Théorie algorithmique des nombres et applications à la cryptanalyse de primitives cryptographiques

Emmanuel Thomé

13 déc. 2012

Algorithmic Number Theory and Applications to the Cryptanalysis of Cryptographical Primitives

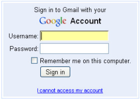
Emmanuel Thomé

Dec. 13rd, 2012

Cryptography is ubiquitous

Numerous applications of cryptography nowadays.





Many public-key cryptographic procotols rely on the hardness of some number-theoretical problems to guarantee their security.

Cryptographic motivation: study algorithms to solve them.

- Purportedly hard problems, so hard work.
- Having an idea about real hardness is important.
  - Bad assessment $\Rightarrow$ bad security.
  - Accurate assessment $\Rightarrow$ well chosen key sizes.

Crypto primitives based on number theory

Among others, two "king" problems:

- Integer factorization (hence RSA). $\boxed{N} \to \boxed{p}, \boxed{q}$
- Discrete logarithm (DL) (El Gamal, DSA). $\boxed{g^x, g} \to \boxed{x}$

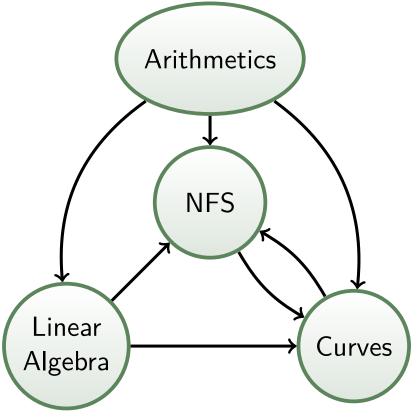
or: $\boxed{xP, P} \to \boxed{x}$

The Number Field Sieve algorithm (NFS) can attack these problems, and is central to our research.

Our research work is at multiple levels:

- algorithms,
- complexity analysis,
- implementation.

The central role of NFS



- DL on curves:
  - large primes.
  - FFS for curves.
- Linear algebra.
  NFS-related problems are our target.
- Efficient arithmetics:
  NFS uses these.

Plan

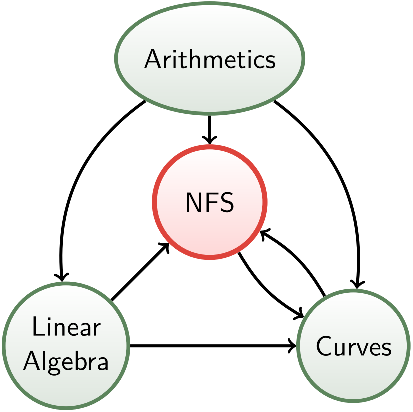
The Number Field Sieve

Curves

Sparse Linear Algebra

Computer Arithmetic

Future directions



The Number Field Sieve (NFS)

NFS is the fastest integer factorization algorithm asymptotically.

Teaching NFS?

- Takes a while (at least a 1-semester course).
- NFS embeds many sub-algorithms (possibly including itself!).
- NFS has many variants.

Our contributions related to NFS

Group effort most of the time, but important own involvement.

- algorithms;
- record computations;
- implementation;
- use NFS to solve other problems.

The diagram

The key to understanding NFS is this diagram.

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl, "x \mapsto m"'] \arrow[dr, "x \mapsto \alpha"] & \\
\mathbb{Q} \supset \mathbb{Z}[m] \arrow[dr, "\varphi_g : t \mapsto t \bmod N"'] & & \mathbb{Z}[\alpha] \subset K \arrow[dl, "\varphi_f : \alpha \mapsto m \bmod N"] \\
 & \mathbb{Z}/N\mathbb{Z} &
\end{tikzcd}$$

The diagram

The key to understanding NFS is this diagram.

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl, "x \mapsto m"'] \arrow[dr, "x \mapsto \alpha"] & \\
\mathbb{Q} \supset \mathbb{Z}[m] \arrow[dr, "\varphi_g\,:\,t \mapsto t \bmod N"'] & & \mathbb{Z}[\alpha] \subset K \arrow[dl, "\varphi_f\,:\,\alpha \mapsto m \bmod N"] \\
 & \mathbb{Z}/N\mathbb{Z} &
\end{tikzcd}$$

NFS searches for many $a - bx$ such that:

- $a - bm \in \mathbb{Q}$ is smooth (product of small primes),
- $(a - b\alpha)$ is smooth (product of small prime ideals).

Combination by linear algebra $\Rightarrow$ congruence of squares $\Rightarrow$ factors.

Implementation of NFS

Best way to learn NFS: implement it.

