Angriffe auf existierende Algorithmen durch Quantencomputer I

Seminar IT-Sicherheit

Eduard Zeller
(ite102658)

Sommersemester 2017
28. August 2017

Betreuer: Prof. Dr. Gerd Beuster

fhwedel
UNIVERSITY OF APPLIED SCIENCES

If you think you understand quantum mechanics, you don't understand quantum mechanics.

Richard Feynman

Diese Ausarbeitung ist im Rahmen des Seminars *IT-Sicherheit: Quanten-Computer und (Post-)Quanten-Kryptographie* bei Prof. Dr. Gerd Beuster im Sommersemester 2017 entstanden. Es behandelt das Thema Angriffe auf existierende Algorithmen durch Quanten-Computer mit Schwerpunkt auf asynchrone Verschlüsselungsalgorithmen. Dabei wird der *Shor-Algorithmus*, den Peter W. Shor 1994 veröffentlicht hat, vorgestellt. Zusätzlich wird der RSA-Verschlüsselungsalgorithmus betrachtet, der die „Beute“ darstellt.

In dem Seminar sollten sich die Studenten der *Fachhochschule Wedel* mit Quanten-Kryptografie und Angriffen auf traditionelle kryptographische Verfahren mittels Quanten-Computern beschäftigen.

Inhaltsverzeichnis

Abbildungsverzeichnis **v**

1 Einleitung **1**

2 RSA-Kryptographie **2**
2.1 Verfahren 2
2.2 Beispiel 3
2.3 Beweis 4

3 Suche nach Perioden **6**
3.1 Verfahren 6
3.2 Beispiel 7

4 Shor-Algorithmus **9**
4.1 Hadamard-Transformation 11
4.2 Orakel 11
4.3 Quanten-Fouriertransformation 12
4.3.1 Fouriertransformation 12
4.3.2 Quanten-Fouriertransformation 13
4.4 Schaltkreis 13
4.5 Laufzeit 14

5 Umsetzung des Shor-Algorithmus **16**

6 Ausblick **17**

Literaturverzeichnis **18**

Abbildungsverzeichnis

3.1 Periode der Sinusfunktion 6
3.2 Der Funktionsgraf von a^x mod n mit den Werten $n = 15$ und $a = 7$ 7

4.1 Shor-Algorithmus Flussdiagramm 10
4.2 Fouriertransformation 12
4.3 Schaltkreis des *Shor-Algorithmus* 14
4.4 Laufzeit der Faktorisierung: rot = Zahlkörpersieb, blau = *Shor-Algorithmus* 15

5.1 Molekül mit sieben Qubits 16

1 Einleitung

Für die Verschlüsselung von Nachrichten für den geheimen Nachrichtenaustausch existiert eine Vielzahl an Verschlüsselungsalgorithmen. Diese können in zwei Kategorien unterteilt werden:

Synchrone Verschlüsselung

Bob und Alice teilen einen gemeinsamen Schlüssel, der sowohl zum Ver- als auch zum Entschlüsseln verwendet wird. Das Problem hierbei ist, dass ein Schlüsselaustausch erfolgen muss, bei dem Eve den Schlüssel abfangen kann.

Asynchrone Verschlüsselung

Alice verschlüsselt ihre Nachrichten mit einem *public key*, den sie von Bob erhalten hat. Bob entschlüsselt die Nachricht mit dem entsprechenden *private key*, den nur er besitzt. Eve hat zwar auch Zugriff auf den *public key*, kann aber die Nachricht nur mit dem *private key* entschlüsseln.

Allgemein basieren Verschlüsselungsalgorithmen darauf, dass die Berechnung von Schlüsseln in relativ kurzer Zeit möglich ist, ein Angriff jedoch unverhältnismäßig viel Zeit benötigt. Im Falle von asynchronen Verschlüsselungsalgorithmen kann der benötigte *private key* aus den Informationen des *public key* berechnet werden. Beispielsweise ist für RSA eine effiziente Faktorisierung von sehr großen Zahlen[1] notwendig. *Diffie-Hellman* und *Elliptische-Kurven-Kryptografie* können geknackt werden, sobald effiziente Algorithmen zum Berechnen des diskreten Algorithmus gefunden wurden. Die besten bekannten Algorithmen für klassische Computer besitzen jedoch (sub-)exponentielle Laufzeiten[5].
Im Jahr 1994 veröffentlichte Peter W. Shor einen Algorithmus, der Mittel der Quanteninformatik benutzt, um Zahlen in polynomieller Zeit zu faktorisieren[9]. Obwohl in der selben Veröffentlichung ein zweiter Algorithmus beschrieben wurde, der – ebenfalls mit Hilfe der Quanteninformatik – den diskreten Algorithmus berechnet, ist der zuerst genannte als *Shor-Algorithmus* bekannt.

[1] heutzutage werden Schlüssel der Länge 2048 bis 15360 Bit verwendet

2 RSA-Kryptographie

Im Jahr 1978 veröffentlichten Ronald R. Rivest, Adi Shamir und Leonard Adleman ein Verfahren, das zum digitalen Signieren und zum Verschlüsseln verwendet werden kann[8]. Das Verfahren war das erste *Public-Key* Verschlüsselungsverfahren und ist noch heute das wichtigste. Seine Sicherheit hängt eng an der Schwierigkeit zusammen, große Zahlen in ihre Primfaktoren zu zerlegen[1].
Das RSA-Verfahren hat zwei große Vorteile:

1. Es muss niemals ein geheimer Schlüssel ausgetauscht werden.

2. Alle Absender können den gleichen Schlüssel verwenden und der Empfänger benötigt nur einen, um alle empfangenen Nachrichten zu entschlüsseln.

