From: Ward Beullens <ward.beullens@student.kuleuven.be>
Sent: Saturday, December 23, 2017 5:43 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: DME

Dear all,

There seems to be a mistake in the dme_implementation document. The irreducible polynomial S^3 + c S^2 + d S + e defined at page 3 is in fact not irreducible.

The constants c and d in the document do not agree with the value in the reference implementation (which are likely the correct values because they do define an irreducible polynomial).

Kind regards,

From: Ward Beullens <ward.beullens@student.kuleuven.be>
Sent: Thursday, December 28, 2017 6:14 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: DME

Dear all,

I believe that the DME system does not reach the claimed level of security.

The public key is a polynomial map with 6 components in 6 variables over a finite field GF(2^48) of the form P = L_3 ° G_2 ° L_2 ° G_1 ° L_1 , where the L_i are invertible linear maps.

If we represent the map over GF(2), we get a map P' of 6*48 components in 6*48 variables, and the maps G_i become quadratic maps G'_i.

We can decompose the degree 4 map P' into the composition of two quadratic maps with the algorithm of Faugère and Perret [1]. The Complexity of this algorithm is O(n^9), where n is the number of variables, so we can estimate the bit cost of this step to be roughly
(6*48)^9 , which is approximately 2^74, much less than the claimed security level of 256 bits.

Now we have P' = Q2 ° Q1, where we know that the Q1 is isomorphic to G'_2, and Q1 isomorphic to G'_1. Generic algorithms that solve the Isomorphism of Polynomials e.g.[2] can recover this isomorphism with a cost that is O(2^n/2), so approximately 2^144. Once these isomorphisms are recovered, they can be used to invert the public key and thus break the cryptosystem.

Note that both steps (finding the decomposition, and finding the
isomorphisms) are done with generic algorithms. It is conceivable that the structure of the public map can be exploited to do the steps more efficiently. (And this seems very likely for the isomorphism finding step)

Kind regards,
Ward

[1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.

[2] Bouillaguet, Charles, Pierre-Alain Fouque, and Amandine Véber.
"Graph-theoretic algorithms for the “isomorphism of polynomials”
problem." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2013.

From: IGNACIO LUENGO VELASCO <iluengo@ucm.es>
Sent: Friday, December 29, 2017 7:17 PM
To: pqc-forum
Cc: pqc-comments
Subject: Re: OFFICIAL COMMENT: DME

Dear Ward,

thanks again for your remarks. I was not aware of the paper [1]. I will check the paper and i will check also your estimates,
but I believe that they are OK. That means that we need to change the parameters of DME to double the security bits,
we will explore and implement different options:

DME(4,2,64) this scheme will give 64 monomials in each polynomial, double the message and public key size
and I estimate that it will multiply by 4 the computation time.
DME(6,2,44) this scheme will produce 144 monomials, double the message size and multiply by 4 (roughly).
We do not implemented yet this scheme and we do not have estimate for the time.

A better option will be change the matrix of the first exponential by putting 3 non zero entries instead of to 2.
Let's denote by DME3 this scheme then DME3(4,2,36) will give the same message size (288 bits) but
it will produce 256 monomials and multiply by 4 the public key size. In this case when when we represent
the map over F_2 we get degree 6 polynomial map that is a composition of a degree 3 and a degree 2
polynomial map. We will try to compute the complexity of the IP problem for the cubic polynomials to
estimate the complexity of your attack for this scheme.

Once you have the decomposition and the 2 quadratic maps G_1 and G_2 how do you invert them and what is the complexity?

