From:	Martin Tomlinson <mt@post-quantum.com>
Sent:	Sunday, June 2, 2019 5:32 AM
To:	pqc-comments
Cc:	pqc-forum@list.nist.gov
Subject:	ROUND 2 OFFICIAL COMMENT: Classic McEliece
Attachments:	Message signed with OpenPGP using GPGMail.asc

Further to my comment about ISARA's recent US patent covering lattice based submissions, I have discovered that ISARA also have a US patent, US9912479, which is a KEM patent using McEliece, granted in 2018. It appears to cover the IP of Classic McEliece and may well cover other code based submissions.

--Martin

--

From:	D. J. Bernstein <djb@cr.yp.to>
Sent:	Sunday, June 2, 2019 10:25 AM
To:	pqc-comments
Cc:	pqc-forum@list.nist.gov
Subject:	Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: Classic McEliece
Attachments:	signature.asc

I've been working on an analysis of post-quantum patents, and I'm confident that US9912479 won't survive a court case. (Obviously it's easier for everybody if ISARA simply gives up the patent rather than frivolously forcing litigation.)

This patent is similar to the Gaborit--Aguilar Melchor patent in that it covers a broad range of submissions to NIST, not just Classic McEliece.
The big difference is that the Gaborit--Aguilar Melchor patent was filed _before_ the relevant scientific literature, while this patent was filed _after_ the relevant scientific literature.

In case anyone ends up in litigation about patent 9912479, here's prior art that directly kills claim 1 of the patent:

   * https://eprint.iacr.org/2016/461, May 2016 version, Figure 2.2.
     (The patent application was filed June 2017.)

This is a special case of https://eprint.iacr.org/2002/174.pdf, Theorem 4, which builds a KEM that produces a session key and confirmation by hashing a random PKE input. The special case here (Streamlined NTRU Prime 4591^761) uses a particular PKE where the input is a random weight-w vector.

(For people not familiar with patent law: If a previous publication has all the elements of a claim then the claim is automatically invalid. One doesn't need to analyze obviousness. The patent holder can't escape by saying that the prior art is more specific than the claim; patent law doesn't let you patent generalizations of prior art.)

The records of ISARA's discussions with the patent office show that the examiner identified an earlier publication with all the same elements, and that ISARA got around this only by saying that this publication used a _random_ vector while ISARA had invented _pseudorandom_ vectors. I've skipped this ridiculous issue by pointing to prior art where the random vectors are indisputably derived from output of a PRNG.

The same prior art https://eprint.iacr.org/2016/461 also directly kills most of the other claims in the patent, still without an obviousness analysis. Some of the claims are artificially narrowed to McEliece, but there are at least four independent ways to kill those claims:

   * Take the prior art cited by the examiner, and point to any number
     of sources (the Ferguson--Schneier book, NIST DRBG standards, etc.)
     explaining in detail that "random" numbers in cryptography are
     normally generated pseudorandomly. This still doesn't need an
     obviousness analysis.

   * Take https://eprint.iacr.org/2002/174.pdf, Theorem 4, as prior art.
     The theorem has interchangeable parts---it explicitly lets you plug
     in any "deterministic encryption algorithm that is secure in the
     OW-CPA model"---so it's obvious to try anything that's identified
     in the literature as a deterministic OW-CPA encryption algorithm.

https://persichetti.webs.com/Thesis%20Final.pdf identifies McEliece
as a deterministic OW-CPA encryption algorithm.

* Take https://eprint.iacr.org/2016/461 as prior art, and argue that
  replacing NTRU with McEliece is an obvious variation, given the
  literature drawing analogies between NTRU and McEliece. (Of course
  this is how ISARA came up with this "invention" in the first place.
  I recommend that researchers avoid collaborating with ISARA, and
  avoid allowing ISARA people to review paper submissions.)

* After killing claim 1 on the basis of prior art, point out that
  ISARA certainly knew this prior art and thus engaged in what's
  called "inequitable conduct" under patent law. This automatically
  kills all the other claims of the patent.

