From: panyanbin@amss.ac.cn
Sent: Friday, January 05, 2018 7:42 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: Compact LWE

Dear Designers and all,

We group found that the following statement claimed by the designers is not true.

"*even if the hard problems in lattice, such as CVP and SIS, can be effciently solved, the secret values or pri-*

vate key in Compact-LWE still cannot be effciently recovered. This allows Compact-LWE to choose very small dimension parameters, such as n =8 in our experiment"

We group find a ciphertext-only attack against CompactLWE. More precisely, given a ciphertext, we can recover the corresponding message, without knowing the private key, by solving some (approximation-)CVP instance. Since the parameters recommended by the authors are small, we just need to solve (approximation-)CVP with a 128-dimensional lattice, which can be done efficiently with the lattice basis reduction algorithm. In our experiments, we can decrypt all the ciphertexts. So we make sure that CompactLWE with the small parameters recommended in the paper is NOT secure. The designers should enlarge the parameters to ensure the security.

The main steps of our attack is as following:

Step 1: Given a Compact-LWE ciphertext c=(la, d, lpk,lpk'), find a short enough vector l= （l'_1,l'_2,\cdots,l'_m) by lattice basis reduction algorithm, such that
\sum_{i=0}^m l'_i*a_i = la ;
\sum_{i=0}^m l'_i*pk_i = lpk mod q ;
\sum_{i=0}^m l'_i*pk_i' = lpk' mod q.

Step 2: Compute \sum_{i=0}^m l'_i*u_i, if l is short enough, then we can show that \sum_{i=0}^m l'_i*u_i=\sum_{i=0}^m l_i*u_i, where \sum_{i=0}^m l_i*u_i is the correct value that can be used to decrypt the ciphertext. So we can use \sum_{i=0}^m l'_i*u_i to recover the message.

The designers also considered the attack to ciphertexts in their paper. However, they thought that one need to guess the exact l to recover the message. Due to our attack, we find that we do not need to guess the exact l, a short l' can also help recover the message very well.

If there is something we miss, please tell us, thank you!

Best regards,

Haoyu Li, Renzhang Liu, Yanbin Pan, Tianyuan Xie

From: Xagawa Keita <xagawa@gmail.com>
Sent: Friday, January 05, 2018 3:15 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: Compact LWE

Dear designers, dear all,

The following sage script decrypts ciphertexts without the secret key.

https://gist.github.com/xagawa/ee91d51a56bda5292235e52640f57707

This attack is an extension of the plaintext-recovery attack in
https://ia.cr/2017/742
and the same as the Li-Liu-Pan-Xi attack.

Regards,
Jonathan, Mehdi, and Keita

Jonathan Bootle, Mehdi Tibouchi, and Keita Xagawa

From: Dongxi.Liu@data61.csiro.au
Sent: Friday, January 05, 2018 3:33 PM
To: panyanbin@amss.ac.cn; pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Hi Haoyu Li, Renzhang Liu, Yanbin Pan, Tianyuan Xie,

Would like to share you code? So your attack can be confirmed more conveniently. If this attack can be confirmed, we will change our scheme slightly by avoiding the use of u_{i} in the calculation of pk_{i} and pk_{i}' and will try your attack again.

Thanks.

Regards,
Dongxi Liu

From: panyanbin@amss.ac.cn <panyanbin@amss.ac.cn>
Sent: Friday, 5 January 2018 11:42 PM
To: pqc-comments@nist.gov
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear Designers and all,

We group found that the following statement claimed by the designers is not true.
"*even if the hard problems in lattice, such as CVP and SIS, can be effciently solved, the secret values or private key in Compact-LWE still cannot be effciently recovered. This allows Compact-LWE to choose very small dimension parameters, such as n =8 in our experiment*"

We group find a ciphertext-only attack against CompactLWE. More precisely, given a ciphertext, we can recover the corresponding message, without knowing the private key, by solving some (approximation-)CVP instance. Since the parameters recommended by the authors are small, we just need to solve (approximation-)CVP with a 128-dimensional lattice, which can be done efficiently with the lattice basis reduction algorithm. In our experiments, we can decrypt all the ciphertexts. So we make sure that CompactLWE with the small parameters recommended in the paper is NOT secure. The designers should enlarge the parameters to ensure the security.

