Privatdozent Dr. C. Diem
diem@math.uni-leipzig.de

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DICHT BESETZTE MATRIZEN UND POLYNOMFAKTORISIERUNG

1. Implementieren Sie eine Klasse Matrix mit den üblichen arithmetischen Operationen der Linearen Algebra, dem Berechnen von reduzierten Stufennormalformen, von Kernen und dem Invertieren von Matrizen. Für die letzeren drei Aufgaben sollten Sie den Gauß-Algorithmus implementieren.
2. Implementieren Sie eine speicher-effiziente Variante der Algorithmen für lineare Algebra über $\mathbb{F}_2$.
3. Implementieren Sie eine parallelisierte Variante für den Rechner miserv100 und testen Sie sie.
4. Implementieren Sie für das Faktorisieren von Polynomen über endlichen Körpern: den relativen und den absoluten Berlekamp-Algorithmus, den Cantor-Zassenhaus-Algorithmus (auch in Characteristik 2) sowie die quadrat-freie Faktorisierung. Zusammen sollte dies mehrere Algorithmen ergebeben, um Polynome über endlichen Körpern zu faktorisieren.

Empfohlene Literatur. Die Faktorisierungsalgorithmen werden gut beschrieben im Buch von Bach uns Shallit.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

Privatdozent Dr. C. Diem
diem@math.uni-leipzig.de

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DÜNN BESETZTE MATRIZEN UND DER WIEDEMANN-ALGORITHMUS

Schnelle lineare Algebra mit dünn besetzten Matrizen ist ein wesentlicher Bestandteil relevanter Algorithmen für das Faktorisierungs- und das diskrete Logarithmusproblem. Ein Beispiel ist das quadratische Sieb, zu dem es auch ein Projekt gibt.

Der Wiedemann-Algorithmus ist einer der Standard-Algorithmen für dieses Problem. In diesem Projekt beschäftigen Sie sich mit diesem Algorithmus und seiner effizienten Parallelisierung.

1. Implementieren Sie den Berlekamp-Massey Algorithmus und dann den Wiedenmann-Algorithmus über beliebigen endlichen Körpern.
2. Implementieren Sie den parallelisierten Block-Wiedemann-Algorithmus für Matrizen über $\mathbb{F}_2$ für miserv100 und testen Sie ihn.

Empfohlene Literatur. Der Originalartikel von Douglas Wiedemann, Solving sparse linear equations over finite fields, IEEE Transactions on Information Theory, **32** (1986) ist sicher ein guter Einstieg. Der Originalartikel für die Block-Variante ist: Don Coppersmith, Solving Homogeneous Linear Equations Over GF(2) via Block Wiedemann Algorithm, Mathematics of Computation **62** (1994). Beide Artikel sind "auf dem Internet" erhältlich.

Es gibt einen Wikipedia-Artikel zum Block-Wiedemann-Algorithmus, der aber wenig Information enthält (und die Informationen zu "Coppersmith-Algorithmus" scheinen mir genau auf den ursprünglichen Wiedemann-Algorithmus zu passen). Ich finde diese Folien interessant: Sonia Belaid und Sylvain Lachartre, Solving Sparse Systems with the Block Wiedemann Algorithm, Efficient Implementation over GF(2)

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

Wichtig! Dieses Projekt ist von besonderer Relevanz für die Projekte zum Faktorisieren und zur Berechnung diskreter Logarithmen. Aus diesem Grund sollten Sie möglichst schnell, auf jeden Fall noch im Juni, eine erste Version abliefern und das Projekt bis Ende Juli fertig stellen.

Privatdozent Dr. C. Diem
diem@math.uni-leipzig.de

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DÜNN BESETZTE MATRIZEN UND DER LANCZOS-ALGORITHMUS

Schnelle lineare Algebra mit dünn besetzten Matrizen ist ein wesentlicher Bestandteil relevanter Algorithmen für das Faktorisierungs- und das diskrete Logarithmusproblem. Ein Beispiel ist das quadratische Sieb, zu dem es auch ein Projekt gibt.

