PROSEMINAR:
PRIMZAHLEN UND FAKTORISIERUNG FÜR DIE KRYPTOGRAFIE

PROF. DR. G. BÖCKLE, K. FISCHER **WS 2016/17**

Motivation

Dass es unendlich viele Primzahlen gibt, ist lange bekannt, aber bereits das Problem ihre Verteilung in den natürlichen Zahlen genau zu quantifizieren ist ungelöst (und sogar zur berühmten Riemannschen Vermutung equivalent!).

Leichter zu beantworten ist die Frage ob eine gegebene Zahl n prim oder zusammengesetzt ist. Probedivision mit allen Zahlen kleiner n wird jedoch schnell unpraktikabel, wenn n wächst. Erst seit 2002 gibt es einen Algorithmus, der diese Frage immer schnell und zuverlässig beantwortet: Ist n eine beliebige D-stellige Zahl, so liefert das A-K-S-Verfahren nach 'circa' D^6-Rechenschritten eine Ja/Nein-Antwort. Ziel des Proseminars ist es hier zunächst den Begriff 'circa' zu formalisieren und die im Verfahren verwendeten Techniken der Restklassenarithmetik und schnellen Multiplikation vorzustellen.

Bevor wir dann den A-K-S Algorithmus im Detail besprechen, wenden wir uns zunächst dem leichteren Problem zu, die Primeigenschaft von n stochastisch zu prüfen, was in der Praxis ausreicht und extrem schnell geht; außerdem wird eine Methode vorgestellt, die n von der besonderen Form $n = 2^q - 1$ schnell testen kann.

Bei genauem Hinsehen hat Probedivision jedoch gegenüber obigen Tests auch einen entscheidenden Vorteil: Sie liefert für zusammengesetzte n nicht nur eine Ja/Nein-Antwort sondern einen echten Teiler. Auch für diese Fragestellung gibt es aber viel schnellere Methoden, namentlich Siebverfahren, auf die wir in der zweiten Hälfte des Seminars eingehen. Trotz aller Anstrengungen ist mit ihnen eine polynomielle Laufzeit (wie die $\sim D^6$ bei A-K-S) aber noch nicht erreicht worden.

Das lässt die Multiplikation zweier großer Primzahlen als geeigneten Kandidat für eine sogenannte 'Einwegfunktion' erscheinen: Leicht auszurechnen, schwierig umzukehren. Andere Kandidaten für solche Funktionen und ihre Anwendungen in der Kryptografie werden wir in den letzten beiden Vorträgen kennen lernen.

Literatur

[CP] R. Crandall and C. Pomerance, *Prime numbers – A computational perspective*, 2nd ed., Springer

[DH] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Information Theory, 1976 (**6**), pp. 644–654

[G] A. Granville, *It is easy to determine whether a given integer is prime*, Bull. AMS; **42**(1), 35p.

[HAC] A. Menezes, P. van Oorschot and Scott A. Vanstone, *Handbook of applied cryptography*, available at `cacr.uwaterloo.ca/hac/`

[M] D. Marcus, *Number fields*, Universitext, 3rd ed., 1995, Springer

[N] J. Neukirch, *Algebraische Zahlentheorie*, 1992, Springer

[P] C. Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, 1987, available at `math.dartmouth.edu/~carlp`

[S] P. W. Shor, *Polynomial-time algorithms for Prime factorization and discrete logarithms on a quantum computer*, 1996, available at `arxiv.org/abs/quant-ph/9508027`

[W] R. de Wolf, *Quantum Computation and Shor's Factoring Algorithm*, 1999, available at `homepages.cwi.nl/~rdewolf`

Vorträge

Wenn nichts anderes angegeben ist, beziehen sich Quellenangaben auf das Buch [CP] von Crandall und Pomerance[1].

