)^{f(1)}(0.0 - 0.1 - 1.0 + 1.1)$
- $|\psi_4\rangle = ((-1)^{f(0)} + (-1)^{f(1)})0.0 + (-(-1)^{f(0)} - (-1)^{f(1)})0.1 + ((-1)^{f(0)} - (-1)^{f(1)})1.0 + (-(-1)^{f(0)} + (-1)^{f(1)})1.1$
- If f is constant, $(-1)^{f(0)} + (-1)^{f(1)} = \pm 2$ and $(-1)^{f(0)} - (-1)^{f(1)} = 0$ and $(-1)^{f(0)} - (-1)^{f(1)} = 0$, so $|\psi_4\rangle = 0.0 - 0.1$ the measure of the first qubit 0 in both cases
- If f is balanced, check that the first bit is 1

Deutsch-Jozsa Circuit for $n = 2$





- Check that if f is constant, the final state before the measurement is $\pm |0.0\rangle \left| \frac{1}{\sqrt{2}}(0 - 1) \right\rangle$, and the 2 first bits are 0.0
- if f is balanced, the final state does not contain qubits starting with 0.0, so no measurement of these qubits will give 0.0.

Simon algorithm

Problem
Let $f : \{0,1\}^n \to \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Simon algorithm

Problem
Let $f : \{0,1\}^n \to \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Example

$$\begin{array}{ll} \text{f}(000) = 101 & \text{f}(100) = 011 \\ \text{f}(001) = 010 & \text{f}(101) = 100 \\ \text{f}(010) = 011 & \text{f}(110) = 101 \\ \text{f}(011) = 100 & \text{f}(111) = 010 \end{array}$$

What is c ?

Simon algorithm

Problem
Let $f : \{0,1\}^n \to \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Example

$$\begin{array}{ll} \mathrm{f}(000) = 101 & \mathrm{f}(100) = 011 \\ \mathrm{f}(001) = 010 & \mathrm{f}(101) = 100 \\ \mathrm{f}(010) = 011 & \mathrm{f}(110) = 101 \\ \mathrm{f}(011) = 100 & \mathrm{f}(111) = 010 \end{array}$$

What is c ? $c = 110$

Simon algorithm

Problem
Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Example

$$\begin{array}{ll} \text{f}(000) = 101 & \text{f}(100) = 011 \\ \text{f}(001) = 010 & \text{f}(101) = 100 \\ \text{f}(010) = 011 & \text{f}(110) = 101 \\ \text{f}(011) = 100 & \text{f}(111) = 010 \end{array}$$

What is c ? $c = 110$

Classical algorithms

Simon algorithm

Problem
Let $f : \{0,1\}^n \to \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Example

$$\begin{array}{ll} \text{f}(000) = 101 & \text{f}(100) = 011 \\ \text{f}(001) = 010 & \text{f}(101) = 100 \\ \text{f}(010) = 011 & \text{f}(110) = 101 \\ \text{f}(011) = 100 & \text{f}(111) = 010 \end{array}$$

What is c ? $c = 110$

Classical algorithms

- Compute $f(x)$ until a collision $f(x_1) = f(x_2)$ ... and then $c = x_1 \oplus x_2$

Simon algorithm

Problem
Let $f : \{0,1\}^n \to \{0,1\}^n$ a 2-to-1 function so that there exists $c \in \{0,1\}^n$ such that

$$f(x) = f(x \oplus c), \text{ where } \oplus \text{ is bitwise exclusive or}$$

Example

$$\begin{array}{ll} \mathrm{f}(000) = 101 & \mathrm{f}(100) = 011 \\ \mathrm{f}(001) = 010 & \mathrm{f}(101) = 100 \\ \mathrm{f}(010) = 011 & \mathrm{f}(110) = 101 \\ \mathrm{f}(011) = 100 & \mathrm{f}(111) = 010 \end{array}$$

What is c ? $c = 110$

Classical algorithms

- Compute $f(x)$ until a collision $f(x_1) = f(x_2)$ ... and then $c = x_1 \oplus x_2$
- Another solution: since $f(000) \neq f(001)$, $c \neq 001$, ...

Simon Quantum Algorithm

Hadamard Transform

- $H^{\otimes n} \left|\underline{j}\right\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} (-1)^{j \cdot k} \left|\underline{k}\right\rangle$
- $H^{\otimes n} \left|\underline{0}\right\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} \left|\underline{k}\right\rangle$

Simon Quantum Algorithm

Hadamard Transform

- $H^{\otimes n}\left|\underline{j}\right\rangle = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}(-1)^{j\cdot k}\left|\underline{k}\right\rangle$
- $H^{\otimes n}\left|\underline{0}\right\rangle = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}\left|\underline{k}\right\rangle$

Simon's algorithm

Start with $2n$ qubits:	$\left|\underline{0}\right\rangle\left|\underline{0}\right\rangle$
Apply $H^{\otimes n}$	$\sum_x \left|\underline{x}\right\rangle\left|\underline{0}\right\rangle$
Apply O_f	$\sum_x \left|\underline{x}\right\rangle\left|\underline{f(x)}\right\rangle$
Measure the second register	$\left|\underline{x_0}\right\rangle + \left|\underline{x_0 + s}\right\rangle$
Apply $H^{\otimes n}$	$\sum_y((-1)^{x_0\cdot y} + (-1)^{(x_0\oplus s)\cdot y})\left|\underline{y}\right\rangle$
	$= \sum_y(-1)^{x_0\cdot y}\cdot(1+(-1)^{s\cdot y})\left|\underline{y}\right\rangle$

Measure y such that $1+(-1)^{s\cdot y} \neq 0$ iff $s\cdot y = 0$

Simon Quantum Algorithm

Hadamard Transform

- $H^{\otimes n} \left|\underline{j}\right\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} (-1)^{j \cdot k} \left|\underline{k}\right\rangle$
- $H^{\otimes n} \left|\underline{0}\right\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} \left|\underline{k}\right\rangle$

Simon's algorithm

Start with $2n$ qubits:	$\left|\underline{0}\right\rangle \left|\underline{0}\right\rangle$
Apply $H^{\otimes n}$	$\sum_x \left|\underline{x}\right\rangle \left|\underline{0}\right\rangle$
Apply O_f	$\sum_x \left|\underline{x}\right\rangle \left|\underline{f(x)}\right\rangle$
Measure the second register	$\left|\underline{x_0}\right\rangle + \left|\underline{x_0 + s}\right\rangle$
Apply $H^{\otimes n}$	$\sum_y ((-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus s) \cdot y}) \left|\underline{y}\right\rangle$
	$= \sum_y (-1)^{x_0 \cdot y} \cdot (1 + (-1)^{s \cdot y}) \left|\underline{y}\right\rangle$

Measure y such that $1 + (-1)^{s \cdot y} \neq 0$ iff $s \cdot y = 0$

Post-processing

- With $n - 1$ values $y_1, \ldots, y_{n-1}$ independent vectors, we obtain a linear system to recover s

Application of Simon to Symmetric-key cryptanalysis

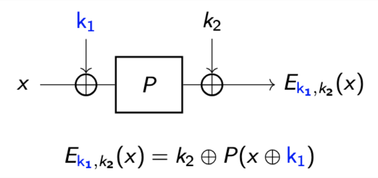
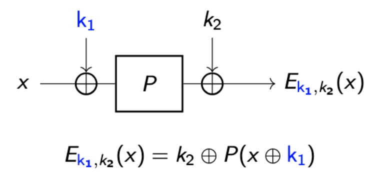