Der Lanczos-Algorithmus ist einer der Standard-Algorithmen für dieses Problem. In diesem Projekt beschäftigen Sie sich mit diesem Algorithmus und einem weiteren für die Lineare Algebra relevanten Schritt, dem Filtern.

1. Implementieren Sie den Lanczos-Algorithmus für "genügend große" endliche Körper. D.h.: Machen Sie zunächst sich keine Gedanken darüber, dass – im Gegensatz zum Skalarprodukt in $\mathbb{R}^n$ – für einen Vektor $\underline{x} \neq 0$ gelten kann: $\underline{x}^t \underline{x} = 0$.

   Die Eingabe sollte eine beliebige (d.h. nicht notwendigerweise symmetrische) Matrix A sein. Stellen Sie sicher, dass die Matrix $A^t A$ nie berechnet wird.

2. Implementieren Sie für kleine Körper (insb. über $\mathbb{F}_2$) die Variante, die nach Theorem 2 in Enge und Gaudry, A general framework for subexponential discrete logarithm algorithms beschrieben wird.

3. Parallelisieren Sie diesen Algorithmus für miserv100 und testen Sie ihn.

4. Implementieren Sie ein (iteratives) "Filtern", mit dem die Matrix kleiner gemacht wird und das Gewicht nach unten geht oder gleich bleibt.

   Achten Sie darauf, dass beim Löschen der Spalten die Indices abgespeichert werden, so dass zum Schluss die korrekte Lösung hergestellt werden kann.

Empfohlene Literatur. Ich denke, dass dieser Artikel ein guter Einstieg ist: Wayne Eberly und Erich Kaltofen, On randomized Lanczos Algorithms. Das sollte dann mit der genannten Bemerkung in Enge und Gaudry, A general framework for subexponential discrete logarithm algorithms kombiniert werden.

Für das Filtern sollten Sie sich Arjen Lenstra, General purpose integer factoring anschauen.

Die Arbeiten sind alle "auf dem Internet" erhältlich.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

Wichtig! Dieses Projekt ist von besonderer Relevanz für die Projekte zum Faktorisieren und zur Berechnung diskreter Logarithmen. Aus diesem Grund sollten Sie möglichst schnell, auf jeden Fall noch im Juni, eine erste Version abliefern und das Projekt bis Ende Juli fertig stellen.

Privatdozent Dr. C. Diem
diem@math.uni-leipzig.de

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DISKRETE LOGARITHMEN IN KLEINER CHARAKTERISTIK

In diesem Projekt geht es um das diskrete Logarithmusproblem in multiplikativen Gruppen endlicher Körper kleiner Charakteristik. Vor etwa vier bis fünf Jahren hat es für diese Frage in kurzer Reihenfolge eine Reihe von Durchbrüchen gegeben.

Im Endergebnis gibt es nun einen recht leicht zu beschreibenenden der Vermutung nach quasi-polynomiellen Algorithmus von Granger, Kleinjung und Zumbrägel.

Das Ziel des Projektes ist es, diesen Algorithmus zu implementieren.

Der Lineare Algebra-Teil soll hierbei in einem anderen Projekt behandelt werden und später integriert werden.

Die Polynomfaktorisierung sollen Hannes Thalheim und Marc Vester implementieren.

Die Aufgabenstellung ist dementsprechend:

1. Implementieren Sie den Algorithmus von Granger, Keinjung und Zumbrägel.
2. Implementieren Sie ggf. noch eine doppelt-große-Primvariation.
3. Integrieren Sie eine schnelle Lineare Algebra.
4. Testen Sie Ihre Implementierung und vergleichen Sie die erzielten Ergebnisse mit den aktuellen Rekorden.

