**From:** Moody, Dustin (Fed)
**To:** (b) (6)
**Cc:** Perlner, Ray A. (Fed)
**Subject:** RE: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
**Date:** Tuesday, January 10, 2017 11 08:03 AM

Daniel,

Yes, Ray and I are think we may as well submit. I have put my code down below. I'm trying to run some averages on q=4, s=4, which Ray predicts should take around 2500 iterations. I don't have enough for a good average yet, but the number of iterations I have so far is: 4300, 861, 937, 160, 1185, 772 which seems okay. He'd like us to try and get some data for q=4, s=5, and q=4, s=6. If our programs are going fast we can try q=8, s=4.

Ray also thinks we can fix the first half of the values of v1 and v2 to be 0, and everything should work fine. That would speed things up. I have a version of the code that does that also, and it does run quicker. I'll send it in a separate email.

Dustin

```
K.<z>=GF(4)
R.<x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23,x24,x25,x26,x27,x28,x29,x30,x31,x32,x33,x34,x35,x36,x37,x38,x39,x40,x41,x42,x43,x44,x45,x46,x47,x48,x49,x50,x51, x52, x53, x54, x55, x56, x57, x58, x59, x60, x61, x62, x63, x64, x65, x66, x67, x68, x69, x70, x71, x72,
x73, x74, x75, x76, x77, x78, x79, x80, x81>=K[]
X=[x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23,x24,x25,x26,x27,x28,x29,x30,x31,x32,x33,x34,x35,x36,x37,x38,x39,x40,x41,x42,x43,x44,x45,x46,x47,x48,x49,x50,x51, x52, x53, x54, x55, x56, x57, x58, x59, x60, x61, x62, x63, x64, x65, x66, x67, x68, x69, x70, x71, x72,
x73, x74, x75, x76, x77, x78, x79, x80, x81]

def randlincomb(s,K):
   lc=0
   for i in range(1,s^2+1):
      lc=lc+K.random_element()*X[i]
   return lc

def randquadform(s,K):
   lc=0
   for i in range(1,s^2+1):
      for j in range(i,s^2+1):
         lc=lc+K.random_element()*X[i]*X[j]
   return lc

def setup(s):
   A=matrix(s,s,lambda i,j: randquadform(s,K))
   B=matrix(s,s,lambda i,j: randlincomb(s,K));
   C=matrix(s,s,lambda i,j: randlincomb(s,K));
   AB=A*B
   AC=A*C
   E=AB.augment(AC)
   U=matrix identity(s^2)
   bv1=False
   while bv1==False:
      U=matrix(s^2,s^2,lambda i,j:K.random_element())
      bv1=U is_invertible()
   T=matrix(K,2*s^2,2*s^2,lambda i,j:K.random_element())
   bv=False
   while bv==False:
      T=matrix(K,2*s^2,2*s^2,lambda i,j:K.random_element())
      bv=T.is_inver ible()
   return E,A,B,C,T,U

s=4
E,A,B,C,T,U=setup(s)

def G(v1,v2):
   ELnew=E.list()
   mnew=matrix(K,2*s^2,2*s^2,lambda i,j:0)
   for h in range(0,2*s^2):
      # elements of E - cubic's
      enew=ELnew[h]
      for k in range(1,s^2+1):
         c3=enew.coefficient(X[k]^3)
         c2=enew.coefficient(X[k]^2)
         c1=enew.coefficient(X[k]^1)
         poly=3*c3*X[k]^2+2*c2*X[k]+c1
         if s==3:
            mnew[h,k-1] = poly(x1=v1[0,0],x2=v1[0,1],x3=v1[0,2],x4=v1[0,3],x5=v1[0,4],x6=v1[0,5],x7=v1[0,6],x8=v1[0,7],x9=v1[0,8])
            mnew[h,k+s^2-1] = poly(x1=v2[0,0],x2=v2[0,1],x3=v2[0,2],x4=v2[0,3],x5=v2[0,4],x6=v2[0,5],x7=v2[0,6],x8=v2[0,7],x9=v2[0,8])
         if s==4:
            mnew[h,k-1] = poly(x1=v1[0,0],x2=v1[0,1],x3=v1[0,2],x4=v1[0,3],x5=v1[0,4],x6=v1[0,5],x7=v1[0,6],x8=v1[0,7],x9=v1[0,8],x10=v1[0,9],x11=v1[0,10],x12=v1[0,11],x13=v1[0,12],x14=v1[0,13],x15=v1[0,14],x16=v1[0,15])
