Large-scale computational records for public-key cryptography: current state of the art, and further directions

material from ia.cr/2020/697 and arxiv.org:2007.02730

Emmanuel Thomé
Université de Lorraine, CNRS, Inria, Nancy, France

Crypto Readind Club meeting (virtual)
March 2024

Plan

Integer factoring and dlog

The Number Field Sieve (NFS)

Some key parameter choices

Moore's law?

Where can we go from there?

Number theoretic problems for public-key crypto

For about 40 years, we've been used to having:

- Integer Factorization (IF)
  Compute prime factors p and q of a large composite integer N.
- and (finite field) discrete logarithms (FF-DLP)
  Compute x such that $g^x = a$ given some p, and $a, g \in \mathbb{Z}/p\mathbb{Z}^\times$.

as prominent mathematical problems for public-key cryptography.

Hardness of IF is the security assumption behind RSA, FF-DLP backs Diffie-Hellman, DSA, and others.

Still relevant ?

Myth: this is all quaint, everybody uses EC or even PQ crypto by now.

Fact: pervasive software and hardware do rely on IF and DLP: TLS, SSH, IPsec, code signing, . . .

. . . now and for quite a few years to come.

How does one choose key sizes?

Tricky questions for public-key crypto

How does one assess the hardness of cryptanalysis, for key sizes that are (fortunately) out of its reach?
How does one make this assessment convincing?

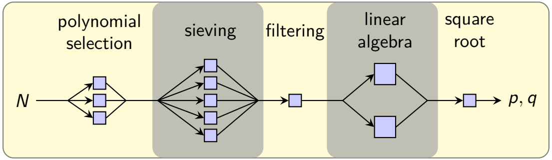
Several possible strategies (non-exclusive).

- Use complexity formulas for the best known cryptanalytic methods, and extrapolate from known computations.
- Simulate the cryptanalysis algorithms, and deduce the cost of cryptanalysis now and tomorrow.
- Improve implementations of algorithms, an demonstrate how/if they can scale.

Summary of NFS

To factor an integer N, or to compute discrete logs in $\mathbb{F}_p^\times$, we use
The Number Field Sieve (NFS).

The NFS algorithm (1990) easily fills a one-semester graduate course.



Complexity of NFS

The time it takes to run NFS

The complexity $C(N)$ of NFS to factor an integer N is

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

$$\text{with } L_N(1/3, (64/9)^{1/3}) = \exp\left((64/9)^{1/3}(\log N)^{1/3}(\log\log N)^{2/3}\right).$$

Computing discrete logs in $\mathbb{F}_p$ for $p \approx N$ can be done with mostly the same algorithm, with the same first-order asymptotic estimate.

We'll come back to this formula later.

Simulation

The major obstacle to simulating NFS is that the algorithm has many steps, with diverse computational requirements.

- Back-of-the-envelope simulations, or claims resembling "in theory, I could do that" are rarely taken seriously.
- Technological stumbling blocks are easily cited as reasons to not believe the simulation results (memory access cost, wafer size for ASIC designs, . . .).
- In record computations, the size of intermediate data is quite often badly anticipated. The credibility of simulations and predicted running times is affected by this.

We need hard facts

Predictions should be based on state-of-the-art software implementation performance.

We need actual software that is fit for large sizes, together with convincing computational results.

- Explore algorithmic ideas that pay off only for large sizes.
- Explore scalability, try to address stumbling blocks.
- Harness large computing power, show that this is more than just theory.
- Make our work reproducible.

In this talk

- State of the art of NFS calculations: how?
- Is it just about Moore's law?
- Where can we go from there?

State of the art

- Integer factoring:
  - RSA-250 (829 bits) factored in February 2020, approx. 2900 core-years;
  - RSA-240 (795 bits) factored in November 2019, approx. 1000 core-years;
  - A 232-digit modulus (RSA-768) factored in December 2009, approx. 1500 core-years.
- Discrete logarithms in prime fields:
  - a 240-digit (795 bits) prime: in November 2019, approx. 3100 core-years.
  - a 232-digit prime (768 bits) prime: in June 2016, approx. 5300 core-years.