Figure 1: Even-Mansour: P public permutation on $\{0,1\}^n$ with $2n$-bit key

Goal: Recover the secret key (k_1, k_2)

Application of Simon to Symmetric-key cryptanalysis

Figure 1: Even-Mansour: P public permutation on $\{0,1\}^n$ with $2n$-bit key



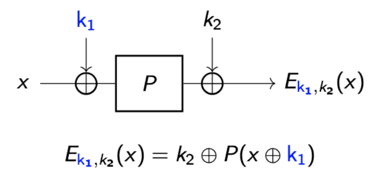
Goal: Recover the secret key (k_1, k_2)

- Classical: If P is random permutation, adversary T queries to P and D to E_{k_1,k_2} needs

$$T \cdot D = 2^n$$

Application of Simon to Symmetric-key cryptanalysis

Figure 1: Even-Mansour: P public permutation on $\{0,1\}^n$ with $2n$-bit key



Goal: Recover the secret key (k_1, k_2)

- Classical: If P is random permutation, adversary T queries to P and D to E_{k_1,k_2} needs

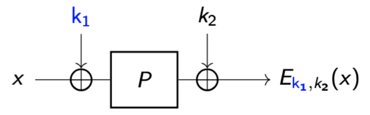
$$T \cdot D = 2^n$$

- Quantum: Define $f(x) = E_{k_1,k_2}(x) \oplus P(x) = P(x \oplus k_1) \oplus P(x) \oplus k_2$:

$$f(x \oplus k_1) = f(x)$$

Application of Simon to Symmetric-key cryptanalysis

Figure 1: Even-Mansour: P public permutation on $\{0,1\}^n$ with $2n$-bit key



$$E_{k_1,k_2}(x) = k_2 \oplus P(x \oplus k_1)$$

Goal: Recover the secret key (k_1, k_2)

- Classical: If P is random permutation, adversary T queries to P and D to E_{k_1,k_2} needs

$$T \cdot D = 2^n$$

- Quantum: Define $f(x) = E_{k_1,k_2}(x) \oplus P(x) = P(x \oplus k_1) \oplus P(x) \oplus k_2$:

$$f(x \oplus k_1) = f(x)$$

- f one query to E_{k_1,k_2} in superposition. Q2 model: Realistic model ?

Shor Algorithm

Arithmetic

- $\mathbb{Z}/N\mathbb{Z}$ is not an integral domain: $N = 15$, $5 \times 3 = 0 \bmod 15$
- $(\mathbb{Z}/N\mathbb{Z})^*$ multiplicative group of invertible elements, not cyclic !
- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$

Arithmetic

- $(\mathbb{Z}/N\mathbb{Z})^*$ multiplicative group of invertible elements, not cyclic !
- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$

Assumptions

1. Assumption 1: $\text{ord}(a) = r$ is even with proba. $1/2$
2. Fact: $(a^{r/2} - 1)(a^{r/2} + 1) = 0 \bmod N$

Arithmetic

- $(\mathbb{Z}/N\mathbb{Z})^*$ multiplicative group of invertible elements, not cyclic !
- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$

Assumptions

1. Assumption 1: $\mathrm{ord}(a) = r$ is even with proba. $1/2$
2. Fact: $(a^{r/2} - 1)(a^{r/2} + 1) = 0 \bmod N$
3. Assumption 2: $a^{r/2} + 1$ is not divisible by N for many a's (CRT)
4. Under Assumption 1 and 2: $d = \gcd(a^{r/2} - 1, N)$ and $d' = \gcd(a^{r/2} + 1, N)$ are non-trivial factors of N

- $(\mathbb{Z}/N\mathbb{Z})^*$ multiplicative group of invertible elements, not cyclic !
- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$

Assumptions

1. Assumption 1: $\mathrm{ord}(a) = r$ is even with proba. 1/2
2. Fact: $(a^{r/2} - 1)(a^{r/2} + 1) = 0 \bmod N$
3. Assumption 2: $a^{r/2} + 1$ is not divisible by N for many a's (CRT)
4. Under Assumption 1 and 2: $d = \gcd(a^{r/2} - 1, N)$ and $d' = \gcd(a^{r/2} + 1, N)$ are non-trivial factors of N

a=2	$(a, N) = 1$	$r = 4, 2^4 = 16 = 1 \bmod 15$	$(2^{4/2} - 1, 15) = 3$
a=3	no		
a=11	$(a, N) = 1$	$r = 2, 11^2 = 121 = 1 \bmod 15$	$(11^{2/2} - 1, 15) = 5$

Order and Oracle

- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$
- Oracle $F : (k, 0) \mapsto (k, a^k \bmod N)$
- E.g. $N = 15$ and $a = 2$, $r = 4$

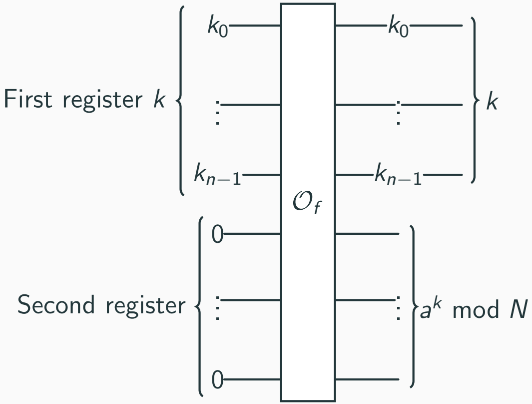
Order and Oracle

- order of a: smallest positive integer r s.t. $a^r = 1 \bmod N$
- $r|\varphi(N)$ Lagrange Theorem in the group $(\mathbb{Z}/N\mathbb{Z})^*$
- r is the smallest period of the function $f : k \mapsto a^k \bmod N$
- Oracle $F : (k, 0) \mapsto (k, a^k \bmod N)$
- E.g. $N = 15$ and $a = 2$, $r = 4$

$$
\begin{array}{llll}
(0,0) \overset{F}{\mapsto} (0,1) & (4,0) \overset{F}{\mapsto} (4,1) & (8,0) \overset{F}{\mapsto} (8,1) & (12,0) \overset{F}{\mapsto} (12,1) \\
(1,0) \overset{F}{\mapsto} (1,2) & (5,0) \overset{F}{\mapsto} (5,2) & (9,0) \overset{F}{\mapsto} (9,2) & (13,0) \overset{F}{\mapsto} (13,2) \\
(2,0) \overset{F}{\mapsto} (2,4) & (6,0) \overset{F}{\mapsto} (6,4) & (10,0) \overset{F}{\mapsto} (10,4) & (14,0) \overset{F}{\mapsto} (14,4) \\
(3,0) \overset{F}{\mapsto} (3,8) & (7,0) \overset{F}{\mapsto} (7,8) & (11,0) \overset{F}{\mapsto} (11,8) & (15,0) \overset{F}{\mapsto} (15,8)
\end{array}
$$

