CSE291-14: The Number Field Sieve

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

Emmanuel Thomé

March 10, 2022

Part 10

Records and some recent stuff

A brief timeline

Hidden SNFS primes, up to kilobit size

Latest factoring and DLP records

Plan

A brief timeline

Hidden SNFS primes, up to kilobit size

Latest factoring and DLP records

- Late 1970s, Schroeppel: first analysis of CFRAC. $L()$ notation.
- 1980s, Pomerance + many: the quadratic sieve and its variants.
- 1983, Coppersmith: $L(1/3)$ algorithm for DLP in $\mathbb{F}_{2^n}$.
- 1985, Lenstra: ECM.
- 1986, Wiedemann: sparse linear algebra over finite fields.
- 1988, Pollard: Factoring with cubic integers.
- 1989, Lenstra, Manasse: factoring with electronic mail.
- 1990, Lenstra+others: The (special) number field sieve.
- 1990-1993, (many): GNFS.
  - Adleman: quadratic characters.
  - Pollard: lattice sieving.
  - Couveignes, Montgomery: square root.

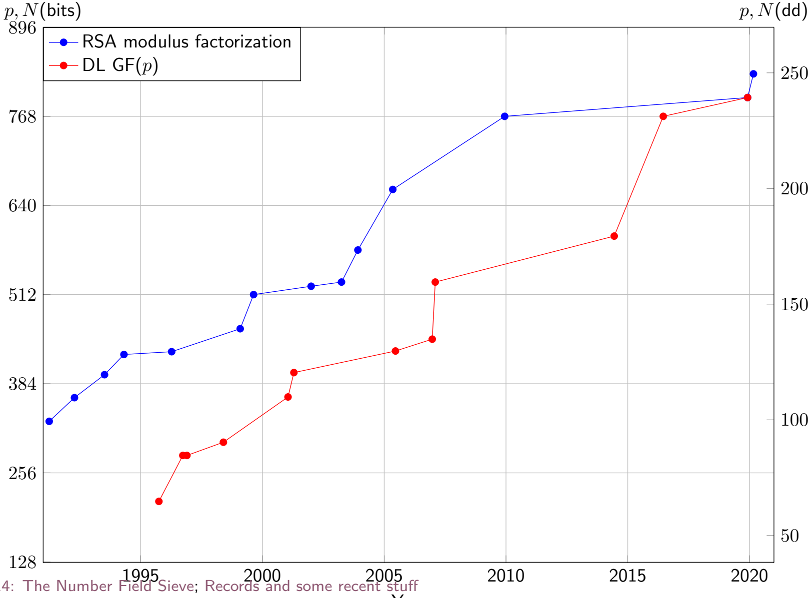
II

- 1992: early days of DSA.
- 1990-1993, Gordon: NFS for discrete logarithms.
- 1993, Schirokauer: Schirokauer maps.
- 1993, Coppersmith: the Multiple Number Field Sieve.
- 1994, Coppersmith: Block Wiedemann.
- 1994, Adleman: FFS.
- 1996, Weber: first practical NFS-DL computations.
- 1999, Lenstra+many: RSA-512.
- 2000, Bernstein: product trees.
- Early 2000s, Kleinjung: improvements to GNFS polynomial selection and to lattice sieving.
- 2002, Joux-Lercier: improvements to NFS-DL.

III

- 2002, Thomé: first use of Block Wiedemann for large computations.
- 2006, Joux-Lercier-Smart-Vercauteren: $L(1/3)$ for all fields.
- 2007: Kilobit SNFS.
- 2007, (many): development of Cado-NFS begins.
- 2008-2009: RSA-768.
- 2013: last big FFS computation $\mathbb{F}_{2^{809}}$.
- 2014: quasi-polynomial in $\mathbb{F}_{2^n}$.
- 2015: The Tower Number Field Sieve.
- 2015: Logjam. Individual logarithms are cheap.
- 2016: DLP-768 (232 digits).
- 2016: hidden SNFS kilobit DLP.
- 2018-2019: DLP-240, RSA-240, RSA-250.
- 2016-2021: several records for extension fields, finally using TNFS.

