Post Quanten Kryptographie

Warum wir neue Verschlüsselungsalgorithmen brauchen

Thomas Jakkel
MatNr. 1001594
01jath1bif@hft-stuttgart.de

Lukas Reinke
MatNr. 1001213
01relu1bif@hft-stuttgart.de

***Zusammenfassung*—Quantenkryptographie wird als ernsthafte Gefahr für die aktuell gängigen Kryptographie-Verfahren gesehen. Speziell die, für das sichere Funktionieren der Netzwerk-Kommunikation benötigten, asymmetrischen Kryptographie-Algorithmen wie beispielsweise *RSA* stehen in der Gefahr mithilfe eines Quantencomputers innerhalb von Stunden gebrochen zu werden. Ein solcher Computer birgt somit eine Gefahr für die gesamte heutige Informationssicherheit.**

Schon 1994 bewies der amerikanische Mathematiker *Peter Shor* in der Theorie, wie mithilfe eines Quantencomputers die Primfaktorzerlegung großer Zahlen in realer Zeit erfolgen kann. Mit einem klassischen, binären Computer wäre dies nicht in der Lebenszeit des Universums möglich, was zentrale Sicherheits-Prinzip asymmetrischer Kryptographie darstellt. Wegen dieser Bedenken hat das US National Institute of Standards and Technology (NIST) seit 2016 eine Ausschreibung zur Entwicklung eines quantensicheren Algorithmus aufgestellt. Der aktuelle Favorit, ein Verfahren auf basis mehrdimensionaler Vektorfelder, soll, soweit sich keine Schwachstellen herausstellen, schon in den nächsten Jahren in der Netzwerk Kryptographie etabliert werden und somit die Kommunikation schon vor der Existenz eines potentiellen Quantencomputers absichern. Es ist also gut möglich, dass wir die aktuellen Kryptographie Standards innerhalb der nächsten 5 Jahre austauschen werden.

ABBILDUNGSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

QC Quantencomputer
ggT größter gemeinsamer Teiler
QFT Quanten-Fourier-Transformation
NIST National Institute of Standards and Technology

I. EINLEITUNG

Das Thema der *Quanten Supremacy*, der Zeitpunkt zu dem ein Quantencomputer (QC) die Fähigkeit besitzt komplexe Probleme besser zu lösen als ein klassischer Computer, ist in den letzten Jahren immer wieder in einschlägigen Medien und diverse Fachpublikationen aufgetaucht. Google zum Beispiel behauptete schon wiederholt einen solchen QC zu besitzen, was jedoch in diversen Publikationen bezweifelt wurde **cho˙ordinary˙2022**. Wenig Zweifel besteht jedoch, dass ein solcher Computer in absehbarer Zeit einsatzbereit sein wird.
Im Folgenden wird beschieden wie ein QC funktioniert, mit welchen Algorithmen, die in realer Zeit laufen, QC die aktuellen, kryptographischen Verfahren brechen können werden. Außerdem werden wir aktuelle, neu Entwickelte quantensichere Algorithmen betrachten und wie sie sie Gefahr durch QC mittigern werden.

II. SHORS-ALGORITHMUS

Die Faktorisierung großer Zahlen [1, S189] ist ein fundamentales mathematisches Problem, bei dem eine gegebene Zahl in ihre Primfaktoren zerlegt wird. Dieses Problem ist von zentraler Bedeutung für die Kryptographie, da viele asymmetrische Verschlüsselungsverfahren, wie beispielsweise RSA, auf der Schwierigkeit der Faktorisierung großer Zahlen beruhen.

Das Problem der Primfaktorzerlegung besteht darin, eine öffentliche Zahl N in zwei oder mehr

geheime Primzahlen p und q zu zerlegen, sodass: $N = p * q$. Für kleine Zahlen kann die Faktorisierung durch Ausprobieren möglicher Primfaktoren relativ einfach sein. Allerdings wird die Faktorisierung bei großen Zahlen exponentiell schwieriger, da es keine effizienten klassischen Algorithmen gibt, welche dies in Polynomialzeit bewältigen können.