Best regards,
Ignacio

El viernes, 29 de diciembre de 2017, 0:14:28 (UTC+1), Ward Beullens escribió:

> Dear all,
>
> I believe that the DME system does not reach the claimed level of
> security.
>
> The public key is a polynomial map with 6 components in 6 variables over
> a finite field GF(2^48) of the form P = L_3 ° G_2 ° L_2 ° G_1 ° L_1 ,
> where the L_i are invertible linear maps.
>
> If we represent the map over GF(2), we get a map P' of 6*48 components
> in 6*48 variables, and the maps G_i become quadratic maps G'_i.
>
> We can decompose the degree 4 map P' into the composition of two
> quadratic maps with the algorithm of Faugère and Perret [1]. The
> Complexity of this algorithm is O(n^9), where n is the number of
> variables, so we can estimate the bit cost of this step to be roughly

From: Ward Beullens <Ward.Beullens@esat.kuleuven.be>
Sent: Friday, January 05, 2018 10:23 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: DME

Dear Ignacio, Dear all,

TL;DR: I the new parameters of the DME scheme are not secure. The following describes an attack that reduces the security in the exponent by more than half.

Recall that the attack on DME that I proposed earlier consisted of 2 steps.

1) Represent the public key map over F_2, wich will make it a polynomial map of degree 4. Decompose this polynomial map into the composition of two quadratic polynomial maps.

2) We know each of the quadratic components is isomorphic to a known quadratic map (the representation of G_1 and G_2 over F_2) which we know how to invert. The second step is to recover this isomorhpism with an IP solver.

The complexity of the first step with the algorithm of [1] is O(n^9), where n is the number of variables of the decomposed map, or equivalently the number of bits of a ciphertext.

The complexity of the second with the algorithm of [2] is O(2^{n/2}).

The adaptations proposed by Ignacio increase n enough to make the second step infeasible. But the complexity of the first step remains much lower than the targeted security level.

I did some experiments and it turns out that the degree of regularity of the quadratic components is very low. In fact, in all experiments the highest degree reached during a GB computation was 3. This means that once a decomposition is found, the public map can be efficiently inverted by doing a GB computations on each component. This means that the complexity of the attack is dominated by the complexity of the first step.

Ignacio also proposed to use 3 (instead of 2) nonzero entries in the rows of matrix of the first exponentiation. With this adaptation, if we represent the public key over F_2 it becomes a degree 6 polynomial map which is the composition of a degree 2 with a degree 3 map. I believe (correct me if I am wrong) that the complexity of decomposing this map with the algorithm of [1] is O(n^15).

According to my experiments the degree of regularity of the cubic component map is 4, so preimages for this map can still be found efficiently with a GB computation, and again we can say that the complexity of the attack is dominated by the complexity of decomposing the public key.

For the different parametrizations of Ignacio I estimate the complexity of the attack as :

DME(4,2,64) -> (4*2*64)^9 ~ 2^81
DME(6,2,44) -> (6*2*44)^9 ~ 2^81

DME3(4,2,36) -> (4*2*36)^15 ~2^122

So none of the proposed parameter sets seem to reach the claimed security level of 256 bits.

I'll finish with a note on the experiments I did. I did not implement the algorithm for decomposing the public key. I generated the two components L_3 o G_2 o L2 and G_1 o L_1, represented them over F_2, composed these quadratic maps with random invertible linear maps over F_2. This is what the output of the decomposition algorithm [1] would output. Then I used the PolyBoRi library [3] to find an inverse for a random point in GF(2)^{6*e}. The highest degree encountered during the GB computation was 3, except in the DME3 case, where the degree was 4. As expected I always found a unique solution.

Finding an inverse for one of the components of DME(3,2,24) (which was originally proposed to have 128 bits of security) took only 565 seconds.

Kind regards,
Ward

[1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.

[2] Bouillaguet, Charles, Pierre-Alain Fouque, and Amandine Véber. "Graph-theoretic algorithms for the "isomorphism of polynomials" problem." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2013.