Timeline of records



Computational cost

Comparing computations is not a trivial task.

- Caveat: we only have published, academic records.
- All record computations generally use a scattered variety of resources.
- The only reasonable thing to do is to give what would have been the total cost if the computation had been run on one single resource type (and document that resource type).
- By definition, the unit of computational power depends on the point in time when the computation is done.
  For about 20 years, the trend of scaling all computational costs to unique computational unit (e.g. MIPS-years) has been all but abandoned.
- Hyperthreading complicates things even more. The usual approach is to count physical CPU time.

Plan

A brief timeline

Hidden SNFS primes, up to kilobit size

Latest factoring and DLP records

Hidden SNFS primes, up to kilobit size

Next few slides: takeaways from a computation done in 2016 (Fried, Gaudry, Heninger, Thomé, EC 2017).

- Relation to NFS in practice.
- We can get something that is cryptographically relevant, for a moderate computational cost.

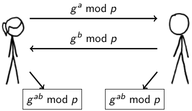
Plan

Hidden SNFS primes, up to kilobit size

- $(\mathbb{Z}/p\mathbb{Z})^*$ in crypto
- Backdooring primes
- Can one unveil the trapdoor?
- Computing DL mod 1024-bit primes with Cado-NFS
- Outcome and lessons

$(\mathbb{Z}/p\mathbb{Z})^*$, a.k.a. MODP groups

For Diffie-Hellman, for DSA: we've been using $(\mathbb{Z}/p\mathbb{Z})^*$ groups for decades.



Today (and whether we like it or not), FF DH and FF DSA are still very widespread.

- TLS
- SSH
- IPsec
- ...

Various measurements show their endured prevalence.

Who says which are the primes we use?

For a given key size, it should be fine if everybody uses the same p.

It is almost "One prime to rule them all"

De facto: a few primes are very widespread, promoted by:

- Standards (RFCs, . . .).
- Implementations (Apache, OpenSSL, . . .), or manufacturers of dedicated equipment (Cisco, Juniper, . . .).

Who has a say on what primes go there?

The 1992 controversy

Beginning of the 1990s = early days of DSA.
Year 1992: panel at Eurocrypt, CACM article in July, article by Gordon at Crypto.

Is it a good idea to standardize primes?

Most important points raised by (Lenstra and) McCurley:

> So far, it has not been demonstrated that trapdoor moduli for the discrete logarithm problem can be constructed such that a) they are hard to detect, and b) knowledge of the trapdoor provides a quantifiable computational advantage for parameter sizes that could actually be computed by known methods, even with foreseeable machines.
>
> —K. S. McCurley, EC92 panel.

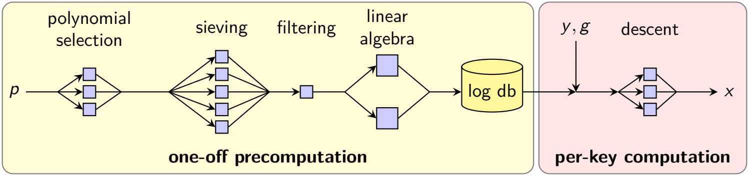
Part of the 1992 discussions focused on why a lower bound on p should be 1024 bits, not 512.

But the above points seemed to suffice to settle the discussion on the trapdoor: too conspicuous, and not a game-changer anyway.

1992 context

In 1992, NFS was still a new algorithm.

- Many practical challenges were yet to be solved.
- Linear algebra appeared a daunting task.
- This is even more true for NFS-DL: first preprint in April 1990.
- Algorithms for individual logs in NFS-DL took years to settle.



All these hurdles have long been passed.

Interlude

Some of the implications of the practice of NFS-DL took a long time to percolate and reach the use of FF-DLP in practice.

Until Logjam, many people overlooked the difference between precomputation (offline) and individual log (online) time for NFS-DL.

		Precomputation core-years	Individual Log core-time
RSA-512	[Cavallar et al. 1999]	1	—
DH-512	[Adrian et al. 2015]	10	10 mins
RSA-768	[Kleinjung et al. 2009]	1,000	—
DH-768	[Kleinjung et al. 2016]	5,000	2 days
RSA-240	[Boudot et al. 2020]	900	—
DH-240	[Boudot et al. 2020]	3,000	1 day

What does it look like in 2016?

