Technische
Hochschule
Rosenheim

Fakultät für Informatik

Studiengang Informatik M.Sc.

Der Shor Algorithmus und seine Auswirkungen auf asymmetrische Verschlüsselung am Beispiel von RSA

Seminar theoretische Informatik

von

Christian Pritzl

Datum der Abgabe: 19.01.2022

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Rosenheim, den 19.01.2022

Christian Pritzl

Kurzfassung

Mit Hilfe von asymmetrischen Verschlüsselungsverfahren wird das Problem des Schlüsselaustausches gelöst. Das bekannteste Kryptosystem aus diesem Bereich stellt RSA dar, welcher 1977 erstmalig vorgestellt wurde und bis heute in vielen Bereichen zur Verschlüsselung eingesetzt wird.

Allerdings konnte die Sicherheit von RSA mathematisch nie bewiesen werden. Vielmehr setzt RSA darauf, dass die Faktorisierung von Zahlen ein sogenanntes „schweres“ Problem ist. Da Computer nicht in der Lage sind, Zahlen effizient zu faktorisieren, kann die RSA-Verschlüsselung nicht in sinnvoller Zeit gebrochen werden.
Das es neben der eigentlichen Faktorisierung mindestens eine weitere Möglichkeiten gibt, die Faktoren eines Produktes zu bestimmen, konnte Victor Miller 1976 beweisen. Allerdings ist die sogenannte Periodenfindung ein ebenso schweres Problem für klassische Computer wie es die Faktorisierung ist.

In den 1990er Jahren wurde eine ganze Reihe von Fortschritten im Bereich der Quantencomputer erzielt. Einer der bis heute bekanntesten neuen Algorithmen ist der, 1994 von Peter Shor publizierte, Shor-Algorithmus. Dieser Algorithmus greift die Arbeit von Miller auf und nutzt die Eigenschaften von Quanten, um die Periodenfindung massiv zu beschleunigen. So muss, zumindest in der Theorie, das RSA-Kryptosystem als unsicher betrachtet werden.
Allerdings konnte der theoretische Fortschritt bis heute nicht in die Praxis übertragen werden. Da sich der Schwerpunkt der Forschungstätigkeit zunehmend aus dem akademischen Umfeld fortbewegt und Unternehmen, sowie Staaten und Organisationen, verstärktes Interesse an Quantencomputern zeigen, kann davon ausgegangen werden, dass bereits in wenigen Jahren skalierbare Quantencomputer verfügbar sind. Diese werden über genügend Qubits verfügen, um anspruchsvolle Algorithmen, wie den Shor-Algorithmus, für große Zahlen auszuführen.

Um weiterhin Verschlüsselung zu ermöglichen wird parallel seit Jahren daran gearbeitet, neue Algorithmen und Kryptosysteme zu entwickeln und zu standardisieren. Diese neuen

Verschlüsselungsverfahren werden sowohl gegen klassische Computer, als auch gegen quantenbasierte Computer sicher sein, da deren Sicherheit mathematisch bewiesen sein wird.

Schlagworte:
Asymmetrische Verschlüsselung, Shor, Quantencomputer, Periodenfindung, RSA

Abkürzungsverzeichnis

BSI Bundesamt für Sicherheit in der Informationstechnik 1

DNSSEC Domain Name System Security Extensions 3

GPG GNU Privacy Guard 3

ISO International Organization for Standardization 25

KTH Kungliga Tekniska högskolan 20

NFS Number Field Sieve 6

NIST National Institute for Standards and Technology 25

PGP Pretty Good Privacy 3

RSA Rivest–Shamir–Adleman 2

SSH Secure Shell 3

SSL Secure Socket Layer 3

TLS Transport Layer Security 3

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung und Abgrenzung

Bereits 1945/1946 hatte der deutsche Ingenieur Konrad Zuse die Idee, dass der Kosmos selbst ein Quantencomputer sein könnte. Diese These stellte er 1969 in seinem Buch „Rechnender Raum“ vor. Seth Loyd hat dies in seinem Buch „Programming the Universe“ 2006 aufgegriffen und weiter fortgeführt. Dieser Ansatz ist naheliegend, da Physiker immer wieder an Grenzen bei der Erklärung diverser Quantenphänomene stießen. Folglich befassten sich Forscher und Entwickler mit der Konzeption eines Quantencomputers, zunächst theoretisch, später dann auch praktisch, mit dem Ziel eine Möglichkeit zur exakten Simulation von Naturphänomenen zur Verfügung zu haben.
1992 wurde von David Deutsch und Richard Jozsa mithilfe des Deutsch-Jozsa-Algorithmus erstmalig der Beweis erbracht, dass Quantencomputer tatsächlich effizienter arbeiten können als klassische Computer. Dan Simon konnte 1994, anhand des nach ihm benannten Simon-Problem, beweisen, dass Quantencomputer spezifische Probleme, wie Primfaktorzerlegung oder Suchen, sogar exponentiell schneller berechnen können. Eine Umsetzung zur effizienten Berechnung der Primfaktorzerlegung publizierte Peter W. Shor, inspiriert von den Algorithmen von Deutsch, Jozsa und Simon, noch im selben Jahr. 1996 konnte Lev Grover einen quantenbasierten Suchalgorithmus, den nach ihm benannten Grover-Algorithmus, für die Suche in einer umsortierten Datenbank, vorstellen.

Bei der Entwicklung von Quantencomputern führend sind bekannte Unternehmen wie Google, IBM und Microsoft, sowie das kanadische Unternehmen D-Wave Systems. Diese verfügen aktuell über Quantencomputer mit etwa 20 bis 70 Qubits, welche aktuell noch weit von jeglicher Relevanz für den Einsatz in der Kryptoanalyse entfernt sind. In seiner Entwicklungseinschätzung zu Quantencomputern von 2020 geht das Bundesamt für Sicherheit in der Informationstechnik (BSI) allerdings davon aus, dass sich die Entwicklung in den nächsten Jahren stark beschleunigen wird. Dies liegt vor allem darin begründet, dass Quantencomputer bisher hauptsächlich im akademischen Umfeld erforscht wurden. Das gesteigerte Interesse großer Unternehmen und Staaten wird zu leistungsfähigeren und fehlertoleranten Quantencomputern führen, die auch im Bereich der Kryptoanalyse an Relevanz gewinnen werden.[1]

1 [14]

Peter W. Shor publizierte in seiner Arbeit „Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer“ zwei Algorithmen, die eine Faktorisierung großer Ganzzahlen beziehungsweise die Berechnung diskreter Logarithmen asymptotisch ermöglichen. Shor griff hierbei auf die, von Victor Miller, 1976 publizierte, Möglichkeit, die Faktorisierung durch eine Reduzierung auf das Problem der Periodenfindung zu lösen, zurück. Dies kann auf einem klassischen Computer nicht effizient durchgeführt werden. Auf einem Quantencomputer allerdings kann die Überlagerung der Zustände zur Periodenfindung genutzt werden. Sobald ein entsprechend leistungsfähiger Quantencomputer verfügbar ist, müssen kryptographische Verfahren aus dem Bereich Rivest–Shamir–Adleman (RSA) und Elliptischen-Kurven als unsicher betrachtet werden, da auch eine Vergrößerung der Schlüssellänge keine zusätzliche Sicherheit mehr bringen wird.

In dieser Arbeit wird der quantenbasierte Faktorisierungsalgorithmus von Peter W. Shor im Kontext der asymmetrischen Verschlüsselung am Beispiel von RSA vorgestellt. Hierzu wird zunächst eine kurze Einführung in den Ablauf von RSA gegeben. Anschließend werden einige zahlentheoretische Grundlagen, die für das weitere Vorgehen nötig sind, eingeführt. Die Vorstellung des Algorithmus erfolgt zweistufig, zunächst anhand einer Vorüberlegung und anschließend detaillierter mit Hilfe eines Beispiels. Abschließend wird der technische Stand bei der Implementierung sowie der Forschungsstand bei der Optimierung des Algorithmus analysiert und ein Fazit gezogen, inwiefern aktuelle Implementierungen des Shor-Algorithmus auf aktuellen Quantencomputern asymmetrische Verschlüsselung bereits in naher Zukunft unsicher machen können.

2 Grundlagen der Asymmetrischen Verschlüsselung

Grundsätzlich lassen sich zwei Typen von Verschlüsselungsalgorithmen unterscheiden. Die sogenannten konventionellen, klassischen symmetrischen Verschlüsselungsalgorithmen und die, 1977 erstmals mit RSA vorgestellte, asymmetrische Verschlüsselung. Bei der symmetrischen Verschlüsselung wird ein Schlüssel sowohl für Verschlüsselung als auch für Entschlüsselung des sogenannten Cyphertextes verwendet.

