From: D. J. Bernstein <djb@cr.yp.to>
To: pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: ROUND 3 OFFICIAL COMMENT: Classic McEliece
Date: Monday, June 06, 2022 12:21:02 PM ET
Attachments: smime.p7m

This is an official comment from the Classic McEliece team regarding a recent talk at Eurocrypt on https://eprint.iacr.org/2021/1634, a paper reporting implementations of the MMT/BJMM algorithms using about 2^60.7 cycles in total on 256 AMD EPYC 7742 CPU cores to break miniature versions of McEliece with length 1284.

For comparison, when the 2021 Bellini--Esser estimator

   https://github.com/Crypto-TII/syndrome_decoding_estimator.git

is told to ignore the cost of memory access ("memory_access=0"), it says 2^66.3 bit operations for 1989 Stern, followed by minor improvements from newer memory-intensive algorithms, such as 2^65.8 bit operations for BJMM and 2^64.6 bit operations for May--Ozerov. These numbers are larger than 2^60.7, but the units are also different: bit operations rather than CPU cycles.

These algorithms were already accounted for in the Classic McEliece submission in 2017, which, regarding the recommended 6960119 parameter set, wrote "ISD variants have reduced the number of bit operations considerably below 2^256". The submission also pointed out that this ignores the cost of memory access, and stated an expectation that "switching from a bit-operation analysis to a cost analysis will show that this parameter set is more expensive to break than AES-256 pre-quantum and much more expensive to break than AES-256 post-quantum."

This expectation was then confirmed by subsequent analysis. For example, the Bellini--Esser estimator with "memory_access=2" reports cost 2^279.2, as noted in the Classic McEliece team email to pqc-forum in November 2021.

Anyone who wants such a high security level _without_ relying on the cost of memory access can instead use the 8192128 parameter set, where the Bellini--Esser estimator with "memory_access=0" says 2^275.6 bit operations for May--Ozerov, assigning zero cost to 2^194.4 memory. The exponent here is about 10% better than attacks from the 1980s (e.g., the estimator says 2^307.4 bit operations for Dumer), but the changes in costs are much smaller if one accounts for the real cost of memory.

The new paper and talk claim that some of the Classic McEliece parameter sets "fail to reach their security level by roughly 20 and 10 bit", that "McEliece Slightly Overestimates Security", etc., giving the impression of challenging the Classic McEliece security analysis and parameter selection. However, these claims are based on three serious errors.

The first error is comparing the cost of a memory-intensive attack to the cost of an AES attack "on our hardware," while ignoring the fact that the machine puts a large volume of hardware into optimizing memory access and only a tiny fraction of the hardware into AES circuits. It is easy to make an AES attack sound much more expensive than it actually is if one uses a highly suboptimal AES attack machine.

The second error is modeling random access to an array of size 2^80 as being as cheap as 80 bit operations. The way the paper arrives at this physically implausible model is by arguing that logarithmic cost "most accurately models our experimental data". The data points were selected from a limited range for which memory access has essentially constant cost on that machine; this is selection bias, not a valid basis for extrapolation.

The third error is jumping from a comparison in this cheap-RAM model to a claim of a Classic McEliece "overestimate". The submission has always said that the number of bit operations is "considerably below 2^256"; obviously this does not reach 2^272 if one assigns merely logarithmic cost to RAM (changing the exponent by at most 8 bits). What the paper says here is matching what the submission says, not disputing it. The

Classic McEliece assignment of category 5 to the 6960119 parameter set has always been explicitly based on the real cost of memory access.

From: Andre Esser <andre.r.esser@gmail.com> via pqc-forum@list.nist.gov
To: pqc-forum <pqc-forum@list.nist.gov>
CC: D. J. Bernstein <djb@cr.yp.to>, pqc-...@list.nist.gov <pqc-forum@list.nist.gov>, pqc-co...@nist.gov <pqc-comments@nist.gov>, pqc-co...@nist.gov <pqc-comments@nist.gov>
Subject: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: Classic McEliece
Date: Wednesday, June 08, 2022 05:20:33 AM ET

The parameter selection process of Classic McEliece takes the asymptotic runtime exponent of Prange

and chooses the code parameters such that it approximately matches the respective AES-keysize, i.e., 128, 192 or 256.

