From:	Dang, Thinh
To:	Lichtinger, Jacob T. (Fed)
Cc:	Dang, Thinh H. (Fed); Thinh Dang - thinh@gwu.edu; Apon, Daniel C. (Fed)
Subject:	Re: Kyber Script
Date:	Friday, October 22, 2021 3:43:22 PM

> ```
> chis_h = {}
> #sum = 0
> for p in hammer_pattern:
>    chis_h[p] = hammer_law(chis, p)
>    #sum += hammer_pattern[p]
> ```

This seems a nicer way to achieve the same effect.

> hammer_bits = len(hammer_pattern)

We want the number of hammered coeffs, not necessarily bits. One coefficient can be hammered in multiple bits. So n_hammer_coefs =   sum(hammer_pattern.values()).

> Also, is there a reason we are using reduce instead of iter_law_convolution for the hammered bits?

iter_law_convolution computes the sum of i.i.d. B3 is a list of independent but not identical distributions.

> For the adversarial DFR, I think you can construct chie_a, chie_na, chiRe_a, chiRe_na like before. Then
>
> ```
> B3 = {}
> if honest:
>    for p in hammer_pattern:
>       B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe), hammer_pattern[p])
> else:
>    for p in hammer_pattern:
>       if p >= 0:
>          B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe_a), hammer_pattern[p])
>       else:
>          B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe_na), hammer_pattern[p])
> ```

I wanted to modularize the computation of adversarial DFR a bit by specifying as input how the adversary wants to sample e_1 (chie_a in the script) depending on how a coef is hammered. This input is called adv_filter and =threshold_law by default, which selects +/- 2 or higher.

> Oh instead of having B3 structured like that, use the convolution law to combine them in the

> for loop:

You're correct.

On Fri, Oct 22, 2021 at 12:06 PM Lichtinger, Jacob T. (Fed) <jacob.lichtinger@nist.gov> wrote:

> Oh instead of having B3 structured like that, use the convolution law to combine them in the for loop:
>
> ```
> B3 = {0: 1.}
> if honest:
>    for p in hammer_pattern:
>       D = iter_law_convolution(law_product(chis_h[p], chiRe), hammer_pattern[p])
>       B3 = law_convolution(B3, D)
> else:
>    for p in hammer_pattern:
>       if p >= 0:
>          D = iter_law_convolution(law_product(chis_h[p], chiRe_a), hammer_pattern[p])
>       else:
>          D = iter_law_convolution(law_product(chis_h[p], chiRe_na), hammer_pattern[p])
>       B3 = law_convolution(B3, D)
> ```
>
> ---
>
> **From:** Lichtinger, Jacob T. (Fed) <jacob.lichtinger@nist.gov>
> **Sent:** Friday, October 22, 2021 11:56 AM
> **To:** Dang, Thinh H. (Fed) <thinh.dang@nist.gov>; Thinh Dang - thinh@gwu.edu <thinh@gwu.edu>; Apon, Daniel C. (Fed) <daniel.apon@nist.gov>
> **Subject:** Re: Kyber Script
>
> For the adversarial DFR, I think you can construct chie_a, chie_na, chiRe_a, chiRe_na like before. Then
>
> ```
> B3 = {}
> if honest:
>    for p in hammer_pattern:
>       B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe), hammer_pattern[p])
> else:
>    for p in hammer_pattern:
>       if p >= 0:
>          B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe_a), hammer_pattern[p])
>       else:
>          B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe_na), hammer_pattern[p])
> ```

I think that should work?

From: Lichtinger, Jacob T. (Fed)
Sent: Friday, October 22, 2021 11:40 AM
To: Dang, Thinh H. (Fed) <thinh.dang@nist.gov>; Thinh Dang - thinh@gwu.edu <thinh@gwu.edu>; Apon, Daniel C. (Fed) <daniel.apon@nist.gov>
Subject: Kyber Script

Hi,

I think I see where a problem is happening in the script.

Line 28: `chis_h = [hammer_law(chis, p) for p in hammer_pattern]`

Once chis_h is built, it is unclear which p corresponds to which distribution. I am assuming that hammer_pattern is a dictionary with keys p (tampered bit value like 32 or 64) and the value at p is the number of bits flipped for that p. I would suggest:

```
chis_h = {}
#sum = 0
for p in hammer_pattern:
   chis_h[p] = hammer_law(chis, p)
   #sum += hammer_pattern[p]
hammer_bits = len(hammer_pattern)
```

Also, is there a reason we are using reduce instead of iter_law_convolution for the hammered bits? If not, I would change the B3 calculations to

```
B3 = {}
if honest:
   for p in hammer_pattern:
      B3[p] = iter_law_convolution(law_product(chis_h[p], chiRe), hammer_pattern[p])
```

I need to think about the else case more.

Thoughts?