ANR CADO (2007–2010): `cado-nfs` implementation [GKM+11]
Joint effort Nancy–LIX.

- Started completely afresh.
- State-of-the-art or close to it almost everywhere.
- A nice playground for new ideas.
  Example: a new NFS square root algorithm. [Tho12]

Largest number factored with `cado-nfs`: RSA-704 [BTZ12]

State of the art NFS: RSA-768 [KAF+10]

RSA-768=1 230 186 684 530 117 755 130 494 958 384 962 720 772 853 569 595 334 792 197 322 452 151 726 400 507 263 657 518 745 202 199 786 469 389 956 474 942 774 063 845 925 192 557 326 303 453 731 548 268 507 917 026 122 142 913 461 670 429 214 311 602 221 240 479 274 737 794 080 665 351 419 597 459 856 902 143 413 = 33 478 071 698 956 898 786 044 169 848 212 690 817 704 794 983 713 768 568 912 431 388 982 883 793 878 002 287 614 711 652 531 743 087 737 814 467 999 489 × 36 746 043 666 799 590 428 244 633 799 627 952 632 279 158 164 343 087 642 676 032 283 815 739 666 511 279 233 373 417 143 396 810 270 092 798 736 308 917.

A key size from old times ? yes and no.

State of the art NFS: RSA-768 [KAF+10]

RSA-768=1 230 186 684 530 117 755 130 494 958 384 962 720 772 853 569 595 334 792 197 322 452 151 726 400 507 263 657 518 745 202 199 786 469 389 956 474 942 774 063 845 925 192 557 326 303 453 731 548 268 507 917 026 122 142 913 461 670 429 214 311 602 221 240 479 274 737 794 080 665 351 419 597 459 856 902 143 413 = 33 478 071 698 956 898 786 044 169 848 212 690 817 704 794 983 713 768 568 912 431 388 982 883 793 878 002 287 614 711 652 531 743 087 737 814 467 999 489 × 36 746 043 666 799 590 428 244 633 799 627 952 632 279 158 164 343 087 642 676 032 283 815 739 666 511 279 233 373 417 143 396 810 270 092 798 736 308 917.

A key size from old times ? yes and no.

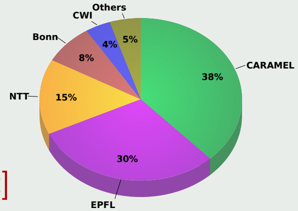
- 768-bit keys were in use by the banking industry until $\approx$ 2007.
- Google's DKIM system was using 512-bit keys until 07/2012. (Most DKIM keys below 768-bit still today).
- Still 2% of the internet SSL servers use 512-bit keys.
- Assumptions like "people are no fools" sometimes doubtful.

State of the art NFS: RSA-768 [KAF+10]

Running the computation has been serious business.

Titanic relation collection

- 64 billion relations, 5 terabytes,
- 1.5 rels/s/core $\Rightarrow$ 1500 core-years.
- Idle time on many clusters.
- Strived to minimize human supervision time. [KBL+12]

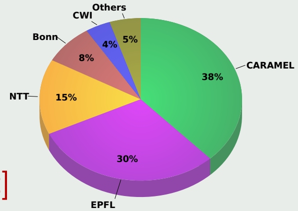


State of the art NFS: RSA-768 [KAF+10]

Running the computation has been serious business.

Titanic relation collection

- 64 billion relations, 5 terabytes,
- 1.5 rels/s/core ⇒ 1500 core-years.
- Idle time on many clusters.
- Strived to minimize human supervision time. [KBL+12]



Enough energy (500MWhr) to boil 2 olympic swimming pools

State of the art NFS: RSA-768 [KAF+10]

Running the computation has been serious business.

Titanic relation collection

Titanic linear algebra

- 193M equations and unknowns, over GF(2).
- Block Wiedemann algorithm key to success. [Tho02]
- Use of computer grids. [KNT10]

State of the art NFS: RSA-768 [KAF+10]

Running the computation has been serious business.

Titanic relation collection

Titanic linear algebra

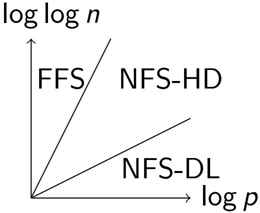
- 193M equations and unknowns, over GF(2).
- Block Wiedemann algorithm key to success. [Tho02]
- Use of computer grids. [KNT10]

It will soon be time to go further ! (see perspectives)

Variants of NFS

A common pattern can be used to describe:

- NFS as a factoring algorithm.
- NFS-DL, for DL in finite fields (large p).
- FFS, for DL in finite fields (small p).
  generalizes Coppersmith's DL algorithm.
- Past own contributions [Tho01]
- Future: new ANR project.



