The CADO-NFS software

Pierrick Gaudry

Caramel – LORIA, Nancy
CNRS, Université de Lorraine, Inria

CATREL Workshop, October 2, 2015

Plan

General presentation of CADO-NFS

Features and algorithms

Concluding remarks

Identity card of CADO-NFS

Name: CADO-NFS – (Crible Algébrique: Distribution, Optimisation)

Date of birth: Around 2007

Authors: Many!

Webpage: `http://cado-nfs.gforge.inria.fr/`

Purpose: Integer factorization and discrete logarithm using the number field sieve.

Language: C / C++.

Build/test manager: CMake / CTest

License: GNU LGPL 2.1 or later.

Latest release: 2.1.1 (October 2014).

Identity card of CADO-NFS

Name: CADO-NFS – (Crible Algébrique: Distribution, Optimisation)

Date of birth: Around 2007

Authors: Many!

Webpage: `http://cado-nfs.gforge.inria.fr/`

Purpose: Integer factorization and discrete logarithm using the number field sieve.

Language: C / C++.

Build/test manager: CMake / CTest

License: GNU LGPL 2.1 or later.

Latest release: 2.1.1 (October 2014).

If you have a laptop, you are encouraged to download CADO-NFS and play with it during my talk!

(Slowly) learning software engineering. . .

Most of the developers had no training in software engineering.

We have made **progress**:

- Version control system (git);
- Continuous integration;
- Bug tracker;
- Mailing lists (discuss, commit-logs);
- Coverage reports;
- Official releases from time to time;
- ~~Documentation~~, README**s**.

Still **missing**:

- Global coding style;
- Code review;
- Better packaging in distributions.

Some statistics

Number of **lines of code**:

- 300 k, including 80 k that are macro-generated.
- "active lines" covered by tests: 100 k.
- 20 k lines of python scripts.

Number of **commits**:

- 10,600 commits since 2007.
- around 2000 in the past year.

Main authors (in number of commits):

Paul Zimmermann	2601
Alex Kruppa	2587
Emmanuel Thomé	2112
Pierrick Gaudry	1030
Cyril Bouvier	727
François Morain	426

Plan

General presentation of CADO-NFS

Features and algorithms

Concluding remarks

Polynomial selection

Feature/algorithm	DL	IF	Comment	Status
Kleinjung's algorithm	X	X	2008 and 2015 improv.	**Prod**
Conj method for $\mathbb{F}_{p^2}$	X		for $p \equiv 7 \mod 8$	**Prod**
JL, GJL, JLSV	X		for $\mathbb{F}_{p^k}$	**Todo**
Two quadratics		X		**Impl**
MNFS	X	X		**Todo**
SNFS	X	X		**Todo**

Rem. It is possible to import a hand-crafted polynomial pair; with or without rational side.

Factor base construction

Feature/algorithm	DL	IF	Comment	Status
Data for sieving primes and powers	X	X		**Prod**
Data for exact ideal factorization	X		uses Magma	**Prod**

Rem. For primes not dividing discriminant nor leading coefficient, very easy.

Rem. For DL, basically need Round 2 at "bad" primes. Currently with Magma, but without using advanced machinery. Translation to C/C++/Python should not be difficult (basic linear algebra).

Relation collection

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
F-K sieving algorithm	X	X	only 2 sides	**Prod**
Cofac using ECM	X	X	fixed sequence	**Prod**
Multi-threaded	X	X	for saving RAM	**Prod**
Cofac strategies	X	X		**Impl**
Batch smoothness cofac	X	X		**At work**
Scaling to large sizes	X	X	say, more than 768	**At work**
Sieving in dim > 2	X		Grémy's PhD	**At work**
Adjust I to q	X	X		**Todo**
Separation sieve / cofac	X	X		**Todo**
MNFS	X	?		**Todo**
(obsolete) Sieving for FFS	X		char 2 and 3	**Impl**

