



Kwantowy algorytm Shora - czyli jak łamać szyfry

Wiesław Płaczek

Zakład Zastosowań Metod Obliczeniowych
Instytut Fizyki Uniwersytetu Jagiellońskiego

Plan

- ◆ Szyfrowanie informacji.
- ◆ Kryptosystem RSA.
- ◆ Algorytm Shora – odsłona 1.
- ◆ Algorytm Shora – odsłona 2.
- ◆ Algorytm Shora – odsłona 3.
- ◆ Kwantowa transformacja Fouriera.
- ◆ Znajdowanie okresu funkcji.
- ◆ Podsumowanie.
- ◆ Literatura.

Szyfrowanie informacji

- ☞ *Powszechnie stosowana metoda ochrony informacji przesyłanej przez przez niepewne łącza.*
- ❖ Standard szyfrowania danych (*data-encryption standard – DES*) przyjęty przez National Bureau of Standards, USA:
 - Tajny klucz (lub klucze) k .
 - Algorytm szyfrowania danych E_k .
 - Algorytm deszyfrowania danych D_k .
 - **Dla każdego komunikatu m muszą być spełnione warunki:**
 1. $D_k(E_k(m)) = m$.
 2. Zarówno E_k jak i D_k można wykonać wydajnie.
 3. Bezpieczeństwo systemu zależy tylko od tajności klucza k , a nie od tajności algorytmów E_k i D_k .

Szyfrowanie informacji – c.d.

- ❖ Podstawowy problem:
 - *Jak bezpiecznie przekazywać klucze do nadawcy i odbiorcy?*
 - Np. „pięta Achillesowa” niemieckiej Enigmy.
- ❖ W środowiskach sieci komunikacyjnych wygodnym rozwiązaniem jest *szyfrowanie z kluczem jawnym (public-key encryption)*:
 - Dwa różne klucze: *klucz jawny (public key)* – do szyfrowania oraz *klucz prywatny (private key)* – do deszyfrowania.
 - Odbiorca informacji ujawnia publicznie klucz do szyfrowania, natomiast klucz do deszyfrowania jest jego ścisłą tajemnicą.
 - Dowolny nadawca może zaszyfrować swoją informację przy użyciu jawnego klucza, ale tylko odbiorca może ją odczytać przy pomocy prywatnego klucza do deszyfrowania.
- *Czy da się skonstruować dobry algorytm realizujący ten schemat?*

Kryptosystem RSA

❖ Powszechnie stosowanym kryptosystemem z kluczem jawnym jest **metoda RSA** (Rivest, Shamir, Adleman: 1977) – oparta na **faktoryzacji** (rozkład na czynniki pierwsze) **liczb naturalnych**.

➤ Podstawy:

□ Niech:

$$N = p \cdot q, \quad \text{gdzie } p \text{ i } q \text{ – liczby pierwsze (duże)}$$

☞ Funkcja Eulera:

$\varphi(N)$ – ilość liczb mniejszych niż N względnie pierwszych z N :

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

□ Notacja:

$$a \equiv b \pmod{N} \quad \text{oznacza: } a \bmod N = b \bmod N.$$

□ Twierdzenie Eulera:

Jeżeli $a < N$ i a jest względnie pierwsze z N (nie mają wspólnych dzielników), to:

$$a^{\varphi(N)} \equiv 1 \pmod{N}$$

Kryptosystem RSA – c.d.