Empfohlene Literatur. Die wichtigste Arbeit ist:

Granger, Kleinjung, Zumbrägel: On the discrete logarithm problem in finite fields of fixed characteristic, Transactions of the AMS, 2018

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: GENERISCHE ALGORITHMEN FÜR DAS FAKTORISIEREN UND DIE BERECHNUNG DISKRETER LOGARITHMEN

Mittels sogenannter *generischer Methoden* kann das diskrete Logarithmusproblem in jeder endlichen Gruppe angegriffen werden. Die Laufzeit ist – unter gewissen Annahmen – stets im Wesentlichen durch die Wurzel des größten Primteilers der Ordnung des Elementes gegeben. Hierfür wird zunächst das Problem auf Untergruppen zyklischer Ordnung reduziert. Der Algorithmus hierfür heißt *Pohlig-Hellman-Algorithmus*. Hiernach gibt es im Wesentlichen drei Möglichkeiten: Man benutzt den *Baby-Step-Giant-Step-Algorithmus*, Pollards *ρ-Methode* oder Pollards λ- oder *Känguru-Methode*.

In diesem Projekt sollen Sie diese für endliche Körper und elliptische Kurven implementieren.

1. Implementieren Sie die Arithmetik auf elliptischen Kurven in Weierstraß- und in Edwards-Darstellung.

   Die folgenden Aufgaben sollten Sie dann sowohl für endliche Körper als auch für elliptische Kurven über endlichen Körpern durchführen.

2. Implementieren Sie die Reduktion von Pohlig und Hellman.

   Hierfür muss die Gruppenordnung faktorisiert werden. Da dies eine Sache für sich ist, sollte die Faktorisierung bei Ihnen mit eingegeben werden. Sie können sie dann "extern" berechnen.

3. Implementieren Sie den Baby-Step-Giant-Step Algorithmus mit einer Hash-Tabelle.

   Achten Sie darauf, dass Sie den Speicher effizient verwalten. Die Gruppenelemente selbst brauchen Sie nicht abzuspeichern, da sie das relevante Element auch nochmal berechnen können.

4. Implementieren Sie die ρ- und die λ-Methoden in verschiedenen Varianten.

5. Untersuchen Sie, welche Varianten die besten Ergebnisse liefern.

   (Die λ-Methode ist einen weiteren Anwendungsbereich als die ρ-Methode, da man mit ihr berücksichtigen kann, dass der gesuchte diskrete Logarithmus in einem Intervall liegt. Deshalb wäre die λ-Methode nicht irrelevant, wenn die ρ-Methode für allgemeine Eingaben in der Regel schneller wäre.)

6. Parallelisieren Sie einen Algorithmus und testen Sie ihn auf miserv100.

7. Implementieren Sie die ρ-Methode für das Faktorisieren so, wie es Ihnen nach dem Vorherigen am geignetsten erscheint.

Empfohlene Literatur. Im Buch von Steven Galbratih werden die "generischen" Algorithmen ausführlich behandelt. Für Optimierungen ist auch dieser Artikel interessant: Joppe Bos, Thorsten Kleinjung, Arjen Lenstra, On the Use of the Negation Map in the Pollard Rho Method, ANTS 2010

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

Privatdozent Dr. C. Diem
diem@math.uni-leipzig.de

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DAS QUADRATISCHE SIEB

Das Ziel diesse Projektes ist, eine effiziente Implementierung des Quadratischen Siebs für Faktorisierung bis auf den Lineare Algebra-Teil und zusätzliche Unterstützung durch elliptische Kurven-Faktorisierung. Diese Teile werden in anderen Projekten behandelt. Sie sollten sich abstimmen, so dass Sie zum Schluss eine effiziente Implementierung erhalten.

1. Implementieren Sie eine erste Variante des Quadratischen Siebs.

   Für die positiven Siebresultate sollten Sie erstmal Probeteilen anwenden.

   Testen Sie, welche Laufzeit das Probeteilen hat und inwiefern diese relevant ist.