Im Folgenden wird das Verfahren beschrieben und abschließend wird ein kleines Beispiel vorgestellt, das aus der Ursprungsarbeit stammt.

2.1 Verfahren

A. Schlüsselerzeugung

Bob wählt nacheinander zufällig zwei Primzahlen p und q. Diese sollten eine Länge von 2048 bis 15360 Bit haben und ungefähr gleich groß gewählt werden. Anschließend muss das Produkt n sowie dessen Eulerfunktionswert $\Phi(n)$ berechnet werden:

$$n = p \cdot q$$
$$\Phi(n) = (p-1)(q-1)$$

Zusätzlich wählt Bob eine natürliche Zahl e, die teilerfremd zu $\Phi(n)$ ist. Die Verschlüsselung wird effizienter, wenn e klein gewählt wird. Ein beliebter Kandidat ist 65537. Zum Schluss muss noch d berechnet werden:

$$e \cdot d \equiv 1 \mod (p-1)(q-1)$$

Die natürliche Zahl kann mit Hilfe des erweiterten euklidischen Algorithmus berechnet werden. Der öffentliche Schlüssel besteht aus dem Paar (n, e), der private Schlüssel ist d.

Die Zahl n heißt RSA-Modul, e heißt Verschlüsselungsexponent und d heißt Entschlüsselungsexponent.

B. Verschlüsselung

Alice verschlüsselt ihren Klartext m zu

$$c = m^e \mod n$$

Dabei ist zu beachten, dass $0 \leq m < n$ gilt. Ist dies nicht der Fall muss m in Blöcke von Ziffern $m_1, m_2, ..., m_k$ zerlegt werden, so dass jeder der k Zahlen kleiner n ist. Jeder der Blöcke wird einzeln verschlüsselt, um die Chiffre $c = c_1, c_2, ..., c_k$ zu erhalten.

C. Entschlüsselung

Bob entschlüsselt die Chiffre und erhält den Klartext

$$m = c^d \mod n.$$

2.2 Beispiel

Das folgende Beispiel stammt aus der Ursprungsarbeit von Rivest, Shamir und Adleman[8].

A Bob wählt die Primzahlen $p = 47, \quad q = 49$ und berechnet

$$n = p \cdot q = 2773 \quad \text{und} \quad \Phi(n) = (p-1)(q-1) = 2668.$$

Er wählt $e = 17$ und zeigt mit Euklids Algorithmus, dass e und $\Phi(n)$ teilerfremd sind: $\mathrm{ggT}(2668, 17) = 1$. Mit der erweiterten Variante von Euklids Algorithmus berechnet er, dass

$$d \cdot 17 \equiv 1 \mod 2668$$

die Lösung $d = 157$ hat. Bob veröffentlicht $e = 17$ und $n = 2773$.

B Alice möchte Bob die Nachricht ITS ALL GREEK TO ME schicken und verwendet den öffentlichen Schlüssel $e = 17, n = 2773$. Sie wählt die Kodierung

$$\text{Leerzeichen} = 00,\ \mathsf{A} = 01,\ \mathsf{B} = 02,\ ...,\ \mathsf{Z} = 26.$$

Damit erhält sie

$$m = 0920\ 1900\ 0112\ 1200\ 0718\ 0505\ 1100\ 2015\ 0013\ 0500.$$

Die Nachricht m wird in 10 Blöcke der Länge 4 aufgeteilt, damit jede Zahl kleiner $n = 2773$ entspricht. Die Blöcke werden einzeln verschlüsselt. $m_1 = 920$ wird

beispielsweise zu

$$c_1 = 0920^{17} = 0948 \mod 2773$$

verschlüsselt. Damit hat die Chiffre den Wert

$$c = 0948\ 2342\ 1084\ 1444\ 2663\ 2390\ 0778\ 0774\ 0219\ 1655$$

C Bob empfängt c und wendet den geheimen Schlüssel $d = 157$ an. Die ersten beiden Buchstaben werden folgendermaßen entschlüsselt:

$$0948^{157} \mod 2773 = 0920.$$

2.3 Beweis

In diesem Abschnitt wird die korrekte Entschlüsselung bewiesen. Für den Beweis wird der kleine Satz von Fermat benötigt, der durch

$$a^{p-1} \equiv 1 \mod p$$

beschrieben wird. Dieser Satz gilt für jede natürliche Zahl $a \neq 0$ und eine Primzahl p, die a nicht teilt. Die Formel zur Berechnung von d

$$ed \equiv 1 \mod (p-1)(q-1)$$

kann auch durch

$$ed = 1 + k(p-1)(q-1)$$

für eine ganze Zahl k ausgedrückt werden. Ausgangszustand für den Beweis, ist die Tatsache, dass die entschlüsselte Chiffre den gleichen Wert hat, wie der entsprechende Klartext, der ver- und entschlüsselt wird:

$$\begin{aligned} c^d &= m^{ed} \\ &= m^{1+k(p-1)(q-1)} \\ &= m \cdot \left(m^{p-1}\right)^{k(q-1)} \\ &\equiv m \cdot 1^{k(q-1)} \mod p \\ &\equiv m \mod p \end{aligned}$$

Im vorletzten Schritt wurde für den Fall von $m \neq 0 \mod p$ der kleine Satz von Fermat angewendet. Falls $m = 0 \mod p$ ist die Aussage ohnehin klar. Mit der gleichen Rechnung folgt

$$c^d \equiv m \mod q.$$

An dieser Stelle wird eine aus der Zahlentheorie bekannte Tatsache angewandt, die aus dem *chinesischen Restsatz* folgt:

Sind k und l teilerfremd, so folgt für alle ganzen Zahlen x und a:

$$x \equiv a \mod k \quad \text{und} \quad x \equiv a \mod l$$

genau dann, wenn

$$x \equiv a \mod k \cdot l.$$

Daraus folgt

$$c^d \equiv m \mod n.$$

Damit wurde gezeigt, dass die Entschlüsselung der Form $c^d \mod n$ die Nachricht m als Ergebnis hat.

