From:	建方 牛 <niux_dannyniu@icloud.com>
Sent:	Monday, June 10, 2019 4:06 AM
To:	pqc-comments
Cc:	pqc-forum@list.nist.gov
Subject:	ROUND 2 OFFICIAL COMMENT: CRYSTALS-DILITHIUM

While going through the codes for Dilithium, I found a non-portable assumption made about the C language.

Namely, it’s assumed when right-shifting signed integer types from the <stdint.h> header, higher-order bits will be filled with the sign of the original operand. AFAIK, nowhere in ISO 9899 was it mentioned such behavior is mandated on implementations.

Would the Crystals team clarify that what had been intended in the reference implementation?

From: 'Thomas Pornin' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Monday, June 17, 2019 3:16 PM
To: pqc-forum
Subject: [pqc-forum] Re: Official comment for Dilithium not received by the forum.

(I am not part of the Dilithium team and I speak only for myself here.)

In ISO C, right-shifting a signed negative integer yields an "implementation-defined result". Crucially, this is not an "undefined behavior"; you will get a value and nothing else will break. It is true that the C standard does not mandate a specific result, but it does mandate the "implementation" (i.e. the C compiler) to do something and also to *document* that something.

In practice, most if not all compilers do an arithmetic shift, because that's what is most useful in such a case: the underlying CPU knows how to do an arithmetic shift and there is no other C operator that would produce it. This is not a risky bet. I would personally be OK with an implementation documenting that it relies on arithmetic shift for signed integers. In an ideal world, there would be a mechanism to detect proper support at compilation time and abort if right-shifting a signed negative value is not (always) an arithmetic shift; but in an ideal world, we would not be writing C code either.

If you want to write code which would work even if the compiler did not do an arithmetic shift, you can use something like this:

```
static inline int32_t
arsh(int32_t x, int n)
{
#if ROBUST_ARITHMETIC_RIGHT_SHIFT
    uint32_t y = (uint32_t)x >> n;
    y |= -(y & (0x80000000u >> n));
    return *(int32_t *)&y;
#else
    return x >> n;
#endif
}
```

Note that the cast from uint32_t to int32_t uses pointer aliasing, and this is guaranteed correct by the C standard because exact-width types like int32_t use two's complement and have no padding bits and no trap representation. That would not work with other types such as plain 'int', though.

--Thomas Pornin

Le dimanche 16 juin 2019 16:53:56 UTC-4, Danny Niu a écrit :

> Hi, all.
>
> I've posted this as an official comment on the website, but didn't get a reply obviously because it didn't reach the forum.
>
> The question is simple, did the Crystals team intend arithmetic shift when shifting signed integer types from <stdint.h>? Because that's not mandated as a standard behavior in ISO C right now.
>
> Thanks.

--

From: Peter Schwabe <peter@cryptojedi.org>
Sent: Tuesday, June 18, 2019 1:36 AM
To: Thomas Pornin
Cc: pqc-forum
Subject: Re: [pqc-forum] Re: Official comment for Dilithium not received by the forum.
Attachments: signature.asc

'Thomas Pornin' via pqc-forum <pqc-forum@list.nist.gov> wrote:

Dear Thomas, dear all,

> (I am not part of the Dilithium team and I speak only for myself
> here.)

The Dilithium team wouldn't have been able to explain it any better.
Yes, our code assumes that right shifting a signed integer is an arithmetic right shift.

All the best,

Peter

From:	赵运磊 <ylzhao@fudan.edu.cn>
Sent:	Tuesday, February 11, 2020 1:26 AM
To:	pqc-comments
Cc:	pqc-forum
Subject:	ROUND 2 OFFICIAL COMMENT: CRYSTALS-DILITHIUM： Generalization and Optimization of Dilithium

Dear Dilithium team:

In the presentation of Dilithim at the first PQC standardization conference, whether better trade-offs on the already remarkable performance of Dilithium can be made is left as an interesting open question.

In cryptology eprint archive report 2018/1180, which is available from https://eprint.iacr.org/2018/1180 (with recent update by providing new parameters), we provide new insights in interpreting the design of Dilithium, in terms of key consensus previously proposed in the literature for key encapsulation mechanisms (KEM) and key exchange (KEX). Based on the deterministic version of the optimal key consensus with noise (OKCN) mechanism, originally developed in [JZ16] for KEM/KEX, we present \emph{signature from key consensus with noise} (SKCN), which could be viewed as generalization of Dilithium. The construction of SKCN is generic, modular and flexible.

We made much efforts to search new parameters for SKCN as well as Dilithium. We focus on parameters for about 128-bit pq-security, which we believe is the most important set of parameters for practice. on the recommended parameters, compared with Dilithium SKCN is more efficient both in computation and in bandwidth, while preserving the same level of post-quantum security. In addition, using the same routine of OKCN for both KEM/KEX and digital signature eases implementation and deployment in practice, and is useful to simplify the system complexity of lattice-based cryptography in general.