Plan

Integer factoring and dlog

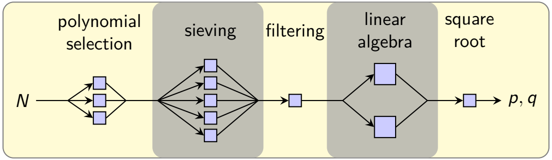
The Number Field Sieve (NFS)

Some key parameter choices

Moore's law?

Where can we go from there?

Summary of NFS



Computational requirements are diverse.

- Sieving (relation collection) is the most expensive. It can be massively distributed.
- (Sparse) Linear algebra comes second.
  It is somewhat cheaper, but needs expensive hardware.
- There are also many auxiliary tasks.

What kind of *relations* does NFS collect?

Polynomial selection finds f with a known root $m \mod N$.
Let $\mathbb{Q}(\alpha)$ be the number field defined by f.

Bird's eye view of strategy for factoring

- Search for pairs of integers (a, b) such that

$$\begin{array}{ccc} a - bm & \text{and} & a - b\alpha \\ \text{(an integer)} & & \text{(ideal in } \mathbb{Q}(\alpha)\text{, of norm } \operatorname{Res}(a - bx, f(x))) \end{array}$$

  are both smooth: they factor into small things. Pairs yield relations.
- Combine relations so that all multiplicities are even.
  We (almost) have squares on both sides.
- With further (easy) work, we find many equalities of squares: $u^2 \equiv v^2 \mod N$
  leads to factors of N with probability at least $\frac{1}{2}$.

Relations in NFS

Relations in NFS

Most of NFS is about collecting:

- pairs of integers (a, b)
- such that two integers derived from (a, b) are smooth.

How do we achieve this for larger and larger problem sizes?

Factoring large numbers?

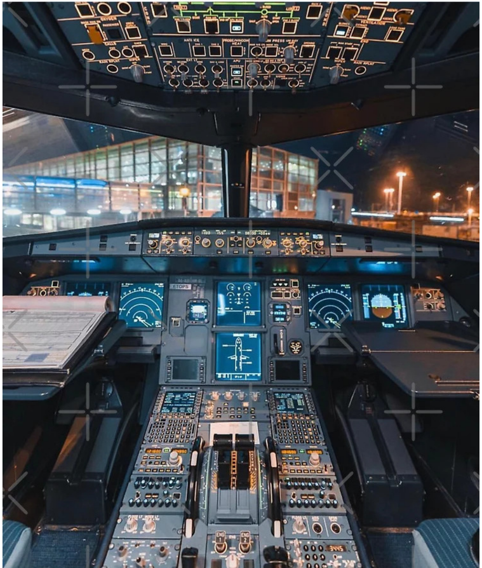
Fantasy: NFS is like a RC airplane.

- You power it up, it flies.
- If you want it to fly higher, the same plane will do.

Factoring large numbers?

Reality: NFS is more like an airliner.

- LOTS of controls.
- Toggling them at random does not get you very far.



Record computations

In contrast with computations that you hope can be one-man efforts with modest resources, record computations are inherently more heavy.

- Software has to include special-purpose stuff:
  - hardware platforms for large computations are not necessarily the same as the resources in a university basement;
  - some improvements start to pay off only for large sizes;
  - for a record computation, we're certainly willing to trade usability for performance.
- How the software gets run is sometimes an important part of the story.

cado-nfs

Enter cado-nfs, a complete, entirely original implementation of NFS.

- Work started in 2007.
- Open development on gitlab/github. LGPL license.
- As of today, about 275,000 lines of code (excluding generated code), about 18,000 commits.
- Three main authors (PG, ET, PZ), and many contributors.
- Has been able to tackle record computations since around 2015.

Free software for record computations: why?

Motivation for doing this with freely available code, which others rarely do (if ever):

- Reproducibility. Data for the latest RSA-240, DLP-240, RSA-250 is online.
- Software can be used by other record computations that use slightly different methods, but otherwise a similar framework.
- Even if not for computing records, having a tool that aims high is a good way to show that some old key sizes are really truly broken.
- Or just "for parts". Carry building blocks of cado-nfs to other areas (and also the other way around).

