From: Leo Ducas <leo.ducas1@gmail.com>
Sent: Tuesday, December 26, 2017 6:12 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: DRS

Dear Authors,

I am trying to understand and reproduce your reduction algorithm for signatures. I've had trouble with the document, where I think there are a few typos (marked by comments in the following python code).

My implementation ends up getting stuck in an infinite loop. While most signatures are produced within less than 10000 iterations of the reduction loop, one sample in (say) 30 does not stop before a million iteration.

I suspect that my implementation is still not conform to yours. In particular I would be grateful if you could specify more precisely the desired behavior for the rounded division w[i]/D (toward 0 ? toward -oo ?). Please also let me know if the typo I suspect are indeed typoes and if I addressed them properly.

Best regards
-- Leo Ducas

==== tentative re-implementation in python

```
import random
from numpy import zeros, int64
import sys

# params
n = 912
D = n
b = 28  // typos ? Sometime referred as B in the document
Nb = 16
N1 = 432
delta = 28

Samples = 100000


def RandomSign():
    return random.randint(0, 1)*2 -1

def KeyGen():
    L = Nb*[b] + N1 *[1] + (n - 1 - Nb - N1) * [0]
    random.shuffle(L)
    t = [D] + L
    S = zeros((n, n), dtype=int64)

    for i in xrange(n):    # Typo ? i should start at 0 ?
        S[i, i] = t[0]
```

```
        for j in xrange(1, n):
            S[i, (i+j) % n] = t[j] * RandomSign() # typo ? t is one-dimensional
    return S


# For efficiency, I'm using directly vector operation rather than loops.
# I rewrote this assuming that the lign w_l <- w_l + q M_{i,j} is typoed,
# and that the correct instruction is w_l <- w_l + q M_{i,l}


# Unlike C, Python does not round toward 0 but toward -oo, adding an option
# to force this behavior

ROUND_TOWARD_ZERO = True

def Sign(S, v):
    w = v
    i = 0
    it = 0
    while True:
        it += 1
        if it % 1000 == 0:
            print it,
            sys.stdout.flush()

        if ROUND_TOWARD_ZERO and w[i] < 0:
            q = - ((-w[i])/D)
        else:
            q = w[i]/D

        w -= q * S[i]

        if (max([abs(x) for x in w]) < D):
            return w

        i = (i + 1) % n


def SignRandom(S):
    v = zeros(n, dtype=int64)
    f = 2**delta - 1
    for i in xrange(n):
        # Simulate random hash of fresh message
        # Always chosen positive, as the version with sign is commented out
        # in the reference implementation
        v[i] = random.randint(0, f)
    return Sign(S, v)

S = KeyGen()

for a in range(100):
    print
    print "Sample ", a
    SignRandom(S)
```