- ◆ Uczestnicy komunikacji: **Alice** i **Bob**
- 1. **Bob** bierze dwie duże liczby pierwsze p i q , i wylicza: $N = p \cdot q$ oraz $\varphi(N)$.
- 2. **Bob** wybiera losowo liczbę $e < \varphi(N)$, która jest względnie pierwsza z $\varphi(N)$, tzn. $\text{GCD}(e, \varphi(N)) = 1$ (GCD – *greatest common divisor*).
- 3. Następnie ogłasza publicznie (np. w CNN): e i N .
- 4. **Alice** zamienia swoją wiadomość na liczbę $a < N$ (dłuższe wiadomości można podzielić na odpowiednie fragmenty i każdy z nich szyfrować oddzielnie) i oblicza:
$$b = f(a) = a^e \pmod{N}$$
- 5. Tak zaszyfrowaną wiadomość **Alice** wysyła do **Boba**.
- 6. **Bob** wylicza: \rightarrow np. przy pomocy algorytmu Euklidesa dla $\text{GCD}(e, \varphi(N))$
$$d = e^{-1} \pmod{\varphi(N)} \quad \text{tzn.} \quad ed \equiv 1 \pmod{\varphi(N)}$$
- 7. Następnie **Bob** odszyfrowuje wiadomość **Alice** przy pomocy prywatnego klucza d :
$$f^{-1}(b) = b^d \pmod{N} = a^{ed} \pmod{N} = a \cdot [a^{\varphi(N)}]^m \pmod{N}, \quad (m - \text{liczba naturalna})$$

$$\equiv a \pmod{N} = a \quad (\text{bo } a < N).$$

tw. Eulera

Kryptosystem RSA – c.d.

- Czy kryptosystem RSA jest bezpieczny?
- ❖ Aby odszyfrować wiadomość trzeba wyliczyć wartość klucza deszyfrującego d , a do tego potrzebna jest znajomość czynników pierwszych p i q liczby N .
- ❖ **Rozkład dużych liczb naturalnych na czynniki pierwsze (faktoryzacja) jest klasycznie trudnym problemem!**
 - ✓ Najlepszy **klasyczny** algorytm, tzw. „sito ciała liczbowego” (J. Pollard, 1988) ma (sub)eksponencjalny czas faktoryzacji:

$$t_F \propto \exp[(64/9)^{1/3} (\ln N)^{1/3} (\ln \ln N)^{2/3}]$$

□ W roku 1994 użyto 1600 szybkich stacji roboczych do faktoryzacji liczby **RSA129**, tzn. znalezienia dwóch dużych czynników pierwszych liczby **129**-cyfrowej – obliczenia zajęły 8 miesięcy (5000 MIPS lat).

- ✓ Można oszacować, że dla liczby **400**-cyfrowej analogiczne obliczenia zajęłyby:

$$t_F \approx 10^{10} \text{ lat} \quad \approx \text{wiek wszechświata!}$$

➔ **Dla dostatecznie dużych N szyfr RSA jest praktycznie nie do złamania!**

Kryptosystem RSA – c.d.

Ale ...

- W roku 1994 **Peter Shor** wynalazł algorytm kwantowy, tzn. taki, który może być zrealizowany tylko na komputerze kwantowym, dla którego czas faktoryzacji liczby naturalnej N wynosi:

$$t_F \propto (\ln N)^3$$

❑ Np. do faktoryzacji liczby **RSA129** komputer kwantowy z zegarem 100 MHz potrzebowałby tylko **kilka sekund!**

✓ ... a dla liczby **400-cyfrowej** **niewiele ponad 1 minutę!**

Ze względu na wielomianową zależność czasu faktoryzacji od rozmiaru (liczby bitów) liczby naturalnej kryptosystem RSA nie jest bezpieczny względem algorytmu Shora!

✓ Kryptosystem RSA jest powszechnie używany w internecie (np. ssh).

Algorytm Shora – odsłona 1

- ◆ Szukamy nietrywialnych rozwiązań kongruencji Legendre'a:

$$x^2 \equiv 1 \pmod{N}$$

czyli: $(x+1)(x-1) \equiv 0 \pmod{N}$

dla $x \in [1, \dots, N-1]$; (rozwiązania trywialne: $x=1$ i $x=N-1$)

Jeżeli: $N = m \cdot n$ i $\text{GCD}(m, n) = 1$,

to na podstawie **chińskiego twierdzenia o resztach** dla kongruencji Legendre'a oprócz rozwiązań trywialnych, tzn. $x \equiv \pm 1 \pmod{N}$, istnieje także rozwiązanie nietrywialne, dla którego czynniki m i n rozdzielone są pomiędzy $(x-1)$ i $(x+1)$.