Die Sicherheit vieler asymmetrischer Verschlüsselungsverfahren, wie beispielsweise das RSA-Verfahren, beruht auf der Schwierigkeit der Faktorisierung großer Zahlen. Wenn ein Angreifer in der Lage wäre, die Primfaktoren einer Zahl zu finden, könnte er den geheimen Schlüssel einer Verschlüsselungsmethode berechnen und die Sicherheit des Systems kompromittieren.

Shors Algorithmus [2] gilt als einer der bedeutendsten Quantenalgorithmen, der die Faktorisierung großer Zahlen in Polynomialzeit ermöglicht. Auch wenn dieser Algorithmus einen Quantencomputer mit vielen, stabilen Qubits voraussetzt, stellt diese revolutionäre Entdeckung bereits heute eine enorme Bedrohung für die moderne Kryptographie dar. Zum einen wird effektiv an QC geforscht, die Anzahl stabiler Qubits in QC steigen. Wenn der Quantencomputer seinen Durchbruch in unserer Gsellschaft erreicht hat, müssen wir darauf vorbereitet sein und uns bereits heute mit den Konsequenzen auseinander setzen. Zum anderen werden verschlüsselten Daten bereits heute gespeichert um sie in der Zukunft entschlüsseln zu können und im Nachinein an wichtige, vertrauliche Informationen zu gelangen. In der Kryptographie spricht man von dem Prinzip: „Safe now, decrypt later“.

Im folgenden [3][4, S. 4-5] werden wir an einem vereinfachten Beispiel näher aufzeigen, wie Shors Algorithmus eine effiziente Lösung für die Faktorisierung großer Zahlen auf einem Quantencomputer bietet. Zunächst betrachten wir die öffentliche Zahl N, welche in ihre Primfaktoren p und q zerlegt werden soll. Für das vereinfachte Beispiel wählen wir $N = 77$. Also:

$$N = p * q \text{ mit } N = 77$$

Beachte das N in Realität eine riesige Zahl ist. Das vereinfachte Beispiel soll ledeglich den Prozess von Shors Algorithmus aufzeigen

Nun stellen wir eine zweite Gleichung

$$g^r = mN + 1$$

auf, welche behauptet, dass man immer einen Exponenten r finden kann, sodass ein Vielfaches einer zufälligen Zahl $g < N$ gleich einem vielfachen m der Zahl $N + 1$ ist. Wir wählen zufällig die Zahl $g = 8$ und erhalten aus 1 $r = 10$

g^x	$g^x/77$	Rest
g^0	0	1
g^1	0	8
g^2	0	64
g^3	6	50
g^4	53	15
g^5	425	43
g^6	3404	36
g^7	27235	57
g^8	217885	71
g^9	1733087	29
g^{10}	13944699	1

Abbildung 1. Zufallszahl g^x mit Rest

Wie extrahiert man aus der Funktion $g^{10} = mN + 1$ die Primfaktoren p, q? Hierfür schreiben wir unsere ursprüngliche Gleichung um und erhalten:

$$\begin{aligned} & g^r = mN + 1 \\ \Leftrightarrow\ & g^r - 1 = mN \\ \Leftrightarrow\ & g^r - 1 = mN \\ \Leftrightarrow\ & (g^{r/2} + 1)(g^{r/2} - 1) = mN \end{aligned}$$

Mit $r = 10$ erhalten wir:

$$\begin{aligned} (g^{r/2} + 1) = (8^5 + 1) \qquad &= 32.769 \\ (g^{r/2} - 1) = (8^5 - 1) \qquad &= 32.767 \end{aligned}$$

Damit haben wir aus einer geschätzten Zahl $g = 8$ zwei Zahlen gefunden, welche vermutlich Faktoren mit N teilen. Um diese Faktoren zu finden,

wenden wir den euklidischen Algorithmus an, um den größter gemeinsamer Teiler (ggT) zu finden:

$$\begin{aligned} 32.769/77 &= 425R44 \\ 77/44 &= 1R33 \\ 44/33 &= 1R11 \\ 33/11 &= 3R0 \\ \Rightarrow\ ggT(32.769, 77) &= 11 \end{aligned}$$

Wir erhalten somit als ersten Faktor $p = 11$. Für den zweiten Faktor kann der selbe Prozess nochmals mit der zweiten Zahl berechnet werden, oder wir erhalten durch $q = N/p \Leftrightarrow q = 7$. Damit haben wir erfolgreich die Zahl $N = 77$ in ihre Primfaktoren $p = 11$ und $q = 7$ zerteilt.