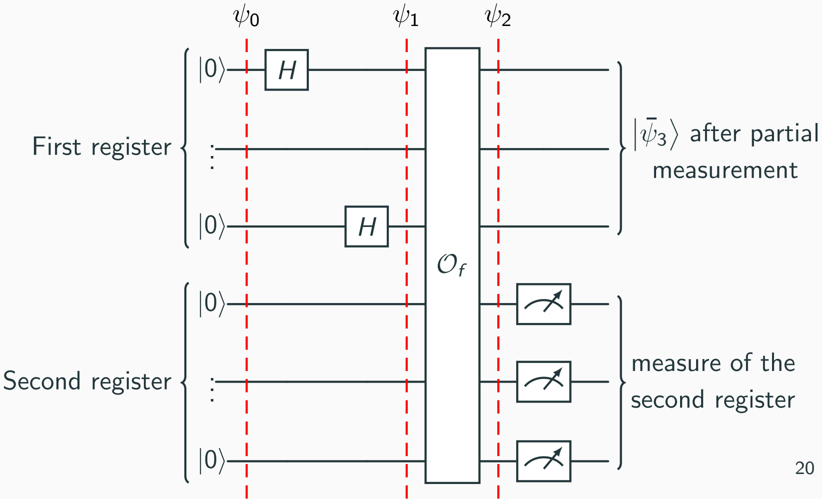
Oracle Circuit $2^n \geq N$

The oracle is composed of 2 registers: the first receives the integer k in binary with n bits, and the second, 0 on n bits. We write $|\underline{k}\rangle$ the register containing k written in binary. For instance, $|\underline{0}\rangle = |0\dots0\rangle$ with n bits. The initial state is $|\underline{k}\rangle \otimes |\underline{0}\rangle$.

- 

Starting the Circuit $2^n \geq N$

- Initialization: $|\psi_0\rangle = |\underline{0}\rangle \otimes |\underline{0}\rangle$.
- Hadamard: $|\psi_1\rangle = H^{\otimes n}(|\underline{0}\rangle) \otimes |\underline{0}\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle \right) \otimes |\underline{0}\rangle$
- Oracle: $|\psi_2\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle \otimes \left|\underline{a^k}\right\rangle$



Using the period to rewrite $|\psi_2\rangle$

- Assumption 3: $\mathrm{ord}(a) = r | 2^n$. This assumption is not true, and can be removed (see later)
- Under Assumption 3: $k = \alpha r + \beta$ with $0 \leq \beta < r$ and $0 \leq \alpha < 2^n/r$,

$$|\psi_2\rangle = \sum_{k=0}^{2^n-1} |\underline{k}\rangle \otimes |\underline{a^k}\rangle = \sum_{\beta=0}^{r-1} \left(\sum_{\alpha=0}^{2^n/r-1} |\underline{\alpha r + \beta}\rangle \right) \otimes |a^\beta\rangle$$

Using the period to rewrite $|\psi_2\rangle$

- Assumption 3: $\text{ord}(a) = r | 2^n$. This assumption is not true, and can be removed (see later)
- Under Assumption 3: $k = \alpha r + \beta$ with $0 \leq \beta < r$ and $0 \leq \alpha < 2^n/r$,

$$|\psi_2\rangle = \sum_{k=0}^{2^n-1} |\underline{k}\rangle \otimes |\underline{a^k}\rangle = \sum_{\beta=0}^{r-1} \left(\sum_{\alpha=0}^{2^n/r-1} |\underline{\alpha r + \beta}\rangle \right) \otimes |a^\beta\rangle$$

- If we measure the second register, we get for a fixed β_0,

$$|\psi_3\rangle = \sum_{\alpha=0}^{2^n/r-1} |\alpha r + \beta_0\rangle \otimes |a^{\beta_0}\rangle$$

Using the period to rewrite $|\psi_2\rangle$

- Assumption 3: $\mathrm{ord}(a) = r | 2^n$. This assumption is not true, and can be removed (see later)
- Under Assumption 3: $k = \alpha r + \beta$ with $0 \leq \beta < r$ and $0 \leq \alpha < 2^n/r$,

$$|\psi_2\rangle = \sum_{k=0}^{2^n-1} |\underline{k}\rangle \otimes \left|\underline{a^k}\right\rangle = \sum_{\beta=0}^{r-1} \left(\sum_{\alpha=0}^{2^n/r-1} \left|\underline{\alpha r + \beta}\right\rangle \right) \otimes \left|a^{\beta}\right\rangle$$

- If we measure the second register, we get for a fixed β_0,

$$|\psi_3\rangle = \sum_{\alpha=0}^{2^n/r-1} |\alpha r + \beta_0\rangle \otimes \left|a^{\beta_0}\right\rangle$$

- Assume we measure the first register, $|\alpha_0 r + \beta_0\rangle$ for fixed α_0 and β_0
- If we redo the computation, we will not the same β_0,
- We cannot do many measures of the first register ...

Example $N = 15$, $a = 2$

- $|\psi_0\rangle = |\underline{0}\rangle \otimes |\underline{0}\rangle$
- Hadamard Transform: $|\psi_1\rangle = (|\underline{0}\rangle + |\underline{1}\rangle + \ldots + |\underline{15}\rangle) \otimes |\underline{0}\rangle$
- Oracle: $|\psi_2\rangle = |\underline{0}\rangle \,.\, |\underline{a^0}\rangle + |\underline{1}\rangle \,.\, |\underline{a^1}\rangle + \ldots + |\underline{15}\rangle \,.\, |\underline{a^{15}}\rangle$

Example $N = 15$, $a = 2$

- $|\psi_0\rangle = |\underline{0}\rangle \otimes |\underline{0}\rangle$
- Hadamard Transform: $|\psi_1\rangle = (|\underline{0}\rangle + |\underline{1}\rangle + \ldots + |\underline{15}\rangle) \otimes |\underline{0}\rangle$
- Oracle: $|\psi_2\rangle = |\underline{0}\rangle \,.\, |\underline{a^0}\rangle + |\underline{1}\rangle \,.\, |\underline{a^1}\rangle + \ldots + |\underline{15}\rangle \,.\, |\underline{a^{15}}\rangle$
- Since $r = 4 | 2^4 = 16$, the values form a rectangular table

$$
\begin{aligned}
|\psi_2\rangle = & \left(|\underline{0}\rangle + |\underline{4}\rangle + |\underline{8}\rangle + |\underline{12}\rangle \right) . |\underline{1}\rangle + \\
& \left(|\underline{1}\rangle + |\underline{5}\rangle + |\underline{9}\rangle + |\underline{13}\rangle \right) . |\underline{2}\rangle + \\
& \left(|\underline{2}\rangle + |\underline{6}\rangle + |\underline{10}\rangle + |\underline{14}\rangle \right) . |\underline{4}\rangle + \\
& \left(|\underline{3}\rangle + |\underline{7}\rangle + |\underline{11}\rangle + |\underline{15}\rangle \right) . |\underline{8}\rangle
\end{aligned}
$$

- If we measure the second register, $|\underline{4}\rangle$, the first register is

$$
\left|\widetilde{\psi_3}\right\rangle = |\underline{2}\rangle + |\underline{6}\rangle + |\underline{10}\rangle + |\underline{14}\rangle
$$

- They are separated by the period $r = 4$, but how can we recover r ?

