Hello NTRU team
Can you say me, why you didn't keep the NTRUencrypt-1024 release, is it because of speed performance or security performance.
Best regards.

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/.
Dear El Hassane Laaji,

* EL HASSANE LAAJI <e.laaji@ump.ac.ma> [2019-05-26 00:34:09 +0000]:
> Can you say me, why you didn't keep the NTRUencrypt-1024 release, is
> it because of speed performance or security performance.

Thanks for your question. To clarify for others, the "NTRUencrypt-1024" parameter set was proposed in the first round NTRUEncrypt submission for use with the ss-ntru-pke and ss-ntru-kem schemes. I'll split your question into two parts:
- Why didn't we recommend ss-ntru?
- Why didn't we recommend an NTRU variant that uses $Z[x]/(x^{1024} + 1)$?

Regarding ss-ntru:

At a fixed security level, NTRU and LWE schemes have a trade-off triangle between
1. the correctness of the decryption procedure,
2. the width of the coefficient distributions,
3. the compactness of public keys and ciphertexts.

The second round NTRUEncrypt team wanted a compact scheme with a correct decryption procedure. The coefficient distribution used in ss-ntru is not compatible with that goal.

Regarding $Z[x]/(x^{1024} + 1)$:

It's not clear to us that there's a real need for an NTRU parameter set with such a large $n$. The largest $n$ that we recommend is 821.

Best,
John (on behalf of the NTRUEncrypt team)
Hello NTRU team;
I have a remark:
In "NTRU Algorithm specification..." document, page 17 DPE_Public_Key algorithm, in operation subsection you compute:

V0=Sq(G.f); // you compute V0 in Sq but in implementation you compute G.f in Rq.
the same for V1=Sq_inverse(V0);// you compute V1 in Sq but in implementation you compute it in Rq.
also hq=Rq(V1.f.f); you compute hq in Rq but in implementation you compute it Sq.
see function:

```c
void owcpa_keypair(unsigned char *pk,
    unsigned char *sk,
    const unsigned char seed[NTRU_SAMPLE_FG_BYTES])
{
    ....
    poly_Rq_mul(Gf, G, f);//v0=f.G should be computed in Sq ?
    poly_Rq_inv(invGf, Gf);//v1=invGf should be computed in Sq ?
    poly_Rq_mul(tmp, invGf, f);
    poly_Sq_mul(invh, tmp, f);//hq=invh should be also computed in Rq ?
    ....
}
```

Can you say me which are the right operations?

Best regards
H.LAAJI
Ph.D student
Mohammed First University Morocco.
Hello El Hassane,

> poly_Rq_inv(invGf, Gf);//v1=invGf should be computed in Sq ?

> Can you say me which are the right operations?

The "poly_Rq" in the name just refers to the fact that Gf and invGf are encoded as elements of Rq. As long as Gf is not zero, our implementation of poly_Rq_inv will return invGf that satisfies invGf * Gf = 1 mod (q, Phi_n).

If Gf happens to be invertible in Rq, then our implementation of poly_Rq_inv will return invGf that satisfies invGf * Gf = 1 mod (q, Phi_1Phi_n).

This condition is not necessary, which is why the specification only asks for an inverse in Sq.

Cheers,

John