Plan

Integer factoring and dlog

The Number Field Sieve (NFS)

Some key parameter choices

Moore's law?

Where can we go from there?

Collecting relations

Relation collection is the most expensive step of NFS.

Description of relation collection

1. How do we divide the work?
2. How do we find smooth $a - bm$ and $a - b\alpha$?
3. How do we choose parameters so that the cost of linear algebra remains under control?

1. (many) needles in a (huge) haystack

Searching a space of size (say) 2^{65} takes long.

- Trivial strategy (loop over a, then b) has unstable yield and does not work well.
- Better: constrain a factor q in one of the factorizations.
  - ✓ Independent tasks per q.
  - ✓ Yield is stable.
  - ✓ The prescribed factor is one thing less to find!

  (old folklore; records have been doing this for decades.)

  ($\Rightarrow$ special-q sieving, lattice sieving, sieving by vectors.)

2. Finding smooth (a, b)

We fix some q.

We explore many (a, b) such that q appears somewhere (it is a lattice in $\mathbb{Z}^2$).

We want $a - bm$ and $a - b\alpha$ to be smooth.
The strategy depends on potential prime factors p in these factorizations.

A prime should appear either often, or very rarely.

- below some bound, for $p < B$, strive to find all pairs (a, b) such that p appears. We typically use sieving. So-called bucket sieving is key.
- "large primes" (LPs) such that $B \leq p < L$ are allowed if we happen to find them. We limit to a few such LPs per relation (e.g., 2, sometimes 3).

The relations that we like to see

✓	$5^2 \cdot 11 \cdot 23 \cdot 287093 \cdot 870953 \cdot 20179693 \cdot 28306698811 \cdot 47988583469$	$2^3 \cdot 5 \cdot 7 \cdot 13 \cdot 31 \cdot 61 \cdot 14407 \cdot 26563253 \cdot 86800081 \cdot 269845309 \cdot 802234039 \cdot 1041872869 \cdot 5552238917 \cdot 12144939971 \cdot 15856830239$
✓	$3 \cdot 1609 \cdot 77699 \cdot 235586599 \cdot 347727169 \cdot 369575231 \cdot 9087872491$	$2^3 \cdot 3 \cdot 5 \cdot 13 \cdot 19 \cdot 23 \cdot 31 \cdot 59 \cdot 239 \cdot 3989 \cdot 7951 \cdot 2829403 \cdot 31455623 \cdot 225623753 \cdot 811073867 \cdot 1304127157 \cdot 78955382651 \cdot 129320018741$
✓	$5 \cdot 1381 \cdot 877027 \cdot 15060047 \cdot 19042511 \cdot 11542780393 \cdot 13192388543$	$2^4 \cdot 5 \cdot 13 \cdot 31 \cdot 59 \cdot 823 \cdot 2801 \cdot 26539 \cdot 2944817 \cdot 3066253 \cdot 87271397 \cdot 108272617 \cdot 386616343 \cdot 815320151 \cdot 1361785079 \cdot 12322934353$
✓	$2^3 \cdot 5^2 \cdot 173 \cdot 971 \cdot 613909489 \cdot 929507779 \cdot 1319454803 \cdot 2101983503$	$2^7 \cdot 3^2 \cdot 5 \cdot 29 \cdot 1021 \cdot 42589 \cdot 190507 \cdot 473287 \cdot 31555663 \cdot 654820381 \cdot 802234039 \cdot 19147596953 \cdot 23912934131 \cdot 52023180217$
✗	$2^2 \cdot 15193 \cdot 232891 \cdot 19514983 \cdot 139295419 \cdot 540260173 \cdot 606335449$	$2^2 \cdot 3^4 \cdot 13 \cdot 19 \cdot 74897 \cdot 1377667 \cdot 55828453 \cdot 282012013 \cdot 802234039 \cdot 3350122463 \cdot 35787642311 \cdot 37023373909 \cdot 128377293101$
✗	$2^2 \cdot 5^4 \cdot 439 \cdot 1483 \cdot 13121 \cdot 21383 \cdot 67751 \cdot 452059523 \cdot 33099515051$	$2^2 \cdot 3^3 \cdot 11 \cdot 13 \cdot 19 \cdot 5023 \cdot 3683209 \cdot 98660459 \cdot 802234039 \cdot 1506372871 \cdot 4564625921 \cdot 27735876911 \cdot 32612130959 \cdot 45729461779$

small primes: abundant $\rightarrow$ dense column in the matrix

large primes: rare $\rightarrow$ sparse colum, limit to 2 or 3 on each side.

