Dear all,

We would like to thank Jingnan He and Xianhui Lu for pointing out a bug in our code.

In the discrete Gaussian sampling algorithm ntru-pke-1024/DGS.c

```c
void DGS ( int64_t *v, /* output vector */
          const uint16_t dim, /* input dimension */
          const uint8_t *stdev) /* input standard deviation */,
```

where the stdev is 724 and therefore requires more than 8 bits to store.

We have fixed this bug. The updated code will be available at

https://www.onboardsecurity.com/nist-post-quantum-crypto-submission

Best regards,

The NTRU team
Hello, I'm very happy to contact you about your implementation in NIST. I have a remark in your implementation: perhaps with your compiler version it works goodly but in mine I find problem:
in file dgs.h:

```c
void DGS (int64_t *v, /* output vector */
          const uint16_t dim, /* input dimension */
          const uint64_t stdev) /* input standard deviation */
{
}
```
and

```c
void DDGS (int64_t *v,
           const uint16_t dim,
           const uint64_t stdev,
           unsigned char *seed,
           size_t seed_len)
```

but in file poly.h:

```c
void DGS (int64_t *v,
          const uint16_t N,
          const uint16_t stdev);
```

`/* deterministic DGS */`

```c
void DDGS (int64_t *v,
           const uint16_t dim,
           const uint64_t stdev,
           unsigned char *seed,
           uint16_t seed_len);
```

You must do the same types of parameters functions in both files: dgs.h; poly.h it work when I changed in both files:

```c
oid DGS (int64_t *v, /* output vector */
         const uint16_t dim, /* input dimension */
         const uint64_t stdev) /* input standard deviation */
{
}
```
and

```c
void DDGS (int64_t *v,
           const uint16_t dim,
           const uint64_t stdev,
           unsigned char *seed,
           uint16_t seed_len)
```

Best regards
Hi Researchers;

About document title: NTRUencrypt A Lattice Based Cryptography Algorithm: Page 5, algorithm 3 NTRU-pke Decrypt. I want to ask you if in line 2, it is \( t = c - m \), or \( t = c - m' \)? Because \( m \) will be computed in line 6.

The same for algorithm 8.

Best regards.
Hi,
I have another remark:
- in doc :algorithm 1 you compute the public key as: h=g/(pf+1)
- but in EESS section 9.1.1 you compute the public key as: h=p*g/(pf+1) or like you write: h=f^(-1) * g*p where f=1+pF
Why you add p factor ?

Best regards

Le jeudi 5 avril 2018, EL HASSANE LAAJI <e.laaji@ump.ac.ma> a écrit :
Hi Researchers;

About document title: NTRUencrypt  A Lattice Based Cryptography Algorithm: Page 5, algorithm 3 NTRU-pke Decrypt. I want to ask you if in line 2, it is \( t = c - m \), or \( t = c - m' \)? Because \( m \) will be computed in line 6.

The same for algorithm 8.

Best regards.
Hi El,

> About document title: NTRUencrypt A Lattice Based Cryptography Algorithm: Page 5, algorithm 3 NTRU-pke Decrypt. I want to ask you if in line 2, it is \( t = c - m \), or \( t = c - m' \)? Because \( m \) will be computed in line 6.

> The same for algorithm 8.

Thanks for pointing this out. Yes it is a typo - should be \( t = c - m' \).

- in doc :algorithm 1 you compute the public key as: \( h = g/(pf+1) \)
- but in EESS section 9.1.1 you compute the public key as: \( h = p^* g/(pf+1) \) or like you write: \( h = f^{-1} g^* p \) where \( f = 1+pF \)

It's an inconsistency of the description in the report and the EESS1 spec.
In the end we will be using \( pg/(pf+1) \) in the encryption scheme - it doesn't really matter if we encode \( g/(pf+1) \) or \( pg/(pf+1) \) in the public key.
In the report we slightly changed it so that both NTRUencrypt and pqNTRUSign uses a same key gen function
- in pqNTRUSign we no longer have the \( p \) factor for the public key.

On that note, I just noticed an another typo: algorithm 2, line 5, it should be: \( t = p^* r * h \).

Regards,
Zhenfei

On Mon, Apr 9, 2018 at 7:31 AM, EL HASSANE LAAJI <e.laaji@ump.ac.ma> wrote:

Hi,
I have another remark:
- in doc :algorithm 1 you compute the public key as: \( h = g/(pf+1) \)
- but in EESS section 9.1.1 you compute the public key as: \( h = p^* g/(pf+1) \) or like you write: \( h = f^{-1} g^* p \) where \( f = 1+pF \)
Why you add \( p \) factor ?

Best regards

Le jeudi 5 avril 2018, EL HASSANE LAAJI <e.laaji@ump.ac.ma> a écrit :

Hi Researchers;

I want to ask you if in line 2, it is \( t = c - m \), or \( t = c - m' \)? Because \( m \) will be computed in line 6.

The same for algorithm 8.

Best regards.
Hi all,

We would like to report that we have fixed the bugs reported in the email. Last week, we were also informed by Ray and Dustin about a bug in key generations. In our code, the fixed weight sparse polynomial generation function within key gen does not always return a fixed weight polynomial with balanced number of +/- 1s. We have also fixed this bug in this revision. For the latest version of our code please see: https://github.com/NTRUOpenSourceProject/ntru-crypto/tree/master/NIST

Regards,
Zhenfei

On Sat, May 19, 2018 at 11:15 AM, Zhenfei Zhang <zzhang@onboardsecurity.com> wrote:
Hi Markku,

Thanks again for the reminder. We do have a patch which was supposed to be available at our website. I make sure they are available next week.

Cheers,
Zhenfei

On Sat, May 19, 2018 at 10:46 AM, Markku-Juhani O. Saarinen <mjos.crypto@gmail.com> wrote:
Hi,

The reference implementation of NTRUEncrypt KEM-1024 does not work -- the encryption and decryption parts do not generate the same shared secret.

I notified the design team more about this more than a month ago, and they rapidly acknowledged the problem, but I haven't seen a bugfix yet.

I don't know what precisely is causing this but there is at least one apparent bug in file NTRUEncrypt/Reference_Implementation/ntru-kem-1024/NTRUEncrypt.c, function mask_m():

```c
274:    /* extract the last bit of rh */
275:    for (i=0;i<LENGTH_OF_HASH*2;i++)
276:    {
277:        seed[i] = (rh[i*8] & 1);
278:        for (j=1;j<8;j++)
279:        {
280:            seed[i] <<= 1;
281:            seed[i] += (rh[i*8+j] & 1);
282:        }
283:    }
```
Note the semicolon at the end of line 278 -- this is not a loop, it just sets j=8 and executes the following bit on lines 280-281 once.

The KAT files are probably useless as the error appears to be on the encrypt side.

The smaller variants of NTRUEncrypt (KEM-443 and KEM-743) do successfully encrypt/decrypt. The corresponding function of these variants looks little different so it is not obvious to me how to correct the error.

Cheers,
- markku

--
You received this message because you are subscribed to the Google Groups "pqc-forum" group.
To unsubscribe from this group and stop receiving emails from it, send an email to pqc-forum+unsubscribe@list.nist.gov.
Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.

--
You received this message because you are subscribed to the Google Groups "pqc-forum" group.
To unsubscribe from this group and stop receiving emails from it, send an email to pqc-forum+unsubscribe@list.nist.gov.
Visit this group at https://groups.google.com/a/list.nist.gov/group/pqc-forum/.