Zatem: $\text{GCD}(x-1, N)$ i $\text{GCD}(x+1, N) \Rightarrow m$ i n .

➤ Jak liczyć $\text{GCD}(a, N)$?

✓ Np. **algorytm Euklidesa**: $t \propto (\log N)^3$.

Algorytm Shora – odsłona 2

➤ Jak znaleźć rozwiązanie kongruencji Legendre'a?

◆ Rozwiązanie kongruencji Legendre'a ma związek z okresem funkcji periodycznej:

$$f_{N,y}(a) = y^a \pmod{N}, \quad \text{gdzie } y \in \{1, \dots, N-1\} \text{ i } \text{GCD}(y, N) = 1.$$

❖ Jeżeli r jest okresem powyższej funkcji, to zachodzi:

$$y^r \equiv 1 \pmod{N} \quad (\text{tzn. } r - \text{tzw. rząd } y \pmod{N}).$$

❖ Jeżeli r – parzyste, to: $y^{r/2} \equiv x \pmod{N}$

jest rozwiązaniem kongruencji Legendre'a: $x^2 \equiv 1 \pmod{N}$.

➤ Skąd bierzemy y ?

✓ Wybieramy losowo ze zbioru $\{1, \dots, N-1\}$.

✓ Można pokazać, że aby znaleźć nietrywialne rozwiązanie dla r parzystego potrzeba wykonać $< 2\log N$ losowań.

✓ Złożoność obliczeniowa $y^a \pmod{N}$: $\mathcal{O}((\log N)^3)$

Algorytm Shora – odsłona 3

➤ Jak znaleźć okres funkcji $f_{N,y}(a) = y^a \bmod N$?

❖ Klasycznie – trudny problem (brak efektywnego algorytmu):

$$t \propto \exp(\log N)$$

❖ Kwantowo – Shor pokazał, że okres funkcji można znaleźć wydajnie przy użyciu **kwantowej transformacji Fouriera**:

$$t_{\text{QFT}} \propto (\log N)^2$$

❖ Dyskretna transformacja Fouriera – klasycznie:

$$F : (x_0, \dots, x_{N-1}) \rightarrow (y_0, \dots, y_{N-1}), \quad x_i, y_i \in \mathbb{C} \text{ (liczby zespolone)}$$

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$$

➤ Złożoność obliczeniowa dyskretnej transformacji Fouriera:

▷ Naiwnie : $\propto [\exp(\ln N)]^2$

▷ Tzw. szybka transformata Fouriera (FFT) : $\propto \ln N \exp(\ln N)$

Kwantowa transformacja Fouriera

- ◆ Kwantowa transformacja Fouriera (QFT) – operator unitarny działający na wektory w N -wymiarowej przestrzeni Hilberta.

Niech $\{|0\rangle, \dots, |N-1\rangle\}$ - baza ortonormalna w $H^{\otimes N}$,

$$\hat{F}_q : |j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle.$$

Niech $N = 2^n$ oraz $|j\rangle = |j_1 j_2 \dots j_n\rangle$, $j_k \in \{0,1\}$, $k = 1, \dots, n$.

Ułamki binarne: $0.j_1 j_{l+1} \dots j_m \equiv \frac{j_l}{2} + \frac{j_{l+1}}{2^2} + \dots + \frac{j_m}{2^{m-l+1}}$.

- Kwantową transformację Fouriera dla wektorów bazowych można przedstawić w postaci:

$$\hat{F}_q |j_1 \dots j_n\rangle = 2^{-\frac{n}{2}} (|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle) \dots \\ \dots (|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle). \quad \leftarrow n \text{ iloczynów}$$

Kwantowa transformacja Fouriera

➤ Czy da się wydajnie zaimplementować kwantową transformację Fouriera?

