From: Alperin-Sheriff, Jacob (Fed)
Sent: Friday, January 12, 2018 12:51 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: LAC

LAC Team:

Your submission seems to incorrectly claim a connection to worst-case hardness problems for ideal lattices. The connection only applies to error distributions (statistically close enough to) the distributions for alpha described in Section 3 of LPR10 (https://eprint.iacr.org/2012/230.pdf), and that requires alpha * q >= omega(\sqrt{log n).

The {-1,0,1} distributions for error you used are not statistically close to any such distribution.

—Jacob Alperin-Sheriff

From: Jacob Alperin-Sheriff <jacobmas@gmail.com>
Sent: Sunday, January 14, 2018 11:38 AM
To: pqc-comments
Cc: pqc-forum
Subject: OFFICIAL COMMENT: LAC

LAC Team:

I believe there is an attack that reduces the security of your scheme by (close to) a factor of 2 in the exponent from the estimates given in your submission.

The key point is that for n=2^k, k>=2

x^{n}+1 = (x^(n/2) + 91*x^(n/4) - 1)(x^(n/2) - 91*x^(n/4) - 1) mod 251
(note that x^n+1 is reducible modulo any prime p when k>=2, and for some choices it reduces to even smaller polynomials that may be more devastating).

The aggressive choice of error distribution {-1,0,1} used in LAC appears to then yield an attack that essentially combines the observations/framework in Section 3.2 of "How (Not) to Instantiate Ring-LWE" by
Peikert https://eprint.iacr.org/2016/351.pdf with algorithms for solving SVP.

Let g=x^(n/2)+91*x^(n/4)-1, h=x^(n/2)-91*x^(n/4)-1, and let Id_g, Id_h be the associated ideals generated by g and 251 (respectively, h and 251)

Then for e a degree at most n polynomial with coefficients in (-1,0,1), the coefficients of e reduced modulo the ideal generated by g and 251

(i.e. e mod Id_g)

will all be of the form

+-{0,1,2} +- {0,91}

(and similarly for h)

Let (a,b=s*a+e (mod R_q) be the public key for LAC.

Now, let a_g = a mod Id_g, b_g = b mod Id_g
a_h = a mod Id_h, b_h = b mod Id_h

Then I believe (I haven't checked yet with easily feasible n (like n=64), will hopefully do next week) that the shortest vector x=(x_1,x_2,x_3,x_4)

solution to the (ring)-I-SIS problem

[a_g, 91*a_g, 1, 91]x = b_g

should be such that x_1+91*x_2 = s (mod Id_g), x_3+91*x_4=e (mod Id_g), and similarly for the case of h.

Assuming this is true, recovering s and e reduces to solving SVP for 2 instances of (n/2) dimensional lattices and then Chinese remainder theorem-ing to recover s and e.

I will try to implement this some time this week in Sage for an easily feasible n to see if I'm correct about the form of the shortest vector.

--
-Jacob Alperin-Sheriff

From: Jacob Alperin-Sheriff <jacobmas@gmail.com>
Sent: Sunday, January 14, 2018 2:40 PM
To: pqc-comments
Cc: pqc-forum
Subject: Re: OFFICIAL COMMENT: LAC

Hi everybody,

Vadim Lyubashevsky pointed out that I made a stupid mistake here about the dimensions of the resulting lattice so disregard everything above except (if it helps) the form of the error modulo the ideals.

On Jan 14, 2018 11:38 AM, "Jacob Alperin-Sheriff" <jacobmas@gmail.com> wrote:

> LAC Team:
>
> I believe there is an attack that reduces the security of your scheme by (close to) a factor of 2 in the exponent from the estimates given in your submission.
>
> The key point is that for n=2^k, k>=2
>
> x^{n}+1 = (x^(n/2) + 91*x^(n/4) - 1)(x^(n/2) - 91*x^(n/4) - 1) mod 251
> (note that x^n+1 is reducible modulo any prime p when k>=2, and for some choices it reduces to even smaller polynomials that may be more devastating).
>
> The aggressive choice of error distribution {-1,0,1} used in LAC appears to then yield an attack that essentially combines the observations/framework in Section 3.2 of "How (Not) to Instantiate Ring-LWE" by
> Peikert https://eprint.iacr.org/2016/351.pdf with algorithms for solving SVP.
>
> Let g=x^(n/2)+91*x^(n/4)-1, h=x^(n/2)-91*x^(n/4)-1, and let Id_g, Id_h be the associated ideals generated by g and 251 (respectively, h and 251)
>
> Then for e a degree at most n polynomial with coefficients in (-1,0,1), the coefficients of e reduced modulo the ideal generated by g and 251
>
> (i.e. e mod Id_g)
>
> will all be of the form
>
> +-{0,1,2} +- {0,91}
>
> (and similarly for h)
>
> Let (a,b=s*a+e (mod R_q) be the public key for LAC.
>
> Now, let a_g = a mod Id_g, b_g = b mod Id_g
> a_h = a mod Id_h, b_h = b mod Id_h
>
> Then I believe (I haven't checked yet with easily feasible n (like n=64), will hopefully do next week) that the shortest vector x=(x_1,x_2,x_3,x_4)