Discrete Fourier Transform

Complex numbers

- $$1 + z + \ldots + z^{n-1} = \begin{cases} n & \text{if } z = 1 \\ \frac{1-z^n}{1-z} & \text{otherwise.} \end{cases}$$

- Crucial Lemma: $n > 0, j \in \mathbb{Z}$,

$$\frac{1}{n}\sum_{k=0}^{n-1} e^{2i\pi\frac{kj}{n}} = \begin{cases} 1 & \text{if } \frac{j}{n} \text{ is an integer} \\ 0 & \text{otherwise.} \end{cases}$$

Discrete Fourier Transform

Complex numbers

- $$1 + z + \ldots + z^{n-1} = \begin{cases} n & \text{if } z = 1 \\ \frac{1-z^n}{1-z} & \text{otherwise.} \end{cases}$$
- Crucial Lemma: $n > 0, j \in \mathbb{Z}$,

$$\frac{1}{n}\sum_{k=0}^{n-1} e^{2i\pi\frac{kj}{n}} = \begin{cases} 1 & \text{if } \frac{j}{n} \text{ is an integer} \\ 0 & \text{otherwise.} \end{cases}$$

Discrete Fourier Transform and Inverse

$$\widehat{F}\,|\underline{k}\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} e^{2i\pi\frac{kj}{2^n}}\,|\underline{j}\rangle \text{ and } \widehat{F}^{-1}\,|\underline{k}\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} e^{-2i\pi\frac{kj}{2^n}}\,|\underline{j}\rangle$$

Discrete Fourier Transform

Complex numbers

- $$1 + z + \ldots + z^{n-1} = \begin{cases} n & \text{if } z = 1 \\ \frac{1-z^n}{1-z} & \text{otherwise.} \end{cases}$$
- Crucial Lemma: $n > 0, j \in \mathbb{Z}$,

$$\frac{1}{n} \sum_{k=0}^{n-1} e^{2i\pi \frac{kj}{n}} = \begin{cases} 1 & \text{if } \frac{j}{n} \text{ is an integer} \\ 0 & \text{otherwise.} \end{cases}$$

Discrete Fourier Transform and Inverse

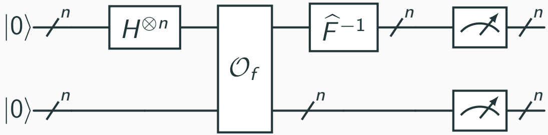
$$\widehat{F} \left| \underline{k} \right\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{2i\pi \frac{kj}{2^n}} \left| \underline{j} \right\rangle \text{ and } \widehat{F}^{-1} \left| \underline{k} \right\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{-2i\pi \frac{kj}{2^n}} \left| \underline{j} \right\rangle$$

The Discrete Fourier Transform is Linear and Unitary

$$\text{If } \left|\psi\right\rangle = \sum_{k=0}^{2^n-1} \alpha_k \left| \underline{k} \right\rangle, \text{ then } \widehat{F} \left|\psi\right\rangle = \sum_{k=0}^{2^n-1} \alpha_k \widehat{F} \left| \underline{k} \right\rangle$$

Shor Circuit

- Initialization: $|\psi_0\rangle = |\underline{0}\rangle \otimes |\underline{0}\rangle$.
- Hadamard: $|\psi_1\rangle = H^{\otimes n}(|\underline{0}\rangle) \otimes |\underline{0}\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle\right) \otimes |\underline{0}\rangle$
- Oracle: $|\psi_2\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle \otimes \left|\underline{a^k}\right\rangle$



- Measure of the first register: $\left|\frac{2^n \ell}{r}\right\rangle$
- Allows (often) to get r (or a factor of r)

Computation

- After measuring the second register $\left|\bar{\psi}_3\right\rangle = \sum_{\alpha=0}^{2^n/r-1} \left|\underline{\alpha r + \beta_0}\right\rangle$

Computation

- After measuring the second register $\left|\bar{\psi}_3\right\rangle = \sum_{\alpha=0}^{2^n/r-1} \left|\underline{\alpha r + \beta_0}\right\rangle$
- Action of $\widehat{F}^{-1}$:

$$\left|\bar{\psi}_4\right\rangle = \widehat{F}^{-1}\left|\hat{\psi}_3\right\rangle = \sum_{\alpha=0}^{2^n/r-1} \widehat{F}^{-1}\left|\underline{\alpha r + \beta_0}\right\rangle$$

$$= \sum_{\alpha}\sum_{j=0}^{2^n-1} e^{-\frac{2i\pi(\alpha r+\beta_0)j}{2^n}}\left|\underline{j}\right\rangle = \sum_{j} \overbrace{\left(\sum_{\alpha} e^{-2i\pi\frac{\alpha j}{2^n/r}}\right)}^{0 \text{ or } 1} e^{-2i\pi\frac{\beta_0 j}{2^n}}\left|\underline{j}\right\rangle$$

$$= \sum_{j \text{ with } j/(2^n/r) \text{ integer}} e^{-2i\pi\frac{\beta_0 j}{2^n}}\left|j\right\rangle = \sum_{\ell=0}^{r-1} e^{-2i\pi\beta_0\frac{\ell}{r}}\left|\frac{2^n\ell}{r}\right\rangle$$

Computation

- After measuring the second register $\left|\bar{\psi}_3\right\rangle = \sum_{\alpha=0}^{2^n/r-1} \left|\underline{\alpha r + \beta_0}\right\rangle$
- Action of $\widehat{F}^{-1}$:

$$\left|\bar{\psi}_4\right\rangle = \widehat{F}^{-1}\left|\hat{\psi}_3\right\rangle = \sum_{\alpha=0}^{2^n/r-1} \widehat{F}^{-1}\left|\underline{\alpha r + \beta_0}\right\rangle$$

$$= \sum_{\alpha}\sum_{j=0}^{2^n-1} e^{-\frac{2i\pi(\alpha r+\beta_0)j}{2^n}}\left|\underline{j}\right\rangle = \sum_{j} \overbrace{\left(\sum_{\alpha} e^{-2i\pi\frac{\alpha j}{2^n/r}}\right)}^{0 \text{ or } 1} e^{-2i\pi\frac{\beta_0 j}{2^n}}\left|\underline{j}\right\rangle$$

$$= \sum_{j \text{ with } j/(2^n/r) \text{ integer}} e^{-2i\pi\frac{\beta_0 j}{2^n}}\left|j\right\rangle = \sum_{\ell=0}^{r-1} e^{-2i\pi\beta_0\frac{\ell}{r}}\left|\frac{2^n\ell}{r}\right\rangle$$

- Measure the first register: $\left|\frac{2^n\ell}{r}\right\rangle$, for $\ell \in \{0, 1, \dots, r-1\}$
- We get $m = \frac{2^n\ell}{r}$ for one of the states $\left|\frac{2^n\ell}{r}\right\rangle$

Measure the first register

$m = \frac{2^n \ell}{r}$ **integer with** n **known and** ℓ **unknown**

- Divide m by 2^n to obtain the rational $x = \frac{m}{2^n} = \frac{\ell}{r}$
- If $x \in \mathbb{Z}$, we get no information on r, and we redo the quantum circuit
- If $\gcd(\ell, r) = 1$, then $\frac{\ell}{r}$ is irreducible and we get r.
- If $\gcd(\ell, r) \neq 1$, then $x = \frac{m}{2^n} = \frac{\ell'}{r'} = \frac{\ell}{r}$ and we get r' a factor of r. We redo the computation with $a' = a^{r'}$ which is of period r/r'.