❖ Tak, ale tylko na komputerze kwantowym!

➤ Co to jest komputer kwantowy?

❖ Komputer kwantowy jest to maszyna operująca na stanach kwantowych, tzw. **qubitach**, w oparciu o prawa mechaniki kwantowej.

▪ Qubit – wektor w 2-wymiarowej przestrzeni Hilberta:

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad |a|^2 + |b|^2 = 1,$$

gdzie $a, b \in \mathbb{C}$, a $\{|0\rangle, |1\rangle\}$ - baza ortonormalna w H^2

▪ Stan n -qubitowy – wektor w przestrzeni Hilberta o wymiarze 2^n :

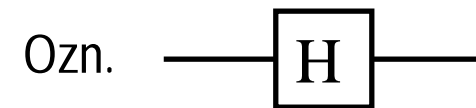
$$|\Psi\rangle = \sum_{k=0}^{2^n-1} c_k |k\rangle, \quad \sum_{k=0}^{2^n-1} |c_k|^2 = 1, \quad c_k \in \mathbb{C}.$$

Kwantowa transformacja Fouriera

- ◆ Obliczenia kwantowe – transformacje unitarne wektorów w przestrzeni Hilberta.
- ◆ Dowolny układ obliczeniowy można zbudować z kilku podstawowych elementów, tzw. bramek kwantowych.
- ◆ Bramki używane do kwantowej transformacji Fouriera:

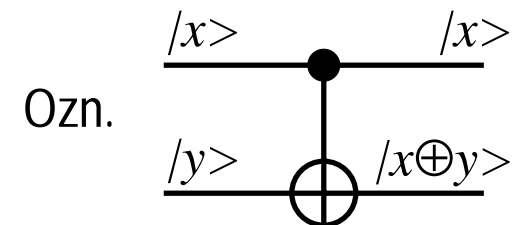
- ◆ Bramka **Hadamarda** – 1-qubitowa:

$$\hat{H}|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.x}|1\rangle), \quad x \in \{0,1\}.$$



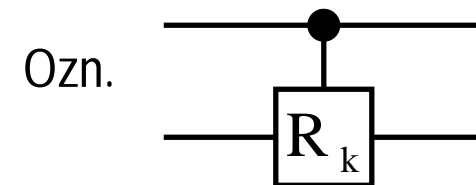
- ◆ Bramka **Control-NOT (CN)** – 2 -qubitowa:

$$\hat{U}_{\text{CN}}|x,y\rangle = |x, x \oplus y\rangle, \quad x, y \in \{0,1\}.$$

