CSE291-14: The Number Field Sieve

https://cseweb.ucsd.edu/classes/wi22/cse291-14

Emmanuel Thomé

March 8, 2022

Part 9

Variants of NFS

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

Main arguments about complexity of NFS

The cost the factorization of N with GNFS is:

$$L_N(1/3, (64/9)^{1/3} + o(1))$$

as a function of N, as N grows to infinity.

However, NFS for factoring is not the whole story.

- We mentioned NFS-DL over finite fields.
- SNFS is also a special case.
- and there are more weird ways to use NFS, including algebraic curves.

The complexity is $L(1/3, c + o(1))$ most of the time, but with varying c.

CEP and analogues

The Canfield-Erdős-Pomerance theorem is fundamentally important for the analysis of NFS.

CEP with the L function

A random integer $n \leq L_x(a, \alpha)$ is $L_x(b, \beta)$-smooth with probability:

$$L_x\left(a - b, -\frac{\alpha}{\beta}(a-b)(1 + o(1))\right).$$

This theorem has useful analogues in other contexts:

- Number fields: we saw it with NFS.
  The smoothness probability according to given norm bounds obeys the same rules.
- Polynomials over finite fields.
- Function fields.

This goes very heuristic, very quickly

Significant caveat about the extension of CEP to number fields or function fields:

- we can only say that this is valid for a fixed number/function field.
- the uniformity of the result across a range of number/function fields is a more difficult aspect. Note that this typically depends on our input!

As a general rule, a lot of these more subtle things are either proven, or at least more accessible to proofs in the polynomial / function field case than in the integer / number field case.

The more down-to-earth approach is not too much concerned about these subtleties, since anyway a lot of things are really heuristic.

Main questions for complexity analysis

- What is the target on the bottom of the diagram?
- What mathematical objects do we have on both sides?
- What objects are we trying to decompose, along which basis?
- What are the sizes (on both sides)?

Notations

We're reusing some notations that we already used for the analysis:

- $\log_{\log N} L_N(1/3, \delta)$ algebraic degree.
- $L_N(1/3, \beta)$ smoothness bound $\approx \#$ relations.
- $L_N(1/3, \alpha)$ bounds on a and b in $a - bx$.

Asymptotically, relation collection and linear algebra are always in the same ballpark, so we may think of α and β as being equal.

A comparison point

Recall the key asymptotic aspects of the analysis of GNFS.

- One rational side, and one algebraic side of degree

$$d = \log_{\log N} L_N(1/3, \delta + o(1)) = (\delta + o(1)) \left(\frac{\log N}{\log \log N} \right)^{1/3}.$$

- ...

A comparison point

- Coefficients of f_0 and f_1 both $\approx m = L_N(2/3, \frac{1}{\delta} + o(1))$.
- Two norms, of absolute values

$$|\operatorname{Res}(a - bx, f_0)| \approx L_N(2/3, \frac{1}{\delta} + o(1))$$
$$|\operatorname{Res}(a - bx, f_1)| \approx L_N(2/3, \frac{1}{\delta} + \alpha\delta + o(1)).$$

- Choose δ to minimize the product of norms, and eventually equate the remaining things to get the final asymptotic complexity estimate.

The size of the norms matters!

The key thing is really $\frac{1}{\delta} + \frac{1}{\delta} + \alpha\delta$. How it connects to the final complexity is not the important part of the machinery.

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

The Special Number Field Sieve (SNFS)

Remember: NFS started with a very exceptional case.

$$N = 2^{128} + 1, \quad f_0 = x - 2^{43}, \quad f_1 = x^3 + 2.$$

The coefficients of f_1 are extremely small.

- It is highly unlikely to find such a good polynomial pair with other random integers of that size.
- Argument: Say we constrain coefficients of f_0 and f_1 to absolute values 2^{43} and 2: only $2^{43\times 2+1\times 4} = 2^{90}$ integers.

Let us define a family of integers such that this sort of extraordinary things are possible.

An SNFS family

Let $(N_i)_{i \geq 0}$ be a family of integers such that...

- Informally: the NFS analysis works in an almost dream-like way.
- More formally: we'll try to do this dream analysis and come back to what the exact definition of the family should be.