From: Alperin-Sheriff, Jacob (Fed)
Sent: Thursday, January 18, 2018 10:59 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: LAC

Hi all,

New idea on LAC.

Remember, the ring is x^512+1, q=251, error distribution is +- with probability 1/4, 0 with probability 1/2 in LAC.

I noticed (well I had the general idea and then used Sage to check all the possible multipliers) that if you multiply s and e by 11 (so the error distribution becomes \pm 11 each with probability 1/4, 0 with probability 1/2) then each of the error coefficients e_i will be at most +- 25 after reducing modulo g=x^(n/2)+91*x^(n/4)-1, specifically in the distribution

[0, 3, 8, 11, 14, 19, 22, 25, 226, 229, 232, 237, 240, 243, 248], and similarly for s
Moreover, the smaller values are more likely, for each individual coefficient they are distributed with probability {0: 0.111, 3: 0.111, 8: 0.074, 11: 0.074, 14: 0.074, 19: 0.037, 22: 0.037, 25: 0.037, 226: 0.037, 229: 0.037, 232: 0.037, 237: 0.074, 240: 0.074, 243: 0.074, 248: 0.111}

I haven't attempted to prove anything yet for the values of n that are of interest, but exhaustively computing for w=8 reveals that with probability at least .5, the Euclidean norm of the coefficients of e mod g [e being the error], will be at most 21,

For 5000 random samples in the w=512 case, the Euclidean norm
of the coefficient vector of ((11*e) mod g) was 207 or less with probability .5.

So, assuming I didn't make another stupid mistake again, the question is then whether this is going to be short enough that z=(11*s (mod g) ,11*e (mod g),-1)
will be the shortest solution of

Az=0

where A= [(a)mod g | 1 | (11*b) mod g] is a 256 x 513

By the above bound on the coefficient vector (11*e), we can get the (11*s (mod g), 11*e (mod g), -1) should be at most 292.75 with probability at least .25.

Lemma 5.2 in Micciancio-Regev's "Worst-case to Average-case Reductions based on Gaussian Measures"
https://pdfs.semanticscholar.org/9935/ac933507b63a2d6eac41c87aa4ab4835be52.pdf
gives that a solution **has** to exist of norm at least 356.9 in this kind of a SIS instance, but I'm not so up on how tight that bound is and if we can hope that (11*s (mod g), 11 * e(mod g), -1) will be the shortest vector.