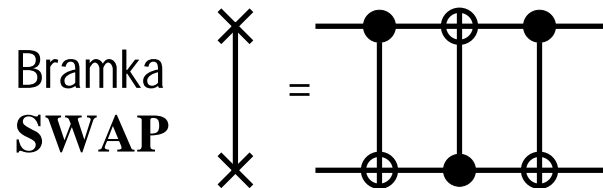
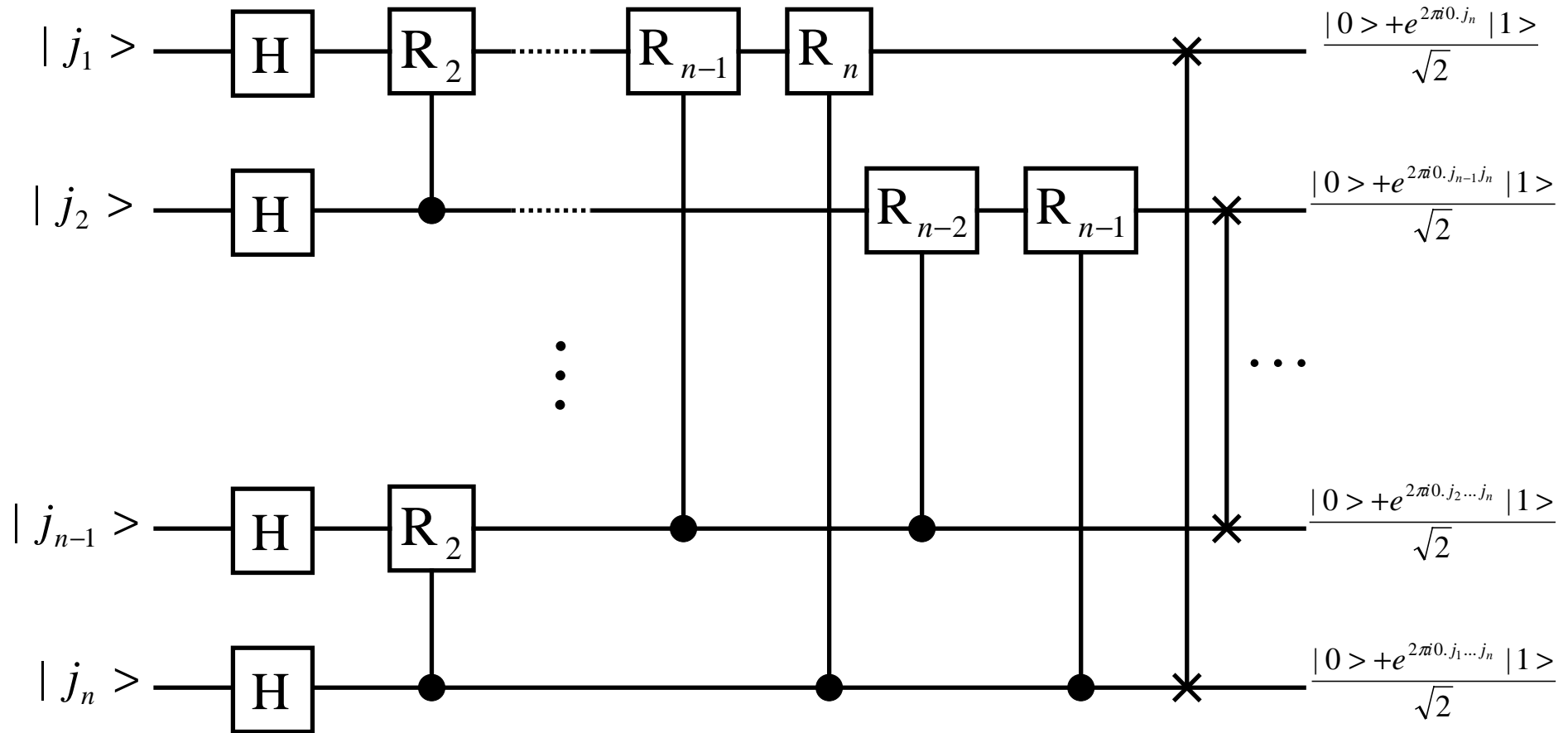


- ◆ Bramka **Control- R_k** :

$$\text{gdzie: } \hat{R}_k|x\rangle = e^{2\pi i x/2^k}|x\rangle, \quad x \in \{0,1\}.$$



Układ do kwantowej transformacji Fouriera



➤ Liczba bramek: $\frac{n(n+1)}{2} + \frac{n}{2} = \frac{n(n+2)}{2} = \mathcal{O}(n^2)$

Znajdowanie okresu funkcji

- ◆ Potrzeba dwóch rejestrów kwantowych:

- Dla argumentu: t -qubitowy, gdzie t powinno być takie, że $N^2 \leq 2^t < 2N^2$
- Dla wartości funkcji: n -qubitowy, gdzie $n = \lceil \log_2 N \rceil$

$$|\psi_0\rangle = \underbrace{|0\dots 0\rangle}_t \underbrace{|0\dots 0\rangle}_n$$

- ◆ Pierwszy rejestr wprowadzamy w stan superpozycyjny działając na każdy z t -qubitów bramką Hadamarda ($\hat{H}|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$):

$$|\psi_1\rangle = \frac{1}{\sqrt{K}} \sum_{j=0}^{K-1} |j\rangle |0\rangle, \quad K = 2^t.$$