Asymmetrische Verschlüsselung, auch bekannt als Public-Key-Verschlüsselung, verwendet je einen Schlüssel für Verschlüsselung und einen separaten Schlüssel zur Entschlüsselung des Cyphertextes. Da sich der Schlüssel für die Entschlüsselung nicht in sinnvoller Zeit aus dem Schlüssel für die Verschlüsselung berechnen lässt, kann dieser öffentlich gemacht werden. Damit lässt sich das Problem der Schlüsselverteilung, welches symmetrische Verfahren haben, lösen. Das bedeutet, jeder kann mithilfe des öffentlichen Schlüssels eine Nachricht verschlüsseln, aber nur der Besitzer des privaten Schlüssels kann diese wieder lesbar machen.

Praktisch implementiert ist dieses Verfahren in weit verbreiteten Protokolle wie zum Beispiel Secure Shell (SSH), Secure Socket Layer (SSL)/Transport Layer Security (TLS), Pretty Good Privacy (PGP)/GNU Privacy Guard (GPG) oder Domain Name System Security Extensions (DNSSEC). Bekannte Public-Key-Algorithmen sind der bereits erwähnte RSA und der ElGamal-Algorithmus aus dem Jahr 1985. Beide Verfahren leiten ihre Sicherheit aus recht ähnlichen Problemstellungen für klassische Computer ab. Bei RSA ist das Problem ein Produkt aus zwei Primzahlen wieder zu faktorisieren, bei ElGamal die Berechnung des diskreten Logarithmus.[1]

1 [20]

2.1 Das RSA-Kryptosystem

Im folgenden Kapitel wird am Beispiel des RSA-Algorithmus das Prinzip der Public-Key-Verschlüsselung vorgestellt. RSA wurde 1977 von Ron Rivest, Adi Shamir und Leonard Adleman vorgestellt und ist bis heute der vermutlich populärste Algorithmus aus der Kategorie asymmetrischer Verschlüsselungen. Zu seiner großen Verbreitung trägt seine gute Verständlichkeit und die Einfachheit der Implementierung bei.

Im Folgenden wird kurz vorgestellt, wie die Einzelschritte zur Verschlüsselung und Entschlüsselung sowie die Generierung der Schlüssel theoretisch abläuft. Dieses Wissen ist die Grundlage um zu verstehen, warum die praktische Sicherheit von RSA die theoretische weit übersteigt.

Um eine Nachricht zu verschlüsseln wird folgende Formel verwendet:

$$c = m^e mod N$$

wobei c für den Cyphertext, m für die Klartextnachricht, e (encryption) für den öffentlichen Teil des Schlüssels und N für das Produkt zweier Primzahlen steht.
Die Entschlüsselung von c erfolgt analog:

$$m = c^d mod N$$

wobei d (decryption) hier für den privaten Teil des Schlüssels steht.
Folglich ist der erste Schritt bei der Generierung des Schlüsselpaares die zufällige Wahl zweier, bestenfalls, gleichgroßer Primzahlen p und q um daraus die Zahl N mittels Multiplikation zu berechnen. Mit Größe ist hier die Länge in Bit gemeint, die entsprechend einer Empfehlung des BSI aktuell mindestens 2000 Bit betragen sollte.[2]
Als nächstes muss die Eulersche Phi-Funktion

$$\varphi(n) = (p-1) * (q-1)$$

angewendet werden. Mit dieser kann ein zufällig gewähltes e für die Verschlüsselung auf Invertierbarkeit geprüft werden. Dies ist erfüllt, wenn e teilerfremd zu $\varphi(n)$ ist:

$$ggT(e, \varphi(n)) = 1$$

.

2 [4]

Aus e lässt sich anschließend der private Schlüssel d zur Entschlüsselung berechnen, hier gilt:

$$e * d \equiv 1 mod \varphi(n)$$

Um eine Kommunikation zu ermöglichen, muss die Kombination (e,n) öffentlich verfügbar gemacht werden. Anhand dieser Formeln lässt sich nachvollziehen, dass der private Schlüssel d berechnet werden kann, falls es gelingt die Zahl N in die beiden Primfaktoren p und q zu zerlegen.[3]

2.2 RSA im Kontext aktueller Faktorisierungsalgorithmen

Wie in Kapitel 2.1 gezeigt, beruht RSA auf der Multiplikation zweier großer Primzahlen. Um den Cyphertext entschlüsseln zu können, ist es nötig, diese Zahl zu faktorisieren.

Die Sicherheit von RSA stützt sich auf zwei Annahmen, die beide bis heute nicht mathematisch bewiesen werden konnten. Die erste Annahme geht davon aus, dass es nicht möglich ist, diese zusammengesetzte Zahl mit polynomialer Laufzeit zu faktorisieren. Dieses Problem wird in der Kryptographie auch als „schwierig“ bezeichnet. Die zweite Annahme ist die Notwendigkeit, überhaupt faktorisieren zu müssen, um den Cyphertext entschlüsseln zu können.[4]

Den Beweis, dass die erste Annahme falsch ist, lieferte Peter W. Shor mit der Vorstellung seines Quantenalgorithmus zur Faktorisierung in polynomialer Laufzeit, welcher in Kapitel 3 vorgestellt wird. Bezüglich der zweiten Annahme ist weitere Forschung nötig, um eine Aussage tätigen zu können.

Im wesentlichen gibt es zwei große Gruppen von Faktorisierungsverfahren. Die erste Gruppe eignet sich für das Finden von kleinen Primfaktoren. Beispiele hierfür sind die Probedivision und die Lenstra Elliptic-Curve-Faktorisierung. Bei der Probedivision wird eine wiederholte Division von N durch alle Primzahlen bis $\sqrt{N}$ durchgeführt, solange bis ein Teiler gefunden wurde. Dieses Verfahren weist exponentielle Laufzeit bezüglich N auf. Die Lenstra Elliptic-Curve-Faktorisierung ist aktuell der schnellste bekannte Algorithmus aus der Gruppe der Algorithmen deren Komplexität hauptsächlich von der Größe von N abhängig ist.

3 [20]
4 [20]

Die zweite Gruppe von Algorithmen basiert auf der sogenannten „Kombination von Kongruenzen“ und stellt die aktuell schnellsten Algorithmen zur Faktorisierung von großen Zahlen. Für N bis circa 130 Dezimalstellen wird vorrangig das sogenannte „Quadratische Sieb “ , welches eine Laufzeit von $2^{O(\sqrt{n})}$ aufweist, verwendet. Mit diesem Algorithmus wurde 1994 eine Zahl mit 129 Dezimalstellen in etwa 8 Monaten faktorisiert (129-digit RSA Challenge).
Eine Verallgemeinerung des Quadratischen Siebs ist das sogenannte „Zahlkörpersieb“ ‚Number Field Sieve (NFS), welches den aktuellen Standard zur Faktorisierung von N in der Kryptographie darstellt. Die Laufzeit dieses Algorithmus beträgt $2^{n^{1/3}}$.
Einfach ausgedrückt wird bei diesen Algorithmen ähnlich zum Sieb des Eratosthenes vorgegangen, bis im letzten Schritt aussichtsreiche Primzahlen mittels Probedivision getestet werden.[5] [6] Im Februar 2020 wurde mit diesem Algorithmus der 829 Bit große RSA-250 Schlüssel faktorisiert.[7] Die aktuell gängige Schlüssellänge beträgt 2048 Bit, was einer Dezimalzahl mit 617 Dezimalstellen entspricht.

Stand heute gibt es kein besseres Verfahren als das Zahlkörpersieb mit subexponentieller Laufzeit, um N zu faktorisieren.[8]

Ein Vergleich der benötigten Operationen zwischen dem Zahlkörpersieb mit dem Shor-Algorithmus kann Abbildung 2.1 entnommen werden. Aus der Grafik lässt sich ablesen, dass auch eine Zunahme der Schlüssellänge keinen zusätzlichen Schutz vor einer Faktorisierung durch den Shor-Algorithmus mehr bedeutet.