I did try something in Sage for smaller values of n, but the results were inconclusive (BKZ didn't find the desired vector, but the desired vector was also of smaller norm than anything in the BKZ-reduced basis, suggesting that I'm doing something wrong). Code is included below; help would be appreciated.

```
#!/usr/bin/env sage

import sys
from sage.all import *

def sample_single_error():
    t=randint(0,3)
    if t==0:
        return -1
    elif t<3:
        return 0
    return 1

def sample_noise(S):
    n=S.degree()
    ret_lst=[sample_single_error() for i in range(n)]
    return S(ret_lst)

def my_gen_lattice2(n=4, q=11, seed=None,
                quotient=None, dual=False, ntl=False, lattice=False,t=5):
    """
    This is a modification of the code for the gen_lattice function from Sage

    Randomness can be set either with ``seed``, or by using
    :func:`sage.misc.randstate.set_random_seed`.

    INPUT:

    - ``type`` -- one of the following strings
        - ``'cyclotomic'`` -- Special case of ideal. Allows for
          efficient processing proposed by [LM2006]_.
    - ``n`` -- Determinant size, primal:`det(L) = q^n`, dual:`det(L) = q^{m-n}`.
      For ideal lattices this is also the degree of the quotient polynomial.
    - ``m`` -- Lattice dimension, `L \subseteq Z^m`.
    - ``q`` -- Coefficient size, `q-Z^m \subseteq L`.
    - ``t`` -- BKZ Block Size
    - ``seed`` -- Randomness seed.
    - ``quotient`` -- For the type ideal, this determines the quotient
      polynomial. Ignored for all other types.
    - ``dual`` -- Set this flag if you want a basis for `q-dual(L)`, for example
      for Regev's LWE bases [Reg2005]_.
    - ``ntl`` -- Set this flag if you want the lattice basis in NTL readable
      format.
    - ``lattice`` -- Set this flag if you want a
```

```
  :class:`FreeModule_submodule_with_basis_integer` object instead
  of an integer matrix representing the basis.

OUTPUT: ``B`` a unique size-reduced triangular (primal: lower_left,
  dual: lower_right) basis of row vectors for the lattice in question.

EXAMPLES:



Cyclotomic bases with n=2^k are SWIFFT bases::

    sage: sage.crypto.gen_lattice(type='cyclotomic', seed=42)
    [11  0  0  0  0  0  0  0]
    [ 0 11  0  0  0  0  0  0]
    [ 0  0 11  0  0  0  0  0]
    [ 0  0  0 11  0  0  0  0]
    [ 4 -2 -3 -3  1  0  0  0]
    [ 3  4 -2 -3  0  1  0  0]
    [ 3  3  4 -2  0  0  1  0]
    [ 2  3  3  4  0  0  0  1]

Dual modular bases are related to Regev's famous public-key
encryption [Reg2005]_::

    sage: sage.crypto.gen_lattice(type='modular', m=10, seed=42, dual=True)
    [ 0  0  0  0  0  0  0  0  0 11]
    [ 0  0  0  0  0  0  0  0 11  0]
    [ 0  0  0  0  0  0  0 11  0  0]
    [ 0  0  0  0  0  0 11  0  0  0]
    [ 0  0  0  0  0 11  0  0  0  0]
    [ 0  0  0  0 11  0  0  0  0  0]
    [ 0  0  0  1 -5 -2 -1  1 -3  5]
    [ 0  0  1  0 -3  4  1  4 -3 -2]
    [ 0  1  0  0 -4  5 -3  3  5  3]
    [ 1  0  0  0 -2 -1  4  2  5  4]


"""
from sage.rings.finite_rings.integer_mod_ring import IntegerModRing
from sage.matrix.constructor import identity_matrix, block_matrix
from sage.matrix.matrix_space import MatrixSpace
from sage.rings.integer_ring import IntegerRing
if seed is not None:
    from sage.misc.randstate import set_random_seed
    set_random_seed(seed)



ZZ = IntegerRing()
ZZ_q = IntegerModRing(q)

A = identity_matrix(ZZ_q, n/2)


from sage.arith.all import euler_phi
from sage.misc.functional import cyclotomic_polynomial

# we assume that n+1 <= min( euler_phi^{-1}(n) ) <= 2*n
found = False
for k in range(2*n,n,-1):
    if euler_phi(k) == n:
```

```
            found = True
            break
        if not found:
            raise ValueError("cyclotomic bases require that n "
                             "is an image of Euler's totient function")
    R = ZZ_q['x'].quotient(cyclotomic_polynomial(k, 'x'), 'x')
    g=x**(n/2)+91*x**(n/4)-1
    T=ZZ_q['x'].quotient(x**(n/2)+91*x**(n/4)-1)


    a_pol=R.random_element()

#    for i in range(m//n):
#        print("i={0}".format(i))
#        A = A.stack(R.random_element().matrix())
    s_pol=sample_noise(R)
    e_pol=sample_noise(R)

    s_pol2=T((11*s_pol).list())
    e_pol2=T((11*e_pol).list())

    Z_mat=s_pol2.matrix().augment(e_pol2.matrix())
    Z_mattop=Z_mat[0:1].augment(matrix(1,1,ZZ.one()*-1))


    b_pol=(a_pol*s_pol+e_pol)*11
    print("s_pol={0}\ne_pol={1}".format((s_pol*11).list(),(e_pol*11).list()))

    a_pol2 = T(a_pol.list())# % S(g.list())
    b_pol2 = T(b_pol.list())# % S(g.list())
#    print("a={0}\nb={1}".format(a_pol,b_pol))

    A=a_pol2.matrix().stack(A)
    A=A.stack(b_pol2.matrix()[0:1])

    print("{0}\n".format(A))
    # switch from representatives 0,...,(q-1) to (1-q)/2,....,(q-1)/2
    def minrepnegative(a):
        if abs(a-q) < abs(a): return (a-q)*-1
        else: return a*-1
    def minrep(a):
        if abs(a-q) < abs(a): return (a-q)
        else: return a
    A_neg = A[0:(n/2)].lift().apply_map(minrepnegative)
    b_neg= A[(n):(n+1)].lift().apply_map(minrepnegative)
    Z_fixed=Z_mattop.lift().apply_map(minrep)
    print("Z_fixed={0}\n||Z_fixed||={1}".format(Z_fixed,float(Z_fixed[0].norm())))
    print('Z_fixed*A={0}\n\n'.format(Z_fixed*A))

    #    print("{0}\n".format(A_neg))
    B=block_matrix([[ZZ(q), ZZ.zero(),ZZ.zero()],[ZZ.one(),A_neg,ZZ.zero()
],[ZZ.zero(),b_neg,ZZ.one()]],
                       subdivide=False)
    #print("B=\n{0}".format(B))
    #print("B*A=\n{0}\n\n".format(B*A))

    B_BKZ=B.BKZ(block_size=t,proof=True)

    print("B_BKZ[0]=\n{0}\n".format(B_BKZ[0]))

    for i in range(0,n+1):
        print('{0}: {1}'.format(i,1.*float(B_BKZ[i].norm())))
```

```
if ntl and lattice:
    raise ValueError("Cannot specify ntl=True and lattice=True ")
if ntl:
    return B._ntl_()
elif lattice:
    from sage.modules.free_module_integer import IntegerLattice
    return IntegerLattice(B)
else:
    return B
```