Our work also shows new applications of NFS.

- 1-sided NFS for oracle-assisted RSA problems [JNT08]
- 1-sided NFS-DL for oracle-assisted DH problems [JLNT09]
- Adapted FFS for DL on high genus curves [EGT11]

1-sided variants

Crypto proofs invoke the hardness of some problems.

- Rather usual situation: somewhat artificial problems.
- Example: "one-more" type RSA problems.
  - Attacker $\mathcal{A}$ allowed a query phase.
    $\mathcal{A}$ learns $\{\sqrt[e]{x} \bmod N\}$ for a query set $X \ni x$.
  - $\mathcal{A}$ receives a challenge c.
  - Claim: infeasible for $\mathcal{A}$ to find $\sqrt[e]{c}$.

1-sided variants

Crypto proofs invoke the hardness of some problems.

- Rather usual situation: somewhat artificial problems.
- Example: "one-more" type RSA problems.
  - Attacker $\mathcal{A}$ allowed a query phase.
    $\mathcal{A}$ learns $\{\sqrt[e]{x} \bmod N\}$ for a query set $X \ni x$.
  - $\mathcal{A}$ receives a challenge c.
  - Claim: infeasible for $\mathcal{A}$ to find $\sqrt[e]{c}$. Is it really hard?

1-sided variants

Crypto proofs invoke the hardness of some problems.

- Rather usual situation: somewhat artificial problems.
- Example: "one-more" type RSA problems.
  - Attacker $\mathcal{A}$ allowed a query phase.
    $\mathcal{A}$ learns $\{\sqrt[e]{x} \bmod N\}$ for a query set $X \ni x$.
  - $\mathcal{A}$ receives a challenge c.
  - Claim: infeasible for $\mathcal{A}$ to find $\sqrt[e]{c}$. Is it really hard?

This is much easier than factoring N [JNT08]

- Use queries to find $\sqrt[e]{p}$ for p in rational factor base.
- Find relations involving $\sqrt[e]{\pi}$ for π in algebraic factor base.
- Linear algebra to find $\{\sqrt[e]{\pi}\}$, descent.
- Key: the relation search needs 1-sided smoothness.

Complexity: $L_N[1/3, (64/9)^{1/3}] \rightsquigarrow L_N[1/3, (32/9)^{1/3}]$. (=SNFS)

Plan

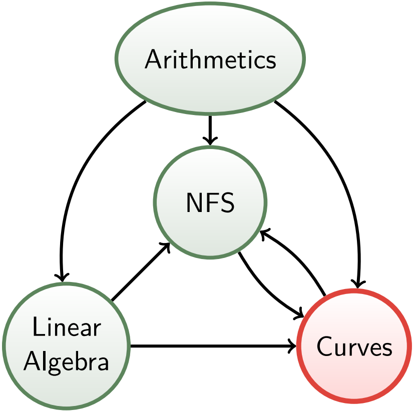
The Number Field Sieve

Curves

Sparse Linear Algebra

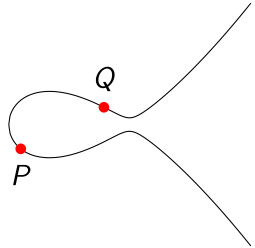
Computer Arithmetic

Future directions



DL problem on (Jacobians of) curves

Algebraic curves: very serious contender for the "best group to do crypto" contest.

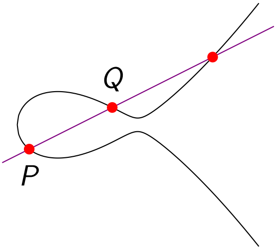


$g = 1.$

- What about higher genus? Any good for crypto?
- What does our background on NFS tell us?

DL problem on (Jacobians of) curves

Algebraic curves: very serious contender for the "best group to do crypto" contest.

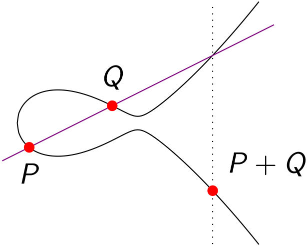


$g = 1.$

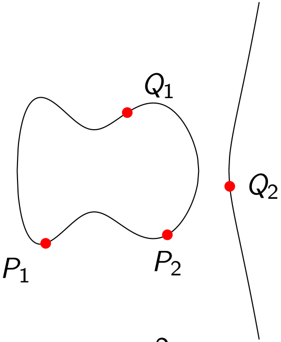
- What about higher genus? Any good for crypto?
- What does our background on NFS tell us?

