Début du message réexpédié :
De : Antoine Joux <Antoine.Joux@m4x.org>
Objet : Discrete Logarithms in GF(2^4080)
Date : 22 mars 2013 07:43:31 UTC+01:00
À : "NMBRTHRY@LISTSERV.NODAK.EDU" <NMBRTHRY@LISTSERV.NODAK.EDU>

Dear Number Theorists,

We are pleased to announce a new record for the computation of discrete logarithms in finite fields. We were able to compute discrete logarithms in GF(2^4080) using about 14100 CPU.hours. This computation was performed using the same index calculus algorithm as in our recent computation [Jo13]. A draft describing the algorithm is available as [Jo13a].

As far as we know, the previous discrete logarithm record in characteristic 2 is GF(2^1971), using a L(1/3) algorithm (see [Go+13a,Go+13b]).

The main features of our new index calculus algorithm are:

- An asymptotic complexity L(1/4+o(1))

- A small smoothness basis of size q^4 for discrete logs in a field GF(q^(2k)) with k close to q. Indeed, this smoothness basis contains polynomials of degree 1 and 2 with coefficients in GF(q^2). As a consequence, the computation of the logarithms of smoothness basis elements takes polynomial time.

- A new descent algorithm that together with classical descent techniques allows to express arbitrary elements in the finite field in terms of smoothness basis elements. This new descent step is essential to reach the announced complexity.

We first defined GF(2^16), from the irreducible polynomial x^16+x^5+x^3+x+1. We denote by 'a' a root of this polynomial and use the polynomial basis 1, a, ..., a^15 to represent elements in GF(2^16).

We then defined GF(2^4080) using the following Kummer extension

GF((2^16)^255) = GF(2^16)[u]/(u^255+A),

where A is the Trace of a [to GF(2^8)], i.e:
A=a^256+a=a^14 + a^12 + a^7 + a^6 + a^5 + a^4 + a.

We choose as basis for the discrete logarithms, the value : g = u+a.

As usual, we set to ourselves the challenge of computing the logarithm of:

Z= sum(i=0,254,u^i*Pol(binary(floor(Pi*Q^(i+1))%Q),a)) [in Pari-gp syntax, with Q=2^16]

The cardinality of the multiplicative group of GF(2^4080) is:

2^4080-1=
3^2*5^2*7*11*13*17^2*31*41*61*97*103*137*151*241*257*307*331*409*673*953*1021*1321*
1361*2143*2857*3061*4421*6529*8161*11119*12241*13669*26317*43691*51001*61681*
106591*131071*354689*383521*550801*949111*12717361*15571321*23650061*40932193*
394783681*1326700741*2949879781*4278255361*4562284561*46908728641*
611787251461*1392971637361*1467129352609*2368179743873*2879347902817*15455023589221*
33910825580641 * 116772720677761 * 418562986357561 * 737539985835313 *
171664686650370481 * 4967178060528306401 * 7226904352843746841 *
9520972806333758431 * 26831423036065352611 * 51366149455494753931 *
373200722470799764577 * 1230412270786066204321 * 8088220746627020943841 *
1014603201108417268835040 1 *
5702451577639775545838643151 * 4251553088834471719044481725601 *
630894905395143528221826310327361 * 1874145702705619946070176801657152 1 *
4202456886288461946911906748730722728656407680497483189224864 01 *
P78 * P116 * C295

where:
P78=116244395157193581337282640791798084114394917399572436767868837818708235649281

P116=59759045572704532151734514229676903701763064698110618010245423428627221235639899045664816790870237783305610352947361

C295=1533490284616846723598415409259297911652711222168775547792306046025942905067111993457984065168194835553976875731082408417656840935335955173202390542791678493470066047287643716215585615327618503502688057052265147924415358240017501698179552850131324250482005642180027027923716821703262527243660161

Since P116 has 385 bits, computing discrete logarithms in GF(2^4080) is clearly out of reach of generic algorithms.

As usual, the computation was done in three steps:
- the generation of multiplicative relations,
- the linear algebra,
- the final computation of individual logarithms.

As mentioned above, the factor basis that has been used contains all irreducible monic polynomials of degree 1 and 2 in u (with arbitrary coefficients in GF(2^16)). Thanks to the action of the 8-th power of Frobenius, this basis can be reduced to approximately 2^22 elements.

Note that performing linear algebra on 2^22 elements would be quite costly. However, as explained in [Jo13a], in the case of Kummer extension, we are in fact able to split the computation into several much smaller ones. In the present case, we have to solve 130 linear systems, the smallest one contains 130 linear polynomials (up to Frobenius), the next system contains 2^14 elements (corresponding to polynomials of the form u^2+u+alpha). Finally, we also have 128 systems containing 2^15 elements. Each of these 128 systems was solved in 9 hours (including the generation of the corresponding equations) on 8-cores. They were run independently in parallel and the total CPU time is less than 9300 CPU.hours.

Individual Logarithms:

We followed a descent approach similar to [JoLe06]. As in our previous computation [Jo13], this descent includes three separate parts. First, we used continued fractions to find an expression of a value related to Z as a product of relatively low degree polynomials. Here, the highest degree polynomial has degree 29. Then using classical descent (rercursively), we expressed these polynomials using polyomials of degree 12 or less. Finally, the new descent phase allows us to continue the descent down to degree 2.