From:	sd lattice
To:	pqc-comments
Cc:	pqc-forum@list.nist.gov
Subject:	OFFICIAL COMMENT: LAC
Date:	Tuesday, January 23, 2018 10:32:58 AM
Attachments:	norms.pdf testpolynomialring.py

Dear Alperin-Sheriff, Jacob

Thank you very much for your comments on LAC.

1: About the connection to worst-case hardness problems: In the design of LAC we considered the requirement that alpha*q>=omega(\sqrt{log n}). The parameters of LAC do not satisfy this requirement directly. However, we notice that, according to the modulus reduction result proposed in [1], varying the dimension n and the modulus q individually while keeping nlogq fixed essentially preserves the hardness of LWE. We estimate the concrete hardness of the RLWE problem by using the method given in [2]. The result shows that the modulus reduction result also works for the hardness of RLWE. Concretely, for the parameter of LAC-128, where q=251,n=512, sigma=1/sqrt(2), the hardness is:

primal attack:
classical cost= 148
quantum cost= 135
dual attack:
classical cost= 147
quantum cost= 133

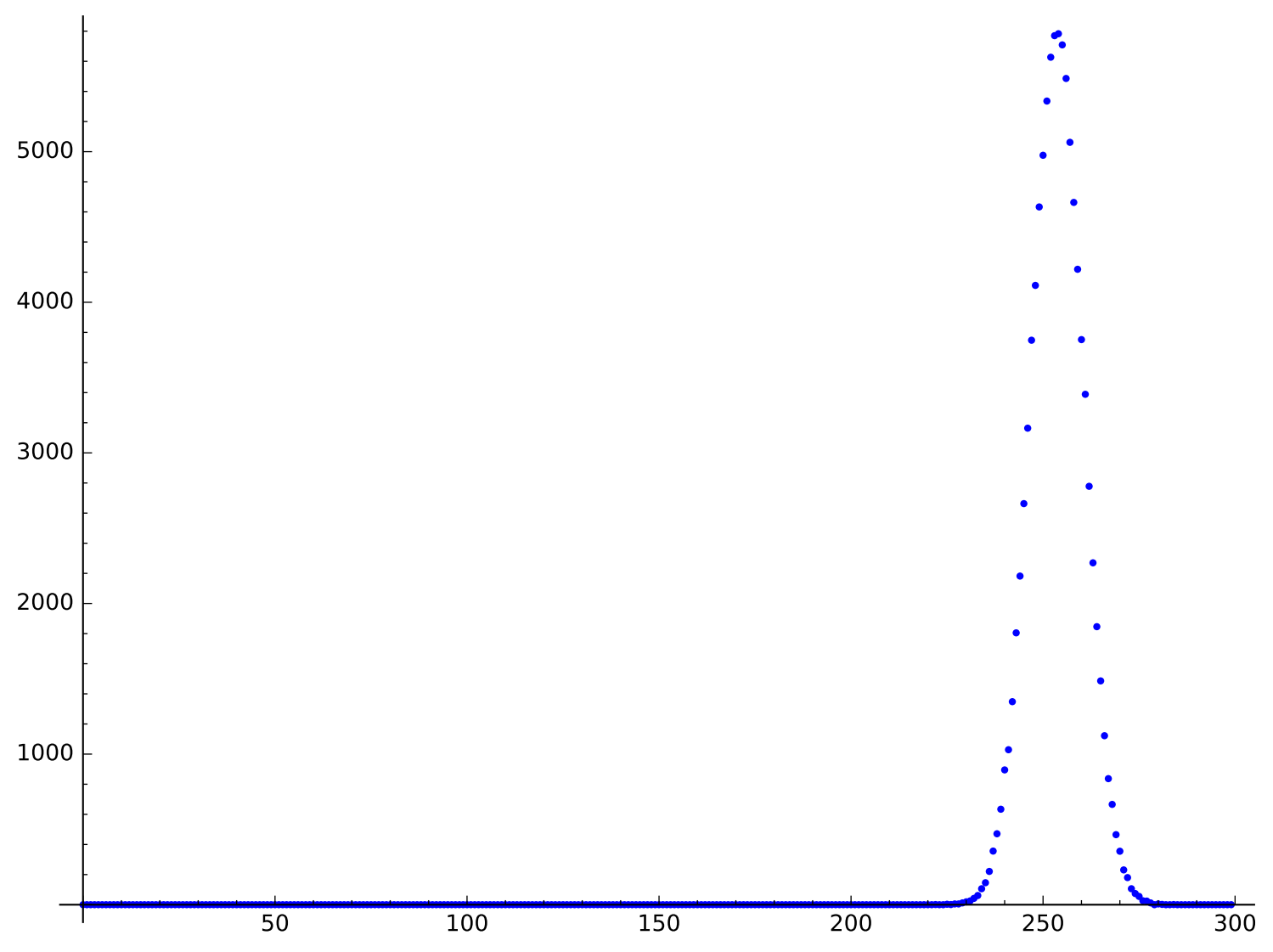
when we set q=251*251, n=256, sigma= 1/sqrt(2) * 251, the hardness is:
primal attack:
classical cost= 156
quantum cost= 141
dual attack:
classical cost= 154
quantum cost= 139
Note that, during the varying of n and q, we need to keep alpha=sigma*sqrt(2*pi)/q fixed, so we set sigma= 1/sqrt(2) * 251. In this case, alpha*q=1/sqrt(2) * 251>omega(\sqrt{log 256}).