5 [2]
6 [17]
7 [7]
8 [17]

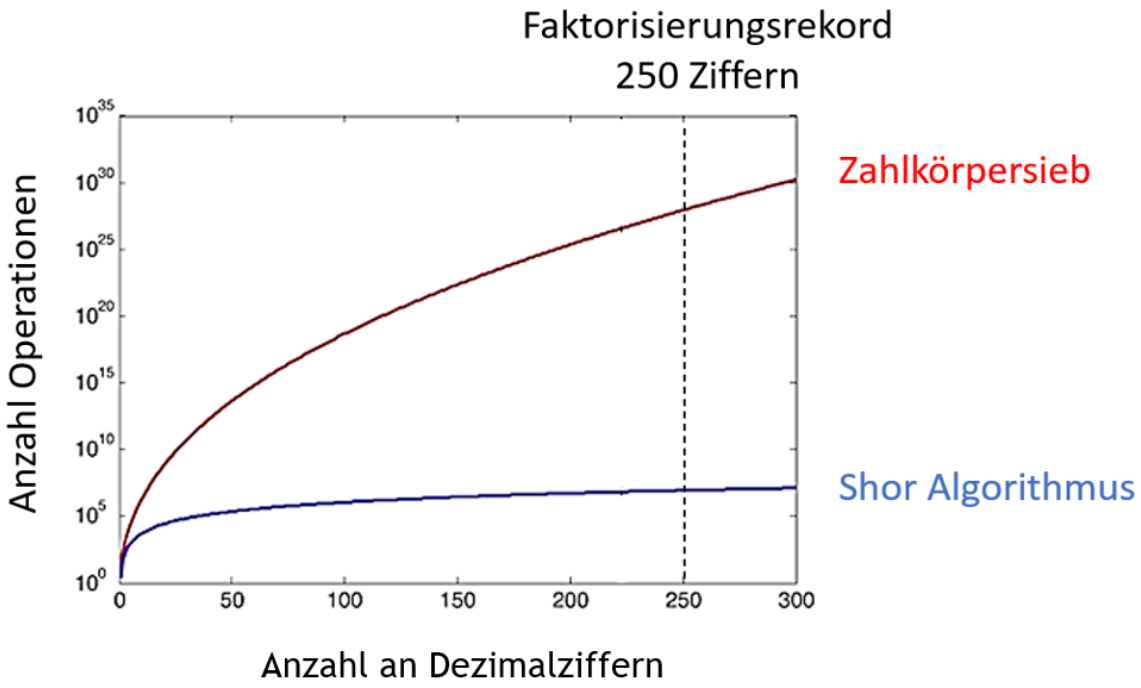

Abbildung 2.1 Vergleich der benötigten Operationen zur Faktorisierung von Zahlen zwischen dem Zahlkörpersieb und dem Shor-Algorithmus, bearbeitete Abbildung [13]

3 Vorstellung des Shor-Algorithmus

Im folgenden Kapitel werden zunächst die mathematischen und zahlentheoretischen Grundlagen geschaffen, um im Anschluss einen detaillierten Einblick in den Ablauf des Shor-Algorithmus zu geben. Der Shor-Algorithmus greift hierbei zwei interessante Ansätze auf. Zunächst im ersten Schritt die Reduzierung der Faktorisierung auf die Periodenfindung und anschließend die Anwendung der Quantenfouriertransformation, um die Ergebnisse der Berechnung aus der Quantenwelt in die Welt klassischer Computer zu führen. Die Reduzierung auf das Problem der Periodenfindung ist nötig, da auch für Quantencomputer Faktorisierung keine triviale Aufgabe ist. Somit stellt der Shor-Algorithmus, im Gegensatz zur landläufigen Meinung, einen Quantenalgorithmus zur Periodenfindung bereit. Dieser Algorithmus ist, wie für Quantenalgorithmen üblich, probabilistisch und wird die Klasse der Monte-Carlo-Algorithmen einsortiert. Das bedeutet, dass ein Ergebnis nur mit einer bestimmten Wahrscheinlichkeit gefunden wird und der Algorithmus auch falsche Ergebnisse zurückliefern darf. Sobald dieses Ergebnis gefunden wurde, kann auf einem klassischen Computer mit Hilfe des euklidischen Algorithmus ein tatsächlicher Faktor der Ausgangszahl bestimmt werden. Während dieser Schritt ebenfalls auf einem Quantencomputer ausgeführt werden könnte, empfiehlt Shor diesen Schritt auf einem klassischen Computer auszuführen, da auf diesen bereits gut optimierte Implementierungen vorhanden sind. [1]

Die Abbildung 3.1 zeigt ein vereinfachtes Ablaufdiagramm des Shor-Algorithmus, welches in diesem Kapitel zur Orientierung verwendet wird.

3.1 Zahlentheoretische Grundlagen

In diesem Kapitel werden einige grundsätzliche Begriffe und Theorien aus der Zahlentheorie eingeführt, die für das weitere Verständnis des Shor-Algorithmus nötig sind.

1 [25]

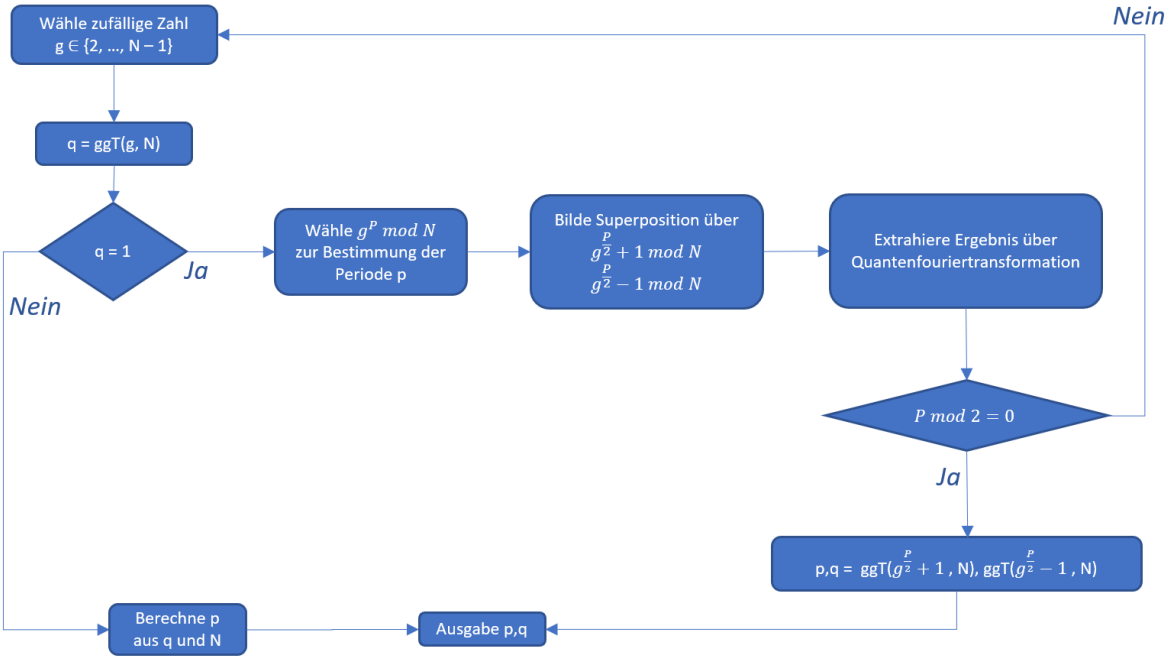

Abbildung 3.1 Vereinfachtes Ablaufdiagramm des Shor-Algorithmus, eigene Darstellung

3.1.1 Primzahlen und Primfaktorzerlegung

Eine Zahl $p \in \mathbb{N}; p > 1$, die nur 1 und sich selbst als Teiler besitzt, wird als Primzahl bezeichnet. Die Darstellung einer Zahl $n \in \mathbb{N}; n > 1$ als Produkt ihrer Primfaktoren

$$n = p_1 * ... * p_n$$

wird als Primfaktorzerlegung, beziehungsweise allgemeiner als Faktorisierung bezeichnet.

Dies wird auch als *Hauptsatz der elementaren Zahlentheorie* bezeichnet.

3.1.2 Periode und Ordnung

Als Ordnung bezeichnet man die kleinste von 0 verschiedene, natürliche Zahl r, für die der Satz von Euler

$$x^r \equiv 1 \ mod \ n, mit \ x \in \mathbb{Z}^n$$

gilt. Wenn x fest ist, dann ist die Ordnung die kleinste Periode der Funktion

$$f : k \mapsto x^k \ mod \ n$$

. Diese Funktion wird im Deutschen auch als diskrete Exponentialfunktion bezeichnet. Anders ausgedrückt ist die Periode die kleinste von 0 verschiedene Zahl mit

$$f(k) = f(k+r)$$

.

