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
    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 uint64_t stdev);
```

`/* deterministic DGS */
void DDGS (int64_t *v,
    const uint16_t N,
    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
void DGS (int64_t *v, /* output vector */
    const uint16_t dim, /* input dimension */
    const uint64_t stdev) /* input standard deviation */
{
    and
    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: \( h = f^\langle -1 \rangle * g^* p \) 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 LAALI <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: \( h = f^\langle -1 \rangle * g^* p \) where \( f = 1+pF \)
Why you add \( p \) factor?

Best regards

Le jeudi 5 avril 2018, EL HASSANE LAALI <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.