[3] http://doc.sagemath.org/html/en/reference/polynomial_rings/sage/rings/polynomial/pbori.html

Sage code for the experiments on the first component of DME(3,2,24), the code for the second component and the experiments for DME3 is very similar. (This was my first time working with sage, so I apologize if my code is not very clean.)

```
from time import time

e = 24;
KS= GF(2);
K = GF(2^e);

R = PolynomialRing(K,'X');
R.inject_variables();

#Find irreducible polynomial
while True:
   a = K.random_element();
   b = K.random_element();
   IP = X^2 + a*X + b;
   if IP.is_irreducible():
      break;

#multiplication over a degree 2 extension field of K
def mulK2(A,B):
   t0 = A[0]*B[0];
   t1 = A[1]*B[0] + A[0]*B[1];
```

```
    t2 = A[1]*B[1];
    return (t0 + b*t2 , t1 + a*t2)

#squaring an element of a degree 2 extension of K a total of n times
def squareK2(A,n):
    if n==0:
        return A;
    B = mulK2(A,A);
    return squareK2(B,n-1);

#find values of Eij such that the exponentiation map is bijective (for vectors with nonzero entries)
while True:
    E11 = ZZ.random_element(0,2*e);
    E12 = ZZ.random_element(0,2*e);
    E21 = ZZ.random_element(0,2*e);
    E23 = ZZ.random_element(0,2*e);
    E32 = ZZ.random_element(0,2*e);
    E33 = ZZ.random_element(0,2*e);
    det = -2^(E11+E23+E32)-2^(E12+E21+E33);
    if gcd(det, 2^(2*e)-1) == 1 :
        break;

#Find L1
while True:
    L1 = matrix([[K.random_element(),K.random_element(),0,0,0,0],
                [K.random_element(),K.random_element(),0,0,0,0],
                [0,0,K.random_element(),K.random_element(),0,0],
                [0,0,K.random_element(),K.random_element(),0,0],
                [0,0,0,0,K.random_element(),K.random_element()],
                [0,0,0,0,K.random_element(),K.random_element()]]);
    if L1.is_invertible():
        break;

# matrix M for conversion between GF(2^e) qnd GF(2)^e
t = K.gen()
M = matrix([[ (t^j)^(2^i) for j in range(0,e)] for i in range(0,e)]);
Minv = M^-1

MM = matrix(K,6,6*e);

Mrow0 = M.matrix_from_rows([0]);

for i in range(0,6):
    MM.set_block(i,e*i,Mrow0);

def G1L1(x):
    x = L1*x;
    y = matrix(K,6,1);
    y[0],y[1] = mulK2(squareK2((x[0],x[1]),E11),squareK2((x[2],x[3]),E12));
    y[2],y[3] = mulK2(squareK2((x[0],x[1]),E21),squareK2((x[4],x[5]),E23));
    y[4],y[5] = mulK2(squareK2((x[2],x[3]),E32),squareK2((x[4],x[5]),E33));
    return y;
```

```
def G1L1Decomposed(x):
   return G1L1(MM*x);

R = BooleanPolynomialRing(e*6,'x')
X = R.gens()
polynomials = [0] * 6*e

#calculating the coefficients of representation of G_1 o L_1 over F_2
for i in range(0,6*e):
   for j in range(i,6*e):
      vec = matrix(K,6*e,1)
      vec[i] = 1;
      vec[j] = 1;
      Q = G1L1Decomposed(vec);
      for k in range(0,6):
         Powers = matrix([ [ (Q[k][0])^(2^n) ] for n in range(0,e)]);
         outVec = Minv * Powers;
         for l in range(0,e):
            if outVec[l,0] != 0:
               polynomials[e*k+l] += X[i]*X[j];

#Pick a random element to forge a signature for
for i in range(0,6*e):
   if (ZZ.random_element(2) == 0):
      #print(i)
      polynomials[i] = polynomials[i] + 1

I = R.ideal(polynomials)

start = time()

GB = I.groebner_basis(prot = True);

print("GB computation took",time()-start,"seconds")

for x in GB:
   print(x)
```

From: IGNACIO LUENGO VELASCO <iluengo@ucm.es>
Sent: Friday, January 05, 2018 7:14 PM
To: pqc-forum
Cc: pqc-comments; ward.beullens@esat.kuleuven.be
Subject: [pqc-forum] Re: OFFICIAL COMMENT: DME