DL problem on (Jacobians of) curves

Algebraic curves: very serious contender for the "best group to do crypto" contest.



$g = 1.$

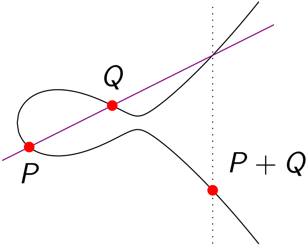


$g = 2.$

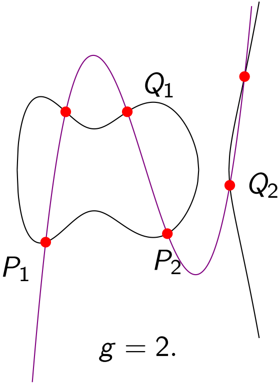
- What about higher genus? Any good for crypto?
- What does our background on NFS tell us?

DL problem on (Jacobians of) curves

Algebraic curves: very serious contender for the "best group to do crypto" contest.



$g = 1.$

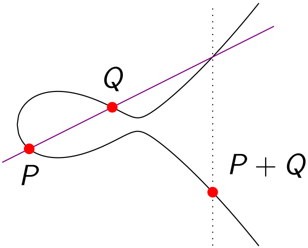


$g = 2.$

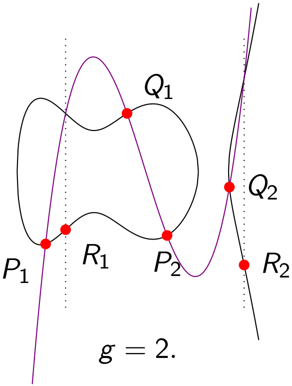
- What about higher genus? Any good for crypto?
- What does our background on NFS tell us?

DL problem on (Jacobians of) curves

Algebraic curves: very serious contender for the "best group to do crypto" contest.



$g = 1.$



$g = 2.$

- What about higher genus? Any good for crypto?
- What does our background on NFS tell us?

Arithmetic in Jacobians

Elements of $\mathrm{Jac}_{\mathcal{C}}(\mathbb{F}_q)$: divisors

$$D = (P_1) + \cdots + (P_r) - r(\infty).$$

- Formal sums of at most g points (over $\overline{\mathbb{F}_q}$, Galois stable).
- Group law easy: polynomial arithmetic.
- Easy to tell into how many points D splits.
- $\# \mathrm{Jac}_{\mathcal{C}}(\mathbb{F}_q) \approx q^g$.

Finding x such that $xD_1 = D_2$:
DL algo (Adleman–DeMarrais–Huang), but for small g (Gaudry):

- Factor base = points over $\mathbb{F}_q$.
- Try to split $xD_1 + yD_2$, for random x, y.
- Linear algebra $\Rightarrow$ relation $\alpha D_1 + \beta D_2 = 0 \Rightarrow$ DL solution.

Complexity of genus 3 DLP

Improvements over the period 2000–2007.

- 1999, Gaudry: $O(q^2)$. Slower than $\sqrt{\#G}$.
- 2000, Harley: $O(q^{3/2})$, balancing relations and linear algebra.
- 2003, Thériault: large prime variation, $O(q^{10/7})$.
  (an old idea from CFRAC times)

Complexity of genus 3 DLP

Improvements over the period 2000–2007.

- 1999, Gaudry: $O(q^2)$. Slower than $\sqrt{\#G}$.
- 2000, Harley: $O(q^{3/2})$, balancing relations and linear algebra.
- 2003, Thériault: large prime variation, $O(q^{10/7})$.
  (an old idea from CFRAC times)
- 2004, Gaudry, T., Thériault, Diem. [GTTD07]
  Two large primes, $O(q^{4/3})$.
  - Old idea from MPQS times.
  - Originality here: analysis shows a win.
  - Implementation: pays off for groups above 60 bits.

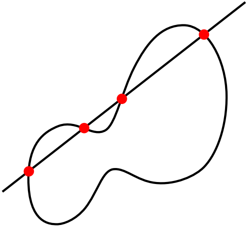
Complexity of genus 3 DLP

Improvements over the period 2000–2007.

- 1999, Gaudry: $O(q^2)$. Slower than $\sqrt{\#G}$.
- 2000, Harley: $O(q^{3/2})$, balancing relations and linear algebra.
- 2003, Thériault: large prime variation, $O(q^{10/7})$.
  (an old idea from CFRAC times)
- 2004, Gaudry, T., Thériault, Diem. [GTTD07]
  Two large primes, $O(q^{4/3})$.
  - Old idea from MPQS times.
  - Originality here: analysis shows a win.
  - Implementation: pays off for groups above 60 bits.
- 2006, Diem: specially for quartics, $O(q)$.
  In-depth study [DT08]

Genus 3 is dead! (almost)

Non-hyperelliptic genus 3 curves may be written as plane quartics.