2. Implementieren Sie das Multiple Prime Quadratic Sieve.

3. Implementieren Sie eine große Primvariation.

4. Integrieren Sie die elliptische Kurven-Faktorisierung.

5. Integrieren Sie eine schnelle Lineare Algebra.

6. Parallelisieren Sie das Sieben für den Rechner miserv100 und testen Sie es.

Empfohlene Literatur. Für eine Einführung empfehle ich das Buch von Buchmann. Danach sollten Sie sich den aktuellen Artkel Arjen Lenstra: General purpose integer factoring anschauen.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: FAKTORISIEREN MIT KETTENBRUCHENTWICKLUNG

Das Ziel diesse Projektes ist, eine effiziente Implementierung des Faktorisierungsalgorithmus mit Faktorbasis und Kettenbruchentwicklung bis auf den Lineare Algebra-Teil und zusätzliche Unterstützung durch elliptische Kurven-Faktorisierung. Diese Teile werden in anderen Projekten behandelt. Sie sollten sich abstimmen, so dass Sie zum Schluss eine effiziente Implementierung erhalten.

1. Implementieren Sie die Kettenbruchentwicklung für die Wurzel einer Zahl n. Betrachten Sie hiermit (durch Experimente), wie man Quadrate erhält, die kleiner als $2\sqrt{n}$ sind. (Wir wissen natürlich schon, dass das geht, es ist ja bewiesen ...)
2. Implementieren Sie eine erste Version des Faktorisierung-Algorithmus mit Faktorbasis und Kettenbruchentwicklung. Verwenden Sie hierbei Probedivision.
3. Integrieren Sie die elliptische Kurven-Faktorisierung.
4. Implementieren Sie eine große Primvariation.
5. Implementieren Sie gegebenenfalls eine doppelt-große Primvariation mit einem Graphen.
6. Integrieren Sie eine schnelle Lineare Algebra.
7. Parallelisieren Sie die Relationensuche für den Rechner miserv100 und testen Sie es.

Empfohlene Literatur. Für eine Einführung empfehle ich das Buch von Neal Koblitz.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: FAKTORISIEREN MIT KETTENBRUCHENTWICKLUNG

Das Ziel diesse Projektes ist, eine effiziente Implementierung des Faktorisierungsalgorithmus mit Faktorbasis und Kettenbruchentwicklung bis auf den Lineare Algebra-Teil und zusätzliche Unterstützung durch elliptische Kurven-Faktorisierung. Diese Teile werden in anderen Projekten behandelt. Sie sollten sich abstimmen, so dass Sie zum Schluss eine effiziente Implementierung erhalten.

1. Implementieren Sie die Kettenbruchentwicklung für die Wurzel einer Zahl n. Betrachten Sie hiermit (durch Experimente), wie man Quadrate erhält, die kleiner als $2\sqrt{n}$ sind. (Wir wissen natürlich schon, dass das geht, es ist ja bewiesen ...)
2. Implementieren Sie eine erste Version des Faktorisierung-Algorithmus mit Faktorbasis und Kettenbruchentwicklung. Verwenden Sie hierbei Probedivision.
3. Integrieren Sie die elliptische Kurven-Faktorisierung.
4. Implementieren Sie eine große Primvariation.
5. Implementieren Sie eine doppelt-große Primvariation mit einem Graphen.
6. Integrieren Sie eine schnelle Lineare Algebra.
7. Parallelisieren Sie die Relationensuche für den Rechner miserv100 und testen Sie es.
8. Wenn "Filtern" für die Lineare Algebra vorhanden (dafür ist eine andere Gruppe zuständig): Testen Sie die Verwendung von dem Graphen gegen direktes Filtern der Matrix (mit zwei großen Primzahlen).

Empfohlene Literatur. Für eine Einführung empfehle ich das Buch von Neal Koblitz.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: FAKTORISIEREN MIT KETTENBRUCHENTWICKLUNG

Das Ziel diesse Projektes ist, eine effiziente Implementierung des Faktorisierungsalgorithmus mit Faktorbasis und Kettenbruchentwicklung bis auf den Lineare Algebra-Teil und zusätzliche Unterstützung durch elliptische Kurven-Faktorisierung. Diese Teile werden in anderen Projekten behandelt. Sie sollten sich abstimmen, so dass Sie zum Schluss eine effiziente Implementierung erhalten.