The main steps of our attack is as following:
Step 1: Given a Compact-LWE ciphertext c=(la, d, lpk,lpk'), find a short enough vector l= （ l'_1,l'_2,\cdots,l'_m) by lattice basis reduction algorithm, such that \sum_{i=0}^m l'_i*a_i = la ;

From: Xagawa Keita <xagawa@gmail.com>
Sent: Friday, January 05, 2018 3:54 PM
To: pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

I already sent a mail but let me send it again.

Dear designers and all,

The following sage script decrypts ciphertexts without the secret key.
https://gist.github.com/xagawa/ee91d51a56bda5292235e52640f57707

This attack is an extension of the plaintext-recovery attack in https://ia.cr/2017/742
and the same as the Li-Liu-Pan-Xi attack.

Regards,
Jonathan, Mehdi, and Keita

Jonathan Bootle, Mehdi Tibouchi, and Keita Xagawa

From: Dongxi.Liu@data61.csiro.au
Sent: Friday, January 05, 2018 3:56 PM
To: xagawa@gmail.com; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear Jonathan, Mehdi, and Keita,

Thanks. Your script will be very helpful. We will have a look.

Regards,
Dongxi

From: Mehdi Tibouchi <tibouchi.mehdi@lab.ntt.co.jp>
Sent: Saturday, January 06, 2018 12:50 AM
To: Xagawa Keita
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear all,

For the sake of completeness, let me add to Keita's comment that the attack below (together with a full key recovery attack on the parameters of ia.cr/2017/685) is accepted for publication at CT-RSA 2018. We will update our eprint paper shortly to provide timings, etc., for the parameters in the NIST submission.

Based on our analysis, it seems very unlikely that the ways in which Compact-LWE departs from standard LWE-based schemes can improve security, and in particular that a reasonable level of security is somehow achievable in the very low dimensions proposed so far, even with further tweaks to the construction.

Best regards,

--
Jonathan, Mehdi and Keita.

From: Yanbin Pan <panyanbin@amss.ac.cn>
Sent: Saturday, January 06, 2018 9:47 AM
To: Tibouchi; Keita
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear all,

Thank Keita very much for sharing his code!

Our paper about the attack against NIST Compact-LWE has been posted by Cryptology ePrint Archive as https://eprint.iacr.org/2018/020 .

Thank Prof. Lepoint very much for telling us Bootle and Tibouchi's work (https://eprint.iacr.org/2017/742) against the former version of Compact-LWE(ia.cr/2017/685) when we submitted our paper to ePrint with exactly the same title with their paper. Prof. Lepoint asked us to change the title, cite their paper and make some comparison, and we did so. We found that the there are many differences between the former version and the current version submitted to NIST, and we guess that some changes are due to Bootle and Tibouchi's former attack.

If we understand Mehdi correctly, we think their work accpeted by CT-RSA2018 is the attack above against the former version of Compact-LWE, but NOT the NIST version, since by the paper submission deadline (Oct 1st (12.00 GMT) 2017) of CT-RSA2018, the NIST version had not appeared anywhere to our best knowledge, and they did not propose any comments to NIST or the designers before ours. We think we should make this point clear so that our attack indeed makes sense.

If we miss something, please tell us, thank you very much!

Best regards,
Haoyu, Renzhang, Yanbin, Tianyuan

From: Mehdi Tibouchi
Date: 2018-01-06 13:49
To: Xagawa Keita
CC: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear all,

For the sake of completeness, let me add to Keita's comment that the attack below (together with a full key recovery attack on the parameters of ia.cr/2017/685) is accepted for publication at CT-RSA 2018. We will update our eprint paper shortly to provide timings, etc., for the parameters in the NIST submission.

Based on our analysis, it seems very unlikely that the ways in which Compact-LWE departs from standard LWE-based schemes can improve security, and in particular that a reasonable level of security is somehow achievable in the very low dimensions proposed so far, even with further tweaks to the construction.

From: Dongxi.Liu@data61.csiro.au
Sent: Saturday, January 13, 2018 6:05 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear All,

As said before, to avoid the attack, we can change our scheme slightly by avoiding the direct use of u_{i} in the calculation of pk_{i} and pk_{i}'. We have done this change and the change is reflected into the Sage script provided by Keita. The changed Sage script is in the end of this email and it is also a complete reference implementation of the revised scheme.