Nachfolgend fassen wir nochmal die Schritte zusammen, wie man eine eine öffentliche Zahl N, ein Produkt aus zwei Primzahlen faktorisiert.

1) Schätze eine Zahl $g < N$
2) Finde einen Exponenten r sodass $g^r = mN + 1$
3) Berechne $(g^{r/2} + 1)$ und $(g^{r/2} - 1)$
4) Nutze den euklidischen Algorithmus um den größter gemeinsamer Teiler (ggT) zu finden

Um diesen Algorithmus auszuführen bräuchte es noch keinen QC, jedoch wäre diese Methode auf einem klassischen Computer nicht schneller, als herkömmliche Methoden. Der Kernprozess, welcher ein QC beschleunigt ist Schritt 2. Um dies zu verstehen, betrachten wir 1 nochmal genauer. Würde man die Tabelle fortführen, würde sich der Restanteil der Zahlen stets wiederholen. In unserem obrigen Beispiel mit der Periode 10, wie sich aus $g^0 Rest1$ und $g^{10} Rest1$ erkennen lässt. Dies gilt ebenfalls für die anderen Zahlen, beispielsweise würde $8^{13} Rest50$ haben. Dies bedeutet, dass sich Schritt 2 beschleunigen lässt indem man die Periode findet, mit der sich der Restanteil wiederholt. An dieser Stelle setzt der Quantenpart ein.

Für die Quantenberechnung [5, S.1-2] werden zwei Quantenregister benötigt. Ein Periodenregister n_p, um den Wert der Periode zu speichern und ein Ergebnisregister n_q um das Ergebnis der Berechnung zu speichern. Die Größe der beiden Register hängt von der Zahl N ab, welche faktorisiert werden soll. Das Periodenregister benötigt eine Anzahl von Qubits n_p, von etwa $\log_2(N^2) \leq n_p \leq log_2(2N^2)$. Das Ergebnisregister muss ledeglich groß genug sein um die Zahl $N - 1$ zu speichern, was $n_q = \log_2(N)$ Qubits benötigt. Zu Beginn wird das Periodenregister fortlaufend von 1 initialisiert:

$$|n_p\rangle = |0\rangle + |1\rangle + |2\rangle + ... + \left|10^{1234}\right\rangle$$

Diese Zahlen stellen den Exponenten r dar. Das Ergebnisregister wird vorerst mit 0 initialisiert:

$$|n_q\rangle = |0\rangle + |0\rangle + |0\rangle + ... + |0\rangle$$

Nun nehmen wir unsere geschätzte Zahl $g < N$ und potenzieren diese mit der Menge des Periodenregisters n_p, teilen diese durch die Zahl N und speichern den Rest R im Ergebnisregister n_q.

$$\left|\frac{g^0}{N}\right\rangle, \left|\frac{g^1}{N}\right\rangle, \left|\frac{g^2}{N}\right\rangle, \ ...$$
$$|R_0\rangle, \ |R_1\rangle, \ |R_2\rangle, \ ...$$

Im jetzigen Stadium haben wir das unveränderte Periodenregister n_p, welches die Zahlen r der Folge $1, 2, 3...$ beinhaltet und das Ergebnisregister n_q in welchem nun der Restanteil R von $\frac{g^r}{N}$ liegt.

$$|0\rangle |R_0\rangle + \ |1\rangle |R_1\rangle + \ |2\rangle |R_2\rangle + \ ...$$

