Euklids Algorithmus

Euklids Algorithmus: erlaubt es, effizient den größten gemeinsamen Teiler $\mathrm{GGT}(a, b)$ zweier ganzer Zahlen a und b zu finden.

Beispiel: $\mathrm{GGT}(1430, 6825) = ?$

1. Falls nötig, ordne die Zahlen a,b so dass $a > b$, also $\mathrm{GGT}(1430, 6825) \rightarrow \mathrm{GGT}(6825, 1430)$.
2. Zerlege a gemäss $a = n\,b + r_1$:
$$6825 = 4 \cdot 1430 + 1105$$
3. Zerlege b gemäss $b = n\,r_1 + r_2$:
$$1430 = 1 \cdot 1105 + 325$$
4. Zerlege r_1 gemäss $r_1 = n\,r_2 + r_3$:
$$1105 = 3 \cdot 325 + 130$$
5. Zerlege r_2 gemäss $r_2 = n\,r_3 + r_4$:
$$325 = 2 \cdot 130 + 65$$
6. Zerlege r_3 gemäss $r_3 = n\,r_4 + r_5$:
$$130 = 2 \cdot 65 + 0$$
7. Wenn $r_j = 0$ (hier r_5), dann ist r_{j-1} (hier $r_4 = 65$) der gesuchte $\mathrm{GGT}(a, b)$ (hier $\mathrm{GGT}(1430, 6825)$).

RSA-Codierung

RSA-Verschlüsselungssystem: benutzt verschiedene Schlüssel für die die Ver- und Entschlüsselung (R. Rivest, A. Shamir und L. Adleman [1978]). Auf diese Weise kann der Codier-Schlüssel öffentlich ausgetauscht werden:

1. Suche zwei große ganze Primzahlen p und q und berechne $n = p\,q$.
2. Finde ein ganze Zahl d mit $\mathrm{GGT}(d, (p-1)\,(q-1)) = 1$.
3. Berechne e über $e\,d \bmod [(p-1)\,(q-1)] = 1$.
4. Die Zahlen e und n sind der öffentliche Schlüssel.
5. Der Versender stellt die Nachricht M als Sequenz ganzer Zahlen $\{M_i\}$ (jede im Intervall $[1, n]$) dar und
6. verschlüsselt sie gemäß $E_i = M_i^e \bmod n$.
7. Der Empfänger entschlüsselt die Nachricht mittels $M_i = E_i^d \bmod n$ und stellt die Nachricht M wieder her.

Wichtig: Ver- und Entschlüsselung sind effizient durchführbar (polynomialer Rechenaufwand), das Herausfinden von p und q dagegen (klassisch) ineffizient (exponentieller Rechenaufwand).

Code-Brechen: Aus 3. folgt direkt $d = [j\,(p-1)\,(q-1) + 1]/e$ mit $j = 1, 2, \ldots,$ bei bekannten p und q (und dem veröffentlichten e) ist der geheime Schlüssel d einfach ermittelbar.

Der Algorithmus von Shor

Einteilung von Rechenproblemen **in Effizienzklassen**:

- **einfach**: Rechenaufwand wächst in **polynomialer** Form (m^n) mit der Dimension m des Problems (n konstant).
- **schwierig**: Rechenaufwand wächst in **exponentieller** Form (n^m).

Die Faktorisierung ganzer Zahlen gilt als schwierig, da jeder bekannte klassische Algorithmus **exponentiell** mit der Zahl m der die Primzahl N darstellenden Bits wächst.
Grundlage der gebräuchlichsten Verschlüsselungssysteme, die daher ziemlich sicher sind.

Der Algorithmus von Shor beruht auf 3 Tricks:

1. Transformation des Problems auf das Auffinden der Periode einer Funktion.
2. Verwendung einer Fourier-Transformation um diese Periode zu finden.
3. Dem Ausnutzen der Quantenparallelität für 1. und 2 (daher polynomialer statt exponentieller Aufwand).