3 Suche nach Perioden

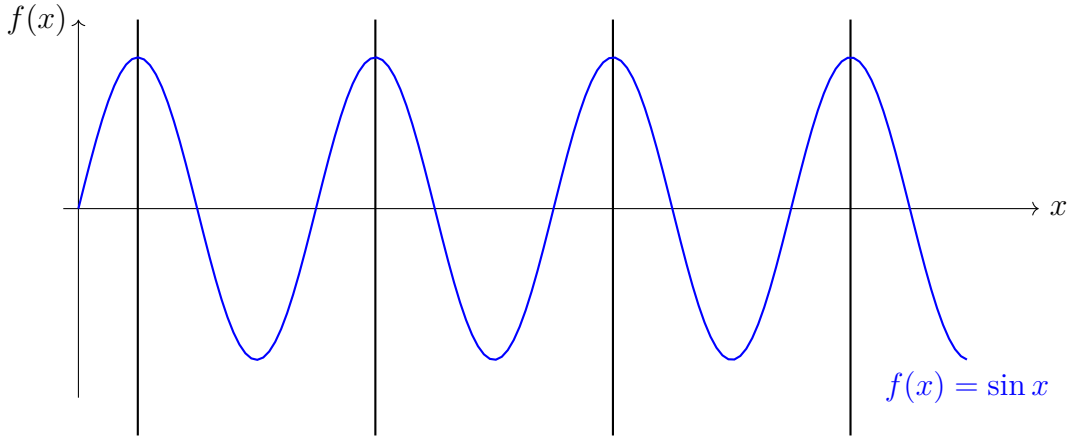


Abbildung 3.1: Periode der Sinusfunktion

Spätestens seit dem Miller-Rabin-Test, der überprüft ob eine Zahl prim ist und beim RSA-Kryptosystem zum Einsatz kommt, ist bekannt, dass die Bestimmung der Periode die Faktorisierung von Zahlen ermöglicht[6]. Die Abbildung 3.1 zeigt zur Veranschaulichung die Sinusfunktion mit der Periode $r = 2\pi$.

3.1 Verfahren

Um Zahlen zu faktorisieren, muss die Periode r der Funktion

$$f(x) = a^x \mod n$$

bestimmt werden. a ist dabei eine beliebige natürliche Zahl kleiner n.
Aus $f(x + r) = f(x)$ folgt $f(r) = f(0)$. Für die Periode r gilt also

$$a^r \equiv 1 \mod n$$
$$a^r - 1 \equiv 0 \mod n$$

Für die natürliche Zahl k gilt

$$a^r - 1 = k \cdot n$$
$$(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = k \cdot n.$$

Da $n = p \cdot q$ ist, können die beiden Faktoren von n bestimmt werden. Dabei ist

$$p = \mathrm{ggT}(a^{\frac{r}{2}} - 1, n) \quad \text{und} \quad q = \mathrm{ggT}(a^{\frac{r}{2}} + 1, n).$$

Es kann jedoch vorkommen, dass die berechnete Periode r ungerade ist. In diesem Fall muss die Periode einer anderen natürlichen Zahl a berechnet werden. Zudem kann es vorkommen, dass $a^{\frac{r}{2}} - 1$ teilerfremd mit n und $a^{\frac{r}{2}} + 1$ ein Vielfaches von n ist. Auch hier muss von vorne begonnen werden. Aus diesem Grund kann die Suche nach einem gültigen a und damit die Faktorisierung recht viel Zeit in Anspruch nehmen. Eine Tatsache der Zahlentheorie kommt dem Problem entgegen:
Ist n ungerade und keine Primpotenz, so gilt für mindestens die Hälfte der Zahlen a in $\{0, ..., n-1\}$, die mit n teilerfremd sind[7]:

- die Periode r der Funktion $a^x \mod n$ ist gerade, und
- n teilt $a^{\frac{r}{2}} + 1$ nicht (anders ausgedrückt: $a^{\frac{r}{2}} \not\equiv -1 \mod n$).

Damit wird deutlich, dass nicht jede Zahl n mit diesem Verfahren faktorisiert werden kann. Das ist jedoch nicht notwendig: gerade Zahlen können einfach halbiert werden. Für einen Rechner ist dieser Schritt besonders trivial, da er im Binärsystem rechnet. Das heißt, dass er alle Bits einer geraden Zahl um eine Stelle nach rechts schieben muss, um durch zwei zu teilen. Auch Primpotenzen können einfach zerlegt werden. Dazu existieren Algorithmen, die solche Zahlen in $\mathcal{O}(\log(n))$ zerlegen[4]. Darüber hinaus werden solche Zahlen im RSA-Kryptosystem nie vorkommen, da sonst die notwendige Sicherheit nicht gegeben ist.