2: About the modulo attack. This is really an interesting observation. We used Sage to check this idea, for 100000 random samples, our result shows that the length of z=[11*s mod g, 11*e mod g, -1] is an Gaussian distribution with mean 253.59 and standard deviation 6.9. The sage code and the figure of the distribution can be found in the attachment. According to Micciancio and Regev's "Lattice Based Cryptography",:
lambda_1(\Lambda_q^{\perp})=sqrt(513/(2*pi*e))*251^{256/513}=86.36. It is clear that, the length of vector z is much longer than lambda_1, and the BKZ algorithm cannot be used to find z.

Note that, the bound of Lemma 5.2 in Micciancio-Regev's "Worst-case to Average-case Reductions based on Gaussian Measures" is not tight enough. Concretely, the bound in this lemma is sqrt(513)*251^{256/513}=356.9, which is much larger than 86.36.

[1]Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, Damien Stehlé: **Classical hardness of learning with errors.** STOC 2013: 575-584
[2]Erdem Alkim, Léo Ducas, Thomas Pöppelmann, Peter Schwabe: **Post-quantum Key Exchange - A New Hope.** USENIX Security Symposium 2016: 327-343
[3] https://cims.nyu.edu/~regev/papers/pqc.pdf

Best Wishes.
LAC Team


5000
4000
3000
2000
1000
50
100
150
200
250
300

From: Alperin-Sheriff, Jacob (Fed)
To: sd lattice; pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: Re: OFFICIAL COMMENT: LAC
Date: Tuesday, January 23, 2018 10:38:34 AM

---

1. [1] doesn't work for ring-LWE.

2. You're right that my idea definitely doesn't work as written, I've since confirmed that the shortest vectors modulo the ideal are several times shorter than the induced (z,e,-1) vector (it may simply never be helpful in any ring, I'm still investigating that)

Thanks for the response.

From: sd lattice <luxianhui@outlook.com>
Date: Tuesday, January 23, 2018 at 10:33 AM
To: pqc-comments <pqc-comments@nist.gov>
Cc: "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>
Subject: OFFICIAL COMMENT: LAC

Dear Alperin-Sheriff, Jacob

Thank you very much for your comments on LAC.