1. Implementieren Sie die Kettenbruchentwicklung für die Wurzel einer Zahl n. Betrachten Sie hiermit (durch Experimente), wie man Quadrate erhält, die kleiner als $2\sqrt{n}$ sind. (Wir wissen natürlich schon, dass das geht, es ist ja bewiesen ...)
2. Implementieren Sie eine erste Version des Faktorisierung-Algorithmus mit Faktorbasis und Kettenbruchentwicklung. Verwenden Sie hierbei Probedivision.
3. Integrieren Sie die elliptische Kurven-Faktorisierung.
4. Implementieren Sie eine große Primvariation.
5. Implementieren Sie eine doppelt-große Primvariation mit einem Graphen.
6. Integrieren Sie eine schnelle Lineare Algebra.
7. Parallelisieren Sie die Relationensuche für den Rechner miserv100 und testen Sie es.
8. Wenn "Filtern" für die Lineare Algebra vorhanden (dafür ist eine andere Gruppe zuständig): Testen Sie die Verwendung von dem Graphen gegen direktes Filtern der Matrix (mit zwei großen Primzahlen).

Empfohlene Literatur. Für eine Einführung empfehle ich das Buch von Neal Koblitz.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: BERECHNUNG DISKRETER LOGARITHMEN IN PRIMKÖRPERN

Das Ziel diesse Projektes ist, eine effiziente Implementierung des sogenannten Gauß'schen-Zahlen Siebs für die Berechnung diskreter Logarithmen bis auf den Lineare Algebra-Teil und zusätzliche Unterstützung durch elliptische Kurven-Faktorisierung. Diese Teile wird in anderen Projekten behandelt. Sie sollten sich abstimmen, so dass Sie zum Schluss eine effiziente Implementierung erhalten.

1. Implementieren Sie das Gauß'sche-Zahlen Sieb.

   Für die positiven Siebresultate sollten Sie erstmal Probeteilen anwenden.

   Testen Sie, welche Laufzeit das Probeteilen hat und inwiefern diese relevant ist.

2. Implementieren Sie eine erste Version des Faktorisierung-Algorithmus mit Faktorbasis und Kettenbruchentwicklung. Verwenden Sie hierbei Probedivision.

3. Integrieren Sie die elliptische Kurven-Faktorisierung.

4. Implementieren Sie eine große Primvariation.

5. Implementieren Sie eine doppelt-große Primvariation mit einem Graphen.

6. Integrieren Sie eine schnelle Lineare Algebra.

7. Parallelisieren Sie die Relationensuche für den Rechner miserv100 und testen Sie es.

8. Wenn "Filtern" für die Lineare Algebra vorhanden (dafür ist eine andere Gruppe zuständig): Testen Sie die Verwendung von dem Graphen gegen direktes Filtern der Matrix (mit zwei großen Primzahlen).

Empfohlene Literatur. Sie sollten sich zunächst mit dem quadratischen Sieb für das Faktorisieren vertraut machen. Dies wird gut erklärt in dem Buch von Buchmann. Danach sollten Sie sich den relevanten Originalartikel anschauen. Dieser ist: Don Coppersmith, Andrew Odlyzko und Richard Schroeppel, Discrete logarithms in GF(p), Algorithmica **1**, 1986

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: DIE ELLIPTISCHE KURVEN-FAKTORISIERUNGSMETHODE

Die elliptische Kurven-Faktorisierungsmethode ist eine überraschende Anwendung elliptischer Kurven mit Bezug zur Kryptographie: Hier wird nicht – wie sonst in der Kryptographie – das diskrete Logarithmusproblem auf elliptischen Kurven über Primkörpern angewandt, sondern elliptische Kurven über einem Restklassenring $\mathbb{Z}/n\mathbb{Z}$ werden benutzt, um n zu faktorisieren. Die Methode ist besonders gut geeignet, um relativ kleine Primfaktoren zu finden. Dies kann dann wiederum in anderen (allgemeinen) Faktorisierungsalgorithmen wie dem Quadratischen Sieb verwendet werden.