---Dan (speaking for myself)

From: pqc-forum@list.nist.gov on behalf of James Howe <jameshoweee@gmail.com>
Sent: Monday, May 18, 2020 11:40 AM
To: pqc-forum
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: Classic McEliece

Dear authors,

I wanted to see if you could add any more insights into the hardware designs quoted in your latest specifications [3]. The paper by Wang et al. [1], cited a lot in the specifications, is for Niederreiter with Goppa codes as an encryption scheme, so I'm interested to see the differences between this and the McEliece KEM submission.

One of my main questions is, as seen in a lot of hardware/software implementations before, how much more SHAKE would cost you in terms of area and/or performance. Your specifications [3] seems to suggest [1] and/or [2] implement the McEliece KEM, as on page 34 of [3] you state that "encapsulation and decapsulation are reasonably fast in software, and impressively fast in hardware, due to the simple nature of the objects (binary vectors) and operations (such as binary matrix-vector multiplications)".

So maybe I'm missing something? Perhaps you have a link to a paper describing your hardware design [2] that's more up-to-date than [1]? Because typically it's also interesting to see how much certain components/modules consume, what are the bottlenecks, throughput performances, and even % of area used on the device.

Looking at [1], which seems to be a basis for the McEliece KEM hardware designs, it seems to be CPA secure instead of CCA? Fig. 2 in [1] doesn't have a re-encryption phase and interestingly on page 15 of [1] they state that "note that we are using two simple 64-bit Xorshift PRNGs in our design to enable deterministic testing. For real deployment, these PRNGs must be replaced with a cryptographically secure random-number generator, e.g., [8]". Checking [2] I also see the same 64-bit XOR shift PRNG verilog file. I assume you'd replace this with SHAKE256 to comply with the specifications and security estimates? Do you have any insights how this would affect area/performance?

Anyway, some more clarifications on this would be helpful :)

Many thanks
James

[1] https://eprint.iacr.org/2017/1180.pdf
[2] http://caslab.csl.yale.edu/code/niederreiter/
[3] https://classic.mceliece.org/nist/mceliece-20190331.pdf
--

From:	pqc-forum@list.nist.gov on behalf of Ruben Niederhagen <ruben.niederhagen@sit.fraunhofer.de>
Sent:	Wednesday, May 27, 2020 11:07 AM
To:	James Howe; pqc-forum
Subject:	Re: [pqc-forum] ROUND 2 OFFICIAL COMMENT: Classic McEliece

Dear James,

thank you for your interest!

The submission document [3] reports "measurements of a separate FPGA implementation of the core mathematical functions (not including, e.g., hashing)".

Page 34 of [3] says that encapsulation, decapsulation, and key generation are fast in hardware, and then says that the "hardware speeds of key generation and decoding are already demonstrated by our FPGA implementation". Decoding is a major subroutine in decapsulation.

A natural next step is to build a complete implementation of the KEM.
There is plenty of literature on hardware implementations of SHAKE. We expect that the total hardware resources for the KEM will be not much more than the core mathematical operations, and that the additional cost in terms of latency due to the KEM construction will be similar to other KEM constructions.

On 18/05/2020 17:39, James Howe wrote:
> I wanted to see if you could add any more insights into the hardware
> designs quoted in your latest specifications [3]. The paper by Wang et al.
> [1], cited a lot in the specifications, is for Niederreiter with Goppa
> codes as an encryption scheme, so I'm interested to see the
> differences between this and the McEliece KEM submission.

I am aware of only two references to [1] in the submission document; neither of them is part of the specification.

As the submission document explains, the FPGA implementation covers only the core mathematical functions. For example, the implementation includes decryption, but it does not include the hashing required for full decapsulation.

> One of my main questions is, as seen in a lot of hardware/software
> implementations before, how much more SHAKE would cost you in terms of
> area and/or performance.

Please see the literature on SHAKE implementations. The focus in [1] and [2] is on the core mathematical computations.

> Your specifications [3] seems to suggest [1] and/or [2] implement the
> McEliece KEM,

No, [3] does not suggest this. It states that the FPGA implementation is "of the core mathematical functions (not including, e.g., hashing)".

> as on page 34 of [3] you state that
> "encapsulation and decapsulation are reasonably fast in software, and

> impressively fast in hardware, due to the simple nature of the objects
> (binary vectors) and operations (such as binary matrix-vector
> multiplications)".

This is correct. The next two sentences say "Key generation is also quite fast in hardware. The hardware speeds of key generation and decoding are already demonstrated by our FPGA implementation."