Briefly, we split u_{i} into u_{1i} and u_{2i}, which are now randomly sampled from {0,…,q-1}, instead of from the much smaller {0,bp-1}, and two noiseless samples are generated in the public key to tackle the change to u_{i} .

Regards,
Dongxi Liu

```
#=========================
# The Sage script (https://gist.github.com/xagawa/ee91d51a56bda5292235e52640f57707)
# implements an attack to Compact-LWE found independently by
# Haoyu Li, Renzhang Liu, Yanbin Pan, Tianyuan Xie, and
# Jonathan Bootle, Mehdi Tibouchi, and Keita Xagawa.
# Based on that script, this script implements a slightly revised
# Compact-LWE and is used to test that attack again.
# In addition, the decryption algorithm of Compact-LWE is
# added in this script to make it a complete reference
# implementation for checking decryption correctness.
# Revisions are marked by "Change:".
# In this script, that attack cannot succeed even if m=32
# (instead of m=128 in the submitted version).
# When m < 32, the attack also fails sometimes due to invalid lcand.
# ========================
# Compact-LWE parameters in NIST PQC Round 1
# ========================
q=2^64
t=2^32
m=32
wPos=224
wNeg=32
b=16
#bp=0  #not needed as a parameter
n=8
ll=8
R=Integers(q)
sk_max = 229119
p_size = 16777216
```

```
e_min  = 457
e_max  = 3200
# ========================
def newkeygen():
   s =vector(R, [R.random_element() for _ in range(n)])
   sp=vector(R, [R.random_element() for _ in range(n)])
   k = 0
   while gcd(k,q)>1:
      k = randint(1,q-1)
   kp= 0
   while gcd(kp,q)>1:
      kp= randint(1,q-1)
   p=0

   #Correctness condition
   while gcd(p,q)>1 or sk_max*p + p* (wPos+wNeg) + e_max * p * (wPos+wNeg) >= q:
      p = int(q/(sk_max+wPos+wNeg+e_max * (wPos+wNeg))) -randint(0,p_size)

   #print "p=", p, sk_max*p + p* (wPos+wNeg) + e_max * p * (wPos+wNeg) < q

   #Change: ensruing sk*ck = skp*ckp, co-prime with p
   #       instead of sk*ck + skp*ckp co-prime with p

   sk  =0
   skp =0
   ck =0
   ckp=0
   while gcd(sk,p)>1:
      sk = randint(1,sk_max)

   while gcd(skp,p)>1:
      skp = randint(1,sk_max)

   while gcd(ck,p)>1:
      ck= randint(1,p)

   Rp = Integers(p)
   ckp= Rp(sk*ck)/Rp(skp)

   return s,k,sk,ck,sp,kp,skp,ckp,p

def newsamplegen(s,k,sk,ck,sp,kp,skp,ckp,p):
   Rp = Integers(p)
   A =random_matrix(ZZ,m,n,x=0,y=b)


   #Change: from u =vector(R,[randint(0,bp-1) for _ in range(m)]) into u1 and u2,
   #       sampled from q instead of bp
   u1 =vector(R,[randint(0,q-1) for _ in range(m)])
   u2 =vector(R,[randint(0,q-1) for _ in range(m)])
```

```
    e =vector(R, [randint(e_min,e_max) for _ in range(m)])
    ep=vector(R, [randint(e_min,e_max) for _ in range(m)])
    r =vector(R, [randint(0,p-1) for _ in range(m)])
    rp=vector(R, [(Rp(-ck * r[i].lift())/Rp(ckp)) for i in range(m)])

    #Change: v using u1 and vp using u2, instead of using the same u