Many primes are found in the wild with unknown provenance.

We cannot tell whether they have been chosen with malice.

- 1024-bit primes in Apache http software;
- RFC 5114 primes ($\geq$1024 bits);
- 2048-bit prime used in IACR 2015 BOD election;
- ...

We wish to investigate how trapdoors can be designed, and how easier they make the DLP computations.

RFC5114

```
Network Working Group                                        M. Lepinski
Request for Comments: 5114                                       S. Kent
Category: Informational                                BBN Technologies
                                                            January 2008

        Additional Diffie-Hellman Groups for Use with IETF Standards

2.  Additional Diffie-Hellman Groups

   This section contains the specification for eight groups for use in
   IKE, TLS, SSH, etc.  There are three standard prime modulus groups
   and five elliptic curve groups.  All groups were taken from
   publications of the National Institute of Standards and Technology,
   specifically [DSS] and [NIST80056A].  Test data for each group is
   provided in Appendix A.

2.1.  1024-bit MODP Group with 160-bit Prime Order Subgroup

   The hexadecimal value of the prime is:

   p = B10B8F96 A080E01D DE92DE5E AE5D54EC 52C99FBC FB06A3C6
       9A6A9DCA 52D23B61 6073E286 75A23D18 9838EF1E 2EE652C0
       13ECB4AE A9061123 24975C3C D49B83BF ACCBDD7D 90C4BD70
       98488E9C 219A7372 4EFFD6FA E5644738 FAA31A4F F55BCCC0
       A151AF5F 0DC8B4BD 45BF37DF 365C1A65 E68CFDA7 6D4DA708
       DF1FB2BC 2E4A4371

   The hexadecimal value of the generator is:

   g = A4D1CBD5 C3FD3412 6765A442 EFB99905 F8104DD2 58AC507F
       D6406CFF 14266D31 266FEA1E 5C41564B 777E690F 5504F213
       160217B4 B01B886A 5E91547F 9E2749F4 D7FBD7D3 B9A92EE1
       909D0D22 63F80A76 A6A24C08 7A091F53 1DBF0A01 69B6A28A
       D662A4D1 8E73AFA3 2D779D59 18D08BC8 858F4DCE F97C2A24
       855E6EEB 22B3B2E5

   The generator generates a prime-order subgroup of size:

   q = F518AA87 81A8DF27 8ABA4E7D 64B7CB9D 49462353
```

Here is p
Here is $q \mid (p-1)$
Please use for crypto.

Supported by:

- 900K (2.3%) HTTPS hosts
- 340K (13%) IPsec hosts

Plan

Hidden SNFS primes, up to kilobit size

- $(\mathbb{Z}/p\mathbb{Z})^*$ in crypto
- Backdooring primes
- Can one unveil the trapdoor?
- Computing DL mod 1024-bit primes with Cado-NFS
- Outcome and lessons

NFS goes very well in special cases

For arbitrary p (or N for factoring), there's a lower bound on how small f and g can be (e.g. by counting).

Factoring knows about especially easy integers

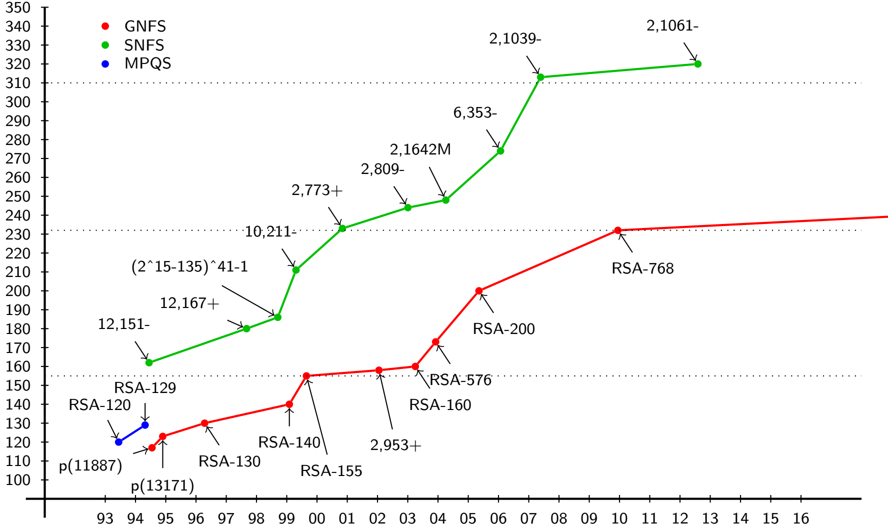
Say if $N = r^e - s$ with r, s small. We pick:

- $f = r^{e \bmod k} X^k - s$ with small k to our liking,
- and $g = X - r^{\lfloor e/k \rfloor}$

This is the special NFS (SNFS, as opposed to GNFS).
Applies in particular to the Cunningham tables.
Likewise, we have an SNFS-DL for "attacker-friendly primes".

Next: timeline of factoring records for SNFS and GNFS, compared.

SNFS versus GNFS (factoring) records



We may ease our task even more

DLP mod attacker-friendly primes may be well within reach while DLP mod “normal” primes of the same size is still remote.

But there is more !

So-called DSA primes

DSS promotes primes with a moderate size subgroup of $(\mathbb{Z}/p\mathbb{Z})^*$
E.g. 1024-bit prime p with 160-bit prime q dividing $p-1$.
RFC5114 promotes examples of such primes.

If a DSA prime is also attacker-friendly, then (S)NFS-DL linear algebra is modulo q, not modulo $p-1$. This is an additional win for the attacker.

Fantasy of a body tinkering with standards

What if we can design attacker-friendly DSA primes?

Heidi hides her polynomials

Heidi, a mischievous protocol designer

- chooses secret polynomials f and g;
- publishes $p = \mathrm{Res}(f, g)$ and pushes for its widespread use.
- p has a (say) 160-bit prime factor q.
- Knowing f and g, Heidi can run SNFS-DL.
  Linear algebra is to be done $\mod q$.

D. Gordon (Crypto 1992): a way to do just that.
This construction is still efficient today.

How to trapdoor a DSA prime [Gordon92]

Want to construct primes p, q such that $q \mid p - 1$ and

$$f(x) = f_6 x^6 + \cdots + f_0, \qquad g(x) = g_1 x - g_0$$

such that $p \mid \mathrm{Res}(f, g)$.

Slow algorithm:

1. Choose random f, g.
2. Check if $p = \mathrm{Res}(f, g)$ prime.
3. Factor $p - 1$ with ECM.
4. Repeat until $p - 1$ has 160-bit prime factor.

How to trapdoor a DSA prime [Gordon92]

Want to construct primes p, q such that $q \mid p-1$ and

$$f(x) = f_6 x^6 + \cdots + f_0, \qquad g(x) = g_1 x - g_0$$

such that $p \mid \mathrm{Res}(f, g)$.

Better algorithm:

1. Choose $f(x)$, q, g_0.
2. Want $q \mid \mathrm{Res}(f(x), g_1 x - g_0) - 1$.
3. Compute $G(g_1) = \mathrm{Res}(f(x), g_1 x - g_0) - 1$.
4. Compute root $G(r) \equiv 0 \bmod q$; $g_1 = r + cq$.
5. Repeat until $\mathrm{Res}(f(x), g_1 x - g_0)$ prime.

Note that this implies that the target size for g_1 is larger than q.

Plan

Hidden SNFS primes, up to kilobit size

- $(\mathbb{Z}/p\mathbb{Z})^*$ in crypto
- Backdooring primes
- Can one unveil the trapdoor?
- Computing DL mod 1024-bit primes with Cado-NFS
- Outcome and lessons

Can we tell whether p has a trapdoor?

This looks nice for Heidi, but won't work if the primes she pushes for is conspicuously weird.

E.g. you shouldn't do DLP in $(\mathbb{Z}/p\mathbb{Z})^*$ for $p = 2^{1024} - 105$.

However if Heidi allows herself sufficient freedom in choosing the coefficients of f, then p looks innocuous.