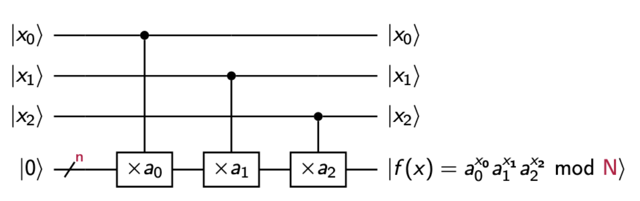
Implementation of the oracle

Reduce exponentiation to controlled multi-product modulo N:

$$f(x) = a^x = \prod_i \left(a^{2^i}\right)^{x_i} = \prod_i \left(a_i\right)^{x_i} \bmod N, \text{ where } a_i = a^{2^i} \bmod N$$

The constants a_i are precomputed:

- Asymptotic best: $O(n \times (n \log n))$ operations
- Typical: $O(n \times (n^2))$ operations



Continued Fractions

Definition

- $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ldots + \frac{1}{a_n}}}}$, noted $[a_0, a_1, \ldots, a_n]$
- E.g., $[5, 2, 1, 4] = 5 + \frac{1}{2 + \frac{1}{1 + \frac{1}{4}}} = 5.3571428\ldots$
- $[5] = 5, [5, 2] = \frac{11}{2} = 5.5, [5, 2, 1] = \frac{16}{3} = 5.33\ldots$

Continued Fractions

Definition

- $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_n}}}}$, noted $[a_0, a_1, \dots, a_n]$
- E.g., $[5, 2, 1, 4] = 5 + \frac{1}{2 + \frac{1}{1 + \frac{1}{4}}} = 5.3571428\dots$
- $[5] = 5, [5, 2] = \frac{11}{2} = 5.5, [5, 2, 1] = \frac{16}{3} = 5.33\dots$

Good Approximation by continued fractions

- $\pi = 3.14159\dots \approx \frac{314}{100}$ (denominator is large)
- $\frac{314}{100} = 3 + \frac{14}{100} = 3 + \frac{1}{\frac{100}{14}} = 3 + \frac{1}{7 + \frac{2}{14}} = 3 + \frac{1}{7 + \frac{1}{7}} = [3, 7, 7]$
- $[3, 7] = 3 + \frac{1}{7} = \frac{22}{7} = 3.1428$
- $[3, 7, 15, 1] = \frac{355}{113} = 3.14159292\dots$ (same order with 6 exact values instead of 2)

Example Shor with $N = 21$

- $N = 21$, $a = 2$, $2^n = 512 = 2^9$
- Circuit outputs $|427\rangle$, so $x = \frac{427}{512}$
- $\frac{427}{512} \approx \frac{4}{5}$ so order 5 ??
- $\frac{427}{512} = [0, 1, 5, 42, 2]$ and $[0, 1] = 1, [0, 1, 5] = \frac{5}{6}, [0, 1, 5, 42] = \frac{211}{253}$
- We keep the best fraction whose denominator is $\leq N$ and it gives r or a fraction of r

Example Shor with $N = 21$

- $N = 21,\ a = 2,\ 2^n = 512 = 2^9$
- Circuit outputs $|427\rangle$, so $x = \frac{427}{512}$
- $\frac{427}{512} \approx \frac{4}{5}$ so order 5 ??
- $\frac{427}{512} = [0, 1, 5, 42, 2]$ and $[0, 1] = 1, [0, 1, 5] = \frac{5}{6}, [0, 1, 5, 42] = \frac{211}{253}$
- We keep the best fraction whose denominator is $\leq N$ and it gives r or a fraction of r

Shor algorithm with arbitrary order

- $N = 21,\ a = 2,\ 2^n = 512 = 2^9 \geq N^2$
- $|\psi_0\rangle = |\underline{0}\rangle \otimes |\underline{0}\rangle$
- $|\psi_1\rangle = \sum_{k=0}^{r-1} |\underline{k}\rangle \otimes |\underline{0}\rangle$
- $|\psi_2\rangle = \sum_{k=0}^{r-1} |\underline{k}\rangle \otimes |\underline{a^k \bmod N}\rangle$
- $r = 6$ and $\frac{2^n \ell}{r} \notin \mathbb{Z}$

Example

The first two lines have 86 terms and 85 in the others

- The state $|\psi_2\rangle$ is not rectangular:

$$\begin{aligned}|\psi_2\rangle &= \frac{1}{\sqrt{512}}(|\underline{0}\rangle + |\underline{6}\rangle + \ldots + |\underline{504}\rangle + |\underline{510}\rangle)\,|\underline{1}\rangle \\ &+ \frac{1}{\sqrt{512}}(|\underline{1}\rangle + |\underline{7}\rangle + \ldots + |\underline{505}\rangle + |\underline{511}\rangle)\,|\underline{2}\rangle \\ &+ \frac{1}{\sqrt{512}}(|\underline{2}\rangle + |\underline{8}\rangle + \ldots + |\underline{506}\rangle)\,|\underline{4}\rangle \\ &+ \ldots \\ &+ \frac{1}{\sqrt{512}}(|\underline{5}\rangle + |\underline{11}\rangle + \ldots + |\underline{509}\rangle)\,|\underline{11}\rangle\end{aligned}$$

Example

The first two lines have 86 terms and 85 in the others

- The state $|\psi_2\rangle$ is not rectangular:

$$
\begin{aligned}
|\psi_2\rangle = & \frac{1}{\sqrt{512}}(|\underline{0}\rangle + |\underline{6}\rangle + \ldots + |\underline{504}\rangle + |\underline{510}\rangle)\,|\underline{1}\rangle \\
& + \frac{1}{\sqrt{512}}(|\underline{1}\rangle + |\underline{7}\rangle + \ldots + |\underline{505}\rangle + |\underline{511}\rangle)\,|\underline{2}\rangle \\
& + \frac{1}{\sqrt{512}}(|\underline{2}\rangle + |\underline{8}\rangle + \ldots + |\underline{506}\rangle)\,|\underline{4}\rangle \\
& + \ldots \\
& + \frac{1}{\sqrt{512}}(|\underline{5}\rangle + |\underline{11}\rangle + \ldots + |\underline{509}\rangle)\,|\underline{11}\rangle
\end{aligned}
$$

- measure the second register $|2\rangle$: $|\psi_3\rangle = |\underline{1}\rangle + |\underline{7}\rangle + \ldots + |\underline{511}\rangle$
- $|\psi_4\rangle = \hat{F}^{-1}|\psi_3\rangle = \sum_{\alpha=0}^{85} \hat{F}^{-1}|\underline{6\alpha+1}\rangle$
- $|\psi_4\rangle = \sum_{j=0}^{511}\left(\sum_{\alpha=0}^{85} e^{-2i\pi\frac{6\alpha j}{512}}\right) e^{-2i\pi\frac{j}{512}}\,|\underline{j}\rangle$

Example Shor with arbitrary order

$$|\psi_4\rangle = \frac{1}{\sqrt{512}} \sum_{j=0}^{511} \left(\frac{1}{\sqrt{86}} \sum_{\alpha=0}^{85} e^{-2i\pi \frac{6\alpha j}{512}} \right) e^{-2i\pi \frac{j}{512}} \left| \underline{j} \right\rangle$$

Now, $\Sigma(j) = \frac{1}{\sqrt{86}} \sum_{\alpha=0}^{85} e^{-2i\pi \frac{6\alpha j}{512}}$ does not take only 0 /1 values.

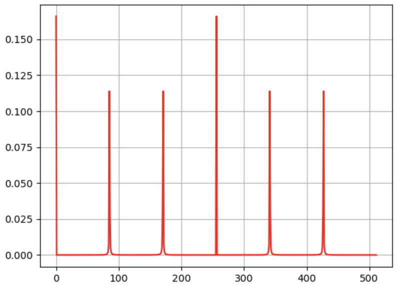
Example Shor with arbitrary order

$$|\psi_4\rangle = \frac{1}{\sqrt{512}} \sum_{j=0}^{511} \left(\frac{1}{\sqrt{86}} \sum_{\alpha=0}^{85} e^{-2i\pi \frac{6\alpha j}{512}} \right) e^{-2i\pi \frac{j}{512}} \left|\underline{j}\right\rangle$$

Now, $\Sigma(j) = \frac{1}{\sqrt{86}} \sum_{\alpha=0}^{85} e^{-2i\pi \frac{6\alpha j}{512}}$ does not take only 0 /1 values.