Dear Ward, dear all

you are right on your comment about the posible decomposition of the map in two exponentials.
If it is easy to do the decomposition then there is not great advantage in to use two exponentials .
Thanks for the practical proof that if one decompose the cuartic map over F_2 in two caudratic maps
the resulting quadratic map are not difficult to invert for the proposed parameters, in fact the make reason
to take to exponential round was to make it safe against this attack over F_2 while keeping the message and key
size small.
Below I will explain how can we make the decomposition very hard with a small modification of DME3 , namely the total map will be
F: Fq^6-->Fq^8 instead of F: Fq^6-->Fq^6. First let me confess that I'm not totally convinced that the
main algorithm of [1] can be applied over F_2 as is. It seems to me that you need to use a big field, in the case of F_2
you can work over an extension F_(2^e) but in this case resulting the polynomials will be of
higher degree. Even if we use the complexity in [1] the following instance of DME3 will be safe:

We can change the initial linear map by adding a first map L0 : Fq^6-->Fq^8 as follows
L0(x1,...,x6)=(x1,...,x6,x2*x4*x6,0) and changing the first exponential map
G1: (Fq^2)^4--> (Fq^2)^4 that now is defined by a 4x4 matrix with with 3 non zero entries A1:=(aij)
and the entries of the last column aregy ai4=2^u+2^v. We use also that x2*x4*x6 is non zero.
Lets keep calling by DME3 the scheme. The parameter for the DME3(3,2,48) are
q=2^48, the total map is F: Fq^6--->Fq^8,
each component has 64 monomials and the size of the pk is 3072 bytes this gives
for kem ct=48 pk=3072 sk=288 bytes=33 instead of the original
ct=36 pk=2304 sk=288 bytes=33

If we represent the composition of the two exponentials over F2 the first one has degree
2+2*3=8 because of the exponents of x2*x4*x6. Then the complexity given by [1] is
O(n^45)=(48)^(3*15)=O(2^367).
Notice that by changing the exponents ai4 one can increase easyly the degree of the first
exponential over F_2, for instance taking ai4=2^u+2^v+2^w the degree of G1 is 2+2*4=10 that
gives a complexity for [1] of O(n^57)=(48)^(3*19)=O(2^465). That means that the scheme is practically
inmune against any algorithm of decomposition whose complexity depends stronglly on the
total degree of the maps over F_2.
We will use the web page http://www.mat.ucm.es/~iluengo/DME/ an updated version of the documentation
with the typos corrected and an apendix given more detailis of the setting of DME3. In a few weeks will
put an optimized implementation of DME3(3,2,48) and DME3(5,2,28), because we will revise first the
complexity of the de differents algorithms of decomposition over F_2.

Best regards,
Ignacio

From: IGNACIO LUENGO VELASCO <iluengo@ucm.es>
Sent: Thursday, January 11, 2018 4:06 PM
To: pqc-forum
Cc: pqc-comments; ward.beullens@esat.kuleuven.be
Subject: Re: OFFICIAL COMMENT: DME

Dear Ward, dear all,

I have found that in the interesting paper [2] the authors prove with a different method
the main results of [1] and they state in Theorem 1 that their result are valid for a
sufficiently large finite field.
Regarding the validity over F_2 in Problem 3 (p. 346) they ask "does there exist
a polynomial-time algorithm for Boolean polynomials?"
Please note that for any other field different than F_2 the complexity given by [1] for DME(3,2,48)
is bigger than 256 bits. For instance for F_4 the two polynomials g1 and g2 have degree 1=d2=4
and n=144 the complexity of [1] is O(n^3*(d1*d2-1))=O(2^322). For F_8, the degrees are d1=d2=8, n=72
and the complexity is O(n^3*(d1*d2-1))=O(2^1166).
Best regards,
Ignacio

[1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm
for decomposing multivariate polynomials and its applications to
cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.