An SNFS family

Let $(N_i)_{i\geq 0}$ be a family of integers such that...

- There exists a polynomial f_1 of degree $d = \log_{\log N} L_N(1/3, \delta + o(1))$, for some δ to be determined...
- with coefficients in $L_N(2/3, o(1))$...
- and such that there is a polynomial f_0 with coefficients just large enough so that $\operatorname{Res}(f_0, f_1) = N$. That means coefficients of size $L_N(2/3, 1/\delta)$.

Then in this case, we end up with:

$$|\operatorname{Res}(a - bx, f_0)| \approx L_N(2/3, \frac{1}{\delta} + o(1))$$
$$|\operatorname{Res}(a - bx, f_1)| \approx L_N(2/3, \alpha\delta + o(1)).$$

Consequence for the analysis

If all of this actually happens:

- $\frac{1}{\delta} + \alpha\delta$ is minimized with $\delta = 1/\sqrt{\alpha}$.
- The smoothness probability is $\frac{2\sqrt{\alpha}}{3\beta}$, and the constraint is:

$$2\alpha - \frac{2\sqrt{\alpha}}{3\beta} = \beta.$$

- We finish with $\alpha = \beta$ and $2\sqrt{\alpha} = 3\beta^2$, which leads $\beta = (2/3)^{2/3} = (4/9)^{1/3}$ and:

$$2\beta = (32/9)^{1/3}.$$

If we go backwards, this gives $\delta = (3/2)^{1/3}$.

What is SNFS

A family of SNFS integers

Let $(N_i)_{i\geq 0}$ be a family of integers such that for each N_i:

- There exists a polynomial f_1 of degree $d = \log_{\log N_i} L_{N_i}(1/3, (3/2)^{1/3} + o(1))$.
- with coefficients in $L_{N_i}(2/3, o(1))\dots$
- and such that there is a polynomial f_0 with coefficients of size $L_{N_i}(2/3, 1/\delta)$ such that $\mathrm{Res}(f_0, f_1) = N_i$ (or a multiple of N_i).

Then the asymptotic cost of factoring an integer in this family is heuristically

$$L_{N_i}(1/3, (32/9)^{1/3} + o(1)).$$

Does this mean anything?

The previous definition is, in essence, an asymptotic one.

The question that gets often asked is

Let $N =$ blah. Is N an SNFS target?

The most useful answer is that if N can be obtained as the resultant of two polynomials with:

- an algebraic polynomial of a reasonable degree;
- an algebraic polynomial with super small coefficients;
- and a rational polynomial with "normal" coefficients

then YES, we're going to obtain something better than GNFS.

Examples of SNFS numbers

Integers of the form $r^e - s$ with r, s bounded by a constant are easily obtained as resultants of an SNFS polynomial pair.

Note however that the same is not true of their composite factors, which might be easier to tackle with GNFS!

Example: the 1214-bit integer $3^{766} + 1$ was factored using some ECM first, and then some (fairly easy) GNFS for the rest.

$$3^{766}+1=2\times 5\times 656822133606644237\times 2102130222907676881\times 687494068024331577741\times$$

$$27742388167909428696921049 69\times 1191349824998215027503985934 9\times$$

$$303865849289906663302787771616459 22849\times$$

$$30388934198614663079171397316787470704219965175523938580742484 2909\times$$

$$62694369818854031569735758211423458086661122500205782560473264292927420 9337\times$$

$$16432199973059274544674873215219567887126295353840312750528946833823987 57857$$

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

Analysis of NFS-DL

If we use the GNFS polynomial selection to compute discrete logarithms modulo a prime p, then the analysis (of the precomputation phase) is exactly the same, and N is replaced by p.

$$L_p(1/3, (64/9)^{1/3} + o(1)).$$

Analysis of NFS-DL (JL)

If d is the GNFS-factoring preferred degree, Joux-Lercier works with degrees D and $D+1$, with $D = d/2$.

- $\deg f_0 = D+1$, ridiculously small coefficients.
- $\deg f_1 = D$, coefficients around $p^{1/D}$.

