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