Trick 1: Transformation des Problems

Für eine ganze Zahl N und eine beliebige ganze Zahl $y \leq N$

(wobei $\mathrm{GGT}(y, N) = 1$ = größter gemeinsamer Teiler von y und N)

gilt mit großer Wahrscheinlichkeit, dass

$$\mathrm{GGT}(y^{r/2} + 1, N) \cdot \mathrm{GGT}(y^{r/2} - 1, N) \;=\; u \cdot v \;=\; N$$

und somit u, v die gesuchten Faktoren sind, wenn

r die Periode der Funktion $\quad f(a) = y^a \bmod N \quad$ ist.

Beispiel: $N = 15 \rightarrow y = 2, 4, 7, 8, 11, 13,$ oder 14

$$\begin{aligned}
11^0 \bmod 15 &= 1 \bmod 15 = (0 \cdot 15 + 1) \bmod 15 = 1 \\
11^1 \bmod 15 &= 11 \bmod 15 = (0 \cdot 15 + 11) \bmod 15 = 11 \\
11^2 \bmod 15 &= 121 \bmod 15 = (8 \cdot 15 + 1) \bmod 15 = 1 \\
11^3 \bmod 15 &= 1331 \bmod 15 = (88 \cdot 15 + 11) \bmod 15 = 11
\end{aligned}$$

$$\longrightarrow r = 2 \text{ (für } y = 11) \longrightarrow \mathrm{GGT}(11^{2/2} \pm 1, 15)$$

$$\longrightarrow \mathrm{GGT}(12, 15) \cdot \mathrm{GGT}(10, 15) = 3 \cdot 5 = 15$$

$$\begin{aligned}
7^0 \bmod 15 &= 1 \bmod 15 = (0 \cdot 15 + 1) \bmod 15 = 1 \\
7^1 \bmod 15 &= 7 \bmod 15 = (0 \cdot 15 + 7) \bmod 15 = 7 \\
7^2 \bmod 15 &= 49 \bmod 15 = (3 \cdot 15 + 4) \bmod 15 = 4 \\
7^3 \bmod 15 &= 343 \bmod 15 = (22 \cdot 15 + 13) \bmod 15 = 13 \\
7^4 \bmod 15 &= 2401 \bmod 15 = (160 \cdot 15 + 1) \bmod 15 = 1 \\
7^5 \bmod 15 &= 16807 \bmod 15 = (1120 \cdot 15 + 7) \bmod 15 = 7
\end{aligned}$$

$$\longrightarrow r = 4 \text{ (für } y = 7) \longrightarrow \mathrm{GGT}(7^{4/2} \pm 1, 15)$$

$$\longrightarrow \mathrm{GGT}(50, 15) \cdot \mathrm{GGT}(48, 15) = 5 \cdot 3 = 15$$

(Die Wahl $y = 14$ wäre aber eine Niete !)

Problem: Der Aufwand, die Periode r zu finden, wächst exponentiell mit der Zahl m der N darstellenden Bits!

Trick 2: Quanten-Fourier-Transformation

Die **Quanten-Fourier-Transformation (QFT)**

$$\sum_{j=0}^{K-1} x_j \,|\, j \,\rangle \xrightarrow{\text{QFT}} \sum_{k=0}^{K-1} y_k \,|\, k \,\rangle \,, \; y_k \;=\; \sum_{j=0}^{K-1} x_j \, \mathrm{e}^{2\pi i \frac{jk}{K}}$$

ist völlig analog zur klassischen diskreten Fourier-Transformation (DFT), außer daß bei der **QFT** die **Amplituden** transformiert werden.