Diese Methode ist eine Weiterentwicklung der sogenannten $p-1$-*Methode*, die auf der Arithmetik in multiplikativen Gruppen beruht.

1. Implementieren Sie alle Darstellungen elliptischer Kurven über Körpern großer Charakteristik von der „Explicit-Formulas Database“ (https://hyperelliptic.org/EFD).
2. Testen Sie sie bezüglich Geschwindigkeit.
3. Implementieren Sie für alle diese Darstellungen die elliptische Kurven-Faktorisierungsmethode und testen Sie sie. Für welche Darstellung ist sie am schnellsten? Wählen Sie dann die entsprechende Darstellung aus.
4. Implementieren Sie eine parallele Variante und testen Sie sie auf miserv100.
5. Stellen Sie Ihre Testergebnisse in Abhängigkeit der Größe der zu faktorisierenden Zahl und des kleinsten Primfaktors dar.

   Soweit möglich vergleichen Sie sie mit denen in Bernstein, Birkner, Lange und Peters: ECM using Edards curves, Mathematics of Computation **82** (2013)
6. Optimieren Sie Ihre Implementierung weiter entsprechend dieses Artikels.
7. Parallelisieren Sie Ihre Implementation und testen Sie Sie auf dem Rechner miserv100.

Empfohlene Literatur. Zum Einstieg empfehle ich das Buch von Neal Koblitz. Später ist der angegebene Artikel relevant. Dieser findet sich auch auf dem Internet.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

Wichtig! Dieses Projekt soll von mehreren Projekten zum Faktorisieren und zur Berechnung diskreter Logarithmen verwendet werden. Sie sollten es deshalb zügig bearbeiten.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: SCHOOFS PUNKTEZÄHLALGORITHMUS

Damit eine elliptische Kurve über einem endlichen Körper kryptographisch geeignet ist, muss auf jeden Fall die Ordnung ihrer Punktegruppe fast prim sein – der sogenannte Kofaktor sollte vielleicht 2 oder 4 sein.

Um dies sicherzustellen, muss die Ordnung der Gruppe berechnet werden. Mit einer passenden Gruppenordnung können dann auch Angriffe mittels Paarungen ausgeschlossen werden.

Der heutzutage klassische Algorithmus hierfür ist derjenige von René Schoof. Varianten dieses Algorithmus sind immer noch die effizientesten Algorithmen für Kurven über Primkörpern. Die Aufgabe besteht darin, diesen zu implementieren und dann noch zu optimieren.

1. Implementieren Sie eine Funktion, die dies macht: Zu einer gegebenen Ordnung einer elliptischen Kurve über einem endlichen Körper mit q Elementen (q eine Primpotenz) und einem $n \in \mathbb{N}$ wird die Ordnung über $\mathbb{F}_{q^n}$ ausgegeben.

   Beachten Sie: Hierfür muss (für den Algorithmus) keine elliptische Kurve betrachtet werden, sondern nur eine Formel ausgewertet werden. Der Hintergrund ist durch L-Polynome oder charakteristische Polynome des Frobenius gegeben (dies ist fast dasselbe, die beiden Polynome sind reziprok zueinander). Die Rechnung selbst ist dann (ohne komplexe Zahlen) – auf den ersten Blick nicht so leicht zu implementieren, selbst wenn man weiß, wie es gehen sollte. Aber es geht ...

2. Implementieren Sie Klassen für elliptische Kurven in Weierstraß-Form und in Edwards-Form sowie für Punkte hierauf. Implementieren Sie einen Transfer zwischen diesen beiden Strukturen (Kurven werden transformiert und Punkte abgebildet).

   Implementieren Sie auch einen Basiswechsel bezüglich einer Körpererweiterung. (Hier passiert algorithmisch ganz wenig, man erhält dann aber eben ein neues Objekt.)

