Estimates for factoring 1024-bit integers

Thorsten Kleinjung, University of Bonn

Contents

GNFS Overview

Polynomial selection, matrix construction, square root computation

Sieving and cofactoring

- Strategies for cofactoring
- Estimates for different factor base sizes

Matrix step

Summary

Problem: factor

$$
\begin{aligned}
N = {} & 1350664108659952233496032162788059699388814756056670275244851 \\
& 4385152651060485953383394028715057190944179820728216447155137 \\
& 3680419703964191743046496589274256239341020864383202110372958 \\
& 7257623585096431105640735015081875106765946292055636855294752 \\
& 1350085287941637732853390610975054433499981115005697723689092 \\
& 7563
\end{aligned}
$$

Available resources:
PC = 2.2 GHz Athlon 64 CPU, $\leq$ 2 GB memory

Time: 1 or 2 years

How many PCs do we need?

GNFS Overview

1. Polynomial selection
2. Collection of relations
3. Construction of the matrix
4. Matrix step
5. Rest of computation (square root)

GNFS Overview

1.	Polynomial selection	easy
2.	Collection of relations	hard
3.	Construction of the matrix	easy
4.	Matrix step	HARD
5.	Rest of computation (square root)	easy

Polynomial selection

$$
\begin{aligned}
f_1 \quad = \quad & 1000000001002023904806000x^6 \\
& +2696978952367681630566064163 40x^5 \\
& -6212838818608524196100227896844747498x^4 \\
& -847105251394275537650757048185246266 8136x^3 \\
& +7386089168513102555044082528893786797012 3111795x^2 \\
& +10323950425845926908896158377241426163762 4065053206x \\
& -1139431985616391987769376205036438729670911719012 77555912
\end{aligned}
$$

of degree $d_1 = 6$ and

$$
\begin{aligned}
f_2 \quad = \quad & 514662055961724717752552412597334861x \\
& -22651198301463826278447637231994318097020 5534545
\end{aligned}
$$

of degree $d_2 = 1$

Much more time for polynomial selection will probably give a polynomial pair whose yield is twice as high.

Construction of the matrix and square root computation

Construction of the matrix

- Have to process 10-500 TB of sieving data
- Some parts can be done during sieving phase
- Much easier than matrix step

Square root computation

- Can be parallelized (easy)
- Can be done in a few months on one PC

Sieving and cofactoring

Aim: Find many pairs (a, b), a, b coprime, such that $F_1(a, b)$ and $F_2(a, b)$ are L-smooth.
(In this talk: $L = 2^{42}$, i.e., smooth=split completely into prime factors $< 2^{42}$)

1. Sieving:
   - finds divisors $< B_i$ of $F_i(a, b)$
   - discards (a, b) if not "enough" divisors are found
2. For each surviving (a, b) compute
   $F_i(a, b) = S_i R_i \quad$ (divisors $< B_i$ in S_i),
   compositeness tests for R_1, R_2
3. Cofactoring:
   - tries to factor (R_1, R_2)
   - discards (a, b) if a factor $> L$ is encountered

Restrictions

Only ≤ 2 GB memory $\Rightarrow$ must choose B_i smaller than "optimal"
("optimal" B_i would require 64 GB)

If B_i are small, traditional bounds for R_i will give a very low yield.

$\Rightarrow$ increase bounds for R_i

$\Rightarrow$ cofactoring needs a lot of time

We need a good strategy for cofactoring.

Cofactoring

Problem: Determine whether (R_1, R_2) is L-smooth.

Many available methods for factoring small numbers:

- MPQS: run time depends on size of input, "always" succeeds
- Pollards $p-1$: additional parameters, run time depends on parameters and size of input, success rate depends on parameters and size of prime factors of input number
- ECM: similar to $p-1$, but can be used several times for the same input number
- others

Which strategy shall we use to factor R_1 and R_2?

Example:

Available factoring algorithms:

- MPQS
- Pollards $p-1$ $\quad (B_1 = 500,\ B_2 = 10000)$

Given: (R_1, R_2), not prime, no prime divisor $< 2^{30}$.

