Cours MPRI 2-12-2

Lecture 1/5: Factoring by combining congruences

(lecturer for part 2/3): E. Thomé

Nov. 5th, 2012

Plan

Tentative plan for the lectures to come

Congruences of squares

Analyzing smoothness-based algorithm

Exercises

Tentative plan

- Nov. 5th (today): Congruences of squares; CFRAC; Adleman's algorithm for discrete logs; Elements for analysis.
- Nov. 12th: The idea of sieving; Implications for analysis; Some improvements.
- Nov. 19th: Sparse linear algebra; The Lanczos and Wiedemann algorithms; Analysis issues; Implementation issues.
- Nov. 26th: Exercises.
- Dec. 3rd: Exam.
- Dec. 10th: Number Field Sieve (I); Factoring with cubic integers; Some algebraic number theory background.
- Dec. 17th: Number Field Sieve (II); Steps being worked on w.r.t NFS; Some record computations; NFS and its cousins.

Copy-paste of a slide from lecture 1, part 1

The difficulty of discrete logarithm computations

Over finite fields:

- $\mathbb{F}_p$:
  - Best algorithm so far: *à la* NFS $O(L_p[1/3, c'])$ (Gordon, Schirokauer).
  - record with $160dd$: T. Kleinjung (2007); 3.3 years of PC 3.2 GHz Xeon64; matrix $2,177,226 \times 2,177,026$ with $289,976,350$ non-zero coefficients, inverted in 14 years CPU.

- $\mathbb{F}_{p^n}$: Adleman-DeMarrais, function field sieve + optimizations.
  - $p = 2$: Coppersmith; record with $\mathbb{F}_{2^{613}}$: Joux/Lercier (2005).
  - record $\mathbb{F}_{3^{6\times 71}}$: Hayashi *et al.* (2010), `http://eprint.iacr.org/2010/090`.
  - Medium p case: Joux+Lercier.

$$L_N[\alpha, c] = \exp((c + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}).$$

F. Morain – École polytechnique – MPRI – cours 2.12.2 – 2011-2012 10/17

Breaking news: DL record in characteristic 2 has just been updated.

DL record for $\mathbb{F}_{2^{619}}$

This was really an old record (2005).

- DL computed using the Function Field Sieve algorithm.
  (a cousin of the Number Field Sieve).
- Computation done in almost a day.
  - About 160 core-hours of sieving.
  - Linear algebra (18hrs) using graphics cards.
  - This entails nice `C` code programming done in Nancy.
- Announced last week at the ECC 2012 workshop in Mexico.

Lecture material

All slides and extra material will appear on the page:

`http://www.loria.fr/~thome/MPRI/`

Lectures are on Monday, expect slides to be posted by Tuesday evening typically.

Plan

Tentative plan for the lectures to come

Congruences of squares

Analyzing smoothness-based algorithm

Exercises

Plan

Factoring ?

Is 8051 a prime, and if not, can you factor it ?

Factoring ?

Is 8051 a prime, and if not, can you factor it ? There's a trick:

$$\begin{aligned} 8051 &= 8100 - 49, \\ &= 90^2 - 7^2, \\ &= 83 \times 97. \end{aligned}$$

Congruences of squares

An early idea (not really with algorithmic intent) due to Fermat.

We try to factor N. Set $r = \left\lceil \sqrt{N} \right\rceil$.

- For $i = 0, \ldots$, compute $f(i) = (r+i)^2 - N$.
- If $f(i)$ is a square, then we have:

$$\begin{aligned} (r+i)^2 - N &= x^2, \\ (r+i-x)(r+i+x) &= N. \end{aligned}$$

Let $N = pq$. This method factors N in time $O(|p-q|)$.
This suceeds if p, q are too close to $\sqrt{N}$. Otherwise hopeless.

Solving the «nearby p, q » case efficiently

Exercise: factor N if $|p - c| < \sqrt[4]{N}$, with $c = \lfloor \sqrt{N} \rfloor$

- Write $N = (c + r)(c - s)$.
- Show that $s \geq r$.
- Give a polynomial-time calculation which recovers rs.
- Deduce $(r - s)$, and finally both r and s.

- This improvement does solve the «too easy case».
- Yet, the key idea of Fermat remains: search for squares.
  Fruitful for many other algorithms.

Congruences of squares

Given a composite N, what does $x^2 \equiv y^2 \mod N$ give ?

$$\begin{aligned} x^2 &\equiv y^2, \\ (x-y)(x+y) &\equiv 0, \\ \left(\frac{x}{y}-1\right)\left(\frac{x}{y}+1\right) &\equiv 0 \quad (\text{we may assume } \gcd(y,N)=1). \end{aligned}$$

N with k distinct prime factors $\Rightarrow 2^k$ square roots of 1

- A "random" congruence $x^2 \equiv y^2$ reveals a factor with prob $1 - \frac{1}{2^{k-1}}$.
- Note that this cannot work for prime powers.