            mnew[h,k+s^2-1] = poly(x1=v2[0,0],x2=v2[0,1],x3=v2[0,2],x4=v2[0,3],x5=v2[0,4],x6=v2[0,5],x7=v2[0,6],x8=v2[0,7],x9=v2[0,8],x10=v2[0,9],x11=v2[0,10],x12=v2[0,11],x13=v2[0,12],x14=v2[0,13],x15=v2[0,14],x16=v2[0,15])
         if s==5:
            mnew[h,k-1] =
poly(x1=v1[0,0],x2=v1[0,1],x3=v1[0,2],x4=v1[0,3],x5=v1[0,4],x6=v1[0,5],x7=v1[0,6],x8=v1[0,7],x9=v1[0,8],x10=v1[0,9],x11=v1[0,10],x12=v1[0,11],x13=v1[0,12],x14=v1[0,13],x15=v1[0,14],x16=v1[0,15],x17=v1[0,16],x18=v1[0,17],x19=v1[0,18],x20=v1[0,19],x21=v1[0,20],x22=v1[0,21],x23=v1[0,22],x24=v1[0,23],x25=v1[0,24])
            mnew[h,k+s^2-1] =
poly(x1=v2[0,0],x2=v2[0,1],x3=v2[0,2],x4=v2[0,3],x5=v2[0,4],x6=v2[0,5],x7=v2[0,6],x8=v2[0,7],x9=v2[0,8],x10=v2[0,9],x11=v2[0,10],x12=v2[0,11],x13=v2[0,12],x14=v2[0,13],x15=v2[0,14],x16=v2[0,15],x17=v2[0,16],x18=v2[0,17],x19=v2[0,18],x20=v2[0,19],x21=v2[0,20],x22=v2[0,21],x23=v2[0,22],x24=v2[0,23],x25=v2[0,24])
   return mnew

DDF=[]
EL=E.list()
for h in range(0,2*s^2):
   # elements of E - cubic's
   e=EL[h]
   V=[]
   for k in range(1,s^2+1):
      c3=e.coefficient(X[k]^3)
      # alpha xk^3 -> 6 alpha in k,k entry
      c2=e.coefficient(X[k]^2)
      # alpha xi xk^2 -> 2 alpha in i,k and k,i entry
      c1=e.coefficient(X[k])
      # term is alpha xi xj xk -> put alpha in i,j j,i entries (i \neq j and both \neq k)
      # alpha xi^2 xk -> 2 alpha in i,i entry (i \neq k)
      m=matrix(K,s^2,s^2,lambda i,j 0)
      m[k-1,k-1]=6*c3
      for i in range(1,s^2+1):
         if i!=k:
            m[k-1,i-1]=2*c2.coefficient(X[i])
            m[i-1,k-1]=m[k-1,i-1]
      for i in range(1,s^2+1):
         if i!=k:
            m[i-1,i-1]=2*c1.coefficient(X[i]^2)
         for j in range(1,s^2+1):
            if i!=j and i!=k and j!=k:
               m[i-1,j-1]=c1.coefficient(X[i]).coefficient(X[j])
               m[j-1,i-1]=m[i-1,j-1]
      V.append(m)
   DDF.append(V)

def lincombDDF(h,v):
   MS=DDF[h]
   sm=0
   for i in range(0,s^2):
      sm=sm+v[0,i]*MS[i]
   return sm

V=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
flg=0
while(flg==0):
   for j in range(0,10000):
      if j%100==0:
         print V
      v1=matrix(K,1,s^2,lambda i,j: K.random_element())
      v2=matrix(K,1,s^2,lambda i,j: K.random_element())
      M=G(v1,v2).transpose()
      MK=M right_kernel()
      mk=s^2
      for t in MK.list():
         if t!=0:
            sm1=0
            for i in range(0,2*s^2):
               sm1=sm1+t[i]*lincombDDF(i,v1)
            sr=sm1.rank()
            mk=min(mk,sr)
            if mk==2*s:
               print j,t
               print v1
               print v2
               flg=1
               break
      V[mk]=V[mk]+1
      if flg==1:
         break
```

**From:** Daniel Smith(b) (6)
**Sent:** Tuesday, January 10, 2017 10:15 AM
**To:** Moody, Dustin (Fed) <dustin.moody@nist.gov>
**Cc:** Perlner, Ray (Fed) <ray.perlner@nist.gov>
**Subject:** Re: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case

Yeah, I got a note a week or two ago telling me the submission deadline for PQCRYPTO. I think that this was all done irresponsibly late, but it's what we have to deal with.