Unter der Annahme, dass r gerade ist, lässt sich die Funktion folgendermaßen umschreiben:

$$(x^{\frac{r}{2}} - 1) * (x^{\frac{r}{2}} + 1) = 0 \ mod \ n$$

und das wiederum ist äquivalent zu

$$(x^{\frac{r}{2}} - 1) * (x^{\frac{r}{2}} + 1) = m * n$$

für eine beliebige positive Ganzzahl m.
Das sich die Faktorisierung auf die Periodenfindung reduzieren lässt ist offensichtlich, wenn $m = 1$ angenommen wird. In diesem Fall gilt

$$(x^{\frac{r}{2}} - 1) * (x^{\frac{r}{2}} + 1) = n$$

. Wenn m nun größer ist als 1, dann sind die beiden gesuchten Faktoren p und q jeweils der größte gemeinsame Teiler von $(x^{\frac{r}{2}} + -1)$ und n:[2]

$$p = ggT((x^{\frac{r}{2}} - 1), n)$$
$$q = ggT((x^{\frac{r}{2}} + 1), n)$$

Da es für klassische Computer extrem schwer ist eine solche Periode zu bestimmen, wird die diskrete Exponentialfunktion als Einwegfunktion in vielen asymmetrischen Verschlüsselungsverfahren eingesetzt. Allerdings wurde bereits 1976 bewiesen, dass Faktorisierung auf das Problem der Periodenfindung für die diskrete Exponentialfunktion reduziert werden kann, womit keine direkte Bestimmung der Primfaktoren mehr nötig ist. [3]

2 [19]
3 [15]

3.1.3 Quantenfouriertransformation

Um aus einem periodischen Signal die Periode zu extrahieren, kann die sogenannte „Fouriertransformation“ angewendet werden. Diese findet Einsatz in unterschiedlichen Fachbereichen, beispielsweise bei der Signalverarbeitung, Datenkomprimierung oder in der Komplexitätstheorie.
In der Quantenphysik ist es nicht möglich, direkte Messungen vorzunehmen, da eine solche Messung lediglich ein potentielles Ergebnis zurückgibt und alle weiteren zerstört. Aus diesem Grund muss dieses Ergebnis in einen anderen Status überführt werden, der eine direkte Messung erlaubt. Eine Möglichkeit dieses Ziel zu erreichen ist der Einsatz der Quantenfouriertransformation.[4]

Eine Besonderheit von Qubits ist ihre Welleneigenschaft. Die Zustände der Qubits entsprechen hierbei Amplituden, welche negative, positive oder komplexe Werte annehmen können. In einem Quantencomputer überlagern sich die Wellen einzelner Qubits und führen im Rahmen einer Berechnung zur Auslöschung von „falschen“ Ergebnissen und zu einer Verstärkung von „richtigen“ Ergebnissen durch Interferenz.
Das Ergebnis ist eine Wellenfunktion von sich überlagernden Wellen einzelner Qubits. [5]
Eine schematische Darstellung der sich überlagernden Amplituden kann Abbildung 3.2 entnommen werden. Die Abbildung zeigt, wie die Periode sich in den Amplituden mit großem positivem Ausschlag widerspiegelt. Der finale Zustand entspricht folglich den Amplituden mit dem größten Ausschlag.[6]

Die Quantenfouriertransformation liefert eine lineare Transformation, die einen Quantenzustand, der einer periodischen Sequenz entspricht, auf einen Quantenzustand abbildet, der der Periode dieser Sequenz entspricht. Sie lässt sich wie folgt als lineare, unitäre Matrix darstellen:[7]

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle\langle j|$$

4 [21]
5 [3]
6 [22]
7 [21]

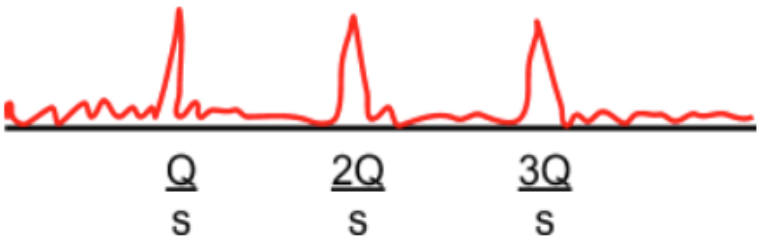

Abbildung 3.2 Schematische Darstellung der überlagerten Amplituden der Quantenzustände zur Darstellung der Periode.[21]

3.2 Vorüberlegungen

In diesem Kapitel werden eine Reihe von Vorüberlegungen angesprochen, die helfen, den detaillierteren Einblick in den Shor-Algorithmus in Kapitel 3.3 besser zu verstehen. Des Weiteren wird kurz dargelegt, warum der Ansatz der Periodenfindung für klassische Computer nicht verwendet werden kann.

Wenn man die Vielfachen einer Zahl g = 2

$$2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, \ldots$$

betrachtet und diese modulo einer anderen beliebigen Zahl, wie zum Beispiel n = 15, rechnet, bekommt man die Zahlenfolge

$$1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, \ldots$$

. Für n = 35 erhält man die Zahlenfolge

$$1, 2, 4, 8, 16, 32, 29, 23, 11, 22, 9, 18, 1, 2, 4, \ldots$$

. Offensichtlich wiederholen sich die Zahlen mit einer Periode von vier beziehungsweise 12, welche in Abhängigkeit des gewählten Modulus steht. Leonhard Euler hat dieses Phänomen bereits 1760 entdeckt und in der Form

$$g \ mod \ N, g^2 \ mod \ N, g^3 \ mod \ N, \ldots$$

verallgemeinert. Die entsprechende Definition kann Kapitel 3.1.2 entnommen werden.

Dieses Prinzip soll nun angewendet werden, um die Faktoren der Zahl 35 zu bestimmen.

Hierzu wird die Basis g = 2 gewählt. Die Basis g muss hierzu zwei Bedingungen erfüllen. Dem Ablaufdiagramm 3.1 lässt sich entnehmen, dass diese beiden Bedingungen, formuliert als Entscheidungen, an unterschiedlichen Stellen des Algorithmus zum Tragen kommen.

1. g und N müssen teilerfremd sein
2. Die Periode P muss gerade sein, was in etwa 50% der Fälle eintritt

Falls g und N nicht teilerfremd sind, kann der Algorithmus angewandt auf RSA terminiert werden, beziehungsweise erneut gestartet werden, um weitere Faktoren zu bestimmen. Falls es sich bei P um eine ungerade Zahl handelt, kann diese nicht verwendet werden, um die Faktoren zu berechnen. Der Algorithmus muss auch in diesem Fall mit einer neuen Basis g wiederholt werden.

Offensichtlich sind in diesem Beispiel beide Bedingungen erfüllt, sodass im Algorithmus jeweils fortgeschritten werden kann. Zunächst wird, wie in Kapitel 3.1.2 gezeigt, die Periode 12 halbiert, womit

$$g^{P/2} mod N = r$$

$$2^6 mod 35 = 29$$

berechnet werden kann. Anschließend wird die Formel $g^n \ mod \ n$ umgeformt zu

$$(2^{(6)} + 1) * (2^{(6)} - 1) mod 35$$

und daraus lässt sich

$$(2^{(6)} + 1) mod 35 = 30$$

und

$$(2^{(6)} - 1) mod 35 = 28$$

bestimmen.
Anschließend lassen sich die beiden Faktoren von 35 aus der Berechnung des größten gemeinsamen Teilers des Modulus mit $g^n \ +- 1 mod n$ bestimmen.

$$ggT(35, 28) = 7$$

$$ggT(35, 30) = 5$$

Diese beiden Zahlen entsprechen den Faktoren von 35.

Falls eines der beiden aufgelisteten Probleme eintritt, muss die Berechnung mit einem neugewählten g erneut gestartet werden. Die Wahrscheinlichkeit, dass P gerade ist und weder $g^{(P/2)}+1$ noch $g^{(P/2)}-1$ ein Vielfaches von N sind, beträgt 37.5 %.

Hierbei lässt sich ein struktureller Zusammenhang zwischen den Faktoren und der Periode feststellen. Wenn sich g nicht durch die beiden Faktoren p und q der Zahl N teilen lässt, dann teilt die Periode das Produkt

$$(p-1)*(q-1)=4*6=24$$

gleichmäßig auf.
Auch wenn dies bei der konkreten Berechnung keinen direkten Vorteil bringt, ist es wichtig strukturelle Zusammenhänge zur Entwicklung eines Algorithmus zu identifizieren.

Der Grund warum klassische Computer mit diesem Ansatz nicht arbeiten können liegt darin, dass sich für große Zahlen die Periode nicht trivial bestimmen lässt. Des Weiteren kann die Periode annähernd so groß wie N selbst sein, womit in der Berechnung nichts gewonnen wäre. Im Gegensatz hierzu kann ein Quantencomputer über die Verallgemeinerung von Euler eine Superposition bilden und somit die Periode, mit der sich die Zahlenfolge wiederholt, schnell finden. Dies ist möglich, da über die Superposition die Funktion an allen Punkten parallel ausgewertet werden kann. Ein klassischer Computer wiederum liefert für eine Eingabe immer nur eine Ausgabe.[8] [9] [10]

3.3 Einblick in den Ablauf des Shor-Algorithmus anhand von Beispielen

In Kapitel 3.2 wurde als Ausgangspunkt die Zahl g = 2 bestimmt, da g teilerfremd zum Modulus sein muss. In der Praxis wird zunächst versucht ein g zu bestimmen, welches einen Faktor mit N teilt. Dieser kann mit dem Algorithmus von Euklid bestimmt werden, sodass anschließend eine Faktorisierung von N möglich ist. Natürlich ist die Wahrscheinlichkeit eine Zahl zu erraten, die einen Faktor mit N teilt sehr gering. Auf die RSA-Verschlüsselung übertragen würde das korrekte Erraten gleichzeitig das Brechen der Verschlüsselung bedeuten. Aus diesem Grund sind einige Umformungen vorzunehmen, um die Wahrscheinlichkeit zu erhöhen, einen Faktor bestimmen zu können.