    v = A*s + R(1)/R(k) * (sk*u1 + r + p*e)
    vp= A*sp+ R(1)/R(kp)* (skp*u2 + rp + p*ep)

    #Change: last two pk samples are new
    A[m-1,:] = vector(R,[0 for _ in range(n)])
    A[m-2,:] = vector(R,[0 for _ in range(n)])
    v[m-1]  = 0
    v[m-2]  = R(sk)/R(k)
    vp[m-1] = R(skp)/R(kp)
    vp[m-2] = 0
    u1[m-1]  = 0
    u1[m-2]  = 1
    u2[m-1]  = 1
    u2[m-2]  = 0

    return A,u1.lift(), u2.lift(), v.lift(),vp.lift(),e,ep

def generate_l():
# the sum of positive entries in l will be psel
# the sum of negative entries in l will be -nsel
    buf = [randint(0,255) for _ in range(512)]
    if wNeg > 0:
        psel  = wPos + (buf[0] % wNeg)
        nsel = buf[1] % wNeg
    else:
        psel  = wPos
        nsel = 0
   l = [0 for _ in range(m)]
    posc = 0
    negc = 0
    count = psel if (nsel == 0) else (psel/nsel + 1)
    for i in range(psel+nsel):
        slot = buf[i+2] % (m-2);
        if (count > 0) and (posc < psel):
            while (l[slot] < 0): # Move away from a negative entry
                slot = (slot + 1) % (m-2);
            l[slot] = l[slot] + 1
            count -= 1
            posc += 1
        else:
            vacant=m-2
            while (l[slot] > 0) and vacant>0: # Move away from a negative entry
                slot = (slot + 1) % (m-2);
                vacant-=1
            if vacant>0: #check whether all entries are positive
                l[slot] = l[slot] - 1
```

```
            count = psel if (nsel == 0) else (psel/nsel + 1)
            negc += 1

    return l

def sample_l(u):
    l = generate_l()
    #Change: checking of l not necessary
    #while not check_l(l,u):
    #    l = generate_l()
    return l

def rol(z,d):
    # cyclic left shift
    return int((z<<d)/t) + (z<<d)%t

def find_zp(z,t):
    zp = int(z/t)
    while gcd(zp,t) > 1:
        zp +=1
    return zp

def ske_encrypt(z,mu):
    zp = find_zp(z,t)
    z  = z%t
    d  = ((mu ^^ rol(z,int(log(t,2)/2))) * zp) % t
    return d

def ske_decrypt(z,d):
    zp = find_zp(z,t)
    z  = z%t
    zpinv = inverse_mod(zp,t)
    return ((zpinv * d) % t) ^^ rol(z,int(log(t,2)/2))

def encrypt(A,u1,u2, v,vp,mu):

    #Change: the last two elements in l are calculated from z1 and z2,
    #       not randomly sampled as other elements in l

    bp = int(q/(sk_max+wPos+wNeg+e_max * (wPos+wNeg))) - 2*p_size
    z1 = randint(0, int(bp/2)-1)
    z2 = randint(0, int(bp/2)-1)
    # z is a session key of KEM
    z = z1 + z2

    l = vector(ZZ,sample_l(u1))
    l[m-2] = (z1- l*u1) % q
    l[m-1] = (z2- l*u2) % q

    a = l * A
    x = (l * v ) % q
```

```
    xp= (l * vp) % q


    d = ske_encrypt(z,mu)
    return a.change_ring(ZZ),x,xp,d, l,z

def dec(s,k,sk,ck,sp,kp,skp,ckp,p,a,x,xp,d):
    d1 = ((x-s*a)*k)%q
    d2 = ((xp-sp*a)*kp)%q

    d3 = (ck*d1.lift()+ckp*d2.lift())%p  #skp*ckp*z

    Rp = Integers(p)
    z = Rp(d3)/Rp(skp*ckp)