3.2 Beispiel

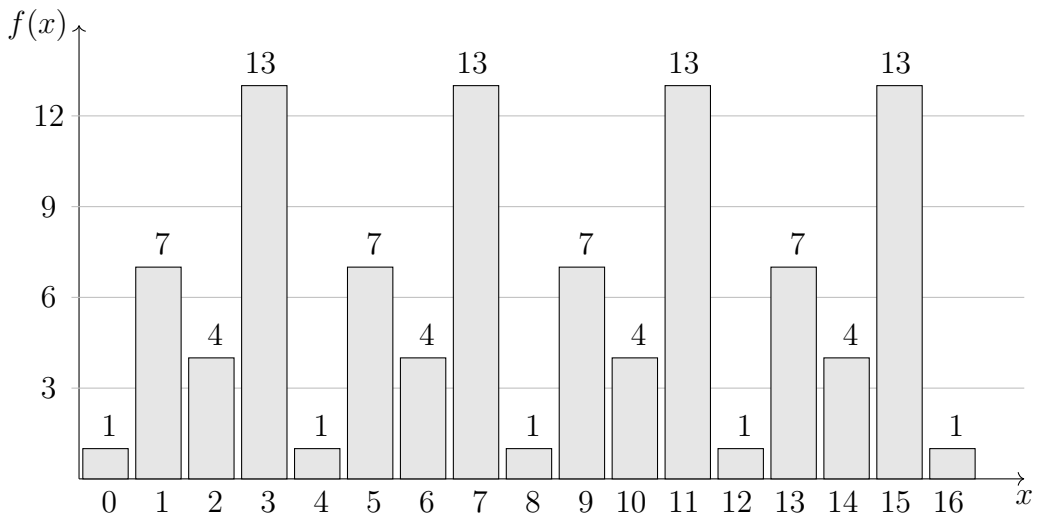


Abbildung 3.2: Der Funktionsgraf von $a^x \mod n$ mit den Werten $n = 15$ und $a = 7$

In diesem Abschnitt wird die Zahl $n = 15$ faktorisiert, um das eben vorgestellte Verfahren zu vertiefen. $n = 15$ ist die kleinste Zahl, die sich mit dieser Methode zerlegen lässt. Alle Zahlen davor sind entweder gerade, Primfaktoren oder Primzahlen. Im ersten Schritt wird eine natürliche Zahl a aus $\{2, ..., 14\}$ ausgewählt, die mit n teilerfremd ist. Dazu wird der euklidische Algorithmus $\mathrm{ggT}(a, 15)$ berechnet. Für $a = 3$ ist das Ergebnis $\mathrm{ggT}(3, 15) = 3$. In diesem Fall wurde direkt ein Teiler gefunden. Das gleiche gilt für $a = 5$.
Alle anderen möglichen Werte für a sind mit 15 teilerfremd. Für $a = 7$ muss die Periode der Funktion $7^x \mod 15$ bestimmt werden. Dazu werden solange alle Potenzen berechnet, bis die Periode erkannt wird. In Abbildung 3.2 werden grafisch die ersten 17 Ergebnisse dargestellt. Die Periode ist $r = 4$. Da r gerade ist, können alle weiteren Berechnung ausgeführt werden, die zum Ergebnis führen.

$$
\begin{aligned}
&\rightarrow a^{\frac{p}{2}} && = 7^{\frac{4}{2}} && = 49 \\
&\rightarrow ggT(n, a^{\frac{p}{2}} + 1) && = ggT(15, 50) && = 5 \\
&\rightarrow ggT(n, a^{\frac{p}{2}} - 1) && = ggT(15, 49) && = 3
\end{aligned}
$$

$$\Rightarrow 15 = 5 \cdot 3 = 3 \cdot 5$$

4 Shor-Algorithmus

Das in Kapitel 3 vorgestellte Verfahren ist als *Shor-Algorithmus* bekannt. Die Abbildung 4.1 fasst alle notwendigen Schritte noch einmal zusammen. Dabei ist die Suche nach der Periode hervorgehoben, da noch nicht hinreichend geklärt wurde, wie diese bestimmt werden kann. In Wirklichkeit ist kein Algorithmus für bekannt, der in sinnvoller Zeit die Periode bestimmen kann; zumindest nicht für klassische Computer.
In seiner Ausarbeitung untersucht Peter W. Shor die Quanteninformatik, insbesondere die Quanten-Fouriertransformation[10]. Diese verwendet er, um auf Quantencomputern die Periode effizient zu berechnen, damit auch sehr große Zahlen in relativ kurzer Zeit faktorisiert werden können, wodurch letzten Endes das RSA-Kryptosystem geknackt werden kann.
Der *Shor-Algorithmus* kann zum größten Teil auf einem klassischen Rechner ausgeführt werden, nur für die Bestimmung der Periode wird ein Quantencomputer, der mindestens $\log_2 n$ Qubit hat, benötigt. Die Schritte, die auf dem Quantencomputer ausgeführt werden müssen, können zu folgenden Schritten zusammengefasst werden:

1. Initialisierung der Register
   $R = |a\rangle\,|b\rangle \leftarrow |0...0\rangle\,|0...0\rangle$

2. Hadamard-Transformation auf $|a\rangle$ anwenden

3. Orakel anwenden

4. Quantenfouriertransformation auf $|a\rangle$ anwenden
   $|a\rangle \leftarrow \mathrm{QFT}_n\,|a\rangle$

5. Register $|a\rangle$ auslesen und Inhalt ausgeben

Die Initialisierung ist ein grundlegender Schritt, bei dem die Quantenbit (kurz: Qubit) vorbereitet werden. Die Hadamard-Transformation versetzt die Qubit in die Superposition, damit die notwendigen Quantenmechanischen Effekte ausgenutzt werden können. Dieser Schritt wird weiter im Abschnitt 4.1 behandelt. Das Orakel veranlasst den Quantencomputer, die Funktion $a^r \mod n$ unter Ausnutzung der Superposition für alle möglichen r gleichzeitig zu berechnen. Dieser Teil wird im Kapitel 4.2 beschrieben. Die Quanten-Fouriertransformation wertet die Werte aus und bestimmt die Periode. Dabei wird die