Detecting the trapdoor

- "Easy" if $g(x) = x + g_0$ or similar.
  1. Brute force leading coefficient f_d of f.
  2. Search values of g_0 near $(p/f_d)^{1/d}$.
  3. Use LLL to search for other small coefficients of f.
- If $g(x) = g_1 x + g_0$ don't know a way that doesn't require brute forcing coefficients of f or g.
- **Open Problem**: Given $p = \mathrm{Res}(f, g_1 x + g_0)$ and f has small coefficients, find f, g.

Crafting the trapdoor

- 1992-era parameters: 512-bit p, 160-bit q
  - Forces $\deg f = 3$; suboptimal for NFS.
  - f chosen from small set so not well hidden.

> ... this trap only makes sense for primes up to [600 bits]. Furthermore, this kind of trap can be detected, although this requires more work than an average user will be able to invest.
>
> —A. Lenstra, EC92 Panel.

- DSA standard: optional "verifiably random" prime generation.

Crafting the trapdoor in the modern era

Gordon's trapdoor construction remains best construction.

- Modern parameters: 1024-bit p, 160-bit q
  - Can choose $\deg f = 6$, optimal for NFS.
  - Choose $|f_i| \approx 2^{11}$.
  - Brute force search to find $f \approx 2^{80} \approx$ cost of Pollard rho for q.
  - Don't know of better way to detect trapdoor.

Exploiting the trapdoor in the modern era

We generated a target 1024-bit prime in 12 core-hours.

The public part:

$$
\begin{aligned}
p &= 1633239872404436791014020700930491550309894398069175 1\\
&\ 91735800707915692277289328503584988628543993514237336\\
&\ 97660534800194492724828721314980248259450358792069235\\
&\ 99182658894420044068709413666950634909369176890244055\\
&\ 53414932372965552542473794227022215159298376298136008\\
&\ 120820061240380894636102392361576512521804 91\\
q &= 1120320311183071261988433674300182306029096710473\ ,
\end{aligned}
$$

and Heidi's hidden polynomials:

$$
\begin{aligned}
f &= 1155\,x^6 + 1090\,x^5 + 440\,x^4 + 531\,x^3 - 348\,x^2 - 223\,x - 1385\\
g &= 567162312818120432489991568785626986771201829237408\,x\\
&\ -663612177378148694314176730818181556491705934826717\ .
\end{aligned}
$$

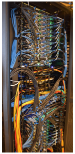
Plan

Hidden SNFS primes, up to kilobit size

- $(\mathbb{Z}/p\mathbb{Z})^*$ in crypto
- Backdooring primes
- Can one unveil the trapdoor?
- Computing DL mod 1024-bit primes with Cado-NFS
- Outcome and lessons

Computation timings

We used only two clusters. Linear algebra was done on higher-end hardware with fast interconnect (Infiniband FDR 56Gbps, Cisco UCS 40Gbps)



Used parameters $m = 24$, $n = 12$ for block Wiedemann.

	sieving	linear algebra			individual log
		sequence	generator	solution	
cores	$\approx$3000	2056	576	2056	500–352
CPU time (core)	240 years	123 years	13 years	9 years	10 days
calendar time	1 month	1 month			80 minutes

Computation went smoothly, of course

On the bright side, our computation took almost exactly the predicted time (both CPU time and wall-clock time).

Yet we did have our share of mishaps.

- UPenn: deal with cluster being kicked out of the computer room with 2-day notice, and moved 2 miles south with no decent network connection.
  raspberry pi's + university wifi + ...
- Nancy: of course not everything was coded yet when we started...

Comparison with other computations

Our computation: $\log_2 p \approx 1024$, $\log_2 q \approx 160$: 400 core-years.

Safe prime of the same size: expect lin.alg $7\times$ harder.

768-bit GNFS-DLP (Kleinjung et al., 2017): $\approx$ 5000 core-years.

2048-bit trapdoored p, like here: expect similar to GNFS-1340.