    mu = ske_decrypt(z.lift(),d)


    return mu

def subsetsumdecrypt(A,v,vp,a,x,xp):
    kappa=q
    kappa2=q
    kappa1=t
    L=block_matrix(ZZ, \
        [[1, 0, -kappa*a.row(), -kappa2 * x, -kappa2 * xp], \
         [0, kappa1*identity_matrix(m), kappa*A, kappa2 * v.column(), kappa2 * vp.column()], \
         [0, 0, 0, kappa2*q, 0],\
         [0, 0, 0, 0, kappa2*q]])
    L=L.BKZ(block_size=10)

    #index of first non-zero entry in the first column of L
    idx=next((i for i,x in enumerate(L.column(0).list()) if x!=0))
    lcand = vector(ZZ,L[idx][1:m+1]/kappa1) if L[idx][0] == 1 else vector(ZZ,-L[idx][1:m+1]/kappa1)
    return lcand

def test_decrypt(trials=10,pairs=10,debug=true):
    succ=0
    correctness=0
    tottime=0.0
    if debug:
        print "Start!!!"
    for npair in range(pairs):
        s,k,sk,ck,sp,kp,skp,ckp,p = newkeygen()
        A,u1,u2,v,vp,e,ep = newsamplegen(s,k,sk,ck,sp,kp,skp,ckp,p)
        print "\n Key pair %d" % (npair)

        succnow=0
        correctnessnow=0
        for _ in range(trials):
            mu = randint(0,t-1)
            a,x,xp,d,l,z = encrypt(A,u1,u2,v,vp,mu)
```

```
        if(dec(s,k,sk,ck,sp,kp,skp,ckp,p,a,x,xp,d)==mu):
                correctnessnow+=1


        tm=cputime(subprocesses=True)
        lcand = subsetsumdecrypt(A,v,vp,a,x,xp)
        tottime+=float(cputime(tm))
        if debug:
           print "\nCipher          :", a,x,xp,d
           print "Cracked         :", lcand*A, (lcand*v) %q, (lcand*vp)%q
           print "l*u%q in Enc    :", z  #(l*u1+l*u2)%q,
           print "l'*u%q in Cracked:", (lcand*u1+lcand*u2)%q
          if (a!=lcand*A) or x!=(lcand*v) %q or xp!=(lcand*vp)%q:
                          print "invalid l' generated by subsetsumdecrypt"

        if z!=(l*u1+l*u2)%q:
           print "\n-----------------Implementation Bug-----------------"

        if z==(lcand*u1+lcand*u2)%q:
           print "\n!!!!!!!!!!!!!!!!!Successful Attack-----------------"


             #print "lcand=", lcand
        z = (lcand*u1) %q + (lcand*u2) %q
        if ske_decrypt(z,d) == mu:
           succnow+=1
        # if z == (lcand*u)%t:
        #    succnow+=1
      succ+=succnow
      correctness+=correctnessnow
      print "Key pair %d complete. \n  Attack  success   rate:  %d / %d" % \
           (npair,succnow,trials)
      print "  Decryption correctness: ", correctnessnow,"/",trials



   print "\nSuccessful recoveries with only ciphertexts: %d/%d (%f)." % \
        (succ,trials*pairs,RR(100*succ/trials/pairs))
   print "Correct decryption with private key: %d/%d (%f)." % \
        (correctness,trials*pairs,RR(100*correctness/trials/pairs))
   print "Average time: %f seconds." % (tottime/trials/pairs)

# use PRG for reproducibility
set_random_seed(0)

test_decrypt(10,10,true)

# =>
# Successful recoveries: 10/10 (100.000000).
# Average time: 3.395185 seconds.
```