- ◆ Na drugi rejestr działamy bramką: \hat{U}_f dla $f(j) = y^j \bmod N$

$$|\psi_2\rangle = \frac{1}{\sqrt{K}} \sum_{j=0}^{K-1} |j\rangle |y^j \bmod N\rangle$$

- ◆ Mierzmy drugi rejestr \rightarrow otrzymujemy pewną liczbę m

\Rightarrow pierwszy rejestr jest superpozycją stanów $|j\rangle$: $y^j \bmod N = m$

Znajdowanie okresu funkcji

- ◆ Niech r – okres funkcji $f(j)$ i założmy, że K jest podzielne przez r (dla uproszczenia analizy; algorytm działa również bez tego założenia, ale rachunki stają się bardziej skomplikowane):

- Liczba stanów, które „przeżyły pomiar” = $K/r \Rightarrow$ stan układu:

$$|\psi_3\rangle = \sqrt{\frac{r}{K}} \sum_{k=0}^{K/r-1} |kr+l\rangle |m\rangle, \quad \text{gdzie } l \in \{0, \dots, r-1\}.$$

- ◆ Jak znaleźć r ?

- Do 1-go rejestru stosujemy odwrotną transformację Fouriera \hat{F}_q^\dagger :

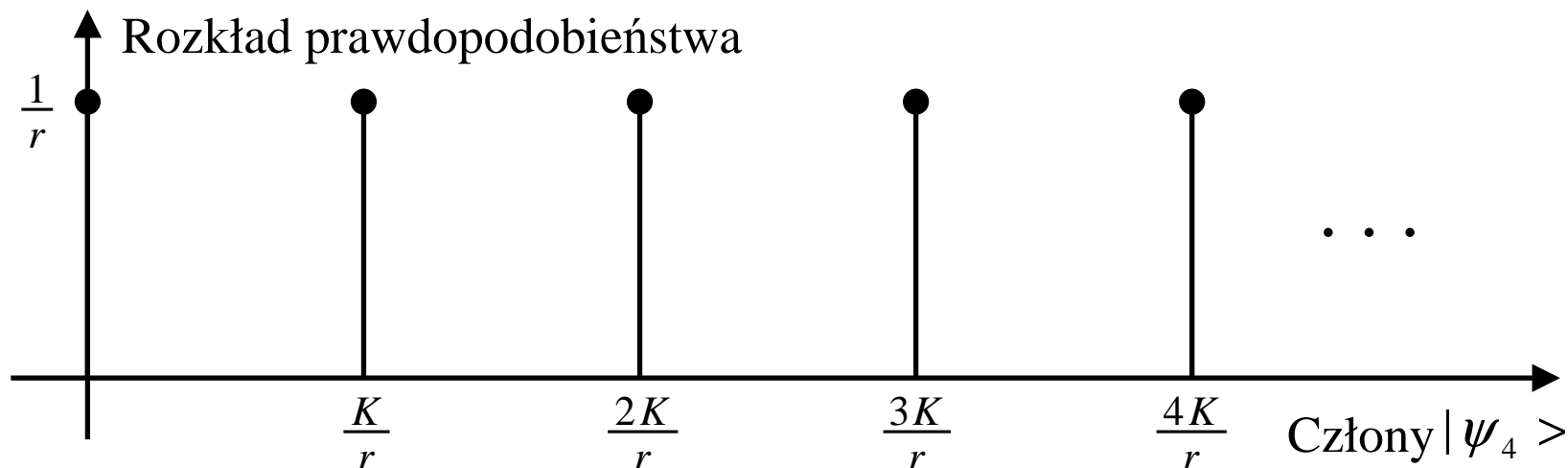
- W oparciu tożsamość:

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{2\pi ijk/N} = \begin{cases} 1 & \text{dla } k \bmod N = 0, \\ 0 & \text{dla } k \bmod N \neq 0, \end{cases}$$