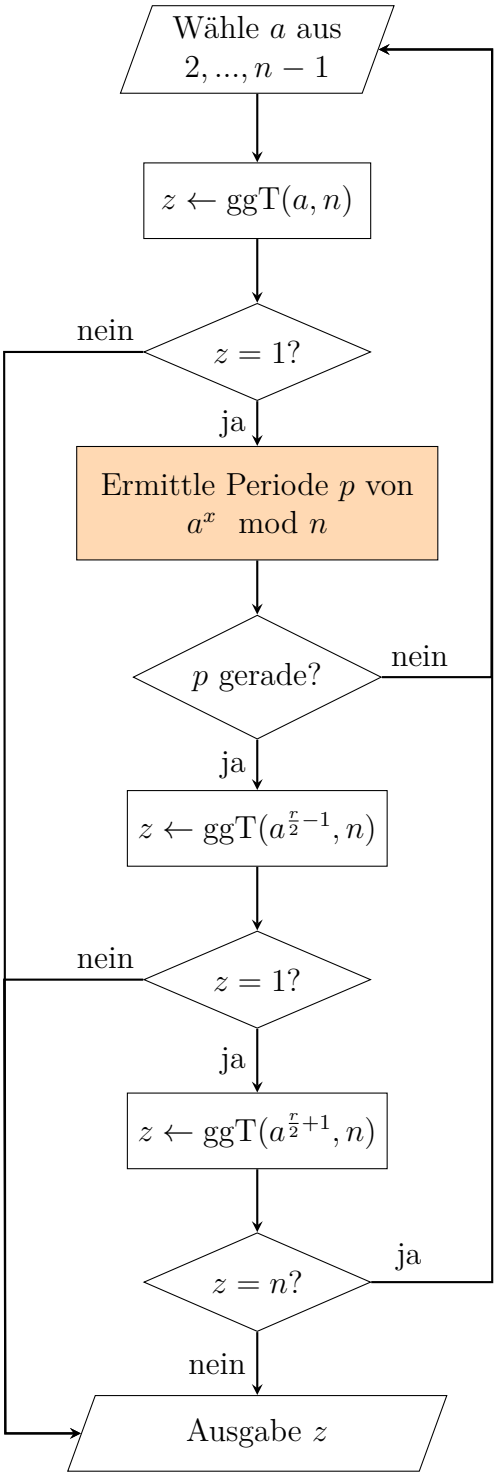


Abbildung 4.1: Shor-Algorithmus Flussdiagramm

Wahrscheinlichkeit des korrekten Ergebnisses angehoben, während die Wahrscheinlichkeiten aller falschen Ergebnisse gesenkt werden. Dies dient dazu, um beim Messvorgang das richtige Ergebnis zu erwischen. Dieser Teil wird im Abschnitt 4.3 beleuchtet. Dazu gibt es eine kleine Einführung in die Fouriertransformation, damit die Vorgänge der Quanten-Fouriertransformation verständlicher werden.
Dabei werden die aus dem vorherigen Kapitel bekannten Werte $n = 15$ und $a = 7$ für Beispiele verwendet.

4.1 Hadamard-Transformation

Auf das erste Register wird eine Hadamard-Transformation H_N angewandt, um die Superposition aller k Zahlen zu erreichen.

$$R = \frac{1}{\sqrt{N}} \sum_{k=0}^{q-1} |k\rangle |0\rangle$$

Beispiel

$$R = \frac{1}{\sqrt{16}} \sum_{k=0}^{15} |k\rangle |0\rangle = \frac{1}{4} \begin{bmatrix} |0\rangle |0\rangle & +|1\rangle |0\rangle & +|2\rangle |0\rangle & +|3\rangle |0\rangle & + \\ |4\rangle |0\rangle & +|5\rangle |0\rangle & +|6\rangle |0\rangle & +|7\rangle |0\rangle & + \\ |8\rangle |0\rangle & +|9\rangle |0\rangle & +|10\rangle |0\rangle & +|11\rangle |0\rangle & + \\ |12\rangle |0\rangle & +|13\rangle |0\rangle & +|14\rangle |0\rangle & +|15\rangle |0\rangle & \end{bmatrix}$$

4.2 Orakel

In diesem Schritt wird die Funktion $a^k \mod n$ für alle k im ersten Register berechnet und in das zweite geschrieben

$$R = \frac{1}{\sqrt{N}} \sum_{k=0}^{q-1} |k\rangle \left| a^k \mod n \right\rangle$$

Beispiel

$$R = \frac{1}{\sqrt{16}} \sum_{k=0}^{15} |k\rangle \left| 7^k \mod 15 \right\rangle = \frac{1}{4} \begin{bmatrix} |0\rangle |0\rangle & +|1\rangle |0\rangle & +|2\rangle |0\rangle & +|3\rangle |0\rangle & + \\ |4\rangle |1\rangle & +|5\rangle |7\rangle & +|6\rangle |4\rangle & +|7\rangle |13\rangle & + \\ |8\rangle |1\rangle & +|9\rangle |7\rangle & +|10\rangle |4\rangle & +|11\rangle |13\rangle & + \\ |12\rangle |1\rangle & +|13\rangle |7\rangle & +|14\rangle |4\rangle & +|15\rangle |13\rangle & \end{bmatrix}$$

4.3 Quanten-Fouriertransformation

Der effizienteste Weg, ein Problem der Mathematik oder Informatik zu lösen, ist, es in ein anderes Problem zu transformieren, für welches eine effiziente Lösung bekannt ist. Es existieren einige Transformationen dieser Art, die so oft in unterschiedlichen Bereichen angewandt werden, dass diese Transformationen sehr genau untersucht und studiert werden. Ein sehr großer Fortschritt für die Quanteninformatik ist, dass solche Transformationen auf einem Quantencomputer sehr viel schneller ausgeführt werden können als auf einem klassischen Computer[7].
Im folgenden Abschnitt wird die klassische Fouriertransformation vorgestellt und anschließend wird mit einem Anwendungsbeispiel die Quanten-Fouriertransformation angesprochen.

