From: Alperin-Sheriff, Jacob (Fed)
Sent: Wednesday, February 14, 2018 11:10 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: Ramstake

Two notes:

1. I see a potential implementation pitfall that should have been at least noted in the specification. Due to the fact 216091%256 = 27, 756839%256=103, the most significant byte of serialize(S) is going to have less entropy (under whatever/most optimistic assumption) than would be necessary to hide encode(seed) . While the concrete implementations don't ever end up using those most significant bytes, there is no clear indication that they are unsafe to use and there ought to be.
2. The specification of the decapsulation algorithm is woefully and inherently non-constant time (the way you determine if one of the decodings is "good" and then break from a loop if you've found a good one). Having run some cycle-counting it seems very feasible to learn from the time Decaps takes with high probability how many "bad" encodings there were. As at least 1 bad encoding occurs with pretty high frequency this could probably yield useful information that could allow mounting a CCA attack (although I haven't worked out the details yet).

   Like any timing issue, it should obviously be fixable without too much trouble at the cost of worse performance on average.

—Jacob Alperin-Sheriff

From: alan szepieniec <alan.szepieniec@gmail.com>
Sent: Sunday, February 18, 2018 11:48 AM
To: Alperin-Sheriff, Jacob (Fed)
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Ramstake

Hi Jacob, hi all,

1. Unless I misunderstand, you mean to mod out by 8 and not by 256. So there are only 216091%8=3 and 756839%8=7 bits left in the most significant byte. Also, serialize(S) is cut into chunks of 255 bytes which are then used to mask codewords of the same length. So there are only 216091%(8*255)=1891 and 756839%(8*255)=2039 bits of masking material left in the last chunk, which obviously is not enough to mask a 255*8=2040 bit codeword. So the most significant byte and the most significant chunk should not be used (and they aren't). I agree that this is worth a cautionary note.

2. Completing the loop regardless of early success will make a timing attack more difficult, but not impossible. The current implementation relies on non-constant-time Reed-Solomon error correction as well as non-constant-time big number arithmetic. It is possible to make everything constant-time, but maybe only with a little more than "not too much" trouble ;)

Alan

On Wed, Feb 14, 2018 at 5:09 PM, Alperin-Sheriff, Jacob (Fed) <jacob.alperin-sheriff@nist.gov> wrote:
> Two notes:
>
>
>
> I see a potential implementation pitfall that should have been at
> least noted in the specification. Due to the fact 216091%256 = 27,
> 756839%256=103, the most significant byte of serialize(S) is going to
> have less entropy (under whatever/most optimistic assumption) than
> would be necessary to hide
> encode(seed) . While the concrete implementations don’t ever end up
> using those most significant bytes, there is no clear indication that
> they are unsafe to use and there ought to be.
> The specification of the decapsulation algorithm is woefully and
> inherently non-constant time (the way you determine if one of the decodings is “good”
> and then break from a loop if you’ve found a good one). Having run
> some cycle-counting it seems very feasible to learn from the time
> Decaps takes with high probability how many “bad” encodings there
> were. As at least 1 bad encoding occurs with pretty high frequency
> this could probably yield useful information that could allow mounting
> a CCA attack (although I haven’t worked out the details yet).
>
>
>
> Like any timing issue, it should obviously be fixable without too much
> trouble at the cost of worse performance on average.
>
>
>