The relations that we like to see

✓	$5^2 \cdot 11 \cdot 23 \cdot 287093 \cdot 870953 \cdot 20179693 \cdot 28306698811 \cdot 47988583469$	$2^3 \cdot 5 \cdot 7 \cdot 13 \cdot 31 \cdot 61 \cdot 14407 \cdot 26563253 \cdot 86800081 \cdot 269845309 \cdot 802234039 \cdot 1041872869 \cdot 5552238917 \cdot 12144939971 \cdot 15856830239$
✓	$3 \cdot 1609 \cdot 77699 \cdot 235586599 \cdot 347727169 \cdot 369575231 \cdot 9087872491$	$2^3 \cdot 3 \cdot 5 \cdot 13 \cdot 19 \cdot 23 \cdot 31 \cdot 59 \cdot 239 \cdot 3989 \cdot 7951 \cdot 2829403 \cdot 31455623 \cdot 225623753 \cdot 811073867 \cdot 1304127157 \cdot 78955382651 \cdot 129320018741$
✓	$5 \cdot 1381 \cdot 877027 \cdot 15060047 \cdot 19042511 \cdot 11542780393 \cdot 13192388543$	$2^4 \cdot 5 \cdot 13 \cdot 31 \cdot 59 \cdot 823 \cdot 2801 \cdot 26539 \cdot 2944817 \cdot 3066253 \cdot 87271397 \cdot 108272617 \cdot 386616343 \cdot 815320151 \cdot 1361785079 \cdot 12322934353$
✓	$2^3 \cdot 5^2 \cdot 173 \cdot 971 \cdot 613909489 \cdot 929507779 \cdot 1319454803 \cdot 2101983503$	$2^7 \cdot 3^2 \cdot 5 \cdot 29 \cdot 1021 \cdot 42589 \cdot 190507 \cdot 473287 \cdot 31555663 \cdot 654820381 \cdot 802234039 \cdot 19147596953 \cdot 23912934131 \cdot 52023180217$
✗	$2^2 \cdot 15193 \cdot 232891 \cdot 19514983 \cdot 139295419 \cdot 540260173 \cdot 606335449$	$2^2 \cdot 3^4 \cdot 13 \cdot 19 \cdot 74897 \cdot 1377667 \cdot 55828453 \cdot 282012013 \cdot 802234039 \cdot 3350122463 \cdot 35787642311 \cdot 37023373909 \cdot 128377293101$
✗	$2^2 \cdot 5^4 \cdot 439 \cdot 1483 \cdot 13121 \cdot 21383 \cdot 67751 \cdot 452059523 \cdot 33099515051$	$2^2 \cdot 3^3 \cdot 11 \cdot 13 \cdot 19 \cdot 5023 \cdot 3683209 \cdot 98660459 \cdot 802234039 \cdot 1506372871 \cdot 4564625921 \cdot 27735876911 \cdot 32612130959 \cdot 45729461779$

small primes: abundant → dense column in the matrix
large primes: rare → sparse colum, limit to 2 or 3 on each side.

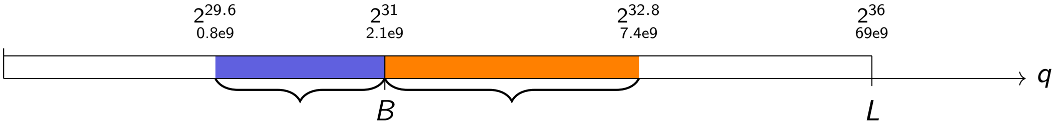
3. Paying attention to the combination cost

Relations with 2 LPs or less are a blessing.