Vortrag 1 (Komplexitätsanalyse)**.**

Wir erinnern zunächst an die Definition einer Primzahl. Ihre Bedeutung wird im Fundamentalsatz (1.1.1) deutlich. Wir wiederholen dann kurz die Restklassenarithmetik, insbesondere das Invertieren mit erw. eukl. Algo. Dann geben wir die Definition eines Quadratischen Rest mod n und des Legendre-/Jacobisymbols. Seine fundamentalen Eigenschaften inklusive Reziprozität (Thm. 2.3.4) werden erwähnt

[1] Dieses findet sich auch als pdf im Internet.

(aber nicht bewiesen). Geschickte Kombination dieser Relationen wird es erlauben das Jacobi-Symbol schnell zu berechnen. Dazu führen wir zunächst O/o – Notation (§1.1.4) ein. Dann untersuchen wir den erw.eukl.Algo. und 2.3.5 auf ihre Laufzeiten: Sie stellen sich als polynomiell heraus. Abschließend kann man einige theoretische Konzepte zu Primzahlen erwähnen: Die Def. von $\pi(x)$ nach §1.1.3, den Primzahlsatz 1.1.4 (o.Bew.), die Riemannsche ζ-Fkt und der Beweis des Eulerprodukts (1.4.1) für ζ sind einige Beispiele.

Vortrag 2 (Schnelle Arithmetik I: DFT)**.**

Wir setzen unsere Analyse von grundlegenden Algorithmen fort: Dazu wird zunächst nach §9 die (balanzierte-)Basis-B Darstellung einer Zahl definiert und wie in §9.1.1 abgeschätzt, wie komplex eine "Schulmultiplikation" ist. Ein schnelleres Verfahren benutzt Methoden der Signalanalyse (§9.5.2): Wir definieren die Diskrete Fourier Transformation und zeigen $\mathrm{DFT}^{-1} \circ \mathrm{DFT} = \mathbf{1}$ sowie Formel (9.22). Anschließend wird ein Divide-and-Conquer Algorithmus (FFT) für die Multiplikation zweier DFT-transformierter Zahlen analysiert (9.5.4).

Vortrag 3 (Schnelle Arithmetik II)**.**

Von der DFT des letzten Vortrags ausgehend gibt es eine ganze Reihe von Faltungsverfahren um schnell zu multiplizieren: Wir erwähnen einige (§9.5.8) und geben ihre Komplexität (im besten Fall $O(n \ln n \ln\ln n)$) an, ohne auf Details der recht aufwendigen Berechnungen einzugehen. Obwohl wir also schnelle Multiplikationen zur Vefügung haben, wollen wir natürlich vermeiden viele ausführen zu müssen um aus a und n etwa $x = a^n$ zu berechnen: Die grundlegende Technik der Leitern (§9.3.1) wird erläutert und ihre Laufzeit untersucht. Sie stellt sich als $O_{\mathrm{op}}(\log n)$ heraus, ist also exponentiell schneller als "Ausmultiplizieren". Umgekehrt kann aus einer perfekten Potenz x sehr schnell die entsprechende Wurzel a gezogen werden: Der Schlüssel hierzu ist eine Technik aus der Analysis, das Newton-Verfahren (9.2.11). Wir beweisen, dass es Laufzeit $O(\ln\ln x)$ hat. Es ist also seinerseits exponentiell schneller als potenzieren!