If we measure the first register, we get $|j\rangle$ with probability $|\Sigma(j)|^2$.

The proba. are ≈ 0, except when $j \approx \frac{2^n \ell}{r}$: for $\ell = 5$, $\frac{512 \times 5}{6} = 426.66$.



j	p_j
422	0.00062...
423	0.00099...
424	0.00186...
425	0.00469...
426	0.02888...
427	**0.11389...**
428	0.00702...
429	0.00226...
430	0.00109...
431	0.00063...

Hardy-Wright Theorem

Theorem
Let $x \in \mathbb{R}$ and a rational $\frac{p}{q}$ such that

$$\left|x - \frac{p}{q}\right| < \frac{1}{2q^2}.$$

Then, $\frac{p}{q}$ is obtained as one of the continued fractions of x.

Theorem
Let $x \in \mathbb{R}$ and a rational $\frac{p}{q}$ such that

$$\left|x - \frac{p}{q}\right| < \frac{1}{2q^2}.$$

Then, $\frac{p}{q}$ is obtained as one of the continued fractions of x.

Let m the closest integer to $\frac{2^n \ell}{r}$. So, $|m - \frac{2^n \ell}{r}| < \frac{1}{2}$.

If $x = \frac{m}{2^n}$, we get $|x - \frac{\ell}{r}| < \frac{1}{2^{n+1}}$.

As we set $2^n \geq N^2 \geq r^2$, $|x - \frac{\ell}{r}| < \frac{1}{2r^2}$.

Using Theorem, we obtain $\frac{\ell}{r}$ as one of the continued fractions of x.

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H
- Shor and Simon algorithms: special case of HSP

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H
- Shor and Simon algorithms: special case of HSP
- Kitaev: any Abelian Group G

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H
- Shor and Simon algorithms: special case of HSP
- Kitaev: any Abelian Group G
- Non-abelian: Kuperberg subexponential algo. for Dihedral HSP

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H
- Shor and Simon algorithms: special case of HSP
- Kitaev: any Abelian Group G
- Non-abelian: Kuperberg subexponential algo. for Dihedral HSP
- LWE (learning with errors problems) can be reduced to (stronger version) Dihedral HSP (with errors)

Conclusion

Generalization

- HSP (Hidden Subgroup Problem): Let G a group and H a subgroup. The function f is constant on each coset of H, find H
- Shor and Simon algorithms: special case of HSP
- Kitaev: any Abelian Group G
- Non-abelian: Kuperberg subexponential algo. for Dihedral HSP
- LWE (learning with errors problems) can be reduced to (stronger version) Dihedral HSP (with errors)

How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits

Craig Gidney[1] and Martin Ekerå[2]

Conclusion

New Results on factorization

- Shor algorithm: $3n$ qubits and $O(n^2)$ gates

Conclusion

New Results on factorization

- Shor algorithm: $3n$ qubits and $O(n^2)$ gates
- Regev algorithm: $O(n^{3/2})$ qubits and $O(n^{3/2})$ gates, runs $n^{1/2}$

Conclusion

New Results on factorization

- Shor algorithm: $3n$ qubits and $O(n^2)$ gates
- Regev algorithm: $O(n^{3/2})$ qubits and $O(n^{3/2})$ gates, runs $n^{1/2}$
- Ragavan-Vaikuntanathan: $10n$ qubits and $O(n^{3/2})$ gates, runs $n^{1/2}$

Conclusion

New Results on factorization

- Shor algorithm: $3n$ qubits and $O(n^2)$ gates
- Regev algorithm: $O(n^{3/2})$ qubits and $O(n^{3/2})$ gates, runs $n^{1/2}$
- Ragavan-Vaikuntanathan: $10n$ qubits and $O(n^{3/2})$ gates, runs $n^{1/2}$
- $n/2 + o(n)$ qubits and $O(n^2)$ gates, runs constants [CFS24]

Regev Quantum factorisation algorithm: reducing the circuit size by $\sqrt{n}$

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Use the period to factorize: $2^{984} = 1163 \bmod 8051$

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Use the period to factorize: $2^{984} = 1163 \bmod 8051$
- $(1163 - 1)(1163 + 1) = 0 \bmod 8051$

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Use the period to factorize: $2^{984} = 1163 \bmod 8051$
- $(1163 - 1)(1163 + 1) = 0 \bmod 8051$
- $\gcd(1162, 8051) = 83$ and $8051 = 83 \times 97$!

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Regev algorithm: $4^{z_1} 9^{z_2} \bmod N$ - order ? (27,15) much shorter !

```
.....
[1, 9, 81, 729, 6561, 2692, 75, 675, 6075, 6369, 964, 625, 5625, 2319, 4769, 2666, 7892, 6620, 3223, 4854, 3431,
[4, 36, 324, 2916, 2091, 2717, 300, 2700, 147, 1323, 3856, 2500, 6398, 1225, 2974, 2613, 7415, 2327, 4841, 3314,
[16, 144, 1296, 3613, 313, 2817, 1200, 2749, 588, 5292, 7373, 1949, 1439, 4900, 3845, 2401, 5507, 1257, 3262, 52
[64, 576, 5184, 6401, 1252, 3217, 4800, 2945, 2352, 5066, 5339, 7796, 5756, 3498, 7329, 1553, 5926, 5028, 4997,
[256, 2304, 4634, 1451, 5008, 4817, 3098, 3729, 1357, 4162, 5254, 7031, 6922, 5941, 5163, 6212, 7602, 4010, 388
[1024, 1165, 2434, 5804, 3930, 3166, 4341, 6865, 5428, 546, 4914, 3971, 3535, 7662, 4550, 695, 6255, 7989, 7493,
[4096, 4660, 1685, 7114, 7669, 4613, 1262, 3307, 5610, 2184, 3554, 7833, 6089, 6495, 2098, 2780, 867, 7803, 581
[282, 2538, 6740, 4303, 6523, 2350, 5048, 5177, 6338, 685, 6165, 7179, 203, 1827, 341, 3069, 3468, 7059, 7174, 1
[1128, 2101, 2807, 1110, 1939, 1349, 4090, 4606, 1199, 2740, 507, 4563, 812, 7308, 1364, 4225, 5821, 4083, 4543,
[4512, 353, 3177, 4440, 7756, 5396, 258, 2322, 4796, 2909, 2028, 2150, 3248, 5079, 5456, 798, 7182, 230, 2070, 2
[1946, 1412, 4657, 1658, 6871, 5482, 1032, 1237, 3082, 3585, 61, 549, 4941, 4214, 5722, 3192, 4575, 920, 229, 20
[7784, 5648, 2526, 6632, 3331, 5826, 4128, 4948, 4277, 6289, 244, 2196, 3662, 754, 6786, 4717, 2198, 3680, 916,
[6983, 6490, 2053, 2375, 5273, 7202, 410, 3690, 1006, 1003, 976, 733, 6597, 3016, 2991, 2766, 741, 6669, 3664, 7
[3779, 1807, 161, 1449, 4990, 4655, 1640, 6709, 4024, 4012, 3904, 2932, 2235, 4013, 3913, 3013, 2964, 2523, 6605
[7065, 7228, 644, 5796, 3858, 2518, 6560, 2683, 8045, 7997, 7565, 3677, 889, 8001, 7601, 4001, 3805, 2041, 2267,
[4107, 4759, 2576, 7082, 7381, 2021, 2087, 2681, 8027, 7835, 6107, 6657, 3556, 7851, 6251, 7953, 7169, 113, 1017
[326, 2934, 2253, 4175, 5371, 33, 297, 2673, 7955, 7187, 275, 2475, 6173, 7251, 851, 7659, 4523, 452, 4068, 4408
[1304, 3685, 961, 598, 5382, 132, 1188, 2641, 7667, 4595, 1100, 1849, 539, 4851, 3404, 6483, 1990, 1808, 170, 15
[5216, 6689, 3844, 2392, 5426, 528, 4752, 2513, 6515, 2278, 4400, 7396, 2156, 3302, 5565, 1779, 7960, 7232, 680,
[4762, 2603, 7325, 1517, 5602, 2112, 2906, 2001, 1907, 1061, 1498, 5431, 573, 5157, 6158, 7116, 7687, 4775, 2720
[2946, 2361, 5147, 6068, 6306, 397, 3573, 8004, 7628, 4244, 5992, 5622, 2292, 4526, 479, 4311, 6595, 2998, 2829,
[3733, 1393, 4486, 119, 1071, 1588, 6241, 7863, 6359, 874, 7866, 6386, 1117, 2002, 1916, 1142, 2227, 3941, 3265,
[6881, 5572, 1842, 476, 4284, 6352, 811, 7299, 1283, 3496, 7311, 1391, 4468, 8008, 7664, 4568, 857, 7713, 5009,
[3371, 6186, 7368, 1904, 1034, 1255, 3244, 5043, 5132, 5933, 5091, 5564, 1770, 7879, 6503, 2170, 3428, 6699, 393
[5433, 591, 5319, 7616, 4136, 5020, 4925, 4070, 4426, 7630, 4262, 6154, 7080, 7363, 1859, 629, 5661, 2643, 7685,
[5630, 2364, 5174, 6311, 442, 3978, 3598, 178, 1602, 6367, 946, 463, 4167, 5299, 7436, 2516, 6542, 2521, 6587, 2
[6418, 1405, 4594, 1091, 1768, 7861, 6341, 712, 6408, 1315, 3784, 1852, 566, 5094, 5591, 2013, 2015, 2033, 2195,
[1519, 5620, 2274, 4364, 7072, 7291, 1211, 2848, 1479, 5260, 7085, 7408, 2264, 4274, 6262, 1, 9, 81, 729, 6561,
[6076, 6378, 1045, 1354, 4135, 5011, 4844, 3341, 5916, 4938, 4187, 5479, 1005, 994, 895, 4, 36, 324, 2916, 2091,
```