1: About the connection to worst-case hardness problems: In the design of LAC we considered the requirement that alpha*q>=omega(\sqrt{log n}). The parameters of LAC do not satisfy this requirement directly. However, we notice that, according to the modulus reduction result proposed in [1], varying the dimension n and the modulus q individually while keeping nlogq fixed essentially preserves the hardness of LWE. We estimate the concrete hardness of the RLWE problem by using the method given in [2]. The result shows that the modulus reduction result also works for the hardness of RLWE. Concretely, for the parameter of LAC-128, where q=251,n=512, sigma=1/sqrt(2), the hardness is:

primal attack:
classical cost= 148
quantum cost= 135
dual attack:
classical cost= 147
quantum cost= 133

when we set q=251*251, n=256, sigma= 1/sqrt(2) * 251, the hardness is:
primal attack:
classical cost= 156
quantum cost= 141
dual attack:
classical cost= 154
quantum cost= 139
Note that, during the varying of n and q, we need to keep alpha=sigma*sqrt(2*pi)/q fixed, so we set sigma= 1/sqrt(2) * 251. In this case, alpha*q=1/sqrt(2) * 251>omega(\sqrt{log 256}).

2: About the modulo attack. This is really an interesting observation. We used Sage to check this idea, for 100000 random samples, our result shows that the length of z=[11*s mod g, 11*e mod g, -1] is an Gaussian distribution with mean 253.59 and standard deviation 6.9. The sage code and the figure of the distribution can be found in the attachment. According to Micciancio and Regev's "Lattice Based Cryptography",:
lambda_1(\Lambda_q^{\perp})=sqrt(513/(2*pi*e))*251^{256/513}=86.36. It is clear that, the length

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Sent: Tuesday, January 23, 2018 11:03 AM
To: Alperin-Sheriff, Jacob (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov; sd lattice
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Tue, Jan 23, 2018 at 10:38 AM Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:

> 1. [1] doesn't work for ring-LWE.

To be clear, modulus switching does work for both LWE and Ring-LWE (among other problems), as claimed below. But the results from [1] do not say anything about (Ring-)LWE with *binary errors*, which is apparently what LAC uses. (Gaussians are the only error distributions considered in [1].) It does not appear that LAC is supported asymptotically by any known worst-case hardness theorem.

Sincerely yours in cryptography, Chris

From: sd lattice <luxianhui@outlook.com>
Date: Tuesday, January 23, 2018 at 10:33 AM
To: pqc-comments <pqc-comments@nist.gov>
Cc: "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>
Subject: OFFICIAL COMMENT: LAC

Dear Alperin-Sheriff, Jacob

Thank you very much for your comments on LAC.

1: About the connection to worst-case hardness problems: In the design of LAC we considered the requirement that alpha*q>=omega(\sqrt{log n}). The parameters of LAC do not satisfy this requirement directly. However, we notice that, according to the modulus reduction result proposed in [1], varying the dimension n and the modulus q individually while keeping nlogq fixed essentially preserves the hardness of LWE. We estimate the concrete hardness of the RLWE problem by using the method given in [2]. The result shows that the modulus reduction result also works for the hardness of RLWE. Concretely, for the parameter of LAC-128, where q=251,n=512, sigma=1/sqrt(2), the hardness is:

primal attack:

classical cost= 148

From: Alperin-Sheriff, Jacob (Fed)
Sent: Tuesday, January 23, 2018 11:06 AM
To: Christopher J Peikert
Cc: pqc-comments; pqc-forum@list.nist.gov; sd lattice
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

LAC uses +-1 with probability 1/4 each, 0 with probability 1/2 so technically not binary. You're right that a better answer would have been things like "Hardness of SIS and LWE with Small Parameters
" https://eprint.iacr.org/2013/069.pdf don't apply to ring-LWE (although it could easily be adapted for the LAC error distribution for the general LWE case).

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Date: Tuesday, January 23, 2018 at 11:02 AM
To: "Alperin-Sheriff, Jacob (Fed)" <jacob.alperin-sheriff@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>, "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>, sd lattice <luxianhui@outlook.com>
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Tue, Jan 23, 2018 at 10:38 AM Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:

> 1. [1] doesn't work for ring-LWE.

To be clear, modulus switching does work for both LWE and Ring-LWE (among other problems), as claimed below. But the results from [1] do not say anything about (Ring-)LWE with *binary errors*, which is apparently what LAC uses. (Gaussians are the only error distributions considered in [1].) It does not appear that LAC is supported asymptotically by any known worst-case hardness theorem.

Sincerely yours in cryptography, Chris

From: sd lattice <luxianhui@outlook.com>
Date: Tuesday, January 23, 2018 at 10:33 AM
To: pqc-comments <pqc-comments@nist.gov>
Cc: "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>
Subject: OFFICIAL COMMENT: LAC

Dear Alperin-Sheriff, Jacob

> Thank you very much for your comments on LAC.