Die Herausforderung besteht nun darin aus der Superposition von allen möglichen Lösungen, die eine richtige zu extrahieren. Der Trick ligt darin sich nur den Rest anzuschauen. Betrachten wir nochmal unser obriges Beispiel. Sobald man eine mögliche Lösung misst, z.B. aus 1 $R = 15.$ wird dieser Rest periodisch erneut auftauchen, nämlich

$$|4\rangle |R_{15}\rangle + \ |14\rangle |R_{15}\rangle + \ |24\rangle |R_{15}\rangle + \ ...$$

Nachdem man also einen Restanteil misst, erhält man eine Menge von Superposition, welche den selben Restanteil besitzen. Und die Exponenten sind genau um die Periode r voneinander getrennt.

$$|i\rangle\,|R\rangle + \ |i+r\rangle\,|R\rangle + \ |i+2r\rangle\,|R\rangle + \ ...$$

Da der Restanteil hier immer der Gleiche ist, kann dieser vernachläsigt werden und man erhält somit eine periodische Superposition.

$$|i\rangle + \ |i+r\rangle + \ |i+2r\rangle + \ ...$$

Im letzten Schritt kann die Quanten-Fourier-Transformation (QFT) angewendet werden, welche es erlaubt die Frequenz einer periodischen Superposition zu ermitteln. [5, S. 2] Wenden wir also auf die obrige periodische Superposition die QFT an, können wir aus dem Ergebnis r auslesen.

III. Lattice-Kryptographie

Im Jahr 2016 hat das US-amerikanische National Institute of Standards and Technology (NIST) einen weltweit einheitlichen Standardisierungsprozess für Post-Quantum-Kryptografie gestartet [6]. 2022 wurden mehrere Algorithmen Ausgewählt. Die meisten dieser Algorithmen basieren auf mathematischen Gitter- (*engl. Lattice*) Problemen.

Lattice-Kryptographie [7] ist ein Bereich der post-quanten Kryptographie. Mit dieser werden Verschlüsselungsalgorithmen auf der Grundlage mathematischer Gitterstrukturen entwickelt. Ein Gitter ist eine diskrete, periodische Anordnung von Punkten im Raum. In der Kryptographie werden meistens Gitter in mehrdimensionalen Vektorräumen verwendet. Die Gitterpunkte werden durch Vektoren repräsentiert, die eine lineare Kombination von Basisvektoren des Gitters sind. Ein Beispiel für ein zweidimensionales Gitter ist die Ebene, in der die Basisvektoren (1, 0) und (0, 1) sind, und die Gitterpunkte alle ganzzahligen Kombinationen dieser Basisvektoren. Ein grundlegendes Problem in der Lattice-Kryptographie ist das Shortest Vector Problem (SVP), bei dem man den kürzesten Vektor in einem Gitter finden muss. Dieses Problem gilt als NP-vollständig. [8, Abs. 2.1]

Ein weiteres Problem ist das Learning With Errors (LWE), bei dem man die Lösung eines linearen Gleichungssystems mit fehlerbehafteten Variablen in einem Gitter finden muss. Die fehlerbehafteten Werte werden durch eine Fehlerfunktion zufällig erzeugt. Diese fügt den Gleichungen Rauschen hinzu und erschwert es, den geheimen Wert aus den Gleichungen zu erlernen.

Nachfolgend wird das LWE-Problem genauer erläutert. Gegeben sei ein Vektor s, der den geheimen Schlüssel darstellt, und ein Vektor e, der den Fehler darstellt. Das LWE-Problem besteht darin, den geheimen Vektor s aus einer Reihe von Gleichungen der Form

$$c = (a * s + e) \mod q$$

zu erlernen, wobei c der Ciphertext und a ein öffentlicher Vektor ist. Das Modulo q repräsentiert eine endliche Gruppe, in der die Rechenoperationen durchgeführt werden.

Um das LWE-Problem zu lösen, werden eine Reihe von Gleichungen erzeugt. Der geheime Vektor s und der Fehlervektor e werden zufällig generiert. Der öffentliche Vektor a wird ebenfalls zufällig gewählt. Für jeden Gitterpunkt i wird die Gleichung

$$c_i = (a * s_i + e_i) \mod q$$

erstellt.

