From: (b) (6)
To: Moody, Dustin (Fed)
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 31, 2017 1:12:49 PM
Attachments: cabc.mgm

I'm 1000 steps behind you. Average of 2000 means average value of "j" in your code?

Something interesting is happening with my code. I can't guarantee that it is doing the right thing, since I basically tried to translate your code into magma (I translated the slower code for G, but I don't think that should be more than a few milliseconds difference per iteration, so a couple of seconds difference for the entire calculation (so not worth it). The input to G are still vectors with first half zero. I ran it without generating elements in the right kernel, but only inputting the basis elements. It seemed to spit out an answer quickly, which I don't quite understand. It makes me think that my code is wrong somehow.

I'm attaching what I've done. Maybe you can spot issues. The one thing I know of is the kernel thing, but if it's correct and still gives answers, that's interesting.

Cheers,
Daniel

On Tue, Jan 31, 2017 at 12:48 PM, Moody, Dustin (Fed) <dustin.moody@nist.gov> wrote:

> You are right to question things. I make no guarantee the code I write is optimal!
>
> M.right_kernel() returns a vector space, with the degree, dimension, and a basis matrix.
>
> ```
> >M.right_kernel()
> Vector space of degree 32 and dimension 1 over Finite Field
> in z of size
> 2^2
> Basis matrix:
> [     0      0      0      1      0      z      1  z + 1      0      0
> 0      1
> 0      z      1  z + 1      0      0      0      0      0      0      0
> 0
> 0      0      0      1      0      z      1  z + 1]
> ```
>
> MK.list() creates a list of all the elements in the kernel. I do agree it is probably quite slow and eats up a bunch of memory. I will try changing it and see how it runs.
>
> By the way, my most updated stats for q=4 and s=4 are an average of just under 2000. I think Ray predicted 2300. For s=5, 3 trials have completed, for an average of 7800. Ray predicted around 9000. I don't know that we need to put detailed computational evidence in, since we don't have a lot. But we could just say we performed some experiments as a

sanity check, which seemed to corroborate our findings.

Dustin

From: Daniel Smith (b) (6)
Sent: Tuesday, January 31, 2017 12:38:28 PM
To: Moody, Dustin (Fed)
Cc: Perlner, Ray (Fed)

Subject: Re: I think I figured out how to extend our cubic ABC attack to the characteristic 2 case

What does M.right_kernel() return? A basis?

If so, what is MK.list()? Is it a list of basis vectors?   Is it a list of every element in the right kernel? If it is the former, then why do you need to test whether t is zero, and if the latter, wouldn't that slow down the algorithm and eat up a lot of memory?

On Tue, Jan 10, 2017 at 12:09 PM, Daniel Smith (b) (6) wrote:

> Thanks. I'll try to look at this this week.
>
> Cheers!
>
> On Tue, Jan 10, 2017 at 11:12 AM, Moody, Dustin (Fed) <dustin.moody@nist.gov> wrote:
>
> > For the faster code, replace the definition of G(v1,v2) with the following:
> >
> > ```
> > def Polylist():
> >
> >   EL2=matrix(R,2*s^2,s^2,lambda i,j:0)
> >
> >   # elements of E - cubic's
> >
> >   for h in range(0,2*s^2):
> >
> >   enew=EL[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
> >
> >   EL2[h,k-1]=poly(x1=0,x2=0,x3=0,x4=0,x5=0,x6=0,x7=0,x8=0)
> >
> >   return EL2
> > ```

```
EL2=Polylist()


def G(v1,v2):

    mnew=matrix(K,2*s^2,2*s^2,lambda i,j:0)

    for h in range(0,2*s^2):

        for k in range(1,s^2+1):

            poly=EL2[h,k-1]

            mnew[h,k-1] = poly(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(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])

    return mnew
```

Define a new function func (which is used to make the first half of v1 and v2 be zero, and the rest random)

```
def func(i):

    if i<=8:

        return 0

    return K.random_element()


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: func(j))
```

```
v2=matrix(K,1,s^2,lambda i,j: func(j))
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) [redacted]
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) [redacted]
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) [redacted]
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) [redacted]
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.)