Vortrag 4 (Pseudoprimzahlen, Miller-Rabin)**.**

Wir wenden uns einigen zahlentheoretischen Grundlagen zu und definieren Fermat-Pseudoprimzahlen zu einer Basis a. Dass es unendlich viele solche gibt zeigt Satz 3.3.4., dessen elementaren Beweis wir kurz besprechen. Lassen wir a variieren, kommen wir zum Begriff der Carmichael-Zahlen. Dass es auch davon unendlich viele gibt, nehmen wir ohne Beweis hin (Satz 3.3.7). Es gib aber einen geeigneten Begriff von 'starker' Pseudoprimzahl, für den es keine 'starken' Carmichael-Zahlen gibt: Wir führen dazu den Begriff des 'Zeugen' und Satz 3.4.4 ein. Bevor wir zum Beweis kommen, stellen wir einige Anwendungen vor: Der Miller-Rabin- Test und 'Industrielle Primzahlen'. Unter Annahme der erweiterten Riemannschen Vermutung (ERH) ist ersterer ein sehr starker Primzahltest. Schließlich gehen wir auf den Beweis von 3.4.4 ein und, falls Zeit bleibt, auf die Beweise der Schlüssel-Lemmata 3.4.8/9

Vortrag 5 (Spezielle (Prim-)Zahlen)**.**

Ohne Annahme von ERH bleibt der Miller-Rabin-Test eine probabilistische Aussage. Für spezielle Zahlen, namentlich die Mersenne-Zahlen $2^q - 1$, gibt es aber sehr schnelle deterministische Tests. Wir gehen zuerst kurz auf diesen Unterschied ein, stellen die Mersenne-Zahlen und vielleicht einige Daten zu Primzahlrekorden vor (§1.3.1). Die kurzen Beweise von 1.3.1 und 1.3.2 machen bereits klar, dass die spezielle Form der Mersenne-Zahlen starke Implikationen hat. Der Hauptteil des Vortrags ist jedoch der Lucas-Lehmer-Test in §4.2: Wir definieren die (konstanten!) Polynome U_k, V_k wie in (3.8) und zeigen Kriterium 4.2.6 zunächst unter Annahme von 4.2.5. Der Beweis des letzteren sollte verstanden, aber vielleicht nicht in allen Details vorgeführt werden.

Vortrag 6 (Satz von Agrawal-Kayal-Saxena)**.**

Für den Satz von Agrawal-Kayal-Saxena (AKS) folgen wir dem Artikel [G]: Die Grundidee ist, dass wir viele Koeffizienten auf beiden Seiten von $(x + a)^n \equiv x^n + a^n \bmod n$ vergleichen. Theorem 1 (in §2.8, mit Beweis) besagt, dass alle Koeffizienten genau dann übereinstimmen, wenn n prim ist. Dann formulieren wir unser Ziel: Der Satz von A-K-S (S. 4), der massive Erleichterungen dieses Primzahlkriteriums bringt. Für den Beweis gehen wir anschließend durch Abschnitte §4 bis 4.2. Alle Rechnungen finden dabei in Restklassenringen $\mathbb{Z}[x]/(p, x^r - 1)$ statt.

Vortrag 7 (Laufzeitanalyse von A-K-S: 'Primes is in P').

Die Laufzeit des A-K-S Kriteriums aus dem letzten Vortrag hängt von einem Parameter r ab. Wie genau, das wollen in diesem Vortrag klären. Dazu gehen wir wiederum nach dem Artikel [G] vor. Die $\widetilde{O}(y)$-Notation aus §3B ist dabei eine nützliche Kurzschreibweise für $O(y(\log y)^{O(1)})$. Wir kombinieren im Folgenden die Laufzeitabschätzungen der ersten Vorträge (Multiplikation von Polynomen und Leitern $\mathrm{mod}(n, x^r - 1)$) zum folgenden Resultat: A-K-S ist $\widetilde{O}(r^{3/2}(\log n)^3)$. Eine naive Suche nach einem geeigneten r führt nach höchstens $\leq 2(\log n)^5$ Schritten zum Erfolg (Lemma 4.4 mit Beweis). Mit analytischer Zahlentheorie kann man für A-K-S sogar $\widetilde{O}(\log^6 n)$ zeigen, worauf wir aber nicht im Detail eingehen können.

Vortrag 8 (Faktorisierung I: Pollard-Rho, BSGS).