Kraitchik

From the 1930's:

- Looking at congruences is enough.
- If $r^2 \bmod N$ and $s^2 \bmod N$ are not squares, but their product is, then we succeed.

This is the principle of combination of congruences.

Combination of congruences

$$
\begin{aligned}
46^2 \bmod 2041 &= 75 = 3 \times 5^2, \\
47^2 \bmod 2041 &= 168 = 2^3 \times 3 \times 7, \\
48^2 \bmod 2041 &= 263 = \text{I am lazy, too hard...} \\
49^2 \bmod 2041 &= 360 = 2^3 \times 3^2 \times 5, \\
50^2 \bmod 2041 &= 459 = 3^3 \times 17, \\
51^2 \bmod 2041 &= 560 = 2^4 \times 5 \times 7, \qquad \ldots
\end{aligned}
$$

This leads to

$$(\underbrace{46 \times 47 \times 49 \times 51}_{x})^2 \equiv 2^{10}3^4 5^4 7^2 \equiv (\underbrace{2^5 3^2 5^2 7}_{y})^2 \mod N.$$

And $\gcd(x - y, N) = 13$.

Combination of congruences

$$
\begin{aligned}
46^2 \bmod 2041 &= 75 = 3 \times 5^2, \\
47^2 \bmod 2041 &= 168 = 2^3 \times 3 \times 7, \\
48^2 \bmod 2041 &= 263 = \text{I am lazy, too hard\ldots} \\
49^2 \bmod 2041 &= 360 = 2^3 \times 3^2 \times 5, \\
50^2 \bmod 2041 &= 459 = 3^3 \times 17, \\
51^2 \bmod 2041 &= 560 = 2^4 \times 5 \times 7, \qquad \ldots
\end{aligned}
$$

Important facts

- We are chiefly interested in smooth numbers.
- Only the parity of exponents really counts.
- We are certainly affected by the size of the residues.

Two distinct prospects

Research towards a «better» factoring based on combination on congruences may focus on:

- Obtaining a (probabilistic) algorithm whose runtime can be analyzed and proven rigorously.
  Example: Dixon's algorithm (next).
- Obtaining a rather fast algorithm, but whose runtime is possibly only heuristic.
  Examples: CFRAC, QS, NFS.

Cryptanalysis is rather biased towards «fast, but heuristic».

Plan

Dixon's random squares algorithm

This was formalized by Dixon in the 1970's. Proven $L(1/2)$.

- We are interested in the factorization of $r^2 \bmod N$ only if it is smooth.
- We fix a smoothness bound B.
- The set of primes $\mathcal{P}_B$ is called the factor base.

Algorithm:

- Pick r at random. Test divisibility by all primes below B.
  If $r^2 \bmod N$ is B-smooth, keep the relation:

$$r_i^2 \equiv p_1^{e_{i,1}} \times \cdots \times p_k^{e_{i,k}} \mod N.$$

- Try to combine these. This is a linear algebra problem over $\mathbb{F}_2$.

Combination by linear algebra

We have a set $\mathcal{R}$ of relations $r_i^2 \equiv p_1^{e_{i,1}} \times \cdots \times p_k^{e_{i,k}}$.

- Consider the matrix $M \in \mathcal{M}_{\#\mathcal{R}\times\#\mathcal{P}}(\mathbb{Z})$, $M = (e_{i,j})$.

$$\begin{aligned} 46^2 \mod 2041 &= 75 = 3 \times 5^2, \\ 47^2 \mod 2041 &= 168 = 2^3 \times 3 \times 7, \\ 49^2 \mod 2041 &= 360 = 2^3 \times 3^2 \times 5, \\ 51^2 \mod 2041 &= 560 = 2^4 \times 5 \times 7, \end{aligned} \Longrightarrow \begin{pmatrix} 0 & 1 & 2 & 0 \\ 3 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ 4 & 0 & 1 & 1 \end{pmatrix}$$

- A vector $V = (v_i)_{1\leq i\leq \#\mathcal{R}}$ yields $VM = (\sum_i v_i e_{i,j})_j$, and:

$$(\prod r_i^{v_i})^2 \equiv \prod_j p_j^{\sum_i v_i e_{i,j}}.$$

- We want V such that coordinates of VM are even: it suffices to search for (left) nullspace elements over the field $\mathbb{F}_2$.