Eigenschaften der (Q)FT:

1. Aus einer Periode r der $x_{0,1,\ldots,K-1}$ wird eine Periode K/r in den $y_{0,1,\ldots,K-1}$.

2. Konstante Verschiebung wird zu Phasenfaktor

$$\sum_{j=0}^{K-1} x_j \,|\, j+l \,\rangle \xrightarrow{\text{QFT}} \sum_{k=0}^{K-1} \mathrm{e}^{2\pi i \frac{kl}{K}} \, y_k \,|\, k \,\rangle \,,$$

   aber die (meßbaren) Wahrscheinlichkeiten bleiben gleich ($|\mathrm{e}^{2\pi i \frac{kl}{K}} \, y_k|^2 = |y_k|^2$).

Beispiel: Shor für die Zahl 15 (I)

Der Algorithmus von Shor für $N = 15$:

Register 1: $k = 3$ Qubits für die Zahlen 0 bis 7 ($\leq N/2$)
(Im Nichtidealfall sind mehr Qubits nötig!)

Register 2: $m = 4$ Qubits für die Zahlen 0 bis 15 ($\leq N$)

Teste (klassisch), ob N durch 2 teilbar oder ob $N = a^b$ ist, also ob 2 oder a (triviale) Teiler von N sind.

Wähle eine Zahl $y \leq 15$ (mit $\mathrm{GGT}(y, 15) = 1$):

z. B. $y = 11$.

1. Initialisierung: Setze alle 7 Qubits auf $|\,0\,\rangle$:

$|\,0000000\,\rangle$ (= $|\,\Psi_1\,\rangle_1 |\,\Phi_1\,\rangle_2$).

2. Präpariere Input: Bringe das 1. Register in eine Überlagerung von $|\,0\,\rangle$ und $|\,1\,\rangle$, also der Zahlen 0 bis 7:

$$|\,0000000\,\rangle \;\rightarrow\; \frac{1}{\sqrt{8}} \Big(\underbrace{|\,000\,\rangle}_{|\,0\,\rangle} + \underbrace{|\,001\,\rangle}_{|\,1\,\rangle} + \underbrace{|\,010\,\rangle}_{|\,2\,\rangle} + \cdots + \underbrace{|\,111\,\rangle}_{|\,7\,\rangle} \Big) |\,0000\,\rangle$$

$$|\,\Psi_1\,\rangle_1 |\,\Phi_1\,\rangle_2 \;\rightarrow\; \frac{1}{\sqrt{2^k}} \sum_{a=0}^{2^k-1} |\,a\,\rangle_1 |\,0\,\rangle_2$$

Beispiel: Shor für die Zahl 15 (II)

3. Berechne $f(a) = y^a \bmod N$ (also $11^a \bmod 15$) für alle a im 1. Register ($0 \ldots 7$) gleichzeitig (Quantenparallelität). Speichere das Ergebnis im 2. Register:

$$\frac{1}{\sqrt{8}}\Big(\underbrace{|000\rangle}_{|0\rangle}\underbrace{|0001\rangle}_{|1\rangle} + \underbrace{|001\rangle}_{|1\rangle}\underbrace{|1011\rangle}_{|11\rangle} + \underbrace{|010\rangle}_{|2\rangle}\underbrace{|0001\rangle}_{|1\rangle} + \ldots + \underbrace{|111\rangle}_{|7\rangle}\underbrace{|1011\rangle}_{|11\rangle}\Big)$$

$$= \frac{1}{\sqrt{8}}\Big(\Big[\underbrace{|000\rangle}_{|0\rangle} + \underbrace{|010\rangle}_{|2\rangle} + \underbrace{|100\rangle}_{|4\rangle} + \underbrace{|110\rangle}_{|6\rangle}\Big]\underbrace{|0001\rangle}_{|1\rangle} + \Big[\underbrace{|001\rangle}_{|1\rangle} + \underbrace{|011\rangle}_{|3\rangle} + \underbrace{|101\rangle}_{|5\rangle} + \underbrace{|111\rangle}_{|7\rangle}\Big]\underbrace{|1011\rangle}_{|11\rangle}\Big)$$

$$|\Psi_3\rangle_1 |\Phi_3\rangle_2 = \frac{1}{\sqrt{2^k}} \sum_{a=0}^{2^k-1} |a\rangle_1 |y^a \bmod N\rangle_2$$

$$\overset{A<\left[\frac{2^k-1}{r}\right]}{=} \sum_{l=0}^{r-1} \Big[\frac{1}{\sqrt{r(A+1)}} \sum_{j=0}^{A} |jr+l\rangle_1\Big] |y^l \bmod N\rangle_2$$

Register 1 enthält jetzt die Periode r, aber nur für jeweils gleiche Meßwerte in Register 2!