Some conspicuous SNFS primes found in the wild ($q = (p-1)/2$):

- $p = 2^{1024} - 1093337$: doable but harder than our p!
  - polynomial not as good as ours: α value is bad; sieving $3\times$ harder
  - linear algebra mod $q = (p-1)/2$.
- $p = 2^{784} - 2^{28} + 1027679$ (exercise) $\approx$ 60 core-years.

Plan

Hidden SNFS primes, up to kilobit size

- $(\mathbb{Z}/p\mathbb{Z})^*$ in crypto
- Backdooring primes
- Can one unveil the trapdoor?
- Computing DL mod 1024-bit primes with Cado-NFS
- Outcome and lessons

Danger of over-interpreting the result

We have found no poorly-hidden trapdoored prime in the wild.

- either because the trap was well hidden (after all, the recipe dates back to 1992).
- or because there was no trapdoor at all.

If Heidi designed RFC5114 and suggested the primes used in Apache and so on, she might be caught red-handed in the future. There is no plausible deniability.

Not clear that Heidi is at ease about such a scenario.

Anyway, now the RFCs have ditched the RFC5114 primes.

Lessons

1024-bit DLP can be easy for an attacker that maliciously chose the prime to his liking.

We found no easy way to prove that a trapdoor is present.

Verifiable randomness is necessary.

- It's not even the question of accusing anyone of wrongdoing. We found no smoking gun.
- But the lack of verifiable randomness is a major hindrance for trust in cryptographic standards.

Of course **people still get it awfully wrong**.

E.g. the standardized French and Chinese elliptic curves are really really bad to this regard.

Plan

A brief timeline

Hidden SNFS primes, up to kilobit size

Latest factoring and DLP records

Pre-2019 state-of-the-art

RSA-768: 02/2008–12/2009. About 1,500 core-years in total.

- large-scale improvement compared to the previous RSA-200 record. RSA-768 was a much larger undertaking.
- coordination of multiple computer clusters.
- fancy block Wiedemann, multi-country.

DLP-768: 06/2016: About 5,300 core-years.

- Much more efficient than previous 180-digit record thanks to Joux-Lercier polynomial selection.
- First apparent involvement of government computational resources (BSI) in an academic computation.

Recent results

Our DLP-240 computation was faster than the previous DLP-768 computation, while of course we tackled a harder challenge.

This is also true if we try to measure the cost on the same hardware that was used for the DLP-768 computation.

What are the important things in this computation?

A simple rule of thumb

We look for smoothness with respect to a bound L.

A prime should appear either often, or very rarely.

- below some bound B, we strive to find all pairs (a, b) such that primes below B appear in the factorization.
  We do this with sieving.
- "large primes" (LPs) such that $B \leq p < L$:
  allowed if we happen to find them.
  Limit to a few 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 $\rightarrow$ dense column in the matrix
large primes: rare $\rightarrow$ sparse colum, limit to 2 or 3 on each side.

Before linear algebra, the filtering step tries to do as many cheap combinations as it can, so as to get a smaller matrix.

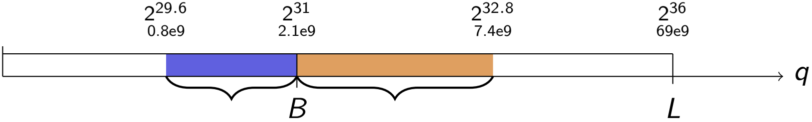
The combination cost

Relations with 2 LPs or less are a blessing.

- They easily participate in cheap combinations.
- If we have only 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 the special-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.

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

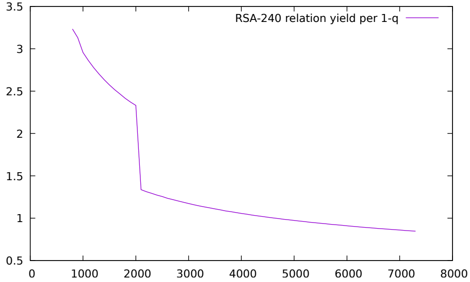
This strategy makes it easy to get rid of most $p \ge B$ on side 0 before we enter linear algebra proper.

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

Unstable yield, but we know what we're doing

Note that we change the relation collection criteria radically depending on q!