4.3.1 Fouriertransformation

Wie bereits beschrieben, werden Transformationen angewandt, um die Berechnung, Beschreibung, Analyse und Auswertung von unterschiedlichen Problemen vereinfachen. Ein Beispiel ist die Wechselstrombetrachtung.
Die Fouriertransformation zerlegt eine periodische Funktion, die durch eine unendliche Reihe von Sinus- und Kosinusfunktionen beschrieben werden kann, in ihre Frequenzanteile. Die Abbildung 4.2 zeigt eine Periodische Funktion, die durch viele Sinusfunktionen mit unterschiedlichen Frequenzen und Amplituden abgebildet werden kann. Der Graph enthält die Amplituden auf der y-Achse und die Frequenzen auf der x-Achse. Häufig wird die x-Achse logarithmisch dargestellt, damit Bereiche von wenigen Hertz bis hin zu Megahertz, Gigahertz oder darüber hinaus vernünftig dargestellt werden können[1].

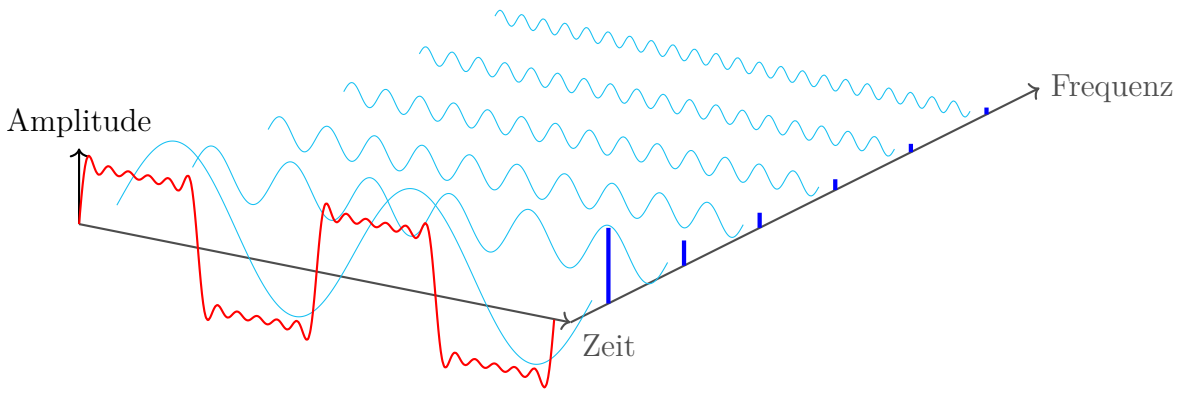


Abbildung 4.2: Fouriertransformation

[1]Für einen Graphen mit einer logarithmischen Achse siehe Abbildung 4.4

4.3.2 Quanten-Fouriertransformation

Die Quanten-Fouriertransformation dient dazu, die richtige Periode aus den Quantenbits zu extrahieren. Dazu wird die Wahrscheinlichkeit der Lösung stark angehoben und gleichzeitig die Wahrscheinlichkeiten der falschen Werte gesenkt, sodass der Wert der Periode beim Messvorgang ermittelt wird.

Das Problem lässt sich mit den Einheitswurzeln der komplexen Zahlen lösen. Eine Lösung der Gleichung $x^n = 1$ in $\mathbb{C}$ heißt komplexe n-te Einheitswurzel. Alle Nullstellen von $x^n - 1$ haben die Länge 1. Der Einheitskreis wird dabei in n gleich große Segmente unterteilt. Die Nullstellen betragen:

$$\omega_k = e^{\frac{2\pi i \cdot k}{N}} \quad (k = 0, ..., n-1).$$

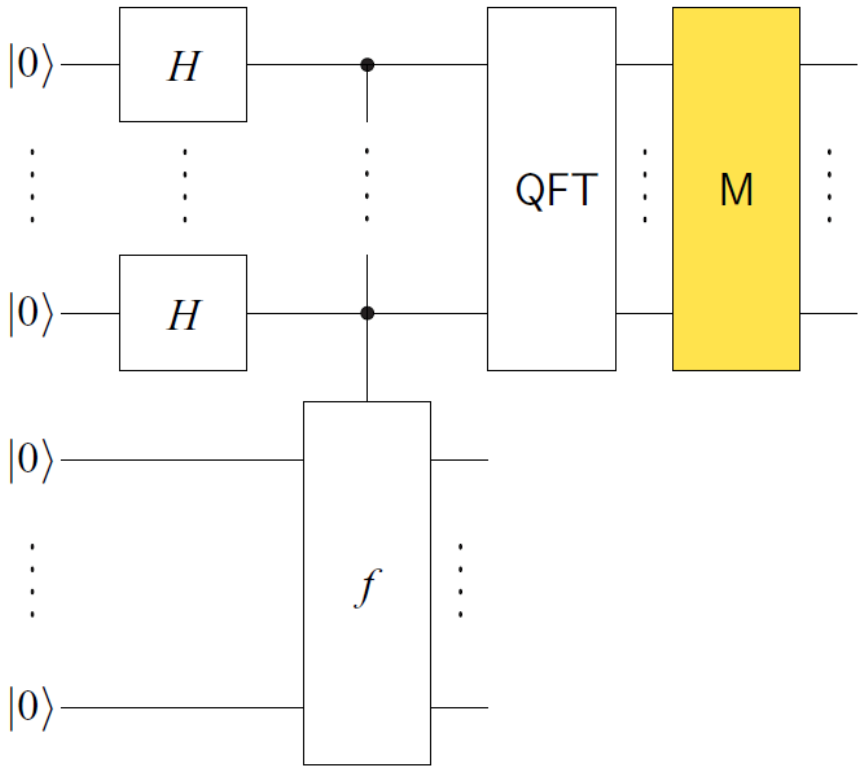
Die Einheitswurzeln kommen bei der Quanten-Fouriertransformation zum Einsatz. Die Ordnung N wird durch die unitäre Matrix