8 [22]
9 [23]
10 [21]

Wenn die Ausgangsgleichung

$$N = p * q$$

zu

$$(g^P + 1)(g^P - 1) = m * N$$

umgeschrieben wird, ist es deutlich wahrscheinlicher eine Lösung zu bestimmen. Bei m handelt es sich hierbei um einen unbekannten multiplikativen Faktor, der zeigt, das es ausreichend ist, eine Zahl zu bestimmen, die einen Faktor mit N teilt.
Der Grund dafür ist, dass man ein Vielfaches von N + 1 bekommt, wenn man g oft genug mit sich selbst multipliziert:

$$g * g * ... * g = m * N + 1$$

$$g^P = m * N + 1$$

$$g^P \equiv 1 \ mod N$$

. Der Summand 1 wird im Folgenden als „r “ bezeichnet. Zusammengefasst erhält man die diskrete Exponentialfunktion aus Kapitel 3.1.2, auf der die Periode bestimmt werden soll.

Mit weiterer geschickter Umformung erhält man anschließend folgende Darstellung:

$$g^P = m * N + 1$$

$$g^p - 1 = m * N$$

. Und nach Anwendung der dritten binomischen Formel lässt sich schreiben:

$$(g^{(P/2)} + 1) * (g^{(P/2)} - 1) = m * N$$

Wie in Kapitel 3.1.2 gezeigt, ist es nötig die Periode zu halbieren um die Faktoren bestimmen zu können. In der weiteren Berechnung kann der Faktor m grundsätzlich vernachlässigt werden, da über Euklid gemeinsame Faktoren erkannt und berücksichtigt werden können. Die Ähnlichkeit zur Ausgangsgleichung

$$N = p * q$$

ist offensichtlich.

Um nun die Periode P bestimmen zu können, wird eine Superposition über die Formel

$$g^{(P/2)} + -1$$

gebildet. So kann der Quantencomputer die linke Seite des Produkts

$$(g^{(P/2)} + 1) * (g^{(P/2)} - 1) = m * N$$

an allen Stellen simultan auswerten. Dies ist der entscheidende Schritt, den ein klassischer Computer nicht durchführen kann. Darüber hinaus ist es nötig, die ermittelten Ergebnisse innerhalb des Quantencomputers so anzuordnen, dass sich die Quantenzustände falscher Ergebnisse über Interferenzen gegenseitig eliminieren und richtige Ergebnisse verstärken.

Tatsächlich besteht die folgende Berechnung auf dem Quantencomputer aus zwei Schritten. Zunächst werden über die Superposition alle möglichen Werte für P evaluiert. Die Obergrenze für die zu evaluierenden Werte wird mit Q angegeben und in der Regel auf das kleinste Vielfache von 2, welches größer als N^2 ist, festgesetzt. Die Superposition wird wie folgt ausgedrückt:

$$\frac{1}{\sqrt{Q}} \sum_{P=0}^{Q-1} |r\rangle \, |f(r)\rangle$$

Die gesuchte Periode steckt, bezeichnet als p, in jedem evaluierten Exponenten und sorgt dafür, dass der Rest r konstant bleibt und sich lediglich das multiplikative m ändert:

$$g^{p+x} = m_0 * N + r$$
$$g^{2p+x} = m_1 * N + r$$
$$g^{3p+x} = m_2 * N + r$$
$$g^{np+x} = m_3 * N + r$$

Mit diesem Wissen lässt sich hierüber eine weitere Superposition bilden, die wie folgt aussieht:

$$|x\rangle + |p + x\rangle + |2p + x\rangle + ... |np + x\rangle$$

Aus dieser Superposition lässt sich über Anwendung der Quantenfouriertransformation die Periode aus der Frequenz ableiten. Da eine Superposition von Zahlen in die Quantenfouriertransformation eingespeist wird, welche alle die Periode p beinhalten, liefert die Quantenfouriertransformation schlussendlich nur ein Ergebnis, welches gemessen

werden kann. Falls dieses Ergebnis eine gerade Zahl ist, kann es in die Formel

$$g^{\frac{P}{2}} + -1$$

eingesetzt werden. Falls die beiden daraus berechneten Zahlen keine Vielfachen von N sind, besitzen sie garantiert je einen gemeinsamen Faktor mit N. Über Euklid lassen sich diese Faktoren ermitteln und somit kann letztendlich N faktorisiert werden.[11] [12] [13]

11[22]
12[23]
13[21]

4 Einschätzung des aktuellen Standes von Forschung und Entwicklung

Wie in Kapitel 3 gezeigt, ist der Shor-Algorithmus theoretisch in der Lage, beliebig große Zahlen zu faktorisieren, womit die RSA-Verschlüsselung in ihrer aktuellen Implementierung zu brechen ist. Im folgenden Kapitel wird der aktuelle technische Stand bei der praktischen Umsetzung des Algorithmus gezeigt, sowie die Schwierigkeiten theoretischer und praktischer Natur dargelegt, die dem Einsatz des Algorithmus noch im Weg stehen. Hierzu wird der Stand der Forschung sowie das Leistungsvermögen aktueller Quantencomputer vorgestellt. Abschließend werden die bisher erzielten Ergebnisse kritisch eingeordnet.

4.1 Herausforderungen

Generell stellt die Anzahl an benötigten Qubits eine große Herausforderung bei der Umsetzung von Algorithmen auf Quantencomputern dar. In der Theorie arbeitet man mit sogenannten logischen Qubits. In der Praxis unterliegen die Qubits allerdings einem Rauschen, sodass hier zur Unterscheidung von physikalischen Qubits gesprochen wird. Als Rauschen werden Umwelteinflüsse bezeichnet, welche die Qubits beim Rechnen stören. Technisch ist es möglich, das Rauschen durch den Einsatz mehrerer physikalischer Qubits zur Darstellung eines logischen Qubits, auszugleichen. Hierbei wird die Information eines logischen Qubits auf viele physikalische Qubits aufgeteilt. Den aktuell besten Ansatz hierzu liefert die Surface-Code-Fehlerkorrektur [1], bei der ein logisches Qubit durch etwa 100 physikalische Qubits dargestellt wird.

Wie hoch diese technische Hürde ist, zeigt der Quantencomputer „Sycamore“ von Google, welcher über 53 logische Qubits verfügt. Mit „Sycamore“ konnte Google 2019 erstmalig die sogenannte Quantum-Supremacy beweisen. Das bedeutet, dass Quantencomputer nicht nur theoretisch, sondern auch in der Praxis schneller rechnen als klassische Computer.[2]

1 [10]
2 [5]

Ein weiteres Kriterium sind die sogenannten Toffoli-Gatter, die zur Fehlerkorrektur benötigt werden. Während ein klassischer Computer bei 10^{16} Operationen auf einen Rechenfehler kommt, liegt das Ziel für Quantencomputer bei einem Fehler auf 10^4 Rechenoperationen. Der Grund für die hohe Fehlerrate liegt in der Instabilität der Qubits.
Zur Fehlerkorrektur ist es erforderlich, die Anordnung der Bits am Eingang des Gatters nachträglich feststellen zu können. Das Toffoli-Gatter verfügt hierzu über 3 Eingänge und 3 Ausgänge, womit es reversibel ist und entsprechende Fehler aus der Quantenberechnung korrigieren kann. [3]

Entsprechend einer Empfehlung des BSI soll die RSA-Schlüsselänge im Jahre 2021 mindestens 2000 Bit betragen, vergleiche hierzu auch Kapitel 2 [4].
2019 untersuchten Forscher von Google und des Kungliga Tekniska högskolan (KTH) Royal Institute of Technology wie sich eine 2048 bit große RSA-Zahl in acht Stunden faktorisieren lassen könnte. Sie kamen zu dem Ergebnis, das mit aktuellen Methoden hierzu etwas 20 Millionen physikalische Qubits nötig wären. Diese Zahl liegt allerdings bereits deutlich unter den etwa 10^9 physikalischen Qubits, die noch 2012 für das selbe Problem veranschlagt wurden.[5] Bei Verlängerung der Laufzeit lässt sich hierbei die Anzahl der benötigten Qubits signifikant verringern, sodass man bei 100 Tagen Laufzeit nur noch 1 Million Qubits benötigen würde. [6]