Look forward into analysis

For analyzing Dixon's algorithm, one needs:

- An estimate on the size of $r^2 \bmod N$, and its probability of B-smoothness given the bound B.
- Time complexity for solving the linear system which arises.

The result of the analysis gives the optimal value for the factor base bound B

Performance handicaps in Dixon's algorithm

Dixon's algorithm is nice for getting a proven algorithm.

However, performance-wise, it suffers from the large size of $r^2 \bmod N$.

Performance handicaps in Dixon's algorithm

Dixon's algorithm is nice for getting a proven algorithm.

However, performance-wise, it suffers from the large size of $r^2 \bmod N$.

- $r^2 \bmod N \approx N$

Performance handicaps in Dixon's algorithm

Dixon's algorithm is nice for getting a proven algorithm.

However, performance-wise, it suffers from the large size of $r^2 \bmod N$.

- $r^2 \bmod N \approx N$
- Hopefully we're trial-dividing...

A faster algorithm would appreciate smaller residues.

- Continued fractions give such a thing.
- The quadratic sieve also does this, and brings the sieving idea.

Plan

Congruences of squares

- The key idea
- Dixon's random squares algorithm
- Continued fractions
- The quadratic sieve idea
- Related stuff: Adleman's algorithm for DL

Continued fractions

Def. The continued fraction expansion of $x \in \mathbb{R}$ is the sequence of expressions

$$[a_0; a_1, \ldots; a_n] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}.$$

Where the expression ends with $1/a_n$ eventually and the integers a_i are obtained by the iteration

$$a_i = \lfloor x_i \rfloor, \qquad x_{i+1} = \frac{1}{x_i - a_i}.$$

Continued fractions (cont'd)

Countless facts and identities.

- The rational number $\frac{p_n}{q_n} = [a_0; a_1, \ldots; a_n]$ is called the n-th convergent. (convergents converge towards x).
- $x \in \mathbb{Q} \Leftrightarrow$ CFE is finite.
- $[\mathbb{Q}(x) : \mathbb{Q}] = 2 \Leftrightarrow$ CFE is eventually periodic.
- $p_n q_{n-1} - p_{n-1} q_n = (-1)^n$ (hence $\gcd(p_n, q_n) = 1$).
- $\left| x - \frac{p_n}{q_n} \right| < \frac{1}{q_n q_{n+1}}$.

The latter yields: $\left| p_n^2 - x^2 q_n^2 \right| < 2x$ for $x > 1$ and $n \geq 0$.

CFRAC (Morrison-Brillhart)

CFRAC follows the same methodology as Dixon's algorithm, but uses the CFE for $x = \sqrt{kN}$ as a source of relations (assume $k = 1$ to start with).

- $Q_n = p_n^2 - x^2 q_n^2$ is an integer, $|Q_n| < 2\sqrt{kN}$.
- Thus Q_n is a square modulo N.
- Q_n, p_n can be computed using integer arithmetic (which is exact).
  (Note: this is slightly difficult to prove).

Algorithm:
- Select a factor base
  - For some k, compute $(p_n)_n$ and $(Q_n)_n$ (q_n not needed) for the CFE of $\sqrt{kN}$.
  - Whenever Q_n is smooth, output a relation.
  - Possibly repeat this with other values of k.
- enough relations ? $\Rightarrow$ solve the linear system.

CFRAC: look forward into analysis

The analysis for CFRAC will proceed the same way as we will do Dixon's.

- We changed the way to form residues.
  These are now $O(\sqrt{N})$ instead of $O(N)$.
- However, we can not prove that the residues are uniformly distributed: rigorous smoothness results will not hold.

Plan for analysis:

- Set B to be optimized.
- The size of the residues is ...
- The smoothness probability is ...
- The per-residue factoring cost is ...
- The total relation collection cost is ...
- The linear system cost is ...

In the end, complexity better than Dixon's (albeit heuristic).

Plan

Our dummy example was not so stupid

The quadratic sieve (Pomerance, 1983) is a combination of two things:

- First idea: pick a simple «naturally small» function:
  - Consider $|f(i)| = \left|\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N\right|$.
  - For $|x| \leq S \ll \sqrt{N}$, we have $|f(i)| \leq 2S\sqrt{N} + \epsilon$
- Second idea: Factor residues completely differently.
  - The process used is known as sieving.
  - Sieving eliminates the per-relation factoring cost.

We will study more size improvements with the MPQS (multiple polynomial QS) algorithm.

Plan

Context switch

We change context completely.

Discrete Logarithm Problem in $\mathbb{F}_p^\times$.

We have $\mathbb{F}_p^\times = \langle g \rangle$, and $a \in \mathbb{F}_p^\times$. Search for ℓ s.t. $a \equiv g^\ell \mod p$.