```
p:=2;
k:=2;
s:=9;
q:=p^k;
F<a>:=GF(q);
R<[x]>:=PolynomialRing(F,s^2,"grevlex");
//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:=[x[i] : i in [1..s^2]];
//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]

randomLinearCombination:= function(s,F)
        temp:=0;
        for i in [1..s^2] do
            temp:=temp+Random(F)*x[i];
        end for;
        return temp;
end function;

randomQuadraticForm:= function(s,F)
        temp:=0;
        for i in [1..s^2] do
            for j in [1..s^2] do
                temp:=temp+Random(F)*x[i]*x[j];
            end for;
        end for;
        return temp;
end function;

setup := function(s)
        A:=[[randomQuadraticForm(s,F) : i in [1..s]] : j in [1..s]];
        B:=[[randomLinearCombination(s,F) : i in [1..s]] : j in [1..s]];
        C:=[[randomLinearCombination(s,F) : i in [1..s]] : j in [1..s]];
        A:=Matrix(A);
        B:=Matrix(B);
        C:=Matrix(C);
        AB:=A*B;
        AC:=A*C;
        E:=VerticalJoin(AB,AC);
        U:=RandomMatrix(F,s^2,s^2);
        while Rank(U) lt s^2 do
            U:=RandomMatrix(F,s^2,s^2);
        end while;
        T:=RandomMatrix(F,2*s^2,2*s^2);
        while Rank(T) lt 2*s^2 do
```

```
            T:=RandomMatrix(F,2*s^2,2*s^2);
        end while;
        return E,A,B,C,T,U;
end function;

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

EL:=ElementToSequence(E);

//PolynomialList:= function()
        //EL2:=ZeroMatrix(F,2*s^2,s^2);
        //for h in [1..2*s^2] do
            //enew:=EL[h];
            //for j in [1..s^2] do
                //c3:=Coefficient(enew,x[j],3);
                //c2:=Coefficient(enew,x[j],2);
                //c1:=Coefficient(enew,x[j],1);
                //poly:=3*c3*x[j]^2+2*c2*x[j]+c1;
                //// don't know what is happening here
                //EL2[h][j]:=

G:=function(v1,v2)
        mnew:=ZeroMatrix(F,2*s^2,2*s^2);
        for h in [1..2*s^2] do
            enew := EL[h];
            for j in [1..s^2] do
                c3:=Coefficient(enew,x[j],3);
                c2:=Coefficient(enew,x[j],2);
                c1:=Coefficient(enew,x[j],1);
                poly:=3*c3*x[j]^2+2*c2*x[j]+c1;
                if s eq 3 then
                   mnew[h,k] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(poly,x[1],v1[1]),x[2],v1[2]),x[3],v1[3]),x[4
],v1[4]),x[5],v1[5]),x[6],v1[6]),x[7],v1[7]),x[8],v1[8]);
                   mnew[h,k+s^2] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(poly,x[1],v2[1]),x[2],v2[2]),x[3],v2[3]),x[4
],v2[4]),x[5],v2[5]),x[6],v2[6]),x[7],v2[7]),x[8],v2[8]);
                elif s eq 4 then
                   mnew[h,k] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evalu
ate(Evaluate(Evaluate(Evaluate(poly,x[1],v1[1]),x[2],v1[2]),x[3],v1[3]),x[4],v1[4]),x[5],v1[5]),x[6],v1[6]),x[7],v1[7]),x
[8],v1[8]),x[9],v1[9]),x[10],v1[10]),x[11],v1[11]),x[12],v1[12]),x[13],v1[13]),x[14],v1[14]),x[15],v1[15]),x[16],v1[16])
;
                   mnew[h,k+s^2] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evalu
ate(Evaluate(Evaluate(Evaluate(poly,x[1],v2[1]),x[2],v2[2]),x[3],v2[3]),x[4],v2[4]),x[5],v2[5]),x[6],v2[6]),x[7],v2[7]),x
[8],v2[8]),x[9],v2[9]),x[10],v2[10]),x[11],v2[11]),x[12],v2[12]),x[13],v2[13]),x[14],v2[14]),x[15],v2[15]),x[16],v2[16])
;
                elif s eq 5 then
                   mnew[h,k] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evalu
ate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(p
oly,x[1],v1[1]),x[2],v1[2]),x[3],v1[3]),x[4],v1[4]),x[5],v1[5]),x[6],v1[6]),x[7],v1[7]),x[8],v1[8]),x[9],v1[9]),x[10],v1[10
]),x[11],v1[11]),x[12],v1[12]),x[13],v1[13]),x[14],v1[14]),x[15],v1[15]),x[16],v1[16]),x[17],v1[17]),x[18],v1[18]),x[19
],v1[19]),x[20],v1[20]),x[21],v1[21]),x[22],v1[22]),x[23],v1[23]),x[24],v1[24]),x[25],v1[25]);
                mnew[h,k+s^2] :=
Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evalu
ate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(Evaluate(p
oly,x[1],v2[1]),x[2],v2[2]),x[3],v2[3]),x[4],v2[4]),x[5],v2[5]),x[6],v2[6]),x[7],v2[7]),x[8],v2[8]),x[9],v2[9]),x[10],v2[10
]),x[11],v2[11]),x[12],v2[12]),x[13],v2[13]),x[14],v2[14]),x[15],v2[15]),x[16],v2[16]),x[17],v2[17]),x[18],v2[18]),x[19
],v2[19]),x[20],v2[20]),x[21],v2[21]),x[22],v2[22]),x[23],v2[23]),x[24],v2[24]),x[25],v2[25]);
            end if;
         end for;
      end for;
      return mnew;
end function;