Der Angreifer erhält eine Menge von Gleichungen, bestehend aus den Ciphertexts c_i und den öffentlichen Vektoren a_i. Das Ziel des Angreifers ist es, den geheimen Vektor s_i zu erlernen, basierend auf den Ciphertexts und den öffentlichen Vektoren. Dies erfordert die Bestimmung der Werte des geheimen Vektors s_i trotz des Vorhandenseins von Fehlern e_i.

Die Schwierigkeit des LWE-Problems beruht darin, die Werte des geheimen Vektors s_i aus den gegebenen Gleichungen zu rekonstruieren. Der Angreifer muss die richtigen Werte des geheimen Vektors von den fehlerbehafteten Gleichungen unterscheiden können.

Hallo

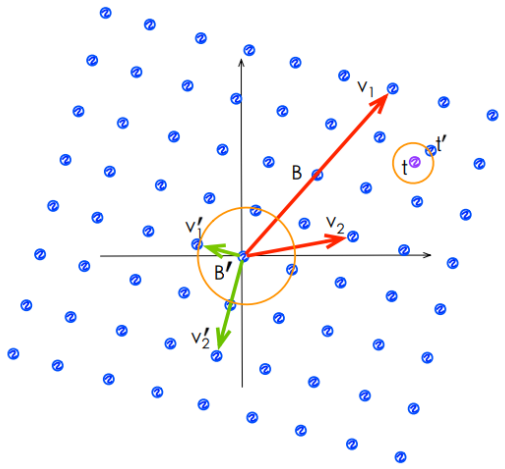

Abbildung 2. Darstellung eines 2D Gitters[9, Fig. 2]

LITERATURVERZEICHNIS

[1] D. J. Bernstein und T. Lange, „Post-quantum cryptography,“ *Nature*, Jg. 549, Nr. 7671, S. 188–194, Sep. 2017, ISSN: 1476-4687. DOI: 10 . 1038 / nature23461. (besucht am 16. 04. 2023).

[2] P. Shor, „Algorithms for Quantum Computation: Discrete Logarithms and Factoring,“ in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, S. 124–134. DOI: 10.1109/SFCS.1994.365700.

[3] A. Kak, „Lecture 12: Public-Key Cryptography and the RSA Algorithm Lecture Notes on “Computer and Network Security”,“

[4] C. Gidney und M. Ekerå, „How to Factor 2048 Bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits,“ *Quantum*, Jg. 5, S. 433, Apr. 2021, ISSN: 2521-327X. DOI: 10.22331/q-2021-04-15-433. arXiv: 1905.09749 [quant-ph]. (besucht am 16. 04. 2023).

[5] M. Amico, Z. H. Saleem und M. Kumph, „An Experimental Study of Shor’s Factoring Algorithm on IBM Q,“ *Physical Review A*, Jg. 100, Nr. 1, S. 012 305, Juli 2019, ISSN: 2469-9926, 2469-9934. DOI: 10 . 1103 / PhysRevA . 100 . 012305. arXiv: 1903 . 00768 [quant-ph]. (besucht am 16. 04. 2023).

[6] D. Moody, „Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process,“ National Institute of Standards and Technology, Gaithersburg, MD, Techn. Ber. NIST IR 8413-upd1, 2022, NIST IR 8413–upd1. DOI: 10.6028/NIST.IR.8413-upd1. (besucht am 16. 04. 2023).

[7] D. Micciancio und O. Regev, „Lattice-based Cryptography,“

[8] A. Wang, D. Xiao und Y. Yu, „Lattice-based cryptosystems in standardisation processes: A survey,“ *IET Information Security*, Jg. 17, Nr. 2, S. 227–243, 2023, ISSN: 1751-8717. DOI: 10 . 1049 / ise2 . 12101. (besucht am 10. 06. 2023).

[9] R. Xu, C. Cheng, Y. Qin und T. Jiang, *Lighting the Way to a Smart World: Lattice-Based Cryptography for Internet of Things*, Mai 2018. arXiv: 1805.04880 [cs]. (besucht am 10. 06. 2023).