- They easily participate in cheap combinations.
- If we have many 2-LP relations, filtering will get rid of most of them.
  We are left with a number of primes to combine that is roughly the number of primes below B.
- Caveat: two sides to deal with.

We must pay attention to q as well! How does it compare to B?

Strategy for RSA-240



$q < B$: allow
2 LPs on side 0,
3 LPs on side 1.

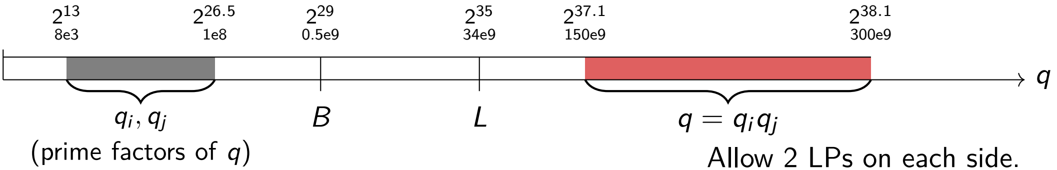
$B \leq q < L$: allow 2 LPs on each side.
(q counts as an extra LP on side 1.)

This strategy makes it easy to get rid of most $p \geq B$ on side 0 before we enter linear algebra proper.

We still have many on side 1, but that is not too bad because linear algebra in the factoring context is reasonable.

Strategy for DLP-240

For DLP-240, we used composite q.



Allow 2 LPs on each side.
(Factors of q are not LPs.)

This strategy reduced the combination work to essentially primes $p < B$ only.

Summary of important choices

For relation collection:

- Choose range of special-q wisely.
- Vary the number of LPs depending on whether $q \leq B$ or $q > B$.
- For dlog, consider composite special-q.

In addition, we combine with batch smoothness detection, which we can use as an alternative to sieving on one of the two sides.

Sparse linear algebra

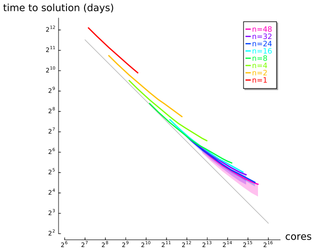
The matrix is always large and very sparse.

- 2019: 795-bit factoring: 282M rows/cols, density $\approx$ 200. $\Rightarrow$212G
- 2019: 795-bit DLP: 36M rows/cols, density $\approx$ 250. $\Rightarrow$68G
- 2020: 829-bit factoring: 405M rows/cols, density $\approx$ 250. $\Rightarrow$382G

Note: we're dealing with exact linear algebra here.

The block Wiedemann algorithm allows reasonable scaling performance on HPC clusters.

- Scaling to 8,000 cores and slightly beyond works ok.
- Running time is very predictable.
- Nothing is automatic, though.



Bringing it to scale

A record computation is a several-month journey.

- On a large, single computing infrastructure with coherent policy and tools, it can be relatively "easy" to keep track of the progress of the computation.
- On the other end of the spectrum, the approach "try to see what we can do with computing power that we can get for free" inevitably puts pressure on the bookkeeping task. It takes a group of crazy academics to do such things, but this model is not tenable much further.

Terabytes of data, zillions of files, many possibilities of transient failures, many hardware/software/policy idiosyncrasies.

Plan

Integer factoring and dlog

The Number Field Sieve (NFS)

Some key parameter choices

Moore's law?

Where can we go from there?

Comparing factoring and DLP

- Integer factoring:
  - RSA-250 (829 bits) in 02/2020, $\approx$2900 core-years, matrix 405M;
  - RSA-240 (795 bits) in 11/2019, $\approx$1000 core-years, matrix 282M;
  - A 232-digit modulus (RSA-768) in 12/2009, $\approx$1500 core-years, matrix 193M.
- Discrete logarithms in prime fields:
  - a 240-digit (795 bits) prime in 11/2019, $\approx$3100 core-years, matrix 36M;
  - a 232-digit prime (768 bits) prime in 06/2016, $\approx$ 5300 core-years, matrix 23M.