> So maybe I'm missing something?

The submission document [3] includes a specification. It also includes a performance analysis. The performance claims are clear and specific. [1] and [2] demonstrate the achievable FPGA performance for the core mathematical operations.

> Perhaps you have a link to a paper
> describing your hardware design [2] that's more up-to-date than [1]?
> Because typically it's also interesting to see how much certain
> components/modules consume, what are the bottlenecks, throughput
> performances, and even % of area used on the device.

Please see [1], [2], and the SHAKE literature for analysis of the main components.

> Looking at [1], which seems to be a basis for the McEliece KEM
> hardware designs, it seems to be CPA secure instead of CCA? Fig. 2 in
> [1] doesn't have a re-encryption phase and interestingly on page 15 of
> [1] they state that "note that we are using two simple 64-bit Xorshift
> PRNGs in our design to enable deterministic testing. For real
> deployment, these PRNGs must be replaced with a cryptographically
> secure random-number generator, e.g., [8]". Checking [2] I also see the same 64-bit XOR shift PRNG verilog file.
> I assume you'd replace this with SHAKE256 to comply with the
> specifications and security estimates? Do you have any insights how
> this would affect area/performance?

I do not know what "the McEliece KEM hardware design" is referring to.
[1] and [2] provide an FPGA implementation of the Niederreiter cryptosystem (as the paper "FPGA-based Niederreiter Cryptosystem using Binary Goppa Codes" clearly states) including key generation, encryption and decryption. This is not the same as the Classic McEliece KEM, but it uses the same core mathematical operations. The performance of those operations is reported in the Classic McEliece performance analysis.

As the quotes from [1] indicate, [1] describes a CPA secure version of the Niederreiter PKE. We also have a CCA secure version of the decryption operation that we can make available. The simple PRNG indeed must be replaced by a secure PRNG as we point out in [1]. Using e.g.
SHAKE in key generation for a PRNG construction, the overhead in time and area compared to the entire key generation is small; parallelism can be exploited to reduce the impact on time, e.g., by performing matrix row reduction during generating the next private key candidate.

> Anyway, some more clarifications on this would be helpful :)

I hope this helped to clear up some misunderstandings.

Best regards
  Ruben (on behalf of the Classic McEliece team)

> [1] https://eprint.iacr.org/2017/1180.pdf
> [2] https://caslab.csl.yale.edu/code/niederreiter/
> [3] https://classic.mceliece.org/nist/mceliece-20190331.pdf

--
Ruben Niederhagen, PhD
Fraunhofer Institute for Secure Information Technology (SIT) Head of Advanced Cryptographic Engineering (ACE)

Address: Rheinstraße 75, 64295 Darmstadt, Germany
Phone: +49 6151 869 - 135
Homepage: https://www.sit.fraunhofer.de/en/pqcryptography/

--

From:	pqc-forum@list.nist.gov on behalf of Tung Chou <blueprint@crypto.tw>
Sent:	Friday, June 5, 2020 11:37 AM
To:	pqc-comments@list.nist.gov
Cc:	pqc-forum
Subject:	[pqc-forum] ROUND 2 OFFICIAL COMMENT: Classic McEliece

I am writing this email to announce the latest keygen speed on Haswell
of Classic McEliece. Compared to the round-2-submission version, the new
code is much faster. The timings (medians) are listed below: the column
"new" shows the latest numbers, which (as mentioned by Dan) can be found
on

https://bench.cr.yp.to/results-kem.html#amd64-hiphop,

and "submission" means the numbers in the round-2 Classic McEliece
submission document.

	new	submission
mceliece348864	52415436	140870324
mceliece460896	181063400	441517292
mceliece6688128	467870488	1180468912
mceliece6960119	417271280	1109340668
mceliece8192128	424239104	933422948

Various optimizations have been applied to achieve the speedup. The same
optimizations have also been applied to our "f" parameter sets, i.e.,
the parameter sets that are "implemented as a possible future proposal";
See below.

	new	submission
mceliece348864f	44064240	82232360
mceliece460896f	141781900	282869316
mceliece6688128f	287968144	625470504
mceliece6960119f	264846568	564570384
mceliece8192128f	319665520	678860388

Tung Chou