Then for $|a| \approx |b| \approx A$, we have:

$$|\operatorname{Res}(a-bx, f_0)| \approx A^{D+1}$$
$$|\operatorname{Res}(a-bx, f_1)| \approx A^D p^{1/D}$$

And then

$$A^{2D+1} p^{1/D} \approx L_p(2/3, \alpha\delta + \frac{2}{\delta}).$$

Same as GNFS-factoring, but the balance is a bit weird:

- $\alpha\delta$ spreads out between both sides,
- and $\frac{2}{\delta}$ is only on one side.

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

The (defunct) Function Field Sieve

FFS is an interesting variation of NFS.

We trade all number fields for function fields.

A function field is attached to a curve, with all the algebraic geometry theory that comes with it.

We are interested in high degree extensions of very small finite fields (think $\mathbb{F}_2$).

Note

The Function Field Sieve is now almost completely superseded by the quasi-polynomial algorithm(s) for DLP in small-characteristic fields.

Analogy

$N \in \mathbb{Z}$, n bits.	$H \in \mathbb{F}_2[x]$, $\deg H = n$.
$N = \text{Res}(f_0, f_1 \in \mathbb{Z}[x])$ $\phi = a - bx \in \mathbb{Z}[x]$ Number field defined by f_0	$H = \text{Res}_y(f_0, f_1 \in \mathbb{F}_2[x, y])$ $\phi = a(x) - b(x)y \in \mathbb{F}_2[x, y]$ Function field defined by f_0 (curve defined by f_0)
Ideal $\langle a - b\alpha \rangle \rightarrow$ prime ideals Norm p	Function $\phi(x, y) \rightarrow$ places $2^{\deg p(x)}$
Smoothness of integers Canfield-Erdős-Pomerance	Smoothness of polynomials (various analogues)

Analogy

$N \in \mathbb{Z}$, n bits.	$H \in \mathbb{F}_2[x]$, $\deg H = n$.
$N = \mathrm{Res}(f_0, f_1 \in \mathbb{Z}[x])$ $\phi = a - bx \in \mathbb{Z}[x]$ Number field defined by f_0	$H = \mathrm{Res}_y(f_0, f_1 \in \mathbb{F}_2[x, y])$ $\phi = a(x) - b(x)y \in \mathbb{F}_2[x, y]$ Function field defined by f_0 (curve defined by f_0)
Ideal $\langle a - b\alpha \rangle \rightarrow$ prime ideals Norm p	Function $\phi(x, y) \rightarrow$ places $2^{\deg p(x)}$
Smoothness of integers Canfield-Erdős-Pomerance	Smoothness of polynomials (various analogues)
All n-bit integers are different	All $\mathbb{F}_{2^n}$ are isomorphic!

Everything works pretty much the same, and actually some tricky things become easier (cofactorization).

FFS allows crazy things

Polynomial selection for FFS is very special.

- Decide on your dream sizes and degrees.
- Choose $f_0(x, y)$ at random.
- Choose $f_1(x, y)$ at random.
- Repeat until $\mathrm{Res}_y(f_0, f_1)$ has an irreducible factor with the desired degree.

This means that FFS reaches the SNFS complexity.

FFS works also for other fields (see later).

FFS allows crazy things

More fun stuff: x and y really have symmetric roles in FFS.

The following works (assuming correct degrees and so on):

- Choose $f_0(x, y) = y -$ (random polynomial in x).
- Choose $f_1(x, y) = x -$ (random polynomial in y).
- Work with $\phi(x, y)$ with well chosen degrees.

In that situation, both f_0 and f_1 define function fields of degree 1, meaning: $\mathbb{F}_2[x]$, really!

(in practice, we might prefer the larger degree choices, so that we can benefit from somewhat distorted smoothness situations, via analogues of the Murphy-α value).

Coppersmith's algorithm

Coppersmith's algorithm is a special case of FFS (but was invented 10 years earlier), working with the polynomials $f_0 = y^{2^k} - R(x)$ and $f_1 = y - x^h$.