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Sent: Tuesday, January 23, 2018 11:16 AM
To: Alperin-Sheriff, Jacob (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov; sd lattice
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Tue, Jan 23, 2018 at 11:05 AM Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:

> LAC uses +-1 with probability 1/4 each, 0 with probability 1/2 so technically not binary. You're right that a better answer would have been things like "Hardness of SIS and LWE with Small Parameters
>
> " https://eprint.iacr.org/2013/069.pdf don't apply to ring-LWE (although it could easily be adapted for the LAC error distribution for the general LWE case).

Yes, that paper gets closer to the mark for plain LWE with binary (or ternary) errors. But to prevent any confusion: it would not support the plain-LWE analogue of LAC either, because the number of samples is limited to smaller than what LAC reveals to the attacker.

Sincerely yours in cryptography, Chris

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Date: Tuesday, January 23, 2018 at 11:02 AM
To: "Alperin-Sheriff, Jacob (Fed)" <jacob.alperin-sheriff@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>, "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>, sd lattice <luxianhui@outlook.com>
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Tue, Jan 23, 2018 at 10:38 AM Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:

> 1. [1] doesn't work for ring-LWE.

To be clear, modulus switching does work for both LWE and Ring-LWE (among other problems), as claimed below. But the results from [1] do not say anything about (Ring-)LWE with *binary errors*, which is apparently what LAC uses. (Gaussians are the only error distributions considered in [1].) It does not appear that LAC is supported asymptotically by any known worst-case hardness theorem.

Sincerely yours in cryptography, Chris

From: Alperin-Sheriff, Jacob (Fed)
Sent: Thursday, February 15, 2018 2:23 PM
To: Christopher J Peikert
Cc: pqc-comments; pqc-forum@list.nist.gov; sd lattice
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

Coming back to this point "Yes, [Hardness of SIS and LWE with Small Parameters] gets closer to the mark for plain LWE with binary (or ternary) errors. But to prevent any confusion: it would not support the plain-LWE analogue of LAC either, because the number of samples is limited to smaller than what LAC reveals to the attacker."

If I'm understanding everything properly, the paper doesn't say anything at all about the plain LWE analogue of LAC and more generally, the public key in any ring-LWE (or plain LWE) scheme using the Lindner-Peikert formulation of primal Regev, because that paper requires m >= n + omega(log n) in order to use the Micciancio-Mol SIS equivalence, and we have m=n for LAC. Is this correct?

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Date: Tuesday, January 23, 2018 at 11:16 AM
To: "Alperin-Sheriff, Jacob (Fed)" <jacob.alperin-sheriff@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>, "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>, sd lattice <luxianhui@outlook.com>
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Tue, Jan 23, 2018 at 11:05 AM Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:

> LAC uses +-1 with probability 1/4 each, 0 with probability 1/2 so technically not binary. You're right that a better answer would have been things like "Hardness of SIS and LWE with Small Parameters
>
> " https://eprint.iacr.org/2013/069.pdf don't apply to ring-LWE (although it could easily be adapted for the LAC error distribution for the general LWE case).

Yes, that paper gets closer to the mark for plain LWE with binary (or ternary) errors. But to prevent any confusion: it would not support the plain-LWE analogue of LAC either, because the number of samples is limited to smaller than what LAC reveals to the attacker.

Sincerely yours in cryptography, Chris

From: Christopher J Peikert <cpeikert@alum.mit.edu>
Date: Tuesday, January 23, 2018 at 11:02 AM
To: "Alperin-Sheriff, Jacob (Fed)" <jacob.alperin-sheriff@nist.gov>
Cc: pqc-comments <pqc-comments@nist.gov>, "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>, sd lattice <luxianhui@outlook.com>
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

From: Mike Hamburg <mike@shiftleft.org>
Sent: Thursday, February 15, 2018 8:39 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: LAC

Hello LAC authors,

I noticed that the LAC reference and optimized code submissions take variable time, at least for the BCH decoding step.

How difficult is it to fix this to run in constant time? Do you plan to implement a constant-time decoding algorithm so that we can test its performance?

Thanks,
— Mike Hamburg

From: Alperin-Sheriff, Jacob (Fed)
Sent: Friday, February 16, 2018 8:54 AM
To: Mike Hamburg; pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: Re: OFFICIAL COMMENT: LAC

Mike,

I had also noticed that the BCH decoding algorithms are written in a variable time manner, but I ended up forgetting to/putting off mention it here after I did some testing of the times/cycles taken by the BCH decoding step and it didn't seem to vary depending on the number of errors it had to decode.

This doesn't mean I or NIST consider the code or the algorithm "okay" (meaning of no potential concern), just that I wasn't able to find a way that it leaked any useful information and so it slipped my mind.

Did you find something different?