Nur die häufige Wiederholung des Experiments ergäbe die Periode r.

Zahl der im Mittel notwendigen Wiederholungen wächst **exponentiell** mit der Zahl $= m$ der Qubits von N !

Beispiel: Shor für die Zahl 15 (III)

4. Anwendung der QFT auf **Register 1:**

$$\longrightarrow \frac{1}{2}\Big(\Big[\underbrace{|000\rangle}_{|0\rangle}+\underbrace{|100\rangle}_{|4\rangle}\Big]\underbrace{|0001\rangle}_{|1\rangle} + \Big[\underbrace{|000\rangle}_{|0\rangle}+\underbrace{e^{i\pi}|100\rangle}_{-|4\rangle}\Big]\underbrace{|1011\rangle}_{|11\rangle}\Big)$$

$$|\Psi_4\rangle_1|\Phi_4\rangle_2 = \frac{1}{r}\sum_{l=0}^{r-1}\Big[\sum_{j=0}^{r-1} e^{2\pi i\frac{lj}{r}}|j\,\frac{2^k}{r}\rangle_1\Big]\,|y^l \bmod N\rangle_2$$

Unabhängig von Register 2 liefert jetzt jede Messung entweder 0 oder ein Vielfaches der Periode $2^k/r$:

- **0:** Versuch ist erfolglos.
- **4:** $r = 2^3/4 = 8/4 = 2$ (Erfolg!).

(Im allg. Fall muß r aus $j \cdot 2^k/r$ über Kettenbruchzerlegung gewonnen werden.)

Obwohl sich in diesem Beispiel die Periode vergrößert, ist die neue Periode $2^k/r$ ein Erfolgsgarant, da zwar 2^k und r exponentiell mit N wachsen, der Bruch aber nur polynomial !

Zahl der Rechenoperationen inkl. Wahrscheinlichkeit für Fehlversuche wächst nur polynomial mit der Zahl m der Qubits, die N darstellen !!!

Shor als Phasenabschätzung

Der Algorithmus von Shor kann auch als eine spezielle Anwendung des allgemeineren Algorithmus der Phasenabschätzung interpretiert werden.

Der Operator, dessen Eigenwerte (genauer die Phase der Eigenwerte) abgeschätzt werden soll, lautet

$$\hat{\mathrm{U}}_x \,|\, y \,\rangle \;=\; |\, xy \,(\mathrm{mod}\, N) \,\rangle$$

mit $y \in \{0,1\}^m$, wobei die Konvention $xy\,(\mathrm{mod} N) \equiv y$ für $N \leq y \leq r-1$ (mit $r = 2^m$) Verwendung findet.

$$|\, u_s \,\rangle \;=\; \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i\, sk/r} \,|\, x^k \,\mathrm{mod}\, N \,\rangle$$

für ganze Zahlen $0 \leq s \leq r-1$ sind Eigenvektoren des Operators $\hat{\mathrm{U}}_x$.

Anstelle der Präparation der einzelnen Eigenvektoren $|\, u_s \,\rangle$ nutzt man die Beziehung

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\, u_s \,\rangle \;=\; |\, 1 \,\rangle$$

aus. $|\, 1 \,\rangle \equiv |\, 000\ldots01 \,\rangle$ ist (in Kontrast zu $|\, u_s \,\rangle$) leicht (und effizient) implementierbar.