The security analysis then relies on the not quantified statement that no algorithmic improvement over Prange needs

to be considered because in a real attack the memory access costs outweigh the improvement.

It does not need much to “challenge this security analysis and parameter selection”. And without a doubt this can not

be sufficient for a final standardization of parameter sets. For this, the McEliece team has to extend its submission

(in the forth round) by the necessary formalisms that support their security arguments.

Instead the team attacks the credibility of the authors of one of the first formal treatments, arriving at the conclusion

of a **slight** security overestimation. It seems unclear, how the team can rule out a security deficit of a few bits with such

certainty, but is failing to give a convincing argument / formalism in the submission.

> The first error is comparing the cost of a memory-intensive attack to
> the cost of an AES attack “on our hardware,” while ignoring the fact
> that the machine puts a large volume of hardware into optimizing memory
> access and only a tiny fraction of the hardware into AES circuits. It is

> easy to make an AES attack sound much more expensive than it actually is
> if one uses a highly suboptimal AES attack machine.
While we would not agree that a processor with AES hardware acceleration is a particularly suboptimal way of attacking

AES, there are of course more specialized systems, but so are for ISD attacks. This goes probably back to the vague

security definitions in form of category I, III and V, which do not specify any benchmarking platform, model or metrics.
But here you are asking **us** to provide the formalisms on which you seem to have based the security of the parameter sets.

We are happy to have a dialogue about it, but the “error” of missing formalisms for such a comparison is not on us.
Also note that the effect of the exact machine we used in comparison to a completely theoretical treatment is insignificant.

We arrive at almost the same security-deficits as Esser-Bellini.

> The second error is modeling random access to an array of size 2^80 as
> being as cheap as 80 bit operations. The way the paper arrives at this
> physically implausible model is by arguing that logarithmic cost “most
> accurately models our experimental data”. The data points were selected
> from a limited range for which memory access has essentially constant
> cost on that machine; this is selection bias, not a valid basis for
> extrapolation.
We are modeling the **amortized** memory access cost. This accounts for the fact that not every memory access is completely

random. Moreover, our data structures are specifically designed to allow for good access patterns.
Of course, by the inherent limitation given by our hardware constraints the data points for extrapolation are selected from a

limited range. However, the high memory experiments were run using about 1.6TB of RAM (which gives significant access times).

And still we find the point where high memory beats low memory at code length 1400. If we theoretically model higher memory

access costs this break even point should not exist.

> The third error is jumping from a comparison in this cheap-RAM model to
> a claim of a Classic McEliece “overestimate”. The submission has always
> said that the number of bit operations is “considerably below 2^256";
> obviously this does not reach 2^272 if one assigns merely logarithmic
> cost to RAM (changing the exponent by at most 8 bits). What the paper
> says here is matching what the submission says, not disputing it. The
> Classic McEliece assignment of category 5 to the 6960119 parameter set
> has always been explicitly based on the real cost of memory access.
The McEliece submission does never specify what the “real cost of memory access” is and to what extend it influences the security

level. Making it easy to rule out any access cost that contradicts the security as “not real”. Moreover, the category 3 set has now

been found repeatedly to not match the security guarantees even under higher memory access costs.

We do not claim that our work is the holy grail in terms of formalizing security arguments for McEliece, but the introduction of

formalisms were overdue. Admittedly, the memory access models are quite idealized, but we are convinced that a cube-root modeling

of the amortized cost overestimates the effort.

We are happy to have discussions about it and to see other models evolving that do a better job in capturing all real world factors.

And finally, of course, we hope to see the results embedded in the Classic McEliece specification and security analysis.

However, we would like all discussions to be peace- and respectful. And preferably in person or in papers, to avoid the time consuming

crafting of such long monologues.