- Fix two points over $\mathbb{F}_q$.
- Draw a line.
- Yields relations faster than before.
- Linear algebra $\rightsquigarrow$ solution.

Outcome for genus 3 [DT08]

- $\widetilde{O}(q)$ algorithm. DLP similar to genus 2 curve over $\mathbb{F}_q$.
- Practical algorithm.
  - Pays off early.
  - Group sizes $\approx$ 110-bit doable.
- Proven complexity, rigorous study of heuristics.
  Relate to random graph properties.

NFS for curves

Large (growing) genus g:
- An $L[1/2]$ algorithm exists.
- What about an $L[1/3]$ algorithm?

Answer: for a special class of curves, YES. [EGT11]

$$\mathcal{C} : f(t,x) = 0, \text{ with} \deg_t f \approx g^{2/3}, \deg_x f \approx g^{1/3}.$$

We try to intersect functions $\phi(t,x) = a(t) - xb(t)$ with the curve:

NFS for curves

Large (growing) genus g:
- An $L[1/2]$ algorithm exists.
- What about an $L[1/3]$ algorithm?

Answer: for a special class of curves, YES. [EGT11]

$$\mathcal{C} : f(t,x) = 0, \text{ with} \deg_t f \approx g^{2/3}, \deg_x f \approx g^{1/3}.$$

We try to intersect functions $\phi(t,x) = a(t) - xb(t)$ with the curve:

- $\deg_x \phi = 1$, $\deg_t \phi = g^{1/3}$: intersect at most $g^{2/3}$ times.
- Hope for smoothness of the intersection.
- Linear algebra as usual.
- Difficult part: the descent for computing logs.

Application to e.g. C_{ab} curves with $a \approx b^2$:

Algorithm of complexity $L_{\#G}[1/3, (64/9)^{1/3}]$.

Plan

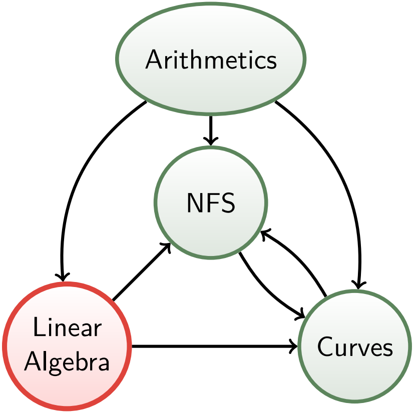
The Number Field Sieve

Curves

Sparse Linear Algebra

Computer Arithmetic

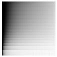
Future directions



Sparse LA over finite fields: challenges

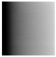
NFS and DL algorithms provide us with large, sparse matrices.

- NFS: matrices defined over $\mathbb{F}_2$.
  - RSA-768: 193M rows/cols, 27G nz



  - RSA-704: 88M rows/cols, 16G nz

- DL: matrices defined over $\mathbb{F}_p$.
  - $\mathbb{F}_{2^{619}}$: 660k rows/cols, 66M nz



100% PDE-free!

- We are talking exact arithmetic.
- No symmetry. No structural zeroes.
- Domain decomposition does not work. No physics.
- Convergence does not make sense.

Algorithms adapted to our problem

- Block Lanczos. Nice if one has a large cluster.
- Block Wiedemann. Offers better distribution opportunities.
  Key to success: fast algorithm for central step. [Tho02]

`cado-nfs` contains an implementation. [GKM+11]

- Hard work.
- Code completely rewritten several times.
  Not quite ready for IOCCC yet.
- Most important parts:
  - Only some thousands of lines of code.
  - Many commits, many hours of work.

The cado-nfs linear algebra

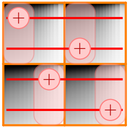
- Core matrix times vector routines in assembly.