Dem Gegenüber steht der aktuelle Entwicklungsstand bei Quantencomputern, welche aktuell maximal 128 Qubits zum Einsatz bringen können. Abbildung 4.1 liefert einen Überblick über die Leistungszunahme von Quantencomputern von 1998 bis 2020, sowie einen Überblick über die aktuell verfügbaren leistungsfähigsten Quantencomputer.[7]

4.2 Aktueller Stand der Forschung

Bereits 2001 konnte IBM eine erste Demonstration des Shor-Algorithmus auf einem Quantencomputer durchführen. Hierbei wurde die Zahl 15 erfolgreich faktorisiert. Weitere signifikante Steigerungen konnten allerdings bis heute nicht erzielt werden.
2019 versuchten Mirko Amico, Zain H. Saleem and Muir Kumph auf einem „ibmqx5“ von IBM mit 16 Qubits die Zahlen 15, 21 und 35 zu faktorisieren. Hierzu benötigten sie 5,6

3 [26]
4 [4]
5 [11]
6 [9]
7 [9]

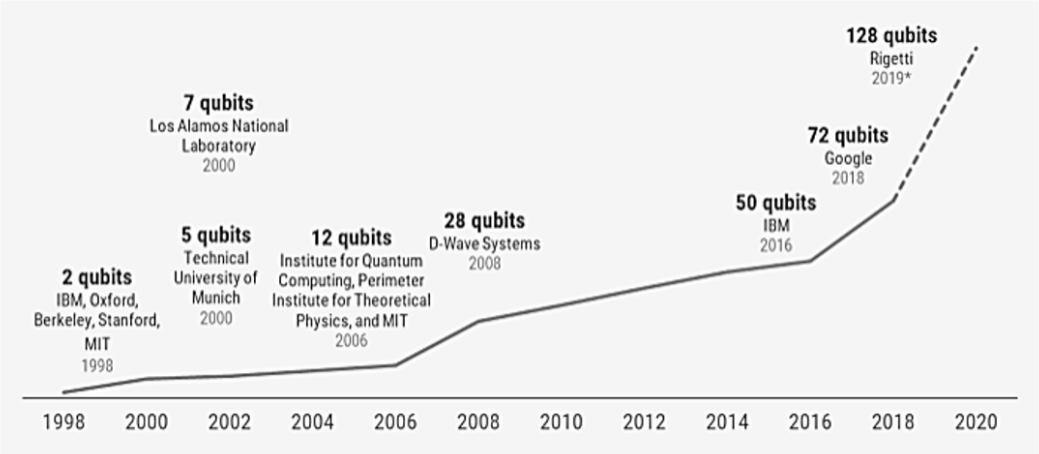

Abbildung 4.1 Übersicht der Leistungszunahme von Quantencomputern hinsichtlich verfügbarer Qubits von 1998 bis 2018 mit Abschätzung bis 2020, bearbeitete Abbildung [8]

beziehungsweise 7 Qubits. Allerdings misslang bereits die Faktorisierung von 35 aufgrund kumulierter Fehler in den Qubit-Gattern in 86% der 1000 Durchläufe für diese Zahl. [8]

Die geringe Anzahl an Qubits, die selbst in den aktuell modernsten Quantencomputern kontrolliert werden können, setzt der Realisierung sehr enge Grenzen. Die Faktorisierung größerer Zahlen scheitert aktuell bereits an der simplen Tatsache, dass diese nicht in den Speichern der Quantencomputern gehalten werden können.
Eine unoptimierte Version des Shor-Algorithmus braucht $3log_2(N)$ Qubits, womit bereits für die Zahl 15 12 Qubits benötigt werden würden. Mit Hilfe verschiedener Optimierungen ist es möglich die Anzahl auf 2n + 3 Qubits zu reduzieren.[9]

Die Forschungsgemeinschaft ist somit bis heute darauf fokussiert die theoretischen Grundlagen weiter zu erforschen und zu optimieren. Hierbei wird vor allem versucht, die Anzahl an benötigten Qubits zu reduzieren. Da jede Operation auf einem Qubit extrem fehleranfällig ist, steigt mit zunehmender Anzahl an Qubits auch die Anzahl benötigter Toffoli-Gatter zur Fehlerkorrektur. Tabelle 4.1 zeigt das Verhältnis von Qubits zu Toffoli-Gattern in Abhängig der RSA-Schlüssellänge in Bits.
Eine Übersicht zur Standortbestimmung kann Abbildung 4.2 entnommen werden. Der gelb markierte Bereich entspricht hierbei dem Stand der aktuellen Forschung, abgeleitet von einzelnen Experimenten auf den in der Abbildung aufgeführten Quantencomputern. Die schwarz gefüllten Quadrate stehen hierbei für Experimente mit einer Laufzeit von einem Tag, die offenen Quadrate für Experimente mit einer Laufzeit von 100 Tagen.

8 [1]
9 [12]

Tabelle 4.1 Übersicht der zur Faktorisierung benötigten Qubits und Toffoli-Gatter zur Fehlerkorrektur in Abhängigkeit der Schlüssellänge N in Bits, [9]

Länge von N in Bits	Anzahl der Qubits	Anzahl der Toffoli-Gatter
1024	2050	$5,81 * 10^{11}$
2048	4098	$5,20 * 10^{12}$
3072	6146	$1,86 * 10^{13}$
7680	15362	$3,30 * 10^{14}$
15360	30722	$2,87 * 10^{15}$

Der blaue Kreis ordnet die, von Google 2019 erstmals bewiesene, Quanten-Supremacy ein. Siehe hierzu auch Kapitel 4.1.
Die Graphen repräsentieren die Anforderungen hinsichtlich physikalischer Qubits, multipliziert mit der Anzahl an benötigten Durchläufen für eine Fehlerkorrektur mit dem Surface-Code Verfahren für Faktorisierung. Orange dargestellt ist die Faktorisierung, blau die Berechnung des diskreten Logarithmus. Die zugrundeliegenden Anforderungen in Bit für die Faktorisierung von RSA sind 1024, 2048, 3072, 7680 und 15360 Bit, die für den diskreten Logarithmus 160, 224, 256, 384 und 521 Bit. Dies ist in die Abbildung mit aufgenommen worden, da ein entsprechender Algorithmus zur Berechnung des diskreten Logarithmus von Shor ebenfalls in der erwähnten Publikation aus Kapitel 1 beschrieben wird. [10]

4.3 Weitere Ansätze und Kritik an bisherigen Implementierungen

Neben dem Shor-Algorithmus gibt es noch weitere Ansätze, Zahlen mithilfe eines Quantencomputers zu faktorisieren.

Ein erfolgversprechender Ansatz ist der Versuch Faktorisierung als Optimierungsproblem anzugehen. So konnte im Jahr 2014 die Zahl 56153 mit nur vier Qubits faktorisiert werden. Dieser Ansatz unterliegt aktuell noch der Einschränkung, dass die zu faktorisierende Zahl nur aus 2 Primfaktoren bestehen darf.
Ein großer Vorteil zu Shor ist andererseits, dass die Umsetzung gelang, ohne das bereits weiteres Wissen über die Primfaktoren bekannt sein muss.[11]

10[25]
11[6]