Best Regards,

Andre, Alex and Floyd

D. J. Bernstein schrieb am Montag, 6. Juni 2022 um 20:21:02 UTC+4:

> This is an official comment from the Classic McEliece team regarding a
> recent talk at Eurocrypt on https://eprint.iacr.org/2021/1634, a paper
> reporting implementations of the MMT/BJMM algorithms using about 2^60.7
> cycles in total on 256 AMD EPYC 7742 CPU cores to break miniature
> versions of McEliece with length 1284.
>
> For comparison, when the 2021 Bellini--Esser estimator
>
> https://github.com/Crypto-TII/syndrome_decoding_estimator.git
>
> is told to ignore the cost of memory access ("memory_access=0"), it says
> 2^66.3 bit operations for 1989 Stern, followed by minor improvements
> from newer memory-intensive algorithms, such as 2^65.8 bit operations
> for BJMM and 2^64.6 bit operations for May--Ozerov. These numbers are
> larger than 2^60.7, but the units are also different: bit operations
> rather than CPU cycles.
>
> These algorithms were already accounted for in the Classic McEliece
> submission in 2017, which, regarding the recommended 6960119 parameter
> set, wrote "ISD variants have reduced the number of bit operations
> considerably below 2^256". The submission also pointed out that this
> ignores the cost of memory access, and stated an expectation that
> "switching from a bit-operation analysis to a cost analysis will show
> that this parameter set is more expensive to break than AES-256
> pre-quantum and much more expensive to break than AES-256 post-quantum."
>
> This expectation was then confirmed by subsequent analysis. For example,
> the Bellini--Esser estimator with "memory_access=2" reports cost
> 2^279.2, as noted in the Classic McEliece team email to pqc-forum in
> November 2021.
>
> Anyone who wants such a high security level _without_ relying on the
> cost of memory access can instead use the 8192128 parameter set, where
> the Bellini--Esser estimator with "memory_access=0" says 2^275.6 bit

> operations for May--Ozerov, assigning zero cost to 2^194.4 memory. The exponent here is about 10% better than attacks from the 1980s (e.g., the estimator says 2^307.4 bit operations for Dumer), but the changes in costs are much smaller if one accounts for the real cost of memory.
>
> The new paper and talk claim that some of the Classic McEliece parameter sets "fail to reach their security level by roughly 20 and 10 bit", that "McEliece Slightly Overestimates Security", etc., giving the impression of challenging the Classic McEliece security analysis and parameter selection. However, these claims are based on three serious errors.
>
> The first error is comparing the cost of a memory-intensive attack to the cost of an AES attack "on our hardware," while ignoring the fact that the machine puts a large volume of hardware into optimizing memory access and only a tiny fraction of the hardware into AES circuits. It is easy to make an AES attack sound much more expensive than it actually is if one uses a highly suboptimal AES attack machine.
>
> The second error is modeling random access to an array of size 2^80 as being as cheap as 80 bit operations. The way the paper arrives at this physically implausible model is by arguing that logarithmic cost "most accurately models our experimental data". The data points were selected from a limited range for which memory access has essentially constant cost on that machine; this is selection bias, not a valid basis for extrapolation.
>
> The third error is jumping from a comparison in this cheap-RAM model to a claim of a Classic McEliece "overestimate". The submission has always said that the number of bit operations is "considerably below 2^256"; obviously this does not reach 2^272 if one assigns merely logarithmic cost to RAM (changing the exponent by at most 8 bits). What the paper says here is matching what the submission says, not disputing it. The Classic McEliece assignment of category 5 to the 6960119 parameter set has always been explicitly based on the real cost of memory access.

--

From: D. J. Bernstein <djb@cr.yp.to>
To: pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] Re: ROUND 3 OFFICIAL COMMENT: Classic McEliece
Date: Tuesday, June 14, 2022 12:07:03 AM ET
Attachments: smime.p7m

Andre Esser writes:

> The parameter selection process of Classic McEliece takes the
> asymptotic runtime exponent of Prange and chooses the code parameters
> such that it approximately matches the respective AES-keysize, i.e.,
> 128, 192 or 256.

No, that's not at all how the Classic McEliece parameters were chosen.
The parameter sets were explicitly selected as follows:

* 348864: "optimal security within 2^18 bytes if n and t are required to be multiples of 32".
* 460896: "optimal security within 2^19 bytes if n and t are required to be multiples of 32".
* 6688128: "optimal security within 2^20 bytes if n and t are required to be multiples of 32".
* 6960119: same without the multiples-of-32 restriction.
* 8192128: "taking both n and t to be powers of 2".