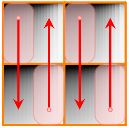
```
#define one_xor(idxreg, bufreg1, bufreg2, offset)    \
     movzwq % ## idxreg, % ## bufreg1            ; \
     shrq $16, %r ## idxreg                      ; \
     movq (%rsi,% ## bufreg1, 8), % ## bufreg2    ; \
     xorq % ## bufreg2, (%rdi,%r ## idxreg, 8)    ; \
     movl offset(%rbp), %e ## idxreg

     one_xor(ax, r14, r8, 0)
     one_xor(bx, r12, r9, 4)
     one_xor(cx, r14, r10, 8)
     one_xor(dx, r12, r11, 12)
```

The cado-nfs linear algebra

- Core matrix times vector routines in assembly.

```
#define one_xor(idxreg, bufreg1, bufreg2, offset)    \
     movzwq % ## idxreg, % ## bufreg1              ; \
     shrq $16, %r ## idxreg                        ; \
     movq (%rsi,% ## bufreg1, 8), % ## bufreg2     ; \
     xorq % ## bufreg2, (%rdi,%r ## idxreg, 8)     ; \
     movl offset(%rbp), %e ## idxreg

     one_xor(ax, r14, r8, 0)
     one_xor(bx, r12, r9, 4)
     one_xor(cx, r14, r10, 8)
     one_xor(dx, r12, r11, 12)
```





- Work with threads and MPI.
  Optimization down to the level of MPI collectives.
- Also adapt to GPUs (H. Jeljeli's work).
  $\mathbb{F}_{2^{619}}$: 660k rows/cols, 66M nz: 17 hours [BBD+12]

Block Wiedemann and distribution

Fact:
- Accessing large, tightly interconnected computers is hard.
- OTOH, mid-size (20-100 nodes) clusters are common.
  E.g. Grid'5000: many mid-size clusters.

Block Wiedemann allows to split the computation:

- Several distinct sites (=clusters), with minimal I/O.
- Asymptotically fast reconstruction needed. [Tho02]

RSA-768: Did ≈40% of linear algebra on Grid'5000. [KNT10]

- 6 clusters ($\leq$ 4 simultaneously), 16 different configs.
- Run on whichever was available at a given time.
- Partly done in best-effort mode.

RSA-704: complete linear algebra on Grid'5000. [BTZ12]

- Exclusively best-effort jobs. Incurred overhead ≈40%.

Plan

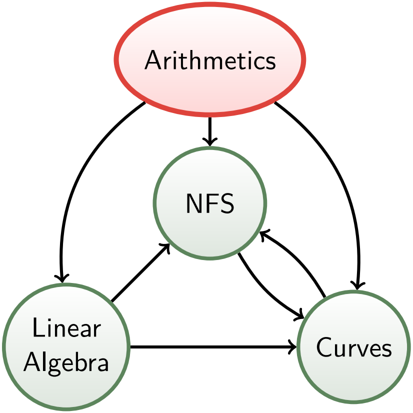
The Number Field Sieve

Curves

Sparse Linear Algebra

Computer Arithmetic

Future directions



Small objects

Fast arithmetic is always a good thing. In the crypto context:

- Cryptography: strive to make a cryptosystem efficient.
- Cryptanalysis: strive to make the attack efficient.

This applies in particular to finite field arithmetic.

- NFS-DL: linear algebra over $\mathbb{F}_p$.
- DL for curves: arithmetic in $\mathbb{F}_q$ and $\mathrm{Jac}_{\mathcal{C}}(\mathbb{F}_q)$.

Given the usage pattern, compile-time optimizations are relevant.

- Avoid generic, one-size-fits-all code.
- Allow modulus-specific code.

Small objects

mpfq: software for fast finite field arithmetic. [GT07]

- Code generating program.
- All field-dependent control flow becomes static.
- Reasonable interface in the end.
- Set some speed records at the outset.
- Now used e.g. within cado-nfs.

The characteristic two part was merged in gf2x. [BGTZ08]

- Use vector instructions;
- Unroll for small sizes;
- CPU-dependent choice of "best" code.
- Used in NTL.

```
__v2di ss1, ss2, s1s, s2s;
__v2di t00, t11, tk;
ss1 = _mm_loadu_si128((__v2di *)s1);
ss2 = _mm_loadu_si128((__v2di *)s2);
t00 = _mm_clmulepi64_si128(ss1, ss2, 0);
t11 = _mm_clmulepi64_si128(ss1, ss2, 17);
s1s = _mm_shuffle_epi32(ss1, 78);
ss1 ^= s1s;
s2s = _mm_shuffle_epi32(ss2, 78);
ss2 ^= s2s;
tk = t00 ^ t11 ^ _mm_clmulepi64_si128(ss1, ss2, 0);
_mm_storeu_si128((__v2di *)t, t00 ^ _mm_slli_si128(tk, 8));
_mm_storeu_si128((__v2di *)(t+2), t11 ^ _mm_srli_si128(tk, 8));
```

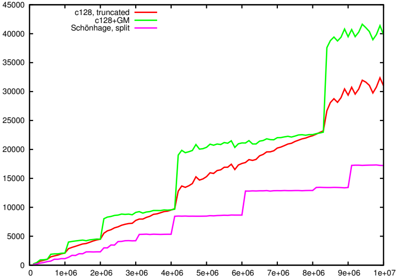
Large objects

Once small poly multiplications are fast, what about large ones?