(more crypto-relevant: work in a subgroup $G < \mathbb{F}_p^\times$ of prime order q).

Similar framework:

- Fix a factor base bound B.
- Pick random values r, and keep those for which $g^r \bmod p$ is B-smooth.
- Aim at $\#\text{rels} = \#\text{FB}$ elements.

Solving the linear system gives logs for FB elements.
To compute $\log_g a$: Find r s.t. ag^r is B-smooth.

Differences DL versus factorisation

The most important difference is in linear algebra:

- No longer «congruences of squares». Matrix is over $\mathbb{F}_q$.
- Look for a right kernel, not a left kernel.

Note: as presented, Adleman's algorithm has:

- a precomputation stage: logs of FB elememts.
- an individual log stage: given a, find $\log_g a$.

Nice DL algorithms nowadays keep this small per-logarithm cost.

Plan

Plan

Analyzing smoothness-based algorithm

Smooth numbers

Analyzing the algorithms

A crucial species: smooth numbers

Def. A B-smooth number N has all its prime factors $\leq B$.

de Bruijn's function: $\psi(x, y) = \mathrm{Card}\{N \leq x, p \mid N \Rightarrow p \leq y\}$.

Main theorem (Canfield, Erdős, Pomerance)

For all $\varepsilon > 0$, uniformly in $y \geq (\log x)^{1+\varepsilon}$, when $x \to \infty$

$$\psi(x, y)/x = u^{-u(1+o(1))}, \quad u = \log x / \log y.$$

A useful function:

$$L_x(\alpha, c) = \exp\left(c(\log x)^\alpha (\log\log x)^{1-\alpha}\right).$$

Prop. For all $\alpha > 0$, $\beta > 0$, when $x \to \infty$

$$\psi(x^\alpha, L_x(1/2, \beta))/x^\alpha = L_x(1/2, \frac{-\alpha}{2\beta} + o(1)).$$

More on L

Handy function introduced (we think) by R. Schroeppel:

$$L_x(\alpha, c) = \exp\left(c(\log x)^\alpha(\log\log x)^{1-\alpha}\right).$$

- $L_x(0, c) =$ polynomial in $\log x$.
- $L_x(1, c) =$ exponential in $\log x$.
- L is often called the sub-exponential function.

Reformulation of C-E-P

An integer $\leq L_x(\alpha, u)$ is $L_x(\beta, v)$-smooth with probability:

$$L_x(\alpha-\beta, -\frac{u}{v}(\alpha-\beta))^{1+o(1)}.$$

For example, a random integer modulo N has a probability $L_N(1/2, \cdot)$ of being $L_N(1/2, \cdot)$-smooth.

Plan

Random squares algorithm: analysis

We have $r^2 \bmod N \approx L_N[1;1]$. Set the smoothness bound $B = L_N[\beta, b]$.

- $\text{Prob}(r^2 \bmod N \text{ is } B\text{-smooth}) = L_N[1-\beta; -\frac{1}{b}(1-\beta) + o(1)]$.
- The cost for testing a relation is $O(B) = L_N[\beta, b]$.

We need $\approx B$ relations $\Rightarrow$ relation collection $L_N[\beta; b]^2 \times L_N[1-\beta; \frac{1}{b}(1-\beta)]$.

- This constrains $\beta = \frac{1}{2} \Rightarrow$ rel. collec. $= L_N[1/2, 2b + \frac{1}{2b}]$.
- Assume an $n \times n$ linear system can be solved in $O(n^\omega)$. Then a kernel element is found in time $L_N[1/2, \omega b]$.
- Total $L_N[1/2, 2b + \frac{1}{2b}] + L_N[1/2, \omega b]$.

For $\omega = 3$, optimum is $b = \frac{1}{2}$. Total complexity $L[1/2, 2]$.

Several remarks

Dixon's algorithm has the advantage of being simply stated, and proven.

Yet, in $L[1/2, \mathbf{2}]$ the constant is large, and affected by several things which can be improved:

- Size of the residues to be factored ;
- Method used for factoring ;
- Method used for solving a linear system.

Plan

Some exercises to finish the lecture

Exercise 1

Consider an algorithm which obtains one relation each time it finds a B-smooth number, of size X, with

$$B = L_N[\beta, b], \quad X = L_N[\sigma, s].$$

Question: how many non-zero entries per row do we have in the matrix, w.r.t. the number of rows/cols ?

Exercise 2

Given Moore's law, and assuming no algorithmic advances, how should the size of a number N evolve as a function of the number of years during which we want it to remain impossible to factor.

More exercises

Exercise 3

Provide an algorithm solving the «nearby p, q» case (see earlier slide).

Exercise 4

Provide a heuristic analysis of CFRAC.