Parameter optimization for any specified key size is simpler and more
robust than trying to work backwards from comparisons between McEliece
and unrelated cryptosystems. There's no inherent power-of-2 requirement
for the key sizes, but 1MB, 0.5MB, 0.25MB are easy to remember.

The quotes above are from the submission document that presented the
current list of proposed parameters:

https://classic.mceliece.org/nist/mceliece-20190331-mods.pdf

Scientifically, this parameter-selection strategy for McEliece goes back
at least to https://eprint.iacr.org/2008/318. See the paragraph in that

paper beginning "For keys limited to", in particular obtaining n = 6960 from a 1MB limit.

The underlying security evaluations have always been explicitly based on concrete analyses, _not_ asymptotics (even though asymptotics are helpful for assessing security stability). This was already emphasized in Section 8.2 of the original submission document:

   https://classic.mceliece.org/nist/mceliece-20171129.pdf

The section begins as follows:

   We emphasize that o(1) does not mean 0: it means something that converges to 0 as n → ∞. More detailed attack-cost evaluation is therefore required for any particular parameters.

That section continues by pointing to the 2008 paper mentioned above, https://eprint.iacr.org/2008/318, as the source of n = 6960. Anyone checking that paper sees that the paper obtained this value of n via a concrete analysis of that paper's attack, not from asymptotics.

Furthermore, that attack is noticeably faster than Prange's algorithm. Asymptotically, this cost difference disappears into a 1+o(1) factor in the exponent, but the parameter selection has never relied on any such simplification.

To be clear, it's not that all of the parameter details are from 2008. For example, to simplify KEM implementations, the Classic McEliece parameter choices avoid list decoding (which would otherwise allow 1 or 2 extra errors). More importantly, the smaller parameter sets were not in the original 2017 proposal; they were added in 2019, in response to NIST making clear in 2019 that it wanted smaller options.

For any particular parameter set, evaluations of the Classic McEliece parameter proposals using the 2008 scripts and various post-2008 scripts show some differences, mostly minor differences regarding exactly which attack overheads are counted and exactly which attacks are covered. The

largest quantitative differences come from the gaps between free memory
and realistic models.

The Classic McEliece team filed an OFFICIAL COMMENT years ago requesting
that NIST "fully define the cost metric to be used" for NISTPQC, so that
all submission teams could evaluate costs in this metric:

   https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/EiwxGnfQgec/m/LqckEVciAQAJ

In the absence of action by NIST to settle on a metric for NISTPQC, the
Classic McEliece team filed another OFFICIAL COMMENT in November 2021
with numbers from a recent estimator for all proposed parameter sets:

   https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/dN_O0rvsLV4/m/QZ4UjtxnAwAJ

For example, that estimator says 2^279.2 for the 6960119 parameter set,
in line with the expectations stated in the original Classic McEliece
submission in 2017. This is not a surprise, given how stable the
landscape of attack algorithms has been.

> The security analysis then relies on the not quantified statement that no
> algorithmic improvement over Prange needs to be considered because in
> a real attack the memory access costs outweigh the improvement.

No. Section 8.2 of the 2017 submission document

   https://classic.mceliece.org/nist/mceliece-20171129.pdf

started from the 2008 numbers (which are already better than Prange),
explicitly considered subsequent algorithms (see also Section 4.1 for
references), observed that the 2008 algorithm and subsequent algorithms
were bottlenecked by memory access, and stated the following regarding
the recommended 6960119 parameter set:

   We expect that switching from a bit-operation analysis to a cost
   analysis will show that this parameter set is more expensive to break
   than AES-256 pre-quantum and much more expensive to break than

AES-256 post-quantum.

Adequate cost quantification wasn't in the literature at that point, but
is now readily available from the November 2021 OFFICIAL COMMENT:

https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/dN_O0rvsLV4/m/QZ4UjtxnAwAJ

The submission has always stated that "ISD variants have reduced the
number of bit operations considerably below 2^256" for 6960119, so the
category-5 assignment for this parameter set relies on accounting for
the costs of memory. For people who want category 5 while ignoring
memory costs, the 8192128 parameter set has always been fully specified
and implemented as part of the Classic McEliece proposal.