[In [Jo13], the maximal degree after the continued fraction step was 18 and the new descent was only used for polynomials of degree 5 or less.]

The continued fraction steps took a few hours on 8 cores. Hitting degree 29 so quickly was quite lucky. The classical descent took 12 hours on one core for the slowest of the polynomials appearing after the continued fraction step. The total cost for classical descent was less than 50 CPU.hours.

After the classical descent, we had 149 polynomials of degree 12, 128 polynomials of degree 11 and 125 polynomials of degree 10 (we neglect polynomials of degree 9 or less, which belongs to a lower level of the descent tree and whose contribution to the total runtime is negliglible). Among those, the most costly where the polynomials of

degree 12. On average, for each of those, it took 4 hours on a Intel
Core i7 processor to find a decomposition into polynomials of degree 9
or less. Once this was done, the remaining time to get the value of
the logarithm was about 13 hours per degree 12 polynomial (using the
same machine as for the linear algebra step). The polynomials of
degree 11 and 10 were less expensive, respectively requiring
a total time of 13 and 4 hours each.

Once this logarithms were collected, concluding the computation took
about 20 CPU.hours.

The total cost of the descent step was less than 4800 CPU.hours.

Finally, we find:
Z = g ^
593537791871423043223099993371775097225725806985608997497794966002388232480207689669841040498259020695864963162877267246612766963427481859635858268330211735283816590918847157995342
025638775868791428528017795465828457233366986043689100592091740290308960776447743054737077011247538124490796554449688480787567205892205650065036371339635472100864592768628245778548
627169993710530248952247502198339102414084716879305058973285967705897824717564625973834423283500191898814926886245805865469139425619857671065003012554407741143232334093943305148519
456757124018567398173204598371497326728353430064760122625256809889244046240196511162297600325959107770470258420076304617198648034933080689987331284620483405839935257400541623168826
1510545134741182277970358473883943958563579015179820120979292270637497907072612180871069400619450857723011268017454116823535827228473296516703273009238893345386444533871542383504242
463001681961734268277378540067885920080290584936097716155329313777328195435585629703275367750105825453097378643622824901407930221204813818805961136841682239404338275246672278987523
1938768330294459381998191220112858134042404497185697219229072411513909004285242224234201221755939491010573105885453826465599986918927823875647571538

Antoine Joux (CryptoExperts and UVSQ, France, Antoine.Joux@m4x.org),

References:
===========

[Go+13a] Discrete logarithms in a GF(2^1971).
Faruk Gologlu, Robert Granger, Gary McGuire and Jens Zumbragel
NMBRTHRY list, Feb. 20th, 2013.
https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1302&L=NMBRTHRY&F=&S=&P=4793

[Go+13a] On the Function Field Sieve and the Impact of Higher Splitting Probabilities:
Application to Discrete Logarithms in GF(2^1971)
Faruk Gologlu, Robert Granger, Gary McGuire and Jens Zumbragel
Eprint Archive. http://eprint.iacr.org/2013/074

[JoLe06] The Function Field Sieve in the Medium Prime Case. Antoine Joux and
Reynald Lercier. EUROCRYPT'2006

[Jo13] Discrete logarithms in GF(2^1778). Antoine Joux.
NMBRTHRY list, Feb. 11th, 2013.
https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1302&L=NMBRTHRY&F=&S=&P=2317

[Jo13a] A new index calculus algorithm with complexity L(1/4+o(1)) in very small characteristic.
Antoine Joux.
Eprint Archive. http://eprint.iacr.org/2013/095

Appendix: Pari/GP verification script
======================
Warning: This verification takes several minutes

```
\p 2000
allocatemem(100000000)
Q=2^16
Z= sum(i=0,254,u^i*Pol(binary(floor(Pi*Q^(i+1))%Q),a))
pola=(a^16+a^5+a^3+a+1)*Mod(1,2)
polu=u^255+ Mod(a^14 + a^12 + a^7 + a^6 + a^5 + a^4 + a,pola)
g=(u+a)*Mod(1,2)*Mod(1,pola)*Mod(1,polu)
lg=5935377918714230432230999933717750972257258069856089974977949660023882324802076896698410404982590206958649631628772672466127669634274818596358582683302117352838165909188471579953
420256387758687914285280177954658284572333669860436891005920917402903089607764477430547370770112475381244907965544496884807875672058922056500650363713396354721008645927686282457785
486271699937105302489522475021983391024140847168793050589732859677058978247175646259738344232835001918988149268862458058654691394256198576710650030125544077411432323340939433051485
194567571240185673981732045983714973267283534300647601226252568098892440462401965111622976003259591077704702584200763046171986480349330806899873312846204834058399352574005416231688
2615105451347411822779703584738839439585635790151798201209792922706374979070726121808710694006194508577230112680174541168235358272284732965167032730092388933453864445338715423835042
424630016819617342682773785400678859200802905849360977161553293137773281954355856297032753677501058254530973786436228249014079302212048138188059611368416822394043382752466722789875
231938768330294459381998191220112858134042404497185697219229072411513909004285242224234201221755939491010573105885453826465599986918927823875647571538
if (g^lg == Z, print("Verification OK"), print("Verification FAILED"))
```