- This is very special because f_0 has degree a power of two. In algebraic geometry terms: a purely inseparable cover of $\mathbb{P}^1$.
- In more concrete terms, this makes it possible to share the factor base between the two sides, which is a unique feature.
- The complexity is $L_{2^n}(1/3, (32/9)^{1/3} + o(1))$ when 2^n is such that 2^k is really the sweet spot for the degree. The general FFS is much more flexible.

Yet, Coppersmith's algorithm is a great landmark, being the very first $L(1/3)$ algorithm!

NFS/FFS on more things

The general methodology of finding smooth things also works on the number fields and function fields that we consider.

- In a family of number fields defined by an NFS-like polynomials (very important assumption!), we can:
  - try to factor $a - b\alpha$.
  - and eventually do fancy things like class group and unit group computations in time $L(1/3)$.
- In a family of high genus algebraic curves with defining polynomials having a prescribed form ($\deg_x H$ and $\deg_y H$ both somewhat large), we can mimic FFS.
  - We can decompose into low-degree prime divisors.
  - Thanks to a descent procedure, we can solve the DLP!

Note that in these cases, we deal with only one side.

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

Multiple kinds of finite fields

Multiple ways to create a k-bit finite field:

- $\mathbb{F}_p$, for a k-bit prime p.
- $\mathbb{F}_{2^k} = \mathbb{F}_2[x]/f(x)$ for an irreducible $f \in \mathbb{F}_2[x]$ of degree $n = k$.
- More generally, $\mathbb{F}_{p^n}$, with $n \log_2 p \approx k$.

$\mathbb{F}_p$ and $\mathbb{F}_{2^k}$ are two opposite ends of the spectrum.

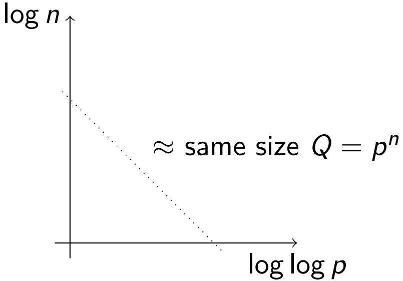
Fields of (roughly) the same size are along the line:

$$\log n + \log \log p = \log(k \log 2) = \text{constant}.$$

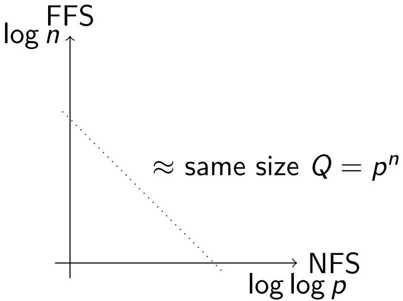
When we compare finite fields, we usually denote $Q = p^n$.

How does the DLP cost evolve as a function of Q?

Multiple kinds of finite fields



Multiple kinds of finite fields



- Prime fields: deg $= 1$, characteristic is large: NFS.
- Binary fields: $p = 2$, everything is in the degree: FFS.
- Does it extend to more fields?
  What happens when $\log\log p \approx \log n$?

Plan

Extensions to other finite fields

- Small characteristic
- Large characteristic
- Medium characteristic
- Small characteristic again

Easy: extending FFS

(reminder: FFS is dead!)

It is easy to reformulate FFS with some characteristic p larger than 2, and see what constraints must be put on p.

The FFS range

Answer: FFS works as long as

$$p \leq L_{Q=p^n}(1/3, o(1))$$

which we can also interpret as

$$\begin{aligned} \log n &\geq 2 \log \log p - \log o(1) + 2 \log \log \log Q. \\ &\geq 2 \log \log p + \text{away from 0 quickly enough.} \end{aligned}$$

FFS range: asymptotic meaning

The FFS range

If $(\mathbb{F}_{Q_i})_{i\geq 0}$ is a family of finite fields, with $Q_i = p_i^{n_i}$, and F is a function such that:

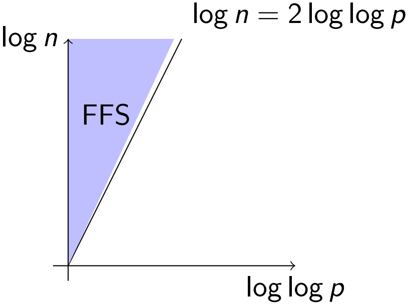
$$\begin{aligned} F(Q) &\leq L_Q(1/3, o(1)) \text{ (as } Q \to \infty) \\ p_i &\leq F(Q_i) \end{aligned}$$

Then precomputation phase of the DLP in $(\mathbb{F}_{Q_i})_{i\geq 0}$ costs:

$$L_{Q_i}(1/3, (32/9)^{1/3} + o(1)).$$

This is informally called the small characteristic range.