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & \omega & \omega^2 & \omega^3 & ... & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & ... & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & ... & \omega^{3(N-1)} \\ ... & ... & ... & ... & ... & ... \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} ... & ... & \omega^{(N-1)(N-1)} \end{bmatrix}$$

beschrieben, wobei ω die N-te Einheitswurzel $e^{\frac{2\pi i}{N}}$ bezeichnet. Die weitere Vorgehensweise wird hier nicht weiter beschrieben, da der Prozess sowie die weiteren Formeln sehr komplex sind. Bei Interesse kann das Buch *Quantum computation and quantum information* von Michael A. Nielsen und Isaac Chuang hinzugezogen werden, das das Thema recht ausführlich behandelt[7].

4.4 Schaltkreis

Der *Shor-Algorithmus* benötigt zwei Quantenregister mit jeweils $log(n)$ Bits. Das erste Register wird in die Superposition gebracht, in dem auf alle Bits die Hadamard-Transformation (H) angewandt wird. Im zweiten Register wird das Orakel mit der Funktion $f = a^k \mod n$ ausgeführt. Anschließend wird die Quanten-Fouriertransformation (QFT) auf beide Register angewandt, um abschließend das Ergebnis zu messen (M).
In der Abbildung 4.3 wird der Schaltkreis des Algorithmus zum Bestimmen der Periode dargestellt, der in einem Quantencomputer zum Einsatz kommt.



Quelle: Homeister[4]

Abbildung 4.3: Schaltkreis des *Shor-Algorithmus*

4.5 Laufzeit

Der *Shor-Algorithmus* hat eine Laufzeit von $L^2 \cdot \log(L) \cdot \log\log(L)$, wobei L die Anzahl der Bits darstellt. Damit gilt $L = \log n$. Hierbei macht es mehr Sinn die Anzahl der Bits – also die Schlüssellänge – zu betrachten als die genaue Zahl n. Zur Einfachheit kann auch gesagt werden, dass der *Shor-Algorithmus* eine Laufzeit von $\mathcal{O}(L^3)$ hat.
Der beste bisher bekannte Algorithmus für klassische Algorithmen hat hingegen eine Laufzeit von $e^{L^{\frac{1}{3}} \cdot \log^{\frac{2}{3}}(L)}$.
Die Abbildung 4.4 zeigt alle drei angesprochenen Funktionen grafisch und verdeutlicht, wie groß der Unterschied im Bezug auf die Dauer zwischen diesen Algorithmen ist.

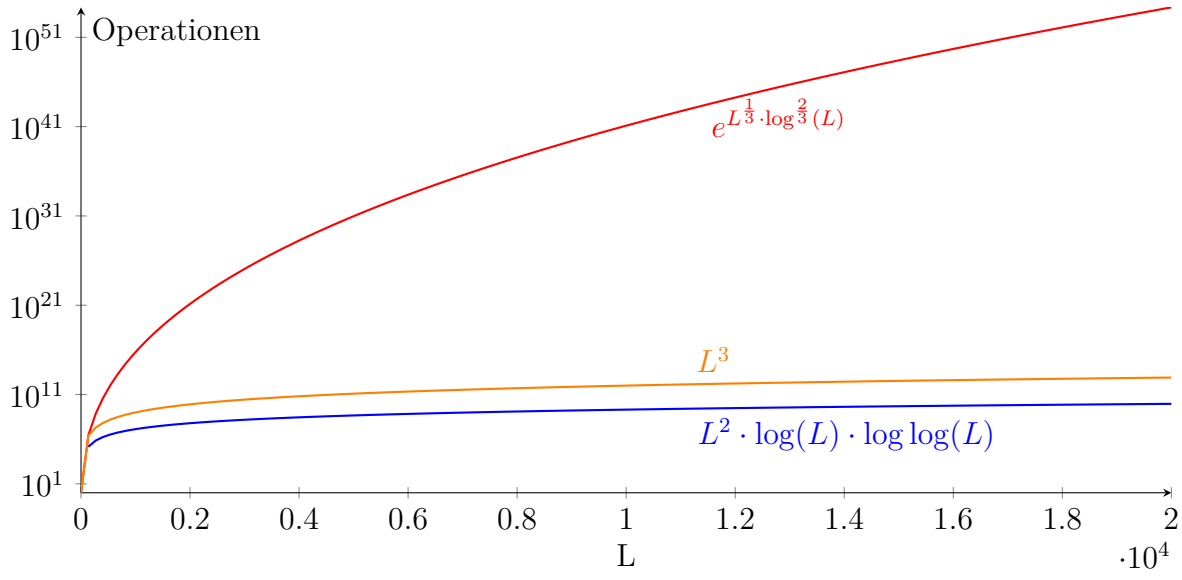
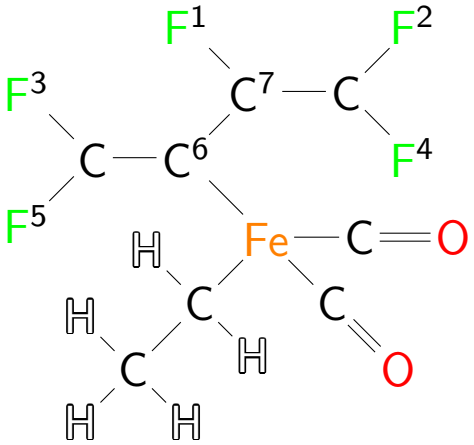