Wie bereits unter Motivation bemerkt, ist das Problem einen Faktor zu finden erwartungsgemäß deutlich rechenintensiver als ein reiner Test auf Zusammengesetztheit. In diesem Vortrag werden zunächst Verfahren mit exponentieller Laufzeit vorgestellt, die aber für kleine Zahlen durchaus ihre Berechtigung in der Praxis haben. Referenzen sind dabei auf das Buch [CP]. Zunächst diskutieren wir Pollards ρ-Verfahren nach §5.2.1 und erläutern was seine Laufzeit mit dem Geburtstagsparadoxon zu tun hat (Algorithmus 5.2.1). Anschließend wird das Problem des Diskreten Logarithmus in einer zyklischen Gruppe eingeführt (Anfang von §5.2.2) und der Baby-Steps-Giant-Steps Algorithmus zu seiner Lösung (§5.3 + Laufzeitabschätzung) besprochen. Falls Zeit bleibt, könnte man auch noch eine Adaption des ρ-Verfahrens für dieses Problem beschreiben: Lambda-Algorithmus (§5.2.3) und 'Distinguished point trick'.

Vortrag 9 (Faktorisierung II: glatte Zahlen, Quadratisches Sieb).

Obwohl das Faktorisierungsproblem bis heute nicht in polynomieller Zeit gelöst werden kann, sind doch wesentlich schnellere Verfahren als die aus dem letzten Vortrag bekannt. Sie gehen ebenfalls auf Pollard zurück und kombinieren auf schöne Weise Lineare Algebra mit dem klassischen Sieb des Eratosthenes. Zunächst formulieren wir das Faktorisierungsproblem für n um: Nach §6.1 reicht es aus in $\mathbb{Z}/n$ eine Wurzel $\neq \pm 1$ aus 1 zu ziehen. Diese wollen wir aus "B-glatten" Teillösungen kombinieren. Eine Abschätzung für die Häufigkeit glatter Zahlen wird durch (1.44) gegeben. Um sie durch Sieben zu finden, ist im Mittel ein Aufwand $O(\ln \ln B)$ erforderlich (§3.2.5). Nach diesen Vorarbeiten können wir das Quadratische Sieb erläutern und erhalten (heuristisch) als Laufzeit der Siebstufe $O(n^{o(1)})$, was subexponentiell ist! Abschließend kann bemerkt werden, dass schnelle Matrix-Verfahren eine ähnliche Komplexität auch für den Lineare-Algebra-Teil ergeben (§6.3.1).

Vortrag 10 (Hintergrund: Zahlkörper).

Dieser Vortrag stellt die Protagonisten der algebraischen Zahlentheorie vor. Eine (mögliche) Referenz ist Marcus [M],§2, oder Neukirch [N], §1.2-1.3. Er ist insofern eine Faktensammlung, die durch Beispiele anschaulich gemacht werden soll. Am Anfang steht der Begriff des Zahlkörpers, seiner Norm und Spur über $\mathbb{Q}$. So wie wir in $\mathbb{Q}$ die ganzen Zahlen $\mathbb{Z}$ untersucht haben, kann man analog den Ring $\mathcal{O}_K$ in einem Zahlkörper K behandeln: etwa seine (Prim-) Ideale und das Analogon des Fundamentalsatzes. Das Beispiel $\mathbb{Z}[i]$ bietet sich hier an. Nach diesem generellen Ausblick untersuchen wir nun die Frage, wann ein Element Quadrat in $\mathcal{O}_K$ ist: §6.2 von [CP] und als Beispiel: $2i$ ist Quadrat in $\mathbb{Z}[i]$, $5i$ jedoch nicht. Als Anwendung wird zum Schluss Lemma 6.2.1 in [CP] bewiesen.

Vortrag 11 (Faktorisierung III: Zahlkörpersieb).