$2^{63} < R_1 < 2^{64}$ (smooth) $\quad R_2 = 1$

Example:

Available factoring algorithms:

- MPQS
- Pollards $p-1$ $\quad (B_1 = 500,\ B_2 = 10000)$

Given: (R_1, R_2), not prime, no prime divisor $< 2^{30}$.

$2^{63} < R_1 < 2^{64}$ (smooth) $\quad R_2 = 1$

Strategy 1: factor R_1 by MPQS

time $= 192\mu$s $\qquad$ yield $= 1$

Strategy 2: use $p-1$, on failure use MPQS

time $=?$ $\qquad$ yield $= 1$

Details for $p-1$ $\quad (B_1 = 500,\ B_2 = 10000)$

time $= 27.3\mu$s (for 64-bit numbers)

probability to find a b-bit factor:

b	probability
31	0.135
32	0.110
33	0.089
34	0.073

64-bit integers (composite, no prime divisor $< 2^{30}$)

(b_1, b_2)	# (64-bit integers being a product of a b_1-bit prime and a b_2-bit prime)
$(31, 34)$	$7.35 \cdot 10^{15}$
$(32, 33)$	$7.33 \cdot 10^{15}$
$(31, 33)$	$5.89 \cdot 10^{15}$
$(32, 32)$	$2.94 \cdot 10^{15}$

64-bit integers (composite, no prime divisor $< 2^{30}$)

(b_1, b_2)	# (64-bit integers being a product of a b_1-bit prime and a b_2-bit prime)
$(31, 34)$	$7.35 \cdot 10^{15}$
$(32, 33)$	$7.33 \cdot 10^{15}$
$(31, 33)$	$5.89 \cdot 10^{15}$
$(32, 32)$	$2.94 \cdot 10^{15}$

$\Rightarrow$ probability of success for $p-1$: 0.2

Example:

Available factoring algorithms: MPQS and Pollards $p-1$

Given: (R_1, R_2), not prime, no prime divisor $< 2^{30}$.

$2^{63} < R_1 < 2^{64}$ (smooth) $\quad R_2 = 1$

Strategy 1: factor R_1 by MPQS

time $= 192\mu$s $\qquad$ yield $= 1$

Strategy 2: use $p-1$, on failure use MPQS

time $= 181\mu$s $\qquad$ yield $= 1$

Example:

Available factoring algorithms: MPQS and Pollards $p-1$

Given: (R_1, R_2), not prime, no prime divisor $< 2^{30}$.