Abbildung 4.4: Laufzeit der Faktorisierung: rot = Zahlkörpersieb, blau = *Shor-Algorithmus*

5 Umsetzung des Shor-Algorithmus

Im Jahr 2001 gelang es IBM die Zahl 15 mit dem *Shor-Algorithmus* in ihre Bestandteile zu zerlegen. Dabei wurde ein Quantencomputer mit einem Fluorkohlenstoff-Molekül verwendet. Damit hatte der Quantencomputer sieben Qubit, die durch die einzelnen Atome des Moleküls realisiert wurden. Die Abbildung 5.1 stellt dieses Molekül dar. In diesem Fall wurde die Kernspinresonanz ausgenutzt, die aus Wechselwirkung von Magnetfeldern und Atomkernen entsteht.



Quelle: IBM

Abbildung 5.1: Molekül mit sieben Qubits

Innerhalb der nächsten Jahre wurde die Zahl 15 auch durch Quantencomputer faktorisiert, die andere quantenmechanische Effekte ausnutzen. 2009 wurde ein *Photonic chip* verwendet, der die Daten mit Licht anstatt mit elektrischem Strom überträgt. 2012 wurde der Josephson-Effekt ausgenutzt. Im selben Jahr und unter Ausnutzung des selben Effektes konnte auch die Zahl 21 faktorisiert werden[2].
Mit einem angepassten Algorithmus konnte 2012 die Zahl 143 sowie 56153 faktorisiert werden[2]
Einer Forschergruppe aus Kanada gelang es 2016, Zahlen über 20 000 auf einem *D-Wave 2X* mit 1000 Qubits zu faktorisieren[3].

6 Ausblick

Aufgrund der geringeren Laufzeit gegenüber klassischen Algorithmen, hat der *Shor-Algorithmus* das Potential große Zahlen in relativ kurzer Zeit zu faktorisieren. Sobald Quantencomputer mit einer ausreichend großen Anzahl an Qubits verfügbar sind, werden einige Kryptosysteme unbrauchbar. Problematisch ist in dieser Hinsicht, dass jeder, der verschlüsselte Daten gespeichert hat, diese auch irgendwann entschlüsseln kann. Die Tabelle 6.1 stellt die Sicherheit einiger asynchroner Verschlüsselungsalgorithmen dar. Wie zu erkennen ist, kann die Schlüssellänge nicht einfach vergrößert werden, da dies nur wenig bringt, die Verschlüsselung an sich aber aufwendiger macht. Auch andere asynchrone Kryptosysteme wie *Diffie-Hellman* (DH) oder die *Elliptische-Kurven-Kryptografie* (ECC) halten Quantencomputern nicht stand. Das heißt jedoch nicht, dass Quantencomputer alle asynchronen Algorithmen knacken können. Dies verdeutlicht beispielhaft *McEliece*.
Inwiefern synchrone Verschlüsselungsalgorithmen Quantencomputern standhalten wird in der Ausarbeitung von Björn Philip Ridder (*ite102659*) beschrieben.

Algorithmus	**Schlüssellänge**	**KonventionelleSicherheit**	**PQ-Sicherheit**
RSA-3072	3072 Bit	128 Bit	35 Bit
RSA-7680	7680 Bit	192 Bit	38 Bit
RSA-15360	15 365 Bit	256 Bit	41 Bit
ECC-256	256 Bit	128 Bit	35 Bit
DH-3072	3072 Bit	128 Bit	35 Bit
McEliece	1 MB	–	128 Bit

Tabelle 6.1: Sicherheit ausgewählte asynchroner Verschlüsselungsalgorithmen

Literaturverzeichnis

[1] Johannes Buchmann. *Einführung in die Kryptographie*. 5. Aufl. Springer-Verlag, 2010.

[2] Nikesh S Dattani und Nathaniel Bryans. „Quantum factorization of 56153 with only 4 qubits“. In: *arXiv preprint arXiv:1411.6758* (2014).

[3] Raouf Dridi und Hedayat Alghassi. „Prime factorization using quantum annealing and computational algebraic geometry“. In: *Scientific Reports* 7 (2017), S. 43048.

[4] Matthias Homeister. *Quantum Computing verstehen: Grundlagen-Anwendungen-Perspektiven*. 4. Aufl. Springer-Verlag, 2015.

[5] Samuel J Lomonaco. „A Lecture on Shor's Quantum Factoring Algorithm“. In: Citeseer, 2000.

[6] Gary L Miller. „Riemann's hypothesis and tests for primality“. In: *Journal of computer and system sciences* 13.3 (1976), S. 300–317.

[7] Michael A Nielsen und Isaac Chuang. *Quantum computation and quantum information*. 10th Anniversary Edition. New York: Cambridge University Press, 2010.

[8] Ronald L Rivest, Adi Shamir und Leonard Adleman. „A method for obtaining digital signatures and public-key cryptosystems“. In: *Communications of the ACM* 21.2 (1978), S. 120–126.

[9] Peter W Shor. „Algorithms for quantum computation: Discrete logarithms and factoring“. In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE. 1994, S. 124–134.

[10] Peter W Shor. „Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer“. In: *arXiv preprint quant-ph/9508027* (1995).