- Schönhage's ternary FFT.
- Cantor's additive FFT.
- Gao-Mateer's modification of Cantor's algorithm.

All tested within `gf2x`. [BGTZ08]

- Ternary FFT fastest unless transforms are reused.
- Cantor pays off quickly e.g. for 4×4 matrix products.
- Gao-Mateer presently not competitive.



Plan

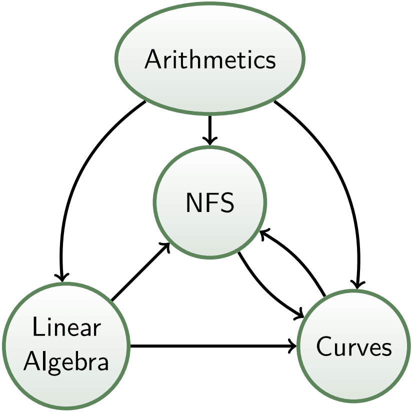
The Number Field Sieve

Curves

Sparse Linear Algebra

Computer Arithmetic

Future directions



NFS and friends

The obvious milestone for integer factoring: RSA-1024.

- Not today, but we have it in sight.
- RSA-896 will be an interesting step. Could start soon.

Code improvements sought.

- We now have several implementations. Pick the best parts.
- Micro-optimizations very effective sometimes.
- Seek better adaptation to existing hardware.
  - grids and best-effort for LA.
  - GPUs and clusters thereof. Other hardware?

New algorithmic ideas?

- NFS is a collection of many steps.
- Even a marginal improvement of one step is worthwhile.
  e.g.: central step in block Wiedemann likes middle product.

NFS and friends

The obvious milestone for integer factoring: RSA-1024.

- Not today, but we have it in sight.
- RSA-896 will be an interesting step. Could start soon.

RSA-1024 is too exciting to be done in "business as usual" fashion.

- Hardware very likely to come into play.
  - Bring hardware people into the game.
  - Seek new CPU designs.
  - Modify algorithms to better suit hardware.

  Will we be able to use fancy hardware for sieving?
- More planning ahead for organization of the computation.
  - Shall we use distributed clients (à la BOINC) ?
  - Which resources for linear algebra ?
- Try new algorithms / strategies when available.

NFS and friends

ANR CATREL project (tomorrow$\rightarrow$2016).

- Invest time on NFS-like algorithms for finite field DLP.
- Try new crazy things.
- NFS for factoring has received significant attention, NFS-DL and FFS less so. Fix this.
- Do new records (easy).
  $\mathbb{F}_{2^{619}}$ done as a warm-up. Plenty ahead.

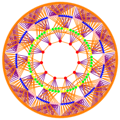
This has an impact on pairings, in particular.

Perspectives for curves

Goals:
- Explicit isogenies in genus 2.
- Walking isogeny graphs for genus 2 Jacobians over $\mathbb{F}_p$.
- Computation of modular polynomials over $\mathbb{F}_p$.

Many tools still to be developed. Current targets:

- Fast computation of theta constants.
  - Existing software by Dupont.
  - Common work with Enge:
    CM at $h = 6000$, well above state of the art.
- Work of R. Cosset (defended 11/2011) very useful.
  Evaluation of $\Theta(\Omega, z) \rightsquigarrow$ isogenies.
- Isogeny graphs in genus 2, beyond `avisogenies`.
  Some nice graphs (with Ionica), for max. RM.
  Not much $\bmod p$ yet.



Arithmetics

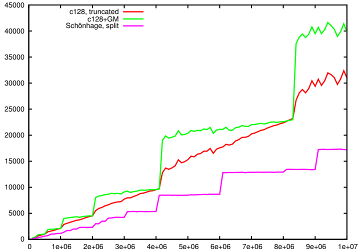
Results always to be made available in software form.

Gao-Mateer algorithm in $\mathbb{F}_2[x]$.

- The sort of beautiful trick one would like to see effective.
- Complexity $O(n \log n \log \log n)$, better than Cantor.
  Yet, no win. Why ?
  - Additions dominate complexity-wise.
    So far, for implementations, multiplications dominate.
    Be alert about new CPUs.
  - An effective truncated variant is yet to be invented.

Any better complexity?