[2] Shangwei Zhao, Ruyong Feng,Xiao-Shan Gao
"On functional decomposition of multivariate polynomials with differentiation
and homogenization" J. of Systems Science and Complexity, V. 25.2 (2012), pp 329–347

From: IGNACIO LUENGO VELASCO <iluengo@ucm.es>
Sent: Thursday, January 11, 2018 4:12 PM
To: pqc-forum
Cc: pqc-comments; ward.beullens@esat.kuleuven.be
Subject: [pqc-forum] Re: OFFICIAL COMMENT: DME

Dear all,

there is a missed parenthesis in the complexity formulas. They should be O(n^(3*(d1*d2-1)))=O(2^322) and O(n^(3*(d1*d2-1)))=O(2^1166).
I apologize for the typos.
Best regards,
Ignacio

El jueves, 11 de enero de 2018, 22:05:39 (UTC+1), IGNACIO LUENGO VELASCO escribió:

> Dear Ward, dear all,
>
> I have found that in the interesting paper [2] the authors prove with a different method
> the main results of [1] and they state in Theorem 1 that their result are valid for a
> sufficiently large finite field.
> Regarding the validity over F_2 in Problem 3 (p. 346) they ask "does there exist
> a polynomial-time algorithm for Boolean polynomials?"
> Please note that for any other field different than F_2 the complexity given by [1] for DME(3,2,48)
> is bigger than 256 bits. For instance for F_4 the two polynomials g1 and g2 have degree 1=d2=4
> and n=144 the complexity of [1] is O(n^3*(d1*d2-1))=O(2^322). For F_8, the degrees are d1=d2=8, n=72
> and the complexity is O(n^3*(d1*d2-1))=O(2^1166).
> Best regards,
> Ignacio
>
> [1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm
> for decomposing multivariate polynomials and its applications to
> cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.
>
> [2] Shangwei Zhao, Ruyong Feng,Xiao-Shan Gao
> "On functional decomposition of multivariate polynomials with differentiation
> and homogenization" J. of Systems Science and Complexity, V. 25.2 (2012), pp 329–347

From: Jacob Alperin-Sheriff <jacobmas@gmail.com>
Sent: Thursday, January 11, 2018 5:09 PM
To: IGNACIO LUENGO VELASCO
Cc: pqc-forum; pqc-comments; ward.beullens@esat.kuleuven.be
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: DME

I would like to remind/clarify to Ward and everyone else that NIST will be evaluating the security of all submissions (including but not limited to DME) with respect to the original algorithm and parameters contained in the submission; for the 1st round, as per the call for proposals, we are not allowing changes to submitted algorithm or parameters to avoid moving targets.

On Thu, Jan 11, 2018 at 4:12 PM, IGNACIO LUENGO VELASCO <iluengo@ucm.es> wrote:

> Dear all,
>
> there is a missed parenthesis in the complexity formulas. They should be O(n^(3*(d1*d2-1)))=O(2^322) and O(n^(3*(d1*d2-1)))=O(2^1166).
> I apologize for the typos.
> Best regards,
> Ignacio
>
> El jueves, 11 de enero de 2018, 22:05:39 (UTC+1), IGNACIO LUENGO VELASCO escribió:
>
>> Dear Ward, dear all,
>>
>> I have found that in the interesting paper [2] the authors prove with a different method
>> the main results of [1] and they state in Theorem 1 that their result are valid for a
>> sufficiently large finite field.
>> Regarding the validity over F_2 in Problem 3 (p. 346) they ask "does there exist
>> a polynomial-time algorithm for Boolean polynomials?"
>> Please note that for any other field different than F_2 the complexity given by [1] for DME(3,2,48)
>> is bigger than 256 bits. For instance for F_4 the two polynomials g1 and g2 have degree 1=d2=4
>> and n=144 the complexity of [1] is O(n^3*(d1*d2-1))=O(2^322). For F_8, the degrees are d1=d2=8, n=72
>> and the complexity is O(n^3*(d1*d2-1))=O(2^1166).
>> Best regards,
>> Ignacio
>>
>> [1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm
>> for decomposing multivariate polynomials and its applications to
>> cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.
>>
>> [2] Shangwei Zhao, Ruyong Feng,Xiao-Shan Gao
>> "On functional decomposition of multivariate polynomials with differentiation
>> and homogenization" J. of Systems Science and Complexity, V. 25.2 (2012), pp 329–347
>
> --
>

From: Ward Beullens <ward@beullens.com>
Sent: Thursday, January 11, 2018 6:03 PM
To: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: DME

Dear Ignacio,

Are you suggesting that the decomposition algorithm of [1] will not work on the PK of DME over F_2?