FFS range: graphically



Plan

Extensions to other finite fields

- Small characteristic
- Large characteristic
- Medium characteristic
- Small characteristic again

Extending NFS

There are various ways to deal with fields of rather large characteristic, and small degree.

- Extension proposed early on by Schirokauer.
- Better version by generalizing Joux-Lercier.
    - Choose d according to Q, pick $D = d/2$.
    - Pick small degree $D+1$ polynomial with a degree n factor mod p (notation: $\psi(x)$).
    - Find small vector in lattice of dimension $D+1$ and determinant Q spanned by $p, \ldots, p^n x$, and $\psi(x)$ and its multiples.
    - This works as long as $n \leq D+1$, which translates to

$$p \geq L_Q(2/3, o(1)).$$

NFS range: asymptotic meaning

The NFS range

If $(\mathbb{F}_{Q_i})_{i\geq 0}$ is a family of finite fields, with $Q_i = p_i^{n_i}$, and F is a function such that:

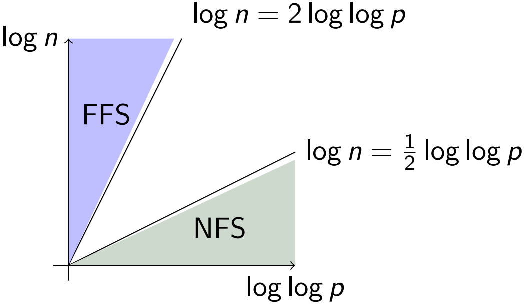
$$\begin{aligned} F(Q) &\geq L_Q(2/3, o(1)) \text{ (as } Q \to \infty) \\ p_i &\geq F(Q_i) \end{aligned}$$

Then precomputation phase of the DLP in $(\mathbb{F}_{Q_i})_{i\geq 0}$ costs:

$$L_{Q_i}(1/3, (64/9)^{1/3} + o(1)).$$

This is informally called the large characteristic range.

NFS range: graphically



Plan

Extensions to other finite fields

- Small characteristic
- Large characteristic
- Medium characteristic
- Small characteristic again

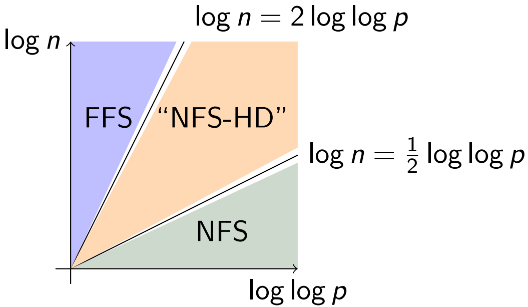
The remaining cases

The “middle case” was long thought as being much harder.

The first $L_Q(1/3)$ algorithm was found in 2006. Since in the middle case, p itself is less than $L_Q(2/3, o(1))$, it is actually quite simple:

- Take small f_0 with $\deg f_0 = n$, and simply take $f_1 = f_0 + p$.
- This is really as simple as it can get. Provided we are away from both boundaries, the complexity is $L_Q(1/3, (128/9)^{1/3} + o(1))$.
- This can be improved: other methods reach $L_Q(1/3, (96/9)^{1/3} + o(1))$

Boundaries are a nightmare



The complexity is easier to express away from the boundaries.

Many other variants

There are multiple variants of the above algorithms, with competing polynomial selection methods.

- Some adjustments for the boundary cases.
- Some finer-grain asymptotic estimates.
- Adaptation to composite degree.
- Adaptation to fields of SNFS-like characteristic.
- Use of towers of number fields.
- . . .