Dieser Vortrag ist wahrscheinlich der technisch komplizierteste, da er die Konzepte des Quadratischen Siebs ins Setting der Zahlkörper aus dem letzten Vortrag verallgemeinert. Ein gutes Verständnis dieser beiden Vorträge ist also Mindestvoraussetzung. Wir erläutern, §6.2.1 folgend, die Strategie des ZKS, stoßen aber auf vier Probleme: §6.2.4. Die abschließende Schwierigkeit Wurzeln in Ganzheitsringen zu ziehen (§6.2.5) kann vielleicht kurz angerissen werden. Als Fazit nach dieser umfassenden Behandlung des Faktorisierungsproblems halten wir fest: Der Begriff der Glattheit macht das Problem für Siebverfahren zugänglich und liefert schließlich subexponentielle Verfahren.

Vortrag 12 (Der Index Kalkül).

Anknüpfend an das Fazit des letzten Vortrags wollen wir auf das Diskrete Logarithmus-Problem (DL) zurückkommen und uns fragen, inwiefern Siebverfahren auch hier angewandt werden können: §6.4.1

folgend behandeln wir den DL in Primkörpern. Die sogenannte Index-Calculus Methode benutzt dort ebenfalls glatte Zahlen und eine Lineare Algebra-Stufe um Teillösungen zu kombinieren (für Details zur Laufzeit siehe §4 in [P]). Wir erinnern dann an endliche Körper und untersuchen DL in deren Einheitengruppen (§6.4.2). Es scheint so, dass, sobald man ein Problem auf Teilprobleme in glatten Zahlen zurückführen kann, Siebverfahren eine subexponentielle Lösung gestatten. Das ist der Grund, warum man heute in der Praxis DL *nicht* in $\mathbb{F}_q^\times$ benutzt!

Vortrag 13 (Asymmetrische Verschlüsselung)**.**

Die grundsätzliche Idee der Asymmetrischen Verschlüsselung wurde 1976 von Diffie und Hellman in ihrer berühmten 11-seitigen Arbeit [DH] beschrieben. Die Lektüre sei jedem Teilnehmer dieses Seminars wärmstens empfohlen. Diffie und Hellman haben sich gefragt wie zwei Parteien, die belauscht werden, Geheimnisse austauschen können, ohne je vorher kommuniziert zu haben. Die Problemstellung und ihr Verfahren wird in §8.1.1 von [CP] beschrieben. Einige Verfahren implementieren diese Ideen: Rivest, Shamir und Adleman entwickelten auf dem Faktorisierungsproblem aufbauend die Methode, die heute ihren Namen trägt. §8.1.2 folgend beschreiben wir wie Schlüsselpaare, Signierung und Verschlüsselung in diesem Verfahren funktionieren. Es gilt: Wer einen RSA-Modul schnell faktorisieren kann, kann dieses Verfahren brechen. Aber wie steht es mit der Umkehrung? (siehe §8.2 von [HAC]).

Vortrag 14 (Shors Algorithmus)**.**

Lässt man statt Turingmaschinen sogenannte Quanten-Turingmaschinen zur Berechnung zu, lassen sich einige Probleme exponentiell beschleunigen: Durch die Arbeit von Shor gibt es seit 1994 einen Algorithmus, der Zahlen in polynomieller Zeit faktorisieren kann. Dieser Vortrag soll kurz die Besonderheiten dieser (bisher grösstenteils hypothetischen) Maschinen vorstellen und dann den Algorithmus von Shor erläutern. Eine gute Übersicht des Materials findet sich in z.B. in [W]. Details finden sich im Original-Artikel [S].

Die folgenden Konzepte betreffend Quanten-Computer sollten kurz vorgestellt werden: Zustand und Überlagerung, Amplitude, Messung und Kollaps eines überlagerten Zustands, Qubit und Quanten-Register, Entwicklung eines Zustands durch (unitäre) Operatoren, Parallelismus und Verschränkung.

Dann kommen wir zum Algorithmus selber: Reduktion auf die Berechnung einer Periode, QFT, Vorbereitung eines gemischten Zustands, Laufzeitabschätzungen.