- Maybe Fürer-like
  $O(n \log n 2^{O(\log^* n)})$?
- As of yet, no $\mathbb{F}_2[x]$ equivalent.



References I

[BTZ12] S. Bai, E. Thomé, and P. Zimmermann, *Factorisation of RSA-704 with CADO-NFS*, 2012. Available at `http://eprint.iacr.org/2012/369`.

[BBD+12] R. Bărbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann, *The relationship between some guy and cryptography*, 2012. Available at `http://ecc.2012.rump.cr.yp.to/`. ECC2012 rump session talk (humoristic).

[BGTZ08] R. Brent, P. Gaudry, E. Thomé, and P. Zimmermann, *Faster Multiplication in GF(2)[x]*. In A. van der Poorten and A. Stein (eds.), ANTS-VIII, vol. 5011 of *Lecture Notes in Comput. Sci.*, 153–166. Springer–Verlag, 2008.

[DT08] C. Diem and E. Thomé, *Index calculus in class groups of non-hyperelliptic curves of genus three*, J. Cryptology **21**(4) (2008), 593–611.

[EGT11] A. Enge, P. Gaudry, and E. Thomé, *An $L(1/3)$ discrete logarithm algorithm for low degree curves*, J. Cryptology **24**(1) (2011), 24–41.

[GKM+11] P. Gaudry, A. Kruppa, F. Morain, L. Muller, E. Thomé, and P. Zimmermann, `cado-nfs`*, An Implementation of the Number Field Sieve Algorithm*, 2011. Available at `http://cado-nfs.gforge.inria.fr/`. Release 1.1.

[GT07] P. Gaudry and E. Thomé, *The mpFq library and implementing curve-based key exchanges*, SPEED: Software Performance Enhancement for Encryption and Decryption, 49–64, 2007.

[GTTD07] P. Gaudry, E. Thomé, N. Thériault, and C. Diem, *A double large prime variation for small genus hyperelliptic index calculus*, Math. Comp. **76**(257) (2007), 475–492.

[JLNT09] A. Joux, R. Lercier, D. Naccache, and E. Thomé, *Oracle-assisted static Diffie-Hellman is easier than discrete logarithms*. In M. G. Parker (ed.), Cryptography and Coding 2009, vol. 5921 of *Lecture Notes in Comput. Sci.*, 351–367. Springer–Verlag, 2009.

[JNT08] A. Joux, D. Naccache, and E. Thomé, *When e-th roots become easier than factoring*. In K. Kurosawa (ed.), Advances in Cryptology – ASIACRYPT 2007, vol. 4833 of *Lecture Notes in Comput. Sci.*, 13–28. Springer–Verlag, 2008. Proc. 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007.

References II

[KAF+10] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, *Factorization of a 768-bit RSA modulus*. In T. Rabin (ed.), Advances in Cryptology – CRYPTO 2010, vol. 6223 of *Lecture Notes in Comput. Sci.*, 333–350. Springer–Verlag, 2010. Proc. 30th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2010.

[KBL+12] T. Kleinjung, J. Bos, A. Lenstra, D. A. Osvik, K. Aoki, S. Contini, J. Franke, E. Thomé, P. Jermini, M. Thiémard, P. Leyland, P. Montgomery, A. Timofeev, and H. Stockinger, *A Heterogeneous Computing Environment to Solve the 768-bit RSA Challenge*, Cluster Comput. **15**(1) (2012), 53–68.

[KNT10] T. Kleinjung, L. Nussbaum, and E. Thomé, *Using a grid platform for solving large sparse linear systems over GF(2)*, 11th ACM/IEEE International Conference on Grid Computing (Grid 2010), 2010.

[Tho01] E. Thomé, *Computation of discrete logarithms in* $\mathbb{F}_{2^{607}}$. In C. Boyd and E. Dawson (eds.), Advances in Cryptology – ASIACRYPT 2001, vol. 2248 of *Lecture Notes in Comput. Sci.*, 107–124. Springer–Verlag, 2001. Proc. 7th International Conference on the Theory and Applications of Cryptology and Information Security, Dec. 9–13, 2001, Gold Coast, Queensland, Australia.

[Tho02] E. Thomé, *Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm*, J. Symbolic Comput. **33**(5) (2002), 757–775.

[Tho12] E. Thomé, *Square Root Algorithms for the Number Field Sieve*. In F. Özbudak and F. Rodríguez-Henríquez (eds.), WAIFI 2012, vol. 7369 of *Lecture Notes in Comput. Sci.*, 208–224. Springer–Verlag, 2012. July 16-19, 2012. Bochum, Germany.