3. Implementieren Sie den Schoof-Algorithmus in "straight-forward"-Art und Weise. Fügen Sie der elliptischen Kurve ein Attribut `ordnung` hinzu, das einen Wert erhält, wenn man einmal die Ordnung berechnet hat. (Das geht aber nur, wenn das Objekt nicht "eingefroren" ist.)

4. Testen Sie die Laufzeiten Ihrer Implementation gegen die Laufzeiten des Kommandos `Order` für elliptische Kurven in `Magma`.

5. Es gibt Varianten des Schoof-Algorithmus basierend auf folgender Idee: Anstatt für eine Primzahl ℓ mit der vollen Gruppe der ℓ-Teilungspunkte zu arbeiten, sucht man zunächst eine zyklische Untergruppe (eigentlich hat man hier Gruppenschemata und Gruppen erst nach einer Körpererweiterung). Für diese Idee sollte man nach Faktoren des ℓ-Teilungspolynoms von Grad $\frac{\ell-1}{2}$ suchen. Dieses Vorgehen ist Abschnitt 9.6 des Skripts von Andrew Sutherland erklärt (s.u.). (Mir erscheint allerdings Sutherlands Aussage über die Grade der Faktoren nicht ganz korrekt, wenn man schon einen Punkt über dem Grundkörper hat?)

   Implementieren Sie diese Variante oder eine geeignete Verbesserung hiervon! (Ein wenig Ausprobieren ist hier vielleicht gefragt.)

(Wie von Sutherland beschrieben, kann man hier noch wesentlich mehr Theorie einsetzen, dies führt dann zum Algorithmus von Schoof, Elkies und Atkins (SEA). Im Projekt würde dies wohl den Rahmen sprengen.)

6. Untersuchen Sie die Statistik der "guten" ℓ.

7. Betrachten Sie auch die folgende Variante des Algorithmus: Wenn ein passener Faktor des ℓ-Teilungspolynoms gefunden worden ist, sagen wir ψ, wird die von ψ definierte Körpererweiterung von Grad d betrachtet und dann die durch y definiert Erweiterung von Grad 2. Die Kurve hat über diesem Grundkörper einen Punkt der Ordung ℓ, dessen Koordinaten schon bekannt sind (die x-Koordinate ist durch die Restklasse von ψ gegeben, die y-Koordinate analog). Dann kann man die Operation auf diesem Punkt auf der Kurve betrachten. Hiermit kann wie beim "nornalen Schoff" die Spur des Frobenius modulo ℓ ausgerechnet werden.

   Implementieren Sie dies! (Wenn man lange darüber nachdenkt, sollte auffallen, dass diese Variante des Algorithmus nah am ursprünglichen Algorithmus ist. Es werden aber andere Datenstrukturen benutzt, nämlich insbesondere die elliptische Kurve.

8. Fügen Sie die Ordnungsberechnung für elliptische Kurven mit dem Algorithmus unter 1. zusammen und erhalten Sie somit eine Funktion, die zu einer elliptischen Kurve über einem endlichen Körper und einem Erweiterungsgrad die entsprechende Ordnung berechnet.

9. Finden Sie "kryptographisch geeignete elliptische Kurven" in Weierstraß-Form und in Edwards-Form über Primkörpern im folgenden Sinne: Die Gruppenordnung ist prim oder der Form $2p$ oder $4p$ mit etwa 256 bit und die Angriffe mittels Paarungen (Reduktion auf das DLP in multiplikativen oder additiven Gruppen endlicher Körper) sind praktisch unmöglich.

Empfohlene Literatur und Hinweise. Für die erste Aufgabe finden Sie die relevanten Zusammenhang beispielsweise in Henning Stichtenoth, Algebraic Function Fields and Codes, Theorem V.1.15. Beachten Sie, dass die Gruppe isomorph zu Grad 0 Klassengruppe ist und die Ordnung somit durch $L(1)$ gegeben ist.