On 2/15/18, 8:39 PM, "Mike Hamburg" <mike@shiftleft.org> wrote:

> Hello LAC authors,
>
> I noticed that the LAC reference and optimized code submissions take variable time, at least for the BCH decoding step.
>
> How difficult is it to fix this to run in constant time? Do you plan to implement a constant-time decoding algorithm so that we can test its performance?
>
> Thanks,
> — Mike Hamburg

From: Alperin-Sheriff, Jacob (Fed)
Sent: Friday, February 16, 2018 10:07 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: LAC

LAC authors:

I believe LAC256 falls significantly short of 256 bits of security under the CCA attack model (that allows up to 2^64 chosen ciphertext queries [see pg 15 of the CFP https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf]).

A simple calculation of binom(2048,1024+310)/(2**2048) gives that (modeling the hash function as a random oracle) for a message chosen uniformly at random, the vector (r,e_1) in LAC.CPA.Enc (which in the LAC.CCA.Enc gets its "randomness" from the message) will have Hamming weight of at least 1024+310 with probability at least 2^{-143.3}, so with high probability, 2^{-210} or so hash function evaluations should yield at least 2^{64} messages giving such high Hamming weights for (r,e_1) in LAC.CPA.Enc.

After doing this 2^{210}*cost(hash) precomputation (which presumably costs signficantly less than 256 bits), we should be able to recover almost any key with 2^{64} chosen ciphertext queries.

For messages that yield Hamming weights of (r,e_1) of 1024+310, testing gives that each bit fails to be correct with probability 2^{-6.0496}, so a simple Python script that enough bits (56 or more) to cause failure on the entire message will occur with probability ~ 2^{-50.51}. As a result, over 2^{64} message queries we should expect with high probability to get far more than the necessary number of failures to recover s based on a rough application of the analysis in https://eprint.iacr.org/2016/085.pdf

—Jacob Alperin-Sheriff

From: Martin Tomlinson <mt@post-quantum.com>
Sent: Friday, February 16, 2018 10:40 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: LAC
Attachments: signature.asc

Dear LAC authors,

Although you decided to use BCH codes as the error correcting code in LAC, a better choice is to use binary Goppa codes since the maximum code length of these codes is 2^m and not 2^m-1. There is an additional information bit that can be corrected and no messy padding. A BCH decoder can be used straightforwardly to implement the error correction decoder for Goppa codes as first pointed out by Charles Retter in his 1975 paper: "Decoding Goppa codes with a BCH decoder.", IEEE Transactions on Information Theory, 21(1):112–112.

One significant advantage is that you can benefit from the extensive research in the last decade that has been applied to the McEliece system to avoid side channel information leakage in the syndrome calculation, Berlekamp-Massey and root finding algorithms. For example the reference implementations of the NTS-KEM submission or the Classic McEliece submission could be used in LAC with only minor modifications to integrate within the procedures the variability of the parameter t.

Best regards

Martin

From: cpeikert@gmail.com on behalf of Christopher J Peikert <cpeikert@alum.mit.edu>
Sent: Friday, February 16, 2018 10:43 PM
To: Alperin-Sheriff, Jacob (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov; sd lattice
Subject: Re: [pqc-forum] Re: OFFICIAL COMMENT: LAC

On Thu, Feb 15, 2018 at 2:22 PM, Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:
> Coming back to this point "Yes, [Hardness of SIS and LWE with Small
> Parameters] gets closer to the mark for plain LWE with binary (or
> ternary) errors. But to prevent any confusion: it would not support
> the plain-LWE analogue of LAC either, because the number of samples is
> limited to smaller than what LAC reveals to the attacker."
>
> If I'm understanding everything properly, the paper doesn't say
> anything at all about the plain LWE analogue of LAC and more
> generally, the public key in any ring-LWE (or plain LWE) scheme using
> the Lindner-Peikert formulation of primal Regev, because that paper
> requires m >= n + omega(log n) in order to use the Micciancio-Mol SIS
> equivalence, and we have m=n for LAC. Is this correct?

Well, those m parameters don't appear to mean the same thing.

I think the main issue is this: for plain LWE with binary (or ternary) errors, [Hardness ... with Small Parameters] proves hardness when the number of LWE samples with *uniform secret over Zq* is

m < n(1 + 1/log n). (See discussion following Thm 4.6.)

However, Lindner-Peikert encryption uses *normal form* LWE, where the entries of the secret are chosen from the error distribution (not uniformly).

Transforming from uniform-secret to normal-form LWE costs at least n samples, which for the above m leaves < n/log n normal-form samples.
But the Lindner-Peikert system reveals at least n normal-form samples.
(So does the original Regev cryptosystem.)

Sincerely yours in cryptography,
Chris