Plan

Extensions to other finite fields

- Small characteristic
- Large characteristic
- Medium characteristic
- Small characteristic again

The quasi-polynomial algorithm

In the FFS range, a different approached was introduced in 2013, reaching eventually quasi-polynomial complexity.

- First QP complexity by BGJT14.
- Multiple variants, practical improvements, and experimental results.
- The algorithm is so different that proving stuff is possible!

Theorem (Quasi-polynomial algorithm)

Given any p and any n, the DLP in $\mathbb{F}_{p^n}^\times$ can be solved in expected time

$$C_{QP}(p^n) = (pn)^{2\log_2 n + O(1)}.$$

This means that FFS is all but dead, except very close to the boundary.

Small, medium, large

Finite fields are not T-shirts, unfortunately.

The small, medium, and large characteristic regimes are asymptotic considerations!!
They only make real sense for an infinite family of target fields.

In practice:

- Comparing $\log \log p$ and $\log n$ may give a rough idea.
- In cases where the answer seems unclear, the only way to decide which method is best is by experimenting!

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

Multiple number fields

MNFS is a variant of NFS, proposed by Coppersmith in 1993.

- Asymptotically, this beats NFS.

  $$L_N(1/3, ((64/9)^{1/3} = 1.923) + o(1)) \rightarrow L_N(1/3, 1.902 + o(1)).$$

- This was later adapted to the discrete logarithm context.

The MNFS diagram

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl] \arrow[dr] \arrow[drr] \arrow[drrrr] & & & & \\
\mathbb{Q} \arrow[dr] & & \mathbb{Q}(\alpha_1) \arrow[dl] & \mathbb{Q}(\alpha_2) \arrow[dll] & \cdots & \mathbb{Q}(\alpha_k) \arrow[dllll] \\
 & \mathbb{Z}/N\mathbb{Z} & & & &
\end{tikzcd}$$

- Pick $a - bx$.
- See if $\mathrm{Res}(a - bx, f_0)$ is smooth (sieve).
- If yes, for $j \in [1, k]$:
  - See if $\mathrm{Res}(a - bx, f_j)$ is smooth.
  - If yes, we have a relation!
- Goal $(R(m))^2 \equiv (A_1(m) \cdots A_k(m))^2 \mod N$.

Basic idea

Informally, the idea is to “share” some computation.

Polynomial selection is not too hard.

- Pick a good polynomial pair (f_0, f_1).
- Derive f_j as $u_j f_0 + v_j f_1$ for a few small multipliers.
- We might as well adjust polynomial selection criteria a little bit.

Orders of magnitude

In Coppersmith's analysis, we need $L_N(1/3, 0.13\ldots)$ number fields (current range: this means dozens of number fields).

The rational side is easier, and is the one that is shared.

The algebraic sides have smaller smoothness bounds, yet the algebraic norms are bigger. Smoothness is rare!

What about practice?

Coppersmith's MNFS has never been used beyond very small proof-of-concept experiments.

Implementation and optimization is certainly not a trivial task, but not insurmountable either.

- Whether it is possible/useful to sieve is not entirely clear. Per-pair ECM, or product trees, can be ways to go.
- Linear algebra has to deal with a weird matrix which deserves special treatment.

The determination of the crossover point between the usual NFS and MNFS is completely open.

Plan

Introduction

SNFS

NFS-DL

The (defunct) Function Field Sieve

Extensions to other finite fields

The Multiple Number Field Sieve

$o(1)$

What is $o(1)$?

GNFS complexity is

$$L_N(1/3, (64/9)^{1/3} + o(1)).$$

Here, $o(1)$ is a function that tends to zero as N tends to ∞.

This is a lot different from an asymptotic complexity bound of the form $O(\text{blah})$. Here, we have a multiplicative unknown function that belongs to $L_N(1/3, o(1))$.