Lesson #1: progress is not about Moore's law

We measured how long our 240-digit DLP computation would have taken on hardware identical to hardware that was used for the 232-digit computation from 2016:

- About 25% less time, even though it is a priori $\approx 2.25\times$ harder.
- This is akin to a 3-fold speedup on identical hardware.

Comparing factoring and DLP

- Integer factoring:
  - RSA-250 (829 bits) in 02/2020, ≈2900 core-years, matrix 405M;
  - RSA-240 (795 bits) in 11/2019, ≈1000 core-years, matrix 282M;
  - A 232-digit modulus (RSA-768) in 12/2009, ≈1500 core-years, matrix 193M.
- Discrete logarithms in prime fields:
  - a 240-digit (795 bits) prime in 11/2019, ≈3100 core-years, matrix 36M;
  - a 232-digit prime (768 bits) prime in 06/2016, ≈ 5300 core-years, matrix 23M.

Lesson #2: factoring vs DLP

Unlike what is commonly thought, DLP is not a LOT harder than factoring.

- Sure, linear algebra is harder.
- But much better number fields can be chosen.
- Adequate parameter choices can balance these effects.

Comparing factoring and DLP

- Integer factoring:
  - RSA-250 (829 bits) in 02/2020, $\approx$2900 core-years, matrix 405M;
  - RSA-240 (795 bits) in 11/2019, $\approx$1000 core-years, matrix 282M;
  - A 232-digit modulus (RSA-768) in 12/2009, $\approx$1500 core-years, matrix 193M.
- Discrete logarithms in prime fields:
  - a 240-digit (795 bits) prime in 11/2019, $\approx$3100 core-years, matrix 36M;
  - a 232-digit prime (768 bits) prime in 06/2016, $\approx$ 5300 core-years, matrix 23M.

Lesson #3: this scales

- Relation collection scales at will.
- Linear algebra scaling results are very good.
  Running over 8,000 cores is easy today, maybe significantly more tomorrow?

Plan

Integer factoring and dlog

The Number Field Sieve (NFS)

Some key parameter choices

Moore's law?

Where can we go from there?

Recall: complexity of NFS

Our primary question was:
How hard is 1024-bit RSA? 1024-bit DLP? How hard are 1536-, 2048-bit, ...?

The time it takes to run NFS

The complexity $C(N)$ of NFS to factor an integer N is

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

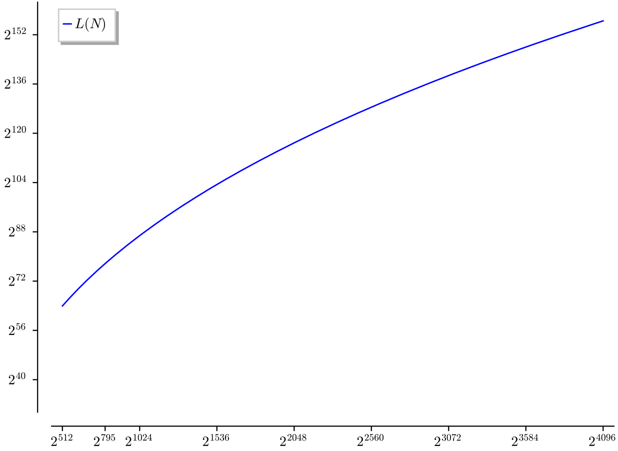
$$\text{with } L_N(1/3, (64/9)^{1/3}) = \exp\left((64/9)^{1/3}(\log N)^{1/3}(\log\log N)^{2/3}\right).$$

Can we extrapolate based on this formula + experimental data?

E.g., can we take $o(1) = 0$ and work from there?