> While we would not agree that a processor with AES hardware
> acceleration is a particularly suboptimal way of attacking AES

Here is one way to see that an AES attacker using the same 7nm chip
technology can do _five orders of magnitude_ better than the paper's AES
attack:

* https://www.ant-miner.store/product/antminer-s17-56th/ describes
  real Bitcoin mining hardware carrying out 56 terahashes/second at
  2520 watts, i.e., 45*10^-12 joules per hash. If these hashes are
  full Bitcoin hashes, 2xSHA-256, then they are roughly the same cost
  as 16xAES-128, so a similar AES attack box would use roughly
  3*10^-12 joules per AES-128.

* The paper in question, https://eprint.iacr.org/2021/1634, reports
  (in Section 7) 2.16*10^9 "AES encryptions per second performed by
  our cluster". The cluster has four EPYC 7742 CPUs (according to
  Section 4). The power consumption of the cluster isn't reported,
  but presumably is on the scale of 1000 watts, i.e., on the scale of
  500000*10^-12 joules per AES-128.

An easier analysis considers AT rather than energy. Each 64-core EPYC
7742 CPU has 32 billion transistors, meaning that each CPU core has half

a billion transistors:

   https://hexus.net/tech/reviews/cpu/133244-amd-epyc-7742-2p-rome-server/?page=2

The AES attack in question uses the AES instructions, which, on each of these cores, can at best carry out two parallel AES rounds per cycle according to the Zen2 AESENC figures in Agner Fog's tables:

   https://agner.org/optimize/instruction_tables.pdf

Each round uses a few thousand bit operations, with a small number of transistors per bit operation, accounting for only a tiny fraction of the transistors in the CPU. Standard key-search circuits instead have almost all transistors performing cipher operations at each moment, with minor overheads for key selection and comparison.

> but so are for ISD attacks.

Quantitatively, compared to their AES hardware, Intel and AMD put _much_ more of their hardware into optimizing memory access——a serious chunk of each core, plus extensive off-chip resources——for obvious reasons.

The point here isn't that it's impossible to use special-purpose hardware to streamline memory-intensive attacks. The point is that a claim regarding the quantitative costs of two attacks, namely AES key search and a memory-intensive ISD attack, was comparing time but neglecting to account for vast differences in the hardware resources used by the attacks. This is not a meaningful security comparison; it does not correctly predict what large-scale attackers can do.

> But here you are asking *us* to provide the formalisms on which you
> seem to have based the security of the parameter sets.

No. The Classic McEliece team asked "NIST to fully define the cost metric to be used for 'categories'". This is not something that should be decided ad-hoc for particular attacks.

The submission has always explicitly advocated accounting for the real cost of memory ("switching from a bit-operation analysis to a cost analysis"). If it turns out that NIST asks for category 5 with free memory: as noted above, the 8192128 parameter set has always been available.

> We are modeling the *amortized* memory access cost.

All of these algorithms are bottlenecked by large-scale data motion for, e.g., sorting b-bit chunks of data, where b is small. It is physically implausible that moving b bits of data large distances through an array of size 2^80 can be as cheap as 80b bit operations.

> Of course, by the inherent limitation given by our hardware constraints the
> data points for extrapolation are selected from a limited range.

The hardware does not require this selection: the same machine offers lower-cost caches (and higher-cost storage beyond DRAM). Real graphs of memory-access cost such as

    https://i0.wp.com/chipsandcheese.com/wp-content/uploads/2022/06/image-4.png
    https://i0.wp.com/chipsandcheese.com/wp-content/uploads/2022/06/image-1.png
    source: https://chipsandcheese.com/2022/06/07/sunny-cove-intels-lost-generation/

are forced by distance constraints to be worse than square-root lines in log space, and are then bumped to locally horizontal line segments (L1 cache, L2 cache, etc.) to simplify the engineering. Taking experiments within one of those horizontal line segments, and then extrapolating horizontally from this selection of data (or logarithmically, which on such a small scale is difficult to robustly distinguish from horizontally), gives incorrect predictions for larger sizes.

——D. J. Bernstein (speaking for myself, beyond the quotes)