Für die weiteren Aufgaben gefällt mir eine Vorlesungsausarbeitung von Andrew Sutherland am MIT sehr gut. Die Kursnummer ist 18.783, die Adresse ist:

https://math.mit.edu/classes/18.783/2015/LectureNotes9.pdf

Der Artikel in der englischsprachigen Wikipedia zum Schoof-Algorithmus ist auch recht gut. Von dort aus finden Sie auch weitere Literatur, insbesondere den Originalartikel von René Schoof Counting Points on Elliptic Curves over Finite Fields aus dem Jahr 1995.

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.

MATHEMATISCHE GRUNDLAGEN DER KRYPTOGRAPHIE MIT ÖFFENTLICHEN SCHLÜSSELN

PROJEKT: SCHNELLE LINEARE ALGEBRA ÜBER KÖRPERN UND POLYNOMRINGEN

Lineare Algebra über Körpern ist in vielerlei Hinsicht grundlegend für die Kryptographie mit öffentlichen Schlüsseln: Sie wird in vielerlei Angriffen benutzt und sie wird auch benötigt, wenn man das diskrete Logarithmusproblem für Kurven höheren Geschlechts (genauer: das diskrete Logarithmusproblem in Grad 0 Klassengruppen von Kurven höheren Geschlechts) betrachtet. (Letzteres ist auch für Angriffe auf das diskrete Logarithmenproblem in elliptischen Kurven relevant.) Genauer sind Normalformen wie die Hermite-Normalform relevant, wenn man nur grundlegende Operationen mit Kurven und ihren Divisoren durchführen will.

In diesem Projekt sollen schnelle Algorithmen für Lineare Algebra über Körpern und über Polynomringen implementiert werden.

1. Implementieren Sie die Matrizenmultiplikation, das Invertieren und das Kern-Berechnen mittels des Gauß-Algorithmus.
2. Implementieren Sie den Strassen-Algorithmus für alle genannten Probleme.
3. Führen Sie Experimente durch. Welcher Algorithmus ist bei welchen Eingaben schneller?
4. Implementieren Sie ggf. noch eine (komplexere und asymptotisch bessere) Variante des Strassen-Algorithmus.
5. Implementieren Sie für Matrizen über Polynomringen (über Körpern) die Algorithmen aus der Arbeit von Mulders und Strohjohann.
6. Implementieren Sie den asymptotisch guten Algorithmus für die Hermite Normalform von Gupta und Storjohann.
7. Testen Sie Ihre Implementierungen. Welche ist schneller?

Empfohlene Literatur. Für die Algorithmen für Matrizen über Körpern sollten Sie sich dieses Buch anschauen: Bürgisser, Clausen, Shokrollahi: Algebraic Complexity Theory. Sie sollten sich besonders Kapitel 16 anschauen.

Ein einfacher einführender Artikel für Matrizen über Polynomringen ist Mulders und Storjohann, On lattice reduction for polynomial matrices, Journal of Symbolic Computation (2002). Die asymptotisch besten Algorithmen scheinen mir im Artikel von Gupta und Storjohann, Computing Hermite forms of polynomial matrices, ISSAC 2011 relevant. Es gibt noch weitere auf den ersten Blick interessante Artikel von Storjohann, besonders High-order lifting and integrality certification, Journal of Symbolic Computation (2002).

Für die Dokumentation. Geben Sie eine knappe Darstellung der Algorithmen und beschreiben Sie Ihre Implementation. Gehen Sie Ihre Implementierung mit Beispielen durch.

Wenn es Probleme gibt. Es ist durchaus möglich, dass einzelne Projektteile unangemessen schwer umzusetzen sind oder sich als nicht sinnvoll herausstellen. Oder vielleicht fällt Ihnen während des Projektes auf, dass Sie in eine etwas andere Richtung gehen wollen. Wenn Sie dieser Auffassung sind, sprechen Sie mich an! Wir können dann gegebenenfalls das Projekt abhändern.