At this point, I think that this enhancement would be appropriate for submission there. What do you guys think?

Can you send me your code? I might be able to write something raw in C that's faster. It will be easier to tinker with yours than to start from scratch. Of course, magma might be reasonable as well, but for now I don't have access either here or at NIST (due to the incompetence of several individuals, including myself, all of whom are in Louisville [surprisingly?]).

I'd like to get some time to work on the extension field cancellation now that stress at not having broken it is setting in, but I doubt I'll have time. I'm trying to get two of my students to complete projects and hopefully submit them as well. I'll try to get back to you if I can on that.

Cheers and Happy New Year!

On Mon, Jan 9, 2017 at 3:10 PM, Moody, Dustin (Fed) <dustin.moody@nist.gov> wrote:

> Daniel,
> FYI, today Ray and I worked on this a bit. We modified our previous code in SAGE to do what Ray wants it to do. We're running some experiments to verify it is behaving as it should. Preliminary indications are that things looks like Ray predicted. He estimated it would take around 2500 trials to find a rank 8 matrix for q=4 and s=4. My program is slow, but it did it in 4300 trials the first time, taking around an hour and a half. I'm doing more experiments to see if we average closer to 2500. If I were a good programmer this might be quicker, but alas….
>
> Also, note the submission deadline to PQCrypto is Valentines Day, Feb. 14th.
>
> Dustin
>
> **From:** Daniel Smith(b) (6)
> **Sent:** Friday, December 16, 2016 12:02 PM
> **To:** Moody, Dustin (Fed) <dustin.moody@nist.gov>; Perlner, Ray (Fed) <ray.perlner@nist.gov>
>
> **Subject:** Re: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
>
> Sorry for being so slow. I'm working on about 5 projects and the limit of my ability is about .5 projects. I'll try to give this a look tonight.
>
> When the result is more mature, I would like to talk to you about a project I'm doing with one of my students on a complexity theoretic proof of security for "big structure" multivariate schemes. I'd like an impartial audience to let me know if it seems too BS, or is essentially equivalent, in terms of credibility, to other types of reduction in pqc.
>
> Another random thought, we can think of an ideal I in F[x1,...,xn] as a lattice in a way. If F is finite of order q and I contains the field equations, x^q-x, then I is radical, and I(V(I))=I by the nullstellensatz. Furthermore, since the field equations are in I, V(I) is in F^n, and not merely in a vector space over an algebraic closure of F. In this case, since V(I) is finite, we can express it as a disjoint union of singleton sets, and so I is the intersection of the corresponding maximal ideals in F[x1,...,xn]. In the special case of encryption, we expect that V(I) is a singleton and so I is maximal. Then I=<x1-a1,x2-a2,...,xn-an>. This is a good basis. If we have a bad basis (which is what we typically have for mpk schemes) in general it is hard to find a good basis. So what if we use some interesting metric, such as the Lee metric on the coefficients of the monomials of a polynomial f. Can we do something like solve a closest vector problem given a good basis which is hard to solve with the bad basis? Bo-Yin had a lattice-like multivariate scheme, but the linear part served as the lattice and the quadratic part was noise. Since F[x1,...,xn] is an integral domain just the same as Z, why can't we accomplish something similar with a more general integral domain?
>
> Cheers,
> Daniel
>
> On Fri, Dec 16, 2016 at 10:16 AM Perlner, Ray (Fed) <ray.perlner@nist.gov> wrote:
>
>> Oh. Forgot to note, whenever I say f, I mean the homogeneous quadratic part (in the regular ABC case) or homogeneous cubic part (in the cubic ABC case.)
>>
>> **From:** Perlner, Ray (Fed)
>> **Sent:** Friday, December 16, 2016 10:06 AM
>> **To:** Moody, Dustin (Fed) <dustin.moody@nist.gov>; Daniel Smith(b) (6)
>> **Subject:** RE: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