I am really not an expert on this functional decomposition stuff, but quoting the results of [2] as saying that the algorithm does not work over F_2 seems far-fetched. They prove that, given a random decomposable system of polynomials, the algorithm succeeds with probability close to one when the finite field is large enough. This is really not the same as saying that the algorithm does not work over F_2. Even if the algorithm of [1] only works for a small fraction of the public keys, this would still be a huge problem for DME.

Of course, the best way to find out is to run the algorithm and see if it works. I'll try to do this when I get the time.

Kind regards,
Ward.

Op 11/01/2018 om 22:05 schreef IGNACIO LUENGO VELASCO:

> Dear Ward, dear all,
>
> I have found that in the interesting paper [2] the authors prove with a different method
> the main results of [1] and they state in Theorem 1 that their result are valid for a
> sufficiently large finite field.
> Regarding the validity over F_2 in Problem 3 (p. 346) they ask "does there exist
> a polynomial-time algorithm for Boolean polynomials?"
> Please note that for any other field different than F_2 the complexity given by [1] for DME(3,2,48)
> is bigger than 256 bits. For instance for F_4 the two polynomials g1 and g2 have degree 1=d2=4
> and n=144 the complexity of [1] is O(n^3*(d1*d2-1))=O(2^322). For F_8, the degrees are d1=d2=8, n=72
> and the complexity is O(n^3*(d1*d2-1))=O(2^1166).
> Best regards,
> Ignacio
>
> [1] Faugère, Jean-Charles, and Ludovic Perret. "An efficient algorithm
> for decomposing multivariate polynomials and its applications to
> cryptography." Journal of Symbolic Computation 44.12 (2009): 1676-1689.
>
> [2] Shangwei Zhao, Ruyong Feng,Xiao-Shan Gao
> "On functional decomposition of multivariate polynomials with differentiation
> and homogenization" J. of Systems Science and Complexity, V. 25.2 (2012), pp 329–347
> --
>

From: Miguel Angel Marco <miguel.a.marco@gmail.com>
Sent: Saturday, January 13, 2018 12:14 PM
To: pqc-forum
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: DME

Dear Ward,

I am no expert in all this field either, so maybe I am missing something, but from a very quick look at paper [2], I see that in the proof of Theorem 5 they use the fact that matrix A is regular, and prove it by relating its determinant to Vandermonde determinants and a constant. This constant turns out to be a power of 2, so this argument fails in characteristic 2.

It could happen, however, that even if the argumemt given in the paper does not apply to characteristic 2, the method or some variant of it coud still find a decomposition of the map. As you said, the best way to know would be to actually implement and test it.

Best,

Miguel

From: Ignacio Luengo Velasco <iluengo@ucm.es>
Sent: Wednesday, October 17, 2018 2:08 PM
To: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: DME - Ward Beullens attack

Dear Ward,

we thank you for your nice work that breaks the implementation we
submitted to the NIST. We thank your final acknowledge that the attack can be
basically prevented by using a stronger padding scheme.

The attack is based on the property that the DME map is quasi-homogeneous for
certain choices of weights in the variables, that is if CT=(y1,..,y6)=F(x1,...,x6), then
for any c in Fq-{0] i we put (a1,a2,a3)=G1^(-1)(c,c,c) and (b1,b2)=G2(c,c), where
G1 and G2 are the matrices of the exponential then
F(a1*x1,a1*x2,a2*x3,a2*x4,a3*x5,a3*x6)=(b1*y1,b1*y2,b1*y3,b2*y4,b2*y5,b2*y6). That is an essential property
of the scheme,
but a proper padding with somee hash can be used to hide the quasi-homogeneity from the attacker.