DDF:=[];

setupDDF:=function()
      DDF:=[];
      for h in [1..2*s^2] do
         e:=EL[h];
         V:=[];
         for j in [1..s^2] do
            c3:=Coefficient(e,x[j],3);
            c2:=Coefficient(e,x[j],2);
            c1:=Coefficient(e,x[j],1);
            m:=ZeroMatrix(F,s^2,s^2);
            m[j,j]:=6*c3;
            for i in [1..s^2] do
               if i ne j then
                  m[j,i]:=2*Coefficient(c2,x[i],1);
                  m[i,j]:=m[j,i];
               end if;
            end for;
            for i in [1..s^2] do
               if i ne j then
                  m[i,i]:=2*Coefficient(c1,x[i],2);
               end if;
               for l in [1..s^2] do
                  if i ne j then
                     if i ne l then
                        if j ne l then
                           m[i,l]:=Coefficient(Coefficient(c1,x[i],1),x[l],1);
                           m[l,i]:=m[i,l];
                        end if;
                     end if;
                  end if;
               end for;
            end for;
            V:=V cat [m];
         end for;
         DDF:=DDF cat [V];
      end for;
      return DDF;
end function;
```

```
DDF:=setupDDF();

linearCombinationDDF:=function(h,v)
        MS:=DDF[h];
        sm:=0;
        for i in [1..s^2] do
            sm:=sm+v[i]*MS[i];
        end for;
        return sm;
end function;

experiment:=function()
        //DDF:=setupDDF();
        ansj:=0;
        V:=[0 : i in [1..s^2]];
        flg:=0;
        t:=Cputime();
        while flg eq 0 do
              for j in [1..10000] do
                  if (j mod 100) eq 0 then
                     V;
                  end if;
                  v1:=[0 : i in [1..Floor(s^2/2)]];
                  v1:=v1 cat [Random(F) : i in [(Floor(s^2/2)+1)..s^2]];
                  v2:=[0 : i in [1..Floor(s^2/2)]];
                  v2:=v2 cat [Random(F) : i in [(Floor(s^2/2)+1)..s^2]];
                  M:=Transpose(G(v1,v2));
                  MK:=NullspaceMatrix(M);
                  MKL:=RowSequence(MK);
                  mk:=s^2;
                  for l in [1..#MKL] do
                      //if not IsZero(Matrix(MKL[l])) then
                      sm1:=0;
                      for i in [1..2*s^2] do
                          sm1:=sm1+MKL[l][i]*linearCombinationDDF(i,v1);
                      end for;
                      sr:=Rank(sm1);
                      mk:=Minimum(mk,sr);
                      if mk eq 2*s then
                         ansj:=j;
                         ansMKV:=MKL[l];
                         ansv1:=v1;
                         ansv2:=v2;
                         flg:=1;
                         break;
                      end if;
                      // end if;
                  end for;
                  V[mk]:=V[mk]+1;
                  if flg eq 1 then
                     break;
                  end if;
              end for;
        end while;
        compTime:=Cputime(t);
        return ansj,ansMKV,ansv1,ansv2,compTime;
end function;

//sample call steps,kervec,v1,v2,compTime:=experiment();
```