Filtering

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
Duplicate removal	X	X	2-step, on disks on-the-fly	**Prod** **Impl**
Singleton and clique	X	X	incl. Bouvier	**Prod**
Merge	X	X		**Prod**
Dble matrix trick	?	X	Kleinjung's idea	**Todo**
Parallel versions	X	X	already multi-thread	**Todo**

Linear algebra

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
Block-Wiedemann	X	X	very flexible	**Prod**
Node and thread parallelism	X	X	distribute matrix	**Prod**
Cluster-level parallelism	X	X	with several seq.	**Impl**
SM incl. as input vectors	X			**Prod**
Berlekamp-Massey step	X	 X	Fast, parallel operational	**Prod** **Prod**
Dble matrix trick	?	X		**Todo**
Lanczos		X		**Todo**
RNS/AVX/GPU arith.	X		fast!	**Impl**
Use Galois action	X		for $\mathbb{F}_{p^k}$	**Todo**

Characters / SM / Sqrt

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
Characters		X		**Prod**
Schirokauer maps	X			**Prod**
Sqrt		X	naive	**Prod**
		X	CRT-based, parallel	**Impl**

Individual logarithm

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
Descent over $\mathbb{F}_p$	X		missing param files	**Prod**
Automatic parameters	X			**Todo**
Init and descent over $\mathbb{F}_{p^k}$	X			**Todo**

Rem. For the descent init over $\mathbb{F}_p$, we use continued fractions and sieving (same binary as for relation collection).

Helper scripts

Feature/algorithm	**DL**	**IF**	**Comment**	**Status**
Single command-line run	X	X	DL only for $\mathbb{F}_p$	**Prod**
Client-server setting	X	X		**Prod**
Automatic sieving parameters		X	OPAL-based	**Impl**
HPC scheduler integration	X	X	we have only OAR	**Todo**
DL in $\mathbb{F}_{p^k}$	X			**At work**

Plan

General presentation of CADO-NFS

Features and algorithms

Concluding remarks

Comparison with QS

Ben Buhrow did some comparisons, using latest release (Oct. 2014).

Some more recent comparison with git version of January 2015:

- CADO-NFS much faster than Magma and Gp/pari for 80dd.
- Crossover point between Cado-nfs and Msieve-qs is around 85dd.
- Crossover point between Cado-nfs and Yafu-qs is around 95dd.

(tests done with just one thread)

Success stories

Computations by CADO-NFS developers:

- 180 digit DL in $\mathbb{F}_p$. June 2014. Bouvier, Gaudry, Imbert, Jeljeli and Thomé.
- 180 digit DL in $\mathbb{F}_{p^2}$. June 2014. Barbulescu, Gaudry, Guillevic, Morain
- 120 digit DL in $\mathbb{F}_{p^4}$. One week ago! Barbulescu, Gaudry, Guillevic, Morain.
- DL in $\mathbb{F}_{2^{809}}$ (with FFS). April 2013. Barbulescu, Bouvier, Detrey, Gaudry, Jeljeli, Thomé, Videau, Paul Zimmermann.
- Many integer factorizations in the 150–190 digits range (aliquot sequences). Zimmermann.
- More to come!

Success stories – 2

Used for **security analysis**:

- Breaking a ransomware using 128 dd RSA keys. Feb 2014. Perigaud, Pernet.
- Breaking Google email DKIM 512-bit RSA key. Oct 2012. Harris.
- PoC for the FREAK attack (Bhargavan et al.). Heninger used Amazon EC2 to factor 512 RSA keys in about 7 hours for 70 USD.
- PoC for the LogJam attack.

Conclusion

CADO-NFS is not the fastest NFS implementation, but:

- Reasonably well packaged, easy to use, even in a parallel context.
- Effort made on portability in Unix world (including MacOS).
- Only "push a button" free implementation of NFS for DL over $\mathbb{F}_p$.

Conclusion

CADO-NFS is not the fastest NFS implementation, but:

- Reasonably well packaged, easy to use, even in a parallel context.
- Effort made on portability in Unix world (including MacOS).
- Only "push a button" free implementation of NFS for DL over $\mathbb{F}_p$.

We welcome new contributors!