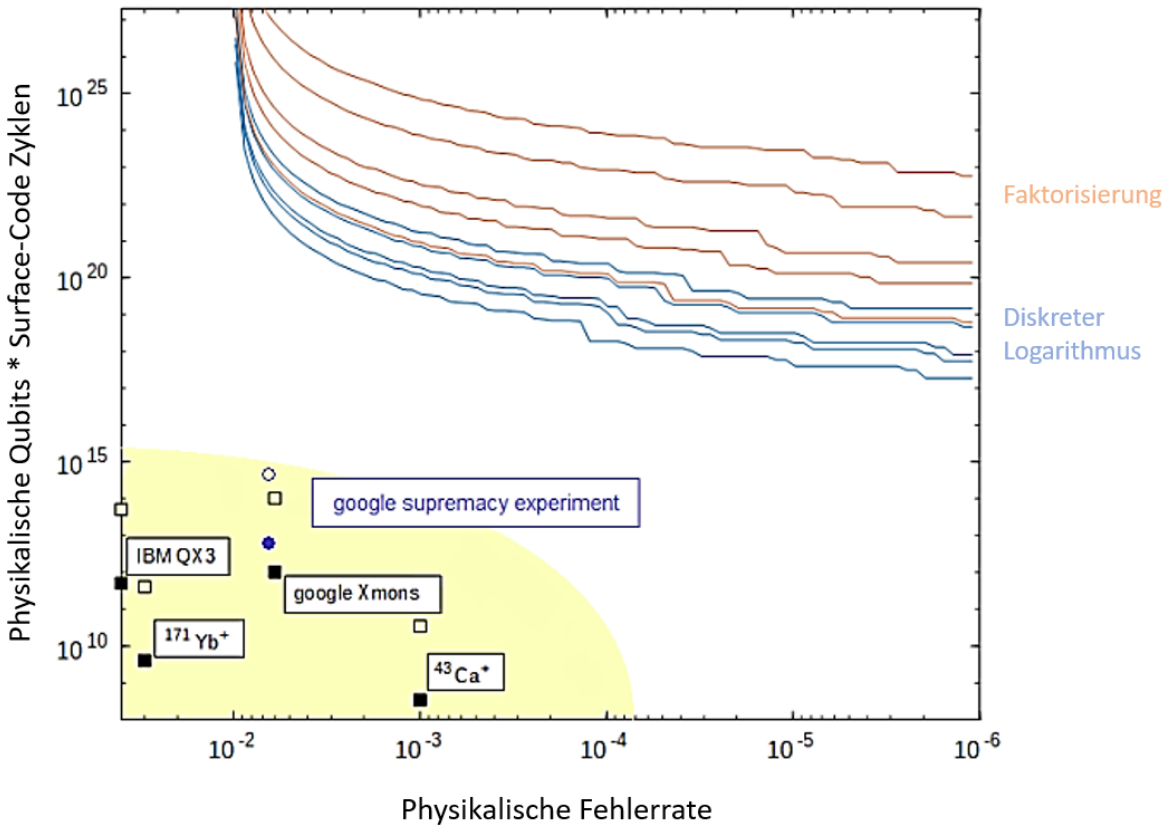

Abbildung 4.2 Übersicht zur Standortbestimmung aktueller Ereignisse im Bereich der Leistungsfähigkeit von Quantencomputern im Kontext von Faktorisierung und Berechnung des diskreten Logarithmus, bearbeitet Abbildung [9]

Tabelle 4.2 Übersicht der benötigten Qubits zur Faktorisierung in Abhängigkeit der Faktoren einer Zahl mit Fokus auf der Frage, ob die Faktoren für die Optimierung der Berechnung bereits bekannt sein müssen [28]

N	Anzahl der Faktoren	Anzahl der Qubits	Algorithmus	Jahr	Vorwissen
15	2	8	Shor	2001	nein
15	2	8	Shor	2007	nein
15	2	8	Shor	209	nein
15	2	8	Shor	2012	nein
15	2	8	Shor	2012	nein
21	2	10	Shor	2012	nein
143	2	4	Optimierung	2012	ja
56153	2	4	Optimierung	2012	ja

Alle erfolgreichen Demonstrationen des Shor-Algorithmus verwendeten bisher eine sogenannte „kompilierte“ Version des Algorithmus. Eine entsprechende Übersicht kann Tabelle 4.2 entnommen werden.[12]
Das bedeutet, dass vorab gewisse Optimierungen durchgeführt werden konnten, da das Ergebnis der Berechnung bereits bekannt war. Konkret wurde jeweils die Basis, die Zahl g aus Kapitel 3.2, so gewählt, dass nur kurze Perioden, welche weniger Qubits benötigen, vorhanden sind.[13].
Entsprechend wird stellenweise vorgeschlagen, in Zukunft die Schwierigkeit der Implementierung nicht alleine von der Größe der zu faktorisierenden Zahl abhängig zu machen, sondern vielmehr von der Länge der gefundenen Periode.[14]

4.4 Zukunft, Post-Quanten-Kryptographie

Ausgehend vom aktuellen Stand, sind Quantencomputer, die den heutigen Stand an asymmetrischer Verschlüsselung brechen können, noch in weiter Ferne. Die Abbildung 4.1 lässt allerdings bereits erahnen, dass die Leistungsfähigkeit von Quantencomputern stark zunimmt.
Hier muss vor allem berücksichtigt werden, dass die aktuelle Entwicklung bisher maßgeblich aus dem akademischen Umfeld vorangetrieben wurde. Sobald Staaten wie USA und China, Organisationen wie die EU und globale Konzerne wie Google, Microsoft oder IBM beginnen, wesentliche Ressourcen in die Entwicklung von Quantencomputern zu stecken, kann davon ausgegangen werden, dass in kurzer Zeit große Leistungssprünge

12[28]
13[27]
14[27]

erzielt werden.
So hat beispielsweise das kalifornische Start-Up Rigetti Computing im Oktober 2021 angekündigt, bis 2024 eine Skalierung ihres Quantencomputers auf 1000 Qubits und bis 2026 sogar 4000 Qubits zu erreichen.[15]

Klassische Kryptosysteme wie RSA, deren grundsätzliches Design angreifbar ist, werden dann keinen Schutz mehr bieten können, sodass neue Algorithmen nötig sind.[16] Ein Abgleich mit den Berechnungen des BSI aus Tabelle 4.1 zeigt, dass Rigetti bereits in wenigen Jahren in der Lage sein könnte, aktuell gängige RSA-Schlüssel zu faktorisieren.

2009 hat das us-amerikanische National Institute for Standards and Technology (NIST) in Zusammenarbeit mit anderen Instituten wie International Organization for Standardization (ISO) mit der koordinierten Forschung und Entwicklung im Bereich der sogenannten „Quantenresistenten Public-Key Kryptosyteme“ begonnen. [17]
Die Arbeit zielt hierbei unter Anderem darauf ab, bereits existierende Lösungen für den praktischen Einsatz im Alltag zu optimieren. 2016 wurde die erste Runde zur Standardisierung von entsprechenden Nachfolgern im Bereich digitaler Signaturen, sowie asymmetrischer Verschlüsselung eröffnet.[18]
Diese neuen Kryptosysteme umfassen beispielsweise

- Gitterbasierte Algorithmen (Lattice-Based), wie zum Beispiel das NTRU-Kryptosystem
- Code-basierte Algorithem, wie dem McElice-Kryptosystem
- Multivariate Polynomen, wie dem sogenannten „Unbalanced Oil-Vinegar-Verfahren“

Zum aktuellen Zeitpunkt gelten diese Verfahren als sicher, sowohl gegen klassische Computer, als auch gegen Quantencomputer, da keine Möglichkeit gefunden wurde, den Shor-Algorithmus entsprechend anzuwenden. [19]

15[18]
16[9]
17[16]
18[16]
19[24]

5 Fazit

Da bis heute nicht mathematisch bewiesen werden konnte, dass Faktorisierung ein „schweres“ Problem ist, war es nur eine Frage der Zeit, bis ein Algorithmus entwickelt wird, der darauf basierende Kryptosyteme wie RSA brechen kann. Ein solcher Algorithmus wurde von Peter W. Shor im Jahr 1996 vorgestellt und setzt den Einsatz von Quantencomputern voraus.

Der Shor-Algorithmus nutzt hierbei den Ansatz Faktorisierung auf das Problem der Periodenfindung zu reduzieren. Das dies theoretisch möglich ist, wurde bereits 1976 publiziert.
Dieser Schritt ist nötig, da Faktorisierung auch für Quantencomputer keine triviale Aufgabe ist. Der zugrunde liegende mathematische Prozess zur Periodenfindung ist dabei ähnlich einfach wie das Faktorisieren von „kleinen“ Zahlen. Bei Zahlen, die größer, also länger werden, kann man sich die Möglichkeit von Quantencomputern, Funktionen parallel an allen möglichen Punkten zu evaluieren, zunutze machen. Durch das anschließende Anwenden der Quantenfouriertransformation kann die Periode aus dem Quantencomputer ausgelesen und in die klassische Welt übermittelt werden.

Der Shor-Algorithmus unterbietet dabei die Laufzeit der besten klassischen Algorithmen wie dem Zahlkörpersieb deutlich. Auch eine Verlängerung der eingesetzten Schlüssel, also eine Vergrößerung der zu faktorisierenden Zahl bedeutet keine weitere Zunahme an Sicherheit mehr.

Der Grund, warum noch keine neuen Verschlüsselungsalgorithmen, die auch von Quantencomputern nicht geknackt werden können, weite Verbreitung erfahren haben, liegt darin, dass bisher keine ausreichend leistungsfähigen Quantencomputer verfügbar sind.
Während für das Brechen der aktuell empfohlenen mindestens 2000 Bit langen Schlüssel etwa 20 Millionen Qubits veranschlagt werden, kommen die besten Quantencomputer auf gerade einmal 100 Qubit.
Eine einfache Skalierung der vorhandenen Technik hat sich als große Herausforderung

erwiesen. Insbesondere die nötige Fehlerkorrektur innerhalb der Berechnung muss von der Forschung verbessert werden.