$2^{63} < R_1 < 2^{64}$ (smooth) $\quad R_2 = 1$

Strategy 1: factor R_1 by MPQS

time $= 192\mu$s $\qquad$ yield $= 1$

Strategy 2: use $p-1$, on failure use MPQS

time $= 181\mu$s $\qquad$ yield $= 1$

Strategy 3: use $p-1$

time $= 27.3\mu$s $\qquad$ yield $= 0.2$

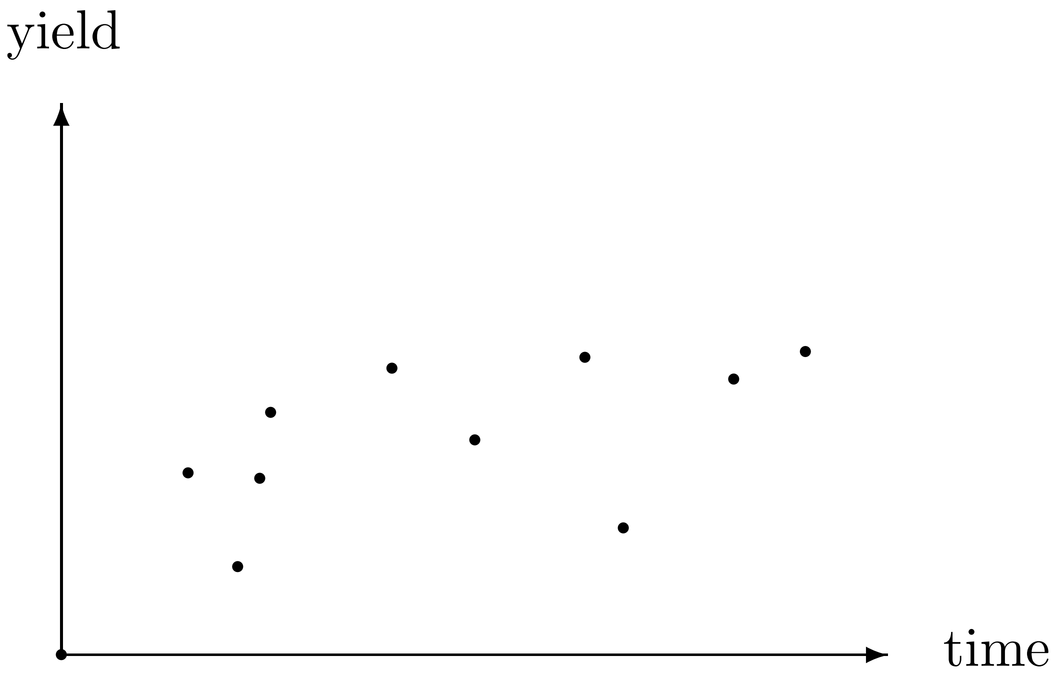
Strategy 4: do nothing

time $= 0\mu$s $\qquad$ yield $= 0$

In general:

many available factoring methods

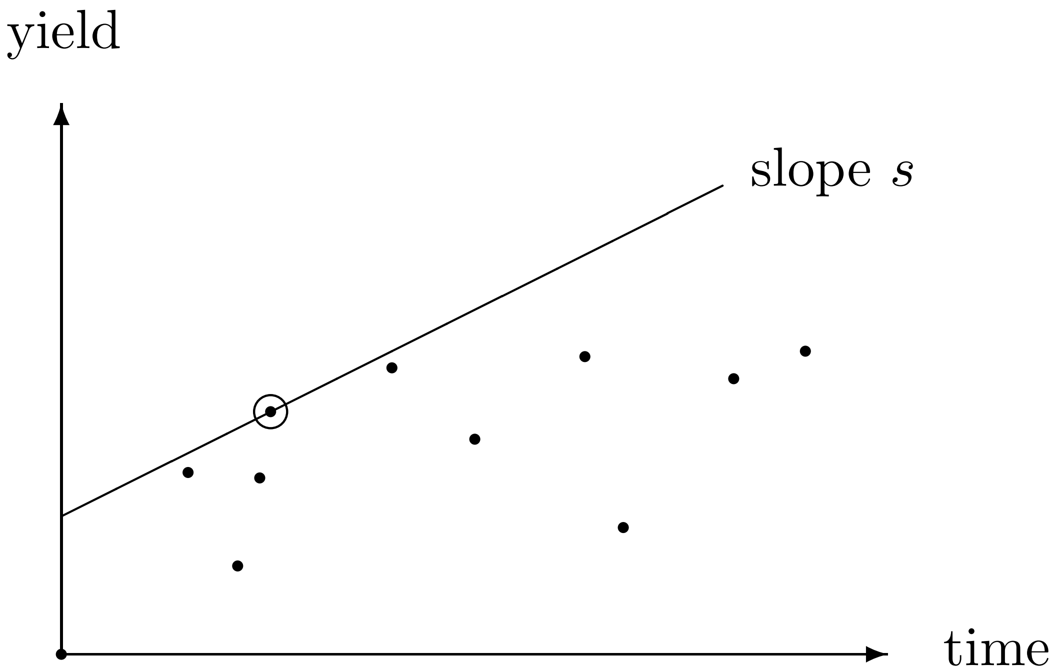
$\Rightarrow$ many strategies



Strategies for bit length (r_1, r_2)

Optimal strategy:

There exists an s such that



Optimal strategy: on line of slope s such that no point above line

Sieving experiments for 1024-bit number N

Large prime bounds: 2^{42}

Lattice sieving area: $2^{16} \times 2^{15}$

Prime factors of special q in $[2^{20}, 2^{32}]$

Memory	(B_1, B_2)	special q	bounds for (R_1, R_2)
2 GB	$(1.15 \cdot 10^9, 250 \cdot 10^6)$	$[50 \cdot 10^{12}, 260 \cdot 10^{12}]$	$(2^{200}, 2^{160})$
1 GB	$(450 \cdot 10^6, 100 \cdot 10^6)$	$[50 \cdot 10^{12}, 350 \cdot 10^{12}]$	$(2^{200}, 2^{160})$
512 MB	$(200 \cdot 10^6, 50 \cdot 10^6)$	$[50 \cdot 10^{12}, 400 \cdot 10^{12}]$	$(2^{210}, 2^{180})$