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Regev algorithm: $4^{z_1} 9^{z_2} \bmod N$ - order ? (27,15) much shorter !
- does not work because $2^{27} \cdot 3^{15} = -1 \bmod 8051$...

Regev's algorithm: idea

- Shor algorithm: compute $4^z \bmod N$ - order $(4 \bmod 8051) = 984$
- Regev algorithm: $4^{z_1} 9^{z_2} \bmod N$ - order ? (27,15) much shorter !
- does not work because $2^{27} \cdot 3^{15} = -1 \bmod 8051$...
- $(19, 47)$: also Period. $2^{19} \cdot 3^{47} = 6888 \bmod 8051$ non-trivial square root of unity
- $gcd(6887, 8051) = 97$

Regev's algorithm: number of gates (1/2)

Hadamard and FFT are free, but oracle function...
The most expensive step is the function evaluation

$$(z_1, \ldots, z_d) \mapsto \prod_{i=1}^{d} a_i^{z_i} \bmod N$$

Regev's algorithm: number of gates (1/2)

Hadamard and FFT are free, but oracle function...
The most expensive step is the function evaluation

$$(z_1, \ldots, z_d) \mapsto \prod_{i=1}^{d} a_i^{z_i} \bmod N$$

In dimension d, we only have to raise to power $2^{n/d}$ to see the period

Regev's algorithm: number of gates (1/2)

Hadamard and FFT are free, but oracle function...
The most expensive step is the function evaluation

$$(z_1, \ldots, z_d) \mapsto \prod_{i=1}^{d} a_i^{z_i} \bmod N$$

In dimension d, we only have to raise to power $2^{n/d}$ to see the period

- so each exponentiation requires only n/d multiplications (pigeonhole principle)

Regev's algorithm: number of gates (1/2)

Hadamard and FFT are free, but oracle function...
The most expensive step is the function evaluation

$$(z_1, \ldots, z_d) \mapsto \prod_{i=1}^{d} a_i^{z_i} \bmod N$$

In dimension d, we only have to raise to power $2^{n/d}$ to see the period

- so each exponentiation requires only n/d multiplications (pigeonhole principle)
- but, we have to do it d times, do we gain something ....

Regev's algorithm: number of gates (2/2)

The trick is to choose $a_1, \ldots, a_d$ as small numbers

- E.g., they can be the squares of the first d primes (4,9,25,49,...)
- To get $a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8$: $((a_1 a_2)(a_3 a_4))((a_5 a_6)(a_7 a_8))$

- Then, we can compute $\prod_{i=1}^{d} a_i^{z_i} \bmod N$ with exponents z_i up to $2^{n/d}$ using only n/d big number multiplications, requiring $\tilde{O}(n^2/d)$ gates
- To get $a_1^{13} a_2^{9} a_3^{3} a_4^{6}$: from $1 = a_1^0 a_2^0 a_3^0 a_4^0$,
  - $a_1^1 a_2^1 a_3^0 a_4^0$ multiply by $a_1 a_2$
  - $a_1^2 a_2^2 a_3^0 a_4^0$ square
  - $a_3^3 a_2^2 a_3^0 a_4^1$ multiply by $a_1 a_4$
  - $a_3^6 a_2^4 a_3^0 a_4^2$ square
  - $a_3^6 a_2^4 a_3^1 a_4^3$ multiply by $a_3 a_4$
  - $a_3^{12} a_2^8 a_3^2 a_4^6$ square
  - $a_3^{13} a_2^9 a_3^3 a_4^6$ multiply by $a_1 a_2 a_3$

Final algorithm

To recover the period, we need a generalization of continued fractions which is LLL for lattice

Final algorithm

To recover the period, we need a generalization of continued fractions which is LLL for lattice

- We need d vectors ($\approx$ Simon's post-processing step)

Final algorithm

To recover the period, we need a generalization of continued fractions which is LLL for lattice

- We need d vectors ($\approx$ Simon's post-processing step)
- LLL has an approximation factor 2^d, exponent larger: $2^{n/d+d}$

Final algorithm

To recover the period, we need a generalization of continued fractions which is LLL for lattice

- We need d vectors ($\approx$ Simon's post-processing step)
- LLL has an approximation factor 2^d, exponent larger: $2^{n/d+d}$
- optimal choice: $d = \sqrt{n}$, $\Rightarrow \tilde{O}(n^{3/2})$ and run the circuit $n^{1/2}$ times

Final algorithm

To recover the period, we need a generalization of continued fractions which is LLL for lattice

- We need d vectors ($\approx$ Simon's post-processing step)
- LLL has an approximation factor 2^d, exponent larger: $2^{n/d+d}$
- optimal choice: $d = \sqrt{n}$, $\Rightarrow \tilde{O}(n^{3/2})$ and run the circuit $n^{1/2}$ times