Let me mention that, in the submission, we only used a very dummy padding
scheme (just setting 3 bytes to the fixed value 0x01) because we were very short of
time and we focused on the kernel of the DME, considering that the padding
could be added later by using some standard method. For that same reason, we
didn't include any secure treatment for the cases were the decapsulation finds
an unproperly padded message. The reason why Ward's attack succeeds is
precisely because we use this dummy padding scheme.

as we mention in the paper, in order to be able to resist against an IND-CCA2
attack were the attacker can perform 2^64 queries to the decapsulation oracle,
we would need at least a padding that involves more than 64 bits of randomness.

We are considering alternatives for a secure padding and
we are open for suggestions. If someone has an idea about a secure way to pad
the message, please get in contact with us.

We will shortly implement DME as described in the paper (with a proper
padding that adds 128 bits to a 256 bit message) and put the reference
implementation in our webpage and in github.

With this changes in the implementation we keep our claim that DME achieves
security level 5.

Let me also mention that in the submission paper we describe the use of DME
for signature but we where unable to finish the implementation on time for the
NIST deadline and that is the reason that we do not submit a candidate for
signature.

Finally, we would like to announce that we have found a new
algorithm for the key generation for the DME that reduces the time for key

generation time by a factor of 28 so an increase in size the field q will not represent a big penalty in the key generation time. Let me explain a little the details of the new algorithm that we will use in the new implementation.

The public key is formed by 6 polynomials {F_i} each of them has 64 monomials in 6 variables. The key generation algorithm in the reference implementation uses interpolation with 64 pairs of (plain_text, cypher_text) computed with private key.

The new key generation algorithm computes directly a 64x6 matrix with the coefficients of the polynomials {F_i} from the matrices of the (secret) linear components using operations on matrices that includes tensor (Kronecker) product of matrices.

Here I put the timing statistics for 1000 tests in a core i5 computer. We give timings and number of field additions and field multiplications:

```
SKEY->PKEY (via interpolation):  mean = 55340.570[us], std_dev =
261.851[us],
                              Fq(add) = 279859, Fq(mul) = 279940

SKEY->PKEY (via tensor product):  mean = 1942.790[us], std_dev = 8.880[us],
                              Fq(add) = 3901, Fq(mul) = 10142
```

As you can see the new algorithm is 28 faster. You can compare it with the time for encrypion and decryption and it is faster than decryption:

```
ENCRYPT: mean = 261.460[us], std_dev = 4.291[us],
          Fq(add) = 384, Fq(mul) = 1368

DECRYPT: mean = 4117.590[us], std_dev = 8.913[us],
         Fq(add) = 25188, Fq(mul) 25188
```

Best regards,
Ignacio

From: Ignacio Luengo Velasco <iluengo@ucm.es>
Sent: Friday, December 21, 2018 8:23 AM
To: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: DME-(4-2-48)

Dear all,

we have uploaded to the github an updated version of the DME KEM with CT=48 bytes, SS=32 bytes, that is we add to the shared key a random 16 bytes vector with a OAEP padding using sha-256 and sha-128. In order to get the 48 bytes of CT we have decided to use 8 variables a keep the same field F_q=F_2^48 instead of using 6 variables and q=2^64 that is DME-(4-2-48) instead DME-(3-2-64).
The link to github is :

https://github.com/miguelmarco/DME

It contain the reference (non optimized) implementation of NIST test and some extra test and material. With this changes in the implementation we keep our claim that DME achieves security level 5 and also that it is IND-CCA2 secure.
The reference implementation of DME-(4-2-48) do not contain the new and fast key generation algorithm computes directly a matrix with the
coefficients of the polynomials {F_i} from the matrices of the (secret) linear components using operations on matrices that includes tensor
(Kronecker) product of matrices. This new algorithm will be added later. For the implementation DME-(3-2-48) (available on request) the new generation algorithm is 28 faster and its running time is in between the ENCRYPT and DECRYPT times.

Best regards,
Ignacio Luengo