- Dostajemy stan układu:

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{\lambda=0}^{r-1} e^{-2\pi i\lambda l/r} \left| \lambda \frac{K}{r} \right\rangle |m\rangle, \quad \text{gdzie } \lambda = 0, \dots, r-1.$$

Znajdowanie okresu funkcji



- ◆ Mierzmy pierwszy rejestr: $\Rightarrow L = \frac{\lambda K}{r}$.
 - Znamy L i K , nie znamy λ i r – jak wyznaczyć r ?
 - Skracamy maksymalnie ułamek: $\frac{L}{K} \Rightarrow \frac{\lambda}{r}$.
 - Jeżeli $\text{GCD}(\lambda, r) = 1$, to wyznaczyliśmy r .
 - Jeżeli $\text{GCD}(\lambda, r) > 1$, to znaleźliśmy jedynie dzielnik r , ozn. r_1 ,
 - Powtarzamy kwantową część algorytmu dla: $z = y^{r_1}$, etc.
aż znajdziemy: $r = r_1 r_2 \dots \rightarrow$ Liczba iteracji: $\leq \log_2 r$.

❖ Ostatecznie: **czas faktoryzacji dla algorytmu Shora:** $t_F \propto (\ln N)^3$

Podsumowanie

- ◆ Kwantowy algorytm Shora umożliwia faktoryzację liczb naturalnych w czasie wielomianowym, co stanowi zagrożenie dla kryptosystemu RSA!
- ◆ Shor podał również algorytm rozwiązujący problem dyskretnych logarytmów w czasie wielomianowym.
 - ◆ Mając daną liczbę pierwszą p oraz dwie liczby naturalne $g, y < p$ znaleźć najmniejszą liczbę naturalną x , taką że: $g^x \equiv y \pmod{p}$.
 - Również szeroko stosowany w kryptografii z kluczem jawnym.
- ◆ Roku 2001: grupa 6 eksperymentatorów z IBM i Uniwersytetu Stanford dokonała implementacji algorytmu Shora na komputerze kwantowym opartym o jądrowy rezonans magnetyczny.
 - Dokonano rozkładu na czynniki pierwsze liczby $15 = 3 \cdot 5$.
- Czy możliwa będzie kryptografia w dobie komputerów kwantowych?
- ❖ **Tak – mechanika kwantowa umożliwia bezpieczne przesyłanie klucza** (np. drogą teleportacji kwantowej).

Literatura

- ◆ P. Shor, „*Algorithms for Quantum Computation: Discrete Logarithm and Factoring*”, Proc. 35th Annual Symposium on Foundations of Computer Science (IEEE, Los Alamitos, CA, 1994), p. 124 and SIAM J. Comput. **26** (1997) 1484.
- ◆ A. Eckert, R. Jozsa, „*Quantum computation and Shor’s factoring algorithm*”, Rev. Mod. Phys. **68** (1996) 733.
- ◆ L.M.K. Vandersypen, M. Steffen, G. Breyta, C.S. Yannoni, M.H. Sherwood, I.L. Chuang, „*Experimental realization of Shor’s quantum factoring algorithm using nuclear magnetic resonance*”, Nature **414** (2001) 883.
- ◆ C. Lavor, L.R.U. Manssur, R. Portugal, „*Shor’s Algorithm for Factoring Large Integers*”, quant-ph/0303175.
- ◆ A.J.G. Hey, ed., „*Feynman and Computation*”, Perseus Books, 1999.
- ◆ M.A. Nielsen, I.L. Chuang, „*Quantum Computation and Quantum Information*”, Cambridge University Press, 2000.
- ◆ J. Preskill, „*Lecture Notes for Physics 229: Quantum Information and Computation*”, <http://www.theory.caltech.edu/~preskill/ph229>.
- ◆ D.E. Knuth, „*Sztuka programowania*”, tom 2, WNT, Warszawa, 2002.