>>
>> I have a much simplified method for recovering the missing linear constraints on t in the minrank equations for characteristic 2. I think the complexity of our attack will be q^(s+2)s^(2 omega) field operations for both even and odd characteristic.:
>>
>> In the quadratic case, in addition to requiring Df(u) = 0 and Df(v) = 0 for band kernel vectors u and v, we can also require f(u) = 0 and f(v) = 0.
>>
>> In the cubic case, instead of requiring that D^f(u, v) = 0, we can require that d/dx_i(f) evaluated at u and v is 0.
>>
>> d/dx_i is just a formal derivative d/dx_i (x_i x_j x_k) is x_j x_k for j and k not equal i.
>> d/dx_i (x_i ^2 x_j) is 2x_i x_j for j not equal i.
>> d/dx_i (x_i ^3) is 3x_i ^2.
>>
>> I believe that all 2s^2 linear constraints you get this way are linearly independent for characteristic 2, but for characteristic 3, we also need to throw in
>>
>> f(u) = 0 and f(v) = 0. Doing so will save us a factor of q work.
>>
>> **From:** Moody, Dustin (Fed)
>> **Sent:** Monday, December 12, 2016 2:45 PM
>> **To:** Daniel Smith <(b) (6) >; Perlner, Ray (Fed) <ray.perlner@nist.gov>
>> **Subject:** RE: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
>>
>> Greetings to you, fellow human colleague. I am inclined to acquiesce to your request.
>>
>> Any chance you will be coming to NIST anytime soon? If not, we can always communicate through other methods, such as a google hangout with no audio!
>>
>> **From:** Daniel Smith(b) (6)
>> **Sent:** Monday, December 12, 2016 2:42 PM
>> **To:** Perlner, Ray (Fed) <ray.perlner@nist.gov>
>> **Cc:** Moody, Dustin (Fed) <dustin.moody@nist.gov>
>> **Subject:** Re: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
>>
>> Hi, human colleagues,
>>
>> Would you like to develop these ideas more fully? I think that we'll need to get some running examples to see the cost of the modification Ray suggested. I'm not sure, offhand if there is any special algebraic structure relating to this or if this is merely a way of breaking the symmetry, as Ray suggested, that produces a benefit because the attack is exponential-ish. (I'm always trying to tie attack ideas to specific principles to propose security metrics.) I'm going to start thinking about that here at the end of the year.
>>
>> It might be good also if we can enrich our paper with more data for an eprint version.
>>
>> Cheers,
>> Daniel
>>
>> On Thu, Sep 15, 2016 at 3:11 PM, Perlner, Ray (Fed) <ray.perlner@nist.gov> wrote:
>>
>>> True enough. It's probably offset somewhat, but not entirely, by the fact that operations over F_2 are cheaper than operations over F_q if you do them right. In any event, the cost is only polylog(q). It should be more than made up for by replacing the q^{2s+6} with q^{s+2}
>>>
>>> **From:** Daniel Smith(b) (6)
>>> **Sent:** Thursday, September 15, 2016 2:50 PM
>>> **To:** Perlner, Ray (Fed) <ray.perlner@nist.gov>
>>> **Cc:** Moody, Dustin (Fed) <dustin.moody@nist.gov>
>>> **Subject:** Re: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case
>>>
>>> Wouldn't that hurt the linear algebra steps considerably, though? The search space should be the same size. I guess that it is still better to have the extra constraint, though, but there is still a slow down compared to higher characteristics.
>>>
>>> On Thu, Sep 15, 2016 at 2:37 PM, Perlner, Ray (Fed) <ray.perlner@nist.gov> wrote:
>>>
>>>> Recall that the problem was that you couldn't impose a meaningful linear constraint on t_i by saying sum (t_i D^2f_i(x1, x1)) = 0 due to the symmetry of the differential. The solution is to use something that looks like a differential, but doesn't have that symmetry.
>>>>
>>>> Instead of having Df(x,a) = f(x+a) - f(x) - f(a) + f(0), pick an element of the base field, s and use D_{s}f(x,a) = f(sx + a) - sf(x) - f(a) + sf(0).
>>>>
>>>> Note that while D_{s}D_{t}f(x,a,b) does not make a cubic map trilinear over the base field, it does make it trilinear over F_2, so we can still use D_{s}D_{t} to do minrank (it's just that the linear algebra will be over F_2 instead of F_q.)