Sieving experiments for 1024-bit number N

Large prime bounds: 2^{42}

Lattice sieving area: $2^{16} \times 2^{15}$

Prime factors of special q in $[2^{20}, 2^{32}]$

Memory	(B_1, B_2)	special q	bounds for (R_1, R_2)
2 GB	$(1.15 \cdot 10^9, 250 \cdot 10^6)$	$[50 \cdot 10^{12}, 260 \cdot 10^{12}]$	$(2^{200}, 2^{160})$
1 GB	$(450 \cdot 10^6, 100 \cdot 10^6)$	$[50 \cdot 10^{12}, 350 \cdot 10^{12}]$	$(2^{200}, 2^{160})$
512 MB	$(200 \cdot 10^6, 50 \cdot 10^6)$	$[50 \cdot 10^{12}, 400 \cdot 10^{12}]$	$(2^{210}, 2^{180})$

Memory	number of sp. q	time per sp. q	number of PCs
2 GB	$1.95 \cdot 10^{12}$	135s	$8.4 \cdot 10^6$
1 GB	$2.85 \cdot 10^{12}$	111s	$10 \cdot 10^6$
512 MB	$3.3 \cdot 10^{12}$	116s	$12 \cdot 10^6$

Matrix step

Extrapolate matrix size from factorisations of large numbers

Get between $6 \cdot 10^9$ and $12 \cdot 10^9$ rows/columns

Assumption: $d = 8 \cdot 10^9$ rows/columns, $w = 1.2 \cdot 10^{12}$ non-zero entries

$\Rightarrow$ need 4-5 TB to store the matrix

Matrix step - Block Wiedemann algorithm

Input: $d \times d$ matrix M over $\mathbb{F}_2$, output: solution(s) of $Mv = 0$

1. Choose random vectors $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$, some conditions.
2. Compute scalar products $\langle x_k, M^i y_l \rangle$ for $i = 1, \ldots, (\frac{1}{m} + \frac{1}{n})d$.
3. Find linear combinations of $M^i y_l$, orthogonal to enough $x_k (M^T)^j$ (Berlekamp-Massey).
4. Compute linear combinations $z_l = \sum_{i=1}^{\frac{d}{n}} a_{il} M^i y_l$.
5. Now $M^{small} z_l = 0$, find up to n solutions from $M^i z_l$, i small.

Matrix step - Parameters

Complexity ($d = 8 \cdot 10^9$, $w = 1.2 \cdot 10^{12}$):

Step	XOR	space
2	$\frac{m+n}{m} dw$	w
3	$O(\frac{(m+n)^3}{n} d^{1+\epsilon})$	$O(\frac{(m+n)^2}{n} d)$
4	dw	w

Choose $m = n = 8192$

Berlekamp-Massey algorithm (step 3):

- 1 PC with 500 TB disk space needs 500 years
- Can be parallelized (might be hard)
- Parts can be done during step 2

Matrix step - Matrix$\times$vector multiplication

Use 1024 clusters, each:

- 64×64 PCs, each 1.5 GB memory
- Gigabit network, torus topology
- handles 8 start vectors, i.e., 2 million multiplications in step 2 and 1 million in step 4

Extrapolation from existing clusters:

- Computation per multiplication: 3s
- Communication per multiplication: 5s

$\Rightarrow$ Time: 6 months for step 2 and 3 months for step 4

Matrix step - Problems

Computing errors:

- Orthogonality checks
- Use linearly dependent start vectors y_l
- Can check intermediate results in Berlekamp-Massey

Hardware failure:

- Store intermediate results frequently
- Backup PCs
- Use linearly dependent start vectors (as above)
- Several Berlekamp-Massey jobs

Summary

Main problems for factoring 1024-bit intergers:

1. Collecting relations
2. Matrix step

- One can do the collecting of relations with 8.4 million PCs in one year.
- One might be able to do the matrix step with 1024 clusters, each consisting of 4096-8192 PCs, in one year.