From: Yanbin Pan <panyanbin@amss.ac.cn>
Sent: Sunday, January 14, 2018 3:36 AM
To: Dongxi.Liu@data61.csiro.au; pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear Dongxi and all,

We found a similar attack against the new version. See the following code. The idea behind the attack is similar to the former one: make the first m-2 components of l small enough, make l satisfy the equaion like before, and make \sum l_i u1_i and \sum_i l_i u2_i small (this is a new condition).
For m=50, we recover all the 10 instances.
For m =128, we can recover 7 instances in the 10 instances.
However, we note that the attack depends on the choice of the parameters in the attack heavily (such as kappa,kappa2), so we believe that if we adjust the parameters well or employ some other tricks, we can recover more, since the idea behind it is clear. Since we think the current attack has shown the weakness of the scheme, we won't do more experiments with more parameters.

Best regards,
Yanbin

```
def subsetsumdecrypt4(A,v,vp,a,x,xp,u1,u2):
   bp = int(q/(sk_max+wPos+wNeg+e_max * (wPos+wNeg))) - 2*p_size
   z1 = randint(0, int(bp/2)-1)
   kappa=q*2^8
   kappa2=q*2^8
   kappa1=t*2^8
   kappa3=1
   L=block_matrix(ZZ, \
      [[1*kappa1, 0, -kappa*a.row(), -kappa2 * x, -kappa2 * xp,0,0], \
       [0, kappa1*identity_matrix(m-2), kappa*A[0:m-2,:], kappa2 * v[0:(m-2)].column(), kappa2 * vp[0:(m-
2)].column(),kappa3*u1[0:(m-2)].column(),kappa3*u2[0:(m-2)].column()], \
       [0, 0, kappa*A[m-2:m,:], kappa2 * v[(m-2):m].column(), kappa2 * vp[(m-2):m].column(),kappa3*u1[(m-
2):m].column(),kappa3*u2[(m-2):m].column()], \
       [0, 0, 0, kappa2*q, 0,0,0],\
       [0, 0, 0, 0, kappa2*q,0,0],\
```

```
        [0, 0, 0, 0, 0, kappa3*q,0],\
        [0, 0, 0, 0, 0,0, kappa3*q]])
    L=L.BKZ(block_size=10)



    vtemp = vector(ZZ, [0 for _ in range(m-2)])
    #index of first non-zero entry in the first column of L
    idx=next((i for i,x in enumerate(L.column(0).list()) if (x/kappa1 == 1) or (x/kappa1 == -1) ))
    print idx;
    if idx==m+5: # never happen in the experiments, although return lcand is not correct, we do not
change
        print "error"
        return lcand
    print L[idx][:];
    lcand = vector(ZZ,L[idx][1:m+1]/kappa1) if L[idx][0] == kappa1 else vector(ZZ,-L[idx][1:m+1]/kappa1)
    vtemp = lcand[0:m-2]
    lcand[m-2] = L[idx,m+n+1]/kappa3 - vtemp*u1[0:m-2] if L[idx][0] == kappa1 else -
1*L[idx,m+n+1]/kappa3 - vtemp*u1[0:m-2]
    lcand[m-1] = L[idx,m+n+2]/kappa3 - vtemp*u2[0:m-2] if L[idx][0] == kappa1 else -
1*L[idx,m+n+2]/kappa3 - vtemp*u2[0:m-2]
    return lcand
```

Yanbin Pan

From: Dongxi.Liu@data61.csiro.au
Date: 2018-01-14 07:04
To: pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear All,

As said before, to avoid the attack, we can change our scheme slightly by avoiding the direct use of u_{i} in the calculation of pk_{i} and pk_{i}'. We have done this change and the change is reflected into the Sage script provided by Keita. The changed Sage script is in the end of this email and it is also a complete reference implementation of the revised scheme.

Briefly, we split u_{i} into u_{1i} and u_{2i}, which are now randomly sampled from {0,…,q-1}, instead of from the much smaller {0,bp-1}, and two noiseless samples are generated in the public key to tackle the change to u_{i} .

From: Dongxi.Liu@data61.csiro.au
Sent: Sunday, January 14, 2018 5:55 AM
To: panyanbin@amss.ac.cn; pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear Yanbin,

Thanks for your finding. I can confirm your new attack. We will look more on the weakness of our scheme.

Regards,
Dongxi

From: Yanbin Pan <panyanbin@amss.ac.cn>
Sent: Sunday, 14 January 2018 7:35 PM
To: Liu, Dongxi (Data61, Marsfield); pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Compact LWE

Dear Dongxi and all,

We found a similar attack against the new version. See the following code. The idea behind the attack is similar to the former one: make the first m-2 components of l small enough, make l satisfy the equaion like before, and make \sum l_i u1_i and \sum_i l_i u2_i small (this is a new condition).
For m=50, we recover all the 10 instances.
For m =128, we can recover 7 instances in the 10 instances.
However, we note that the attack depends on the choice of the parameters in the attack heavily (such as kappa,kappa2), so we believe that if we adjust the parameters well or employ some other tricks, we can recover more, since the idea behind it is clear. Since we think the current attack has shown the weakness of the scheme, we won't do more experiments with more parameters.

Best regards,
Yanbin

```
def subsetsumdecrypt4(A,v,vp,a,x,xp,u1,u2):
   bp = int(q/(sk_max+wPos+wNeg+e_max * (wPos+wNeg))) - 2*p_size
   z1 = randint(0, int(bp/2)-1)
   kappa=q*2^8
   kappa2=q*2^8
   kappa1=t*2^8
   kappa3=1
   L=block_matrix(ZZ, \
      [[1*kappa1, 0, -kappa*a.row(), -kappa2 * x, -kappa2 * xp,0,0], \
```