Is $L_N(1/3, c)^{1+o(1)}$ a useful estimate?

log / log-plot of $L_N(1/3, c)$:

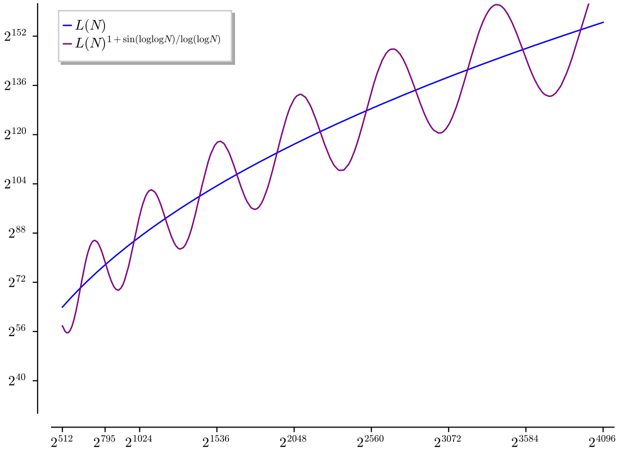


Is $L_N(1/3, c)^{1+o(1)}$ a useful estimate?

log / log-plot of $L_N(1/3, c)$:

$L_N(1/3, c)^{1+o(1)}$ can be weird.

E.g., $o(1) = \dfrac{\sin \log \log N}{\log \log N}$



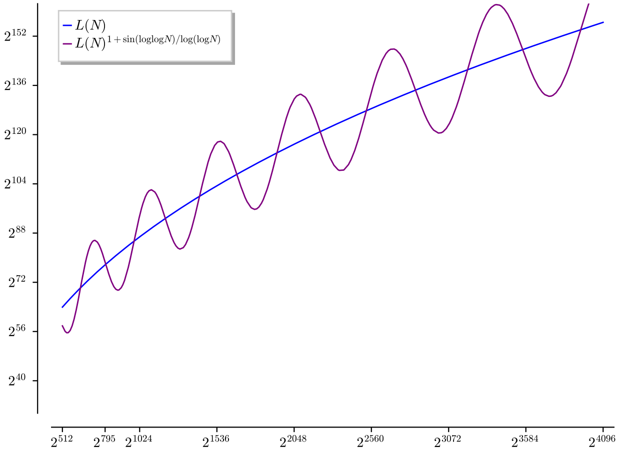
Is $L_N(1/3, c)^{1+o(1)}$ a useful estimate?

log / log-plot of $L_N(1/3, c)$:

$L_N(1/3, c)^{1+o(1)}$ can be weird.

E.g., $o(1) = \dfrac{\sin \log \log N}{\log \log N}$

Worse, NFS does have some oscillatory behaviour



$o(1)$ is not zero!

Blindly relying on the $L(1/3, c)$ formula by setting $o(1) = 0$ is actually dangerous.

- Analytic number theory does not help.
  $o(1)$ is the error term of a series that diverges for all practical values of N.
- The methodology is not sound. There is hardly any justification to relying on an $L(1/3, c)$ fit rather than a linear or quadratic fit, really.
- Transitions such as when the degree of number fields used in NFS-DL goes from 4 to 5 can be dramatic. It turns out that 5 is not quite $+\infty$ yet.

Lesson #4

I do not give any credence to far-fetched extrapolations of NFS hardness.
(Up to 1024-bit, existing software can give estimates. Beyond that, all bets are off).

How about the quantum threat?

Of course, Shor's algorithm breaks both factoring and DLP. My personal take:

- No significant factoring or DLP challenge will be first broken by quantum computers before at least a few decades.
- The sheer computing power of state-level adversaries is a much more concrete risk that should be taken seriously!

PQ is on the verge of being deployed everywhere.

- How long will it take? Deploying the full set of Kyber + Dilithium + Falcon + Sphincs (for best PQ interoperability) is not a mundane task: variety is likely to put off full transition by many years.
- I'm slightly worried that in the meantime, the repeal of RSA and DLP from public-key crypto hasn't happened as fast as it should have (if at all).

On to larger sizes?

We might not see many further record computations done with NFS.

- Gathering disordered resources as academics often do won't scale much further.
- It's a formidable amount of work for just one paper or two.
- Environmental footprint of large scale computing is often brought up.
- Landmarks such as 1024 bit remain attractive, though.
- IMO, the most important message to convey is that IT SCALES.

Factoring and DLP have been studied for long but a breakthrough cannot be ruled out.

- I doubt THE complexity of factoring is something as weird as $L(1/3, c + o(1))$.
- A good share of public-key crypto might be hanging from a cliff.
  A breakthrough would be mostly unnoticed if the author does not disclose it.