For new parameters of future version of Dilithium, we may suggest to use \eta=2 (i.e., the secret and the noise are from [-2, 2]). In this case, q<=1952257 is for hardness of LWE of 128 pq-security. The new parameter set for SKCN (that is also applicable to Dilithium) is summarized as follows.

	SKCN	Dilithium
q	1952257	8380417
n	256	256
(h,l)	(5,4)	(5,4)
eta	2	5
|pk|	1312	1472
|sk|	3056	3504
|sig|	2573	2701
Repetition	5.67	6.6
MLWE	128	128
MSIS	125	125

Best regards
Yunlei

From:	pqc-forum@list.nist.gov on behalf of D. J. Bernstein <djb@cr.yp.to>
Sent:	Tuesday, July 28, 2020 1:06 PM
To:	pqc-comments
Cc:	pqc-forum
Subject:	[pqc-forum] ROUND 2 OFFICIAL COMMENT: CRYSTALS-DILITHIUM
Attachments:	signature.asc

The latest NIST report states that "CRYSTALS-KYBER shares a common framework with the CRYSTALS-DILITHIUM signature scheme, which is also a finalist." I'm filing this comment to request clarification of what this shared "framework" is referring to.

I didn't find a citation or further explanation, so formally this is a question for NIST, but if the statement actually originates with or is endorsed by the Dilithium or Kyber teams then perhaps they can clarify.

I searched for the word "framework" in the Kyber documentation and found a paper title "A framework for efficient and composable oblivious transfer", which is cited for an encryption scheme, which is not relevant to Dilithium. I also searched for the word "framework" in the Dilithium documentation and found it used a few times to refer to signature schemes, which are not relevant to Kyber.

There are huge code differences and huge spec differences. I don't see how the attack analysis is shared, beyond what's shared across all lattice submissions. The round-1 randomness disaster was for Dilithium, not Kyber. The round-2 implementor objections to a non-full NTT were for Kyber, not Dilithium. If NIST seriously believes that 20000 bytes + 2 million cycles isn't "acceptable" for a TLS session (see my separate message regarding Frodo) then, given the number of signatures per TLS session, NIST would seem forced to conclude that Dilithium's performance also isn't "acceptable", while this conclusion isn't forced for Kyber.
NIST alludes to patent issues for Kyber, not Dilithium.

In general, the mention of sharing across finalists seems to be trying to hint that taking Dilithium and Kyber as a package would create some sort of benefit for users, but I'm puzzled as to what the claimed benefit is. If lower-performance Dilithium will potentially be riding on Kyber's coattails then it should be clear for the record why. In the opposite direction, if Kyber is eliminated because of US patent 9094189 or US patent 9246675, then I don't see why this should be held against Dilithium.

I realize that the Dilithium author list is most of the Kyber author list, and that the formal names of the submissions share a prefix.

---Dan

From:	Moody, Dustin (Fed)
Sent:	Wednesday, July 29, 2020 2:04 PM
To:	D. J. Bernstein; pqc-comments
Cc:	pqc-forum
Subject:	Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: CRYSTALS-DILITHIUM

Dan,

A "framework" is the "basic structure of something".

CRYSTALS-KYBER and CRYSTALS-DILITHIUM are both part of a "cryptographic suite based on algebraic lattices" (https://pq-crystals.org.) They both use module lattices, and that "the only operations required for Kyber and Dilithium for *all* security levels are variants of Keccak, additions/multiplications in Z_q for a fixed q, and the NTT (number theoretic transform) for the ring $Z_q[X]/(X^{256}+1)$." We do note they use different values of q. One is a KEM, while the other is a digital signature. However, we think their basic structure is similar enough that we said they share a common framework.

Dustin

From: pqc-forum@list.nist.gov on behalf of D. J. Bernstein
Sent: Tuesday, July 28, 2020 1:05 PM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: CRYSTALS-DILITHIUM

The latest NIST report states that "CRYSTALS-KYBER shares a common framework with the CRYSTALS-DILITHIUM signature scheme, which is also a finalist." I'm filing this comment to request clarification of what this shared "framework" is referring to.

I didn't find a citation or further explanation, so formally this is a question for NIST, but if the statement actually originates with or is endorsed by the Dilithium or Kyber teams then perhaps they can clarify.

I searched for the word "framework" in the Kyber documentation and found a paper title "A framework for efficient and composable oblivious transfer", which is cited for an encryption scheme, which is not relevant to Dilithium. I also searched for the word "framework" in the Dilithium documentation and found it used a few times to refer to signature schemes, which are not relevant to Kyber.

There are huge code differences and huge spec differences. I don't see