What can you put in $L_N(1/3, o(1))$?

$$L_N(1/3, o(1)) = \exp\left(o((\log N)^{1/3}(\log\log N)^{2/3})\right).$$

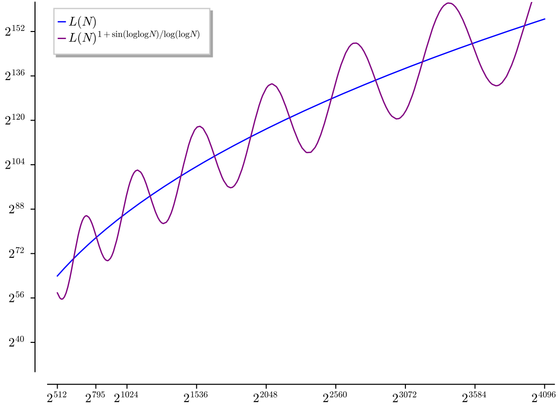
- Any polynomial in $\log N$ is in $L_N(1/3, o(1))$.
- Even super-polynomial $\exp((\log N)^x)$ for $x \leq 1/3$ is.
- Oscillatory behaviour can also be swallowed, e.g. the following completely imaginary multiplier is in $L_N(1/3, o(1))$:

$$2 - \cos\left(2\pi \cdot 3^{1/3}\left(\frac{\log N}{\log\log N}\right)^{1/3}\right).$$

(rationale: after all, what can we say of what happens when the “ideal” degree is halfway between two integers?)

We don't know the truth

Comparison of $L(N) = L_N(1/3, (64/9)^{1/3})$ with something that would be consistent with our generous asymptotic inaccuracy:



Does that matter?

FAQ: how expensive is RSA-$(n+200)$ compared to RSA-n?

We often do as if the cost were $c \cdot L(N)$ for some unknown constant c, instead of the not really useful $L(N)^{1+o(1)}$.

In the 1990s, some words of caution were usually going with these bodacious approximations.

This has since become the accepted practice for establishing key size recommendations.

- The usual RSA key size $\leftrightarrow$ symmetric key size correspondence table comes from here.
- Shorter range extrapolations usually do:

$$\text{cost at } n+k \text{ bits} \approx \frac{L(2^{n+k})}{L(2^n)} \times \text{cost of academic record at } n \text{ bits.}$$

What if we're totally wrong?

Here are two expressions, both consistent with $L(N)^{1+o(1)}$.

$$g_0(N) = L(N) \qquad g_1(N) = L(N)^{1/(1+22/\log\log N)}.$$

- Based on $g_0(N) = L(N)$, we can compute $\frac{g_0(2^{2048})}{g_0(2^{512})} \approx 2^{28}$. We're tempted to think that RSA-2048 is several hundred millions times harder than RSA-512.
- Based on $g_1(N)$, we can compute $\frac{g_1(2^{2048})}{g_1(2^{512})} \approx 2^{8}$. The conclusion is very different!
- What gives?

Where does $o(1)$ come from?

While $o(1)$ is used in several places to simplify calculus, its origin can be traced to smoothness estimates.

Smoothness (Hildebrand)

Under some conditions on x, y:

$$\frac{\Psi(2^x, 2^y)}{2^x} = \rho(x/y)\left(1 + O\left(\frac{\log(1+\frac{x}{y})}{y}\right)\right).$$

Furthermore, the asymptotic behaviour of ρ is

$$\rho(u) = \exp(-u \log u \cdot (1 + o(1))) = u^{-u\cdot(1+o(1))}.$$

The main inaccuracy is the asymptotic expansion of $\log\rho$.

More terms of $\log \rho$

Can we compute more terms of $\log \rho$? YES!

With some computer algebra, computing hundreds of terms in the asymptotic expansion of $\log \rho$ is eminently possible, eventually leading to more terms replacing $o(1)$ in the NFS asymtptotic complexity.

We get a bivariate series in $\frac{\log\log\log N}{\log\log N}$ and $\frac{1}{\log\log N}$.

Is it useful?

What if we replace $o(1)$ by (say) a hundred terms of the series for our estimate?

Bad news: for $N < \exp(\exp(22)) \approx 2^{5 \text{ billion bits}}$, the series diverges!

The (log of the) NFS complexity is approximated by the first term of a divergent series (in the practical range).

Note that this is not unlike our example function $g_1(N)$.

The main problem is our attempt to write a self-contained asymptotic estimate for the probability of smoothness, seeking a closed formula. Explicit numerical estimates of smoothness probabilities should be of better value.