Dies führt dazu, dass selbst 20 Jahre nach der Vorstellung des Shor-Algorithmus, die größte zu faktorisierende Zahl 21 ist. Und selbst dieser Erfolg konnte nur erzielt werden, da aufgrund der bereits bekannten Faktoren gewisse optimierende Maßnahmen hinsichtlich der Periode vorgenommen werden konnten.
Aber auch alternative Ansätze, neben dem Shor-Algorithmus, liegen bis heute weit hinter den Erfolgen, die mit klassischen Computern und Algorithmen, wie dem Zahlkörpersieb, erzielt werden konnte.

Wurde die bisherige Forschung zunächst vor allem von akademischer Seite vorangetrieben, lässt sich seit einiger Zeit beobachten, dass große Unternehmen wie Microsoft, Google und IBM, sowie neue kleinere Unternehmen, wie D-Wave oder Rigetti Computing, ein gesteigertes Interesse im Bereich der Quantencomputer zeigen. Auch Staaten wie die USA oder China und Organisationen wie die EU steigern zunehmend die eingesetzten Ressourcen zur Erforschung und Weiterentwicklung von Quantencomputern.

Entsprechend wurde begonnen eine neue Klasse von Verschlüsselungsalgorithmen, die neben dem Einsatz von klassischen Computern auch den Angriffen von Quantencomputern standhalten sollen, zu standardisieren. So soll der Wechsel von RSA und Anderen zu den neuen Verfahren vorbereitet werden, damit die Welt nicht unvorbereitet ist, wenn ein ausreichend leistungsfähiger Quantencomputer realisiert worden ist.

Literatur

[1] Mirko Amico, Zain H. Saleem und Muir Kumph. “Experimental study of Shor’s factoring algorithm using the IBM Q Experience”. In: *Physical Review A* 100.1 (2019). ISSN: 2469-9926. DOI: 10.1103/PhysRevA.100.012305. URL: https://arxiv.org/pdf/1903.00768.pdf (besucht am 22. 12. 2021).

[2] Daniel J. Bernstein u. a. “Post-quantum RSA”. In: *Post-quantum cryptography*. Hrsg. von Tanja Lange und Tsuyoshi Takagi. Bd. 10346. Lecture Notes in Computer Science. Cham: Springer, 2017, S. 311–329. ISBN: 978-3-319-59878-9. DOI: 10.1007/978-3-319-59879-6{\textunderscore}18. URL: https://eprint.iacr.org/2017/351.pdf (besucht am 30. 10. 2021).

[3] Max Born. “Zur Quantenmechanik der Stoßvorgänge”. In: *Zeitschrift für Physik* 37.12 (1926), S. 863–867. DOI: 10.1007/BF01397477. URL: https://link.springer.com/article/10.1007%2FBF01397477.

[4] Bundesamt für Sicherheit in der Informationstechnik. “Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Version 2021-01”. In: 2021 (). URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile (besucht am 30. 10. 2021).

[5] Davide Castelvecchi. “Quantum-computing pioneer warns of complacency over Internet security”. In: *Nature* 587.7833 (2020), S. 189. DOI: 10.1038/d41586-020-03068-9. URL: https://www.nature.com/articles/d41586-020-03068-9.

[6] Nikesh S. Dattani und Nathaniel Bryans. *Quantum factorization of 56153 with only 4 qubits*. URL: https://arxiv.org/pdf/1411.6758.pdf (besucht am 20. 12. 2021).

[7] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, Paul Zimmermann. *Factorization of RSA-250*. 2020. URL: https://caramba.loria.fr/rsa250.txt (besucht am 22. 12. 2021).

[8] Faisal Khan. “Wie wird Moores Gesetz im Zeitalter des Quantum Computing irrelevant?” In: (1.12.2020). URL: https://ichi.pro/de/wie-wird-moores-gesetz-im-zeitalter-des-quantum-computing-irrelevant-101065848192001 (besucht am 22. 12. 2021).

[9] Federal Office for Information Security. “Status of quantum computer development: Entwicklungsstand Quantencomputer”. In: 2020.283 (). URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Quantencomputer/P283_QC_Studie-V_1_2.pdf?__blob=publicationFile&v=1 (besucht am 19. 12. 2021).

[10] Austin G. Fowler, David S. Wang und Lloyd C. L. Hollenberg. “Surface code quantum error correction incorporating accurate error propagation”. In: *Quant. Info. Comput. 11* (2011). URL: https://arxiv.org/pdf/1004.0255.pdf (besucht am 22. 12. 2021).

[11] Craig Gidney und Martin Ekerå. “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”. In: *Quantum* 5 (2021), S. 433. ISSN: 2521-327X. DOI: 10.22331/q-2021-04-15-433. URL: https://arxiv.org/pdf/1905.09749.pdf (besucht am 21. 12. 2021).

[12] Thomas Häner, Martin Roetteler und Krysta M. Svore. “Factoring using 2n+2 qubits with Toffoli based modular multiplication”. In: *Quantum Information and Computation* (2017). ISSN: 15337146. URL: https://arxiv.org/pdf/1611.07995.pdf (besucht am 22. 12. 2021).

[13] IBM Quantum. *Shor’s algorithm - IBM Quantum: classical vs. quantum factoring algorithms*. 2021-12-22. URL: https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm.

[14] Matthias Matting. “Geschichte des Quantencomputings: Das ganze Universum ist ein Quantencomputer - Golem.de”. In: *Golem.de* (3.05.2017). URL: https://www.golem.de/news/geschichte-des-quantencomputings-das-ganze-universum-ist-ein-quantencomputer-1705-127600.html (besucht am 15. 01. 2022).

[15] Gary L. Miller. “Riemann’s hypothesis and tests for primality”. In: *Journal of Computer and System Sciences* 13.3 (1976), S. 300–317. ISSN: 00220000. DOI: 10.1016/S0022-0000(76)80043-8.

[16] Moody, Dustin. “Cryptography in a Post-Quantum World”. In: (). URL: https://site1.auth.aps.commonspotcloud.com/units/maspg/meetings/upload/moody-05202020.pdf (besucht am 21. 12. 2021).

[17] Carl Pomerance. “A Tale of Two Sieves”. In: *Biscuits of number theory*. Hrsg. von Arthur Benjamin und Ezra Brown. The Dolciani mathematical expositions. Washington, DC: Math. Association of America, 2009, S. 85–104. ISBN:

9780883853405. DOI: 10.1090/dol/034/15. URL: http://www.ams.org/notices/199612/pomerance.pdf (besucht am 07. 11. 2021).

[18] *Rigetti Announces SPAC Deal | Rigetti Computing*. 12.01.2022. URL: https://www.rigetti.com/merger-announcement (besucht am 16. 01. 2022).

[19] Ryan Larose. "Shor's Algorithm Part 1: QuIC Seminar 12". In: (). URL: https://www.ryanlarose.com/uploads/1/1/5/8/115879647/shor1.pdf (besucht am 22. 12. 2021).

[20] Bruce Schneier. *Applied cryptography: Protocols, algorithms, and source code in C*. 20th anniversary edition. Indianapolis, IN: John Wiley & Sons, 2015. ISBN: 978-1-119-09672-6.

[21] Scott Aaronson. "RSA and Shor's Algorithm: Lecture 19". In: (). URL: https://www.scottaaronson.com/qclec/19.pdf (besucht am 23. 10. 2021).

[22] Scott Aaronson. *Shor, I'll do it*. 2007. URL: https://scottaaronson.blog/?p=208 (besucht am 09. 11. 2021).

[23] Scott Aaronson. "Shor, Quantum Fourier Transform: Lecture 20". In: (). URL: https://www.scottaaronson.com/qclec/20.pdf (besucht am 22. 12. 2021).

[24] Kent Seamons. *Proceedings of the 8th Symposium on Identity and Trust on the Internet*. ACM Other conferences. New York, NY: ACM, 2009. ISBN: 9781605584744. DOI: 10.1145/1527017. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901595 (besucht am 22. 12. 2021).

[25] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM J.Sci.Statist.Comput. 26 (* (). URL: https://arxiv.org/pdf/quant-ph/9508027.

[26] Simone Ulmer. *Fehler im Quantencomputing eliminieren*. 2011. URL: https://www.ethlife.ethz.ch/archive_articles/111215_Toffoli_Gatter_su/ (besucht am 22. 12. 2021).

[27] John A. Smolin, Graeme Smith und Alexander Vargo. "Oversimplifying quantum factoring". In: *Nature* 499.7457 (2013), S. 163–165. DOI: 10.1038/nature12290. URL: https://arxiv.org/pdf/1301.7007.pdf (besucht am 22. 12. 2021).

[28] Lisa Zyga. "New largest number factored on a quantum device is 56,153". In: *Phys.org* (28.11.2014). URL: https://phys.org/news/2014-11-largest-factored-quantum-device.html (besucht am 20. 12. 2021).