Algorithm

1. Choose $a_1, \dots, a_d$ squares of the first $d = \sqrt{n}$ primes $4, 9, 25, 49, \dots$
2. Apply the following quantum circuit d times:
   (i) Compute $\prod_{i=1}^{d} a_i^{z_i}$ mod N in superposition over all $(z_1, \dots, z_d) \in \left(0, \dots, 2^{n/d+d}\right)^d$
   (ii) Apply QFT and measure to get an approximate dual lattice vector
3. Use the lattice algorithm LLL to recover the period $(z_1, \dots, z_d)$
4. Use the period to factor N

Ragavan and Vaikuntanathan variant

Solve 2 drawbacks of Regev's algorithm

1. Number of qubits: $O(n \log n) \Rightarrow 10n$: avoid the squaring (not reversible !) while modular multiplications are
   - Fibonacci representation: every number can be written as $\sum_{i \in I} F_i$
   - Kasiski: $(a^{F_k}, a^{F_{k+1}}) \Rightarrow (a^{F_{k+2}}, a^{F_{k+1}})$ using only multiplications
   - Circuit reversible, but check invertible elements $\left|a, b, a^{-1} \bmod N, b^{-1} \bmod N\right\rangle \Rightarrow \left|a, ab, a^{-1} \bmod N, (ab)^{-1} \bmod N\right\rangle$
   - $45.7\sqrt{n}$ modular multiplications while Regev just $6\sqrt{n}$, but the space increases to store the different values to be reversible...
2. Number of runs: Regev requires no errors on the $\sqrt{n}$ runs, while RV using a filtering technique can remove very bad outputs

Reducing the number of qubits

New algorithm[1]

- Factoring RSA moduli using $n/2 + o(n)$ qubits and $O(n^3)$ gates
- Benmarks for RSA-2048: ≤ 1700 qubits and $\leq 60 \times 2^{36}$ Toffoli gates (in 60 runs)
- Based on a completely classical arithmetic circuit

[1] CFS, CRYPTO 2025, "Reducing the Number of Qubits in Quantum Factoring"

New algorithm[1]

- Factoring RSA moduli using $n/2 + o(n)$ qubits and $O(n^3)$ gates
- Benmarks for RSA-2048: ≤ 1700 qubits and $\leq 60 \times 2^{36}$ Toffoli gates (in 60 runs)
- Based on a completely classical arithmetic circuit
- Gidney reduces: qubits down to 1399 logical qubits by computing the MSB rather than the LSB, 2^{32} Toffoli gates as previous counting and 9.2 runs, and update estimates at the physical level

Gidney latest result

How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA

June 9, 2025

[1]CFS, CRYPTO 2025, "Reducing the Number of Qubits in Quantum Factoring"

Discrete logarithm and RSA special case

Find d s.t. $a = g^d$:

[2] Ekerå, Håstad, "Quantum algorithms for computing short discrete logarithms and factoring RSA integers, PQCrypto 2017"
[3] Ekerå, "On post-processing in the quantum algorithm for computing short discrete logarithms", DCC 2020

Discrete logarithm and RSA special case

Find d s.t. $a = g^{d}$: $f(x, y) := g^{x} a^{-y} = g^{x-dy} \bmod N$

- Also a hidden period problem: $f(x + d, y + 1) = f(x, y)$
- Also reduces to controlled multi-product

[2]Ekerå, Håstad, "Quantum algorithms for computing short discrete logarithms and factoring RSA integers, PQCrypto 2017"

[3]Ekerå, "On post-processing in the quantum algorithm for computing short discrete logarithms", DCC 2020

Discrete logarithm and RSA special case

Find d s.t. $a = g^d$: $f(x, y) := g^x a^{-y} = g^{x-dy} \bmod N$

- Also a hidden period problem: $f(x + d, y + 1) = f(x, y)$
- Also reduces to controlled multi-product

Ekerå & Håstad method[2,3]:

- Reduce RSA factorisation (N=pq) to small DLOG of size $n/2$: if we recover $p + q$, we can factor N
- Use an input register of size $n/2 + (n/2)/s$ for some s
- $\approx s + 1$ measurements to find d via an efficient lattice-based post-processing. Typically $s = O(\log n)$.

Space is reduced to: $n/2 +$ workspace

[2] Ekerå, Håstad, "Quantum algorithms for computing short discrete logarithms and factoring RSA integers, PQCrypto 2017"

[3] Ekerå, "On post-processing in the quantum algorithm for computing short discrete logarithms", DCC 2020

Variant Shor's algorithm

Ideas

- Once $p + q$ is known, using $N = pq$, recover p is easy

[4] "Quantum period-finding is compression robust"

Variant Shor's algorithm

Ideas

- Once $p + q$ is known, using $N = pq$, recover p is easy
- $G = \langle g \rangle$ a cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ of order $> (p + q - 2)/2$
- Compute $x = g^{(N-1)/2} = g^{(p+q-2)/2}$ mod N since $(N - \varphi(N) - 1)/2 = (p + q - 2)/2$ as $\varphi(N) = N - p - q + 1$
- Compute short discrete logarithm $d = (p + q - 2)/2$ from g and x

[4] "Quantum period-finding is compression robust"

Variant Shor's algorithm

Ideas

- Once $p+q$ is known, using $N = pq$, recover p is easy
- $G = \langle g \rangle$ a cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ of order $> (p+q-2)/2$
- Compute $x = g^{(N-1)/2} = g^{(p+q-2)/2} \bmod N$ since $(N - \varphi(N) - 1)/2 = (p+q-2)/2$ as $\varphi(N) = N - p - q + 1$
- Compute short discrete logarithm $d = (p+q-2)/2$ from g and x
- Get many pairs (j, k) s.t. k is the ℓ most significant bits of $dj \bmod 2^m$: Hidden Number Problem (HNP)

[4] "Quantum period-finding is compression robust"

Variant Shor's algorithm

Ideas

- Once $p+q$ is known, using $N = pq$, recover p is easy
- $G = \langle g \rangle$ a cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ of order $> (p+q-2)/2$
- Compute $x = g^{(N-1)/2} = g^{(p+q-2)/2} \bmod N$ since $(N - \varphi(N) - 1)/2 = (p+q-2)/2$ as $\varphi(N) = N - p - q + 1$
- Compute short discrete logarithm $d = (p+q-2)/2$ from g and x
- Get many pairs (j, k) s.t. k is the ℓ most significant bits of $dj \bmod 2^m$: Hidden Number Problem (HNP)
- May, Schlieper[4]: we can replace f by $h \circ f$ where h is a universal hash function is still periodic

[4] "Quantum period-finding is compression robust"

Variant Shor's algorithm

Ideas

- Once $p+q$ is known, using $N = pq$, recover p is easy
- $G = \langle g \rangle$ a cyclic subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ of order $> (p+q-2)/2$
- Compute $x = g^{(N-1)/2} = g^{(p+q-2)/2} \bmod N$ since $(N - \varphi(N) - 1)/2 = (p+q-2)/2$ as $\varphi(N) = N - p - q + 1$
- Compute short discrete logarithm $d = (p+q-2)/2$ from g and x
- Get many pairs (j, k) s.t. k is the ℓ most significant bits of $dj \bmod 2^m$: Hidden Number Problem (HNP)
- May, Schlieper[4]: we can replace f by $h \circ f$ where h is a universal hash function is still periodic
- How to compute some bits of $a^k \bmod N \bmod 2^r$ with $o(\log n)$ space

[4] "Quantum period-finding is compression robust"