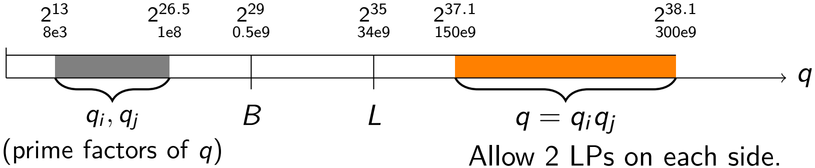
The yield changes (plot from this data)



This is expected, and fits well with our goal!

Strategy for DLP-240

For DLP-240, we used composite q, to avoid the disadvantage of having q in the LP range.



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

This strategy was efficient in reducing the combination work to essentially primes $p < B$ only.

Alternative to sieving

In all cases, we have an "easy" and a "hard" side, depending on the size of the norms.

Relation collection is about restricting attention to a subset of (a, b)'s. There's one side that we have to do first.

If we do the "hard" side first, not very many of the (a, b) pairs are left.

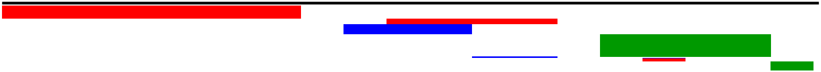
- In some situations, this selection is so drastic that it may make sense to process these few pairs one by one instead of doing sieving on the other side.
- This is exactly what we did for the previous records, using product trees (for some parameter ranges).

Summary of relation collection

- Tried-and-true techniques do work. Many low-level improvements in the deep aspects of special-q sieving.
- Seldom used techniques such as composite special-q or batch smoothness detection played a key role.
- We tailored the relation collection step so that the subsequent filtering step works well. (choice of q ranges, number of LPs.)

Relation collection is by far the most expensive step, which ran over several months. The distribution of the work raises several interesting issues as well.

Approximative timeline and core-hours



2018/08 - 2019/03	**DLP-240** relation collection. 4k cores working in parallel.	21M $c \cdot h$
2019/05 - 2019/08	**DLP-240** linear algebra (sequences)	5M $c \cdot h$
2019/04 - 2019/06	**RSA-240** relation collection. 4.3k cores working in parallel.	7M $c \cdot h$
2019/10 - 2020/02	**RSA-250** relation collection. 12k cores working in parallel.	21M $c \cdot h$
2019/07 - 2019/08	**RSA-240** linear algebra (sequences)	0.6M $c \cdot h$
2019/11	**RSA-240** linear algebra (wrap up)	0.1M $c \cdot h$
	DLP-240 linear algebra (wrap up)	0.7M $c \cdot h$
2020/02	**RSA-250** linear algebra	2M $c \cdot h$

caveat: time windows often include partially idle periods

Relations, matrix size, core-years timings

	RSA-240	DLP-240	RSA-250
polynomial selection deg f_0, deg f_1	76 core-years 1, 6	152 core-years 3, 4	150 core-years 1, 6
relation collection raw relations unique relations	794 core-years 8.93G 6.01G	2400 core-years 3.82G 2.38G	2450 core-years 8.75G 6.13G
filtering after singleton removal after clique removal after merge, + density	days 2.60G $\times$ 2.38G 1.18G $\times$ 1.18G 282M, $d = 200$	days 1.30G $\times$ 1.00G 150M $\times$ 150M 36M, $d = 253$	days 2.74G $\times$ 2.62G 1.82G $\times$ 1.82G 405M, $d = 252$
linear algebra	83 core-years	625 core-years	250 core-years
m, n	512,256	48,16	1024,512
characters, sqrt, ind log	days	days	days

Data & reproducibility info: gitlab.inria.fr/cado-nfs/records.

Conclusions

- More than just records, we developed efficient parameterization strategies for further computations.
- We developed an extensive simulation framework to guide the parameter choices. Not perfect.
- We show that our implementation scales well and can tackle larger problems. No technology barrier at this point.

Comparisons:

- Comparison with previous record (DLP-768, 232 digits, 2016): On identical hardware, our DLP-240 computation would have taken less time than the 232-digits computation.
- FF-DLP is not much harder than integer factoring.

For future projects, we intend to keep the focus on our capacity to anticipate the computational cost, and to harness large computing power.