From: pqc-forum@list.nist.gov on behalf of Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Tuesday, July 18, 2023 12:04 AM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

The eMLE-Sig scheme is a lattice-based scheme that seems to be a spiritual successor to the original Round 1 encryption candidate Compact LWE. The design principle of this type of schemes is essentially to say:
“we will use as a trapdoor a lattice vector that isn't particularly short, so that there is no hope to recover it by lattice reduction, but that has some sort of additional hidden structure that still allows it to be used as a secret; moreover, since lattice attacks must fail, we will use a ridiculously small lattice dimension”.

The underlying belief is that the additional hidden structure cannot be taken advantage of in lattice attacks. This belief, however, proved incorrect in the case of Compact LWE (which co-authors and I broke shortly after submission), and again in the first instance of eMLE which was broken on this mailing list by Lorenz Panny a couple of years ago.

It is incorrect again in eMLE-Sig 2.0. Basically, a secret key element x in that scheme is a short element of the “convolution” ring Z[x]/(x^n-1) satisfying an equation of the form:

$$h' = g' \cdot x + p_0 \cdot k'_0 + p_1 \cdot k'_1 + p_2 \cdot k'_2 \quad (*)$$

where g' and h' are known ring elements, the p_i's are rational primes (with p_2 >> p_1 >> p_0) and the k'_i's are unknown but short.

The break on the original version of the scheme was based on the observation that the equation (*) above can be seen as an Inhomogeneous-SIS instance mod p_2 recovering the short vector (x,k'_0,k'_1). This attack is expected to succeed when the length of that vector is substantially smaller than the first minimum of the SIS lattice, which by the Gaussian heuristic is approximately:

$$\lambda_1(L_{SIS}) = \sqrt{n/2\pi e}\, p_2^{1/3}.$$

The countermeasure proposed in the eMLE-Sig 2.0 submission is to add noise making k'_1 larger, so that (x,k'_0,k'_1) is no longer below
lambda_1 above. Concretely speaking, the “level-I” parameter have roughly the following vector Euclidean norms:

$$|x| \approx 20$$
$$|k'_0| \approx |k'_2| \approx 45$$
$$|k'_1| \approx 1300$$
$$\lambda_1 \approx 800$$

So indeed, (x,k'_0,k'_1) is of length larger than lambda_1, and one expects exponentially many vectors of that length in the SIS lattice.
However, it is a very structured in the sense that x and k'_0 are themselves much shorter, and this can obviously be taken advantage of.

The highbrow way to do this is to scale the Euclidean norm itself and adjust the volume computations, etc., accordingly. A similar, more pedestrian way is to scale the subvectors by integer factors: if a, b are coprime to p_2, (a·x, b·k'_0, k'_1) is a solution to the ISIS problem:

$$h' = a^{-1} g'\cdot u + b^{-1} p_0\cdot v + p_1\cdot k'_1 \pmod{p_2}$$

that moreover lies in $aZ^n \times bZ^n \times Z^n$. The normalized volume of the corresponding lattice increases from $p_2^{1/3}$ to $(ab\, p_2)^{1/3}$, and hence the lambda_1 is multiplied by $(ab)^{1/3}$ compared to the above.
Picking a = 65, b = 29 to balance out all the subparts of the vector, we end up looking for a vector of norm $\sqrt{3}\cdot 1300 \sim 2250$ in a lattice with

$$\lambda_1 \approx 800\cdot(ab)^{1/3} \approx 10000$$

which is now a well-defined problem with a unique solution whp. Given the very small lattice dimension 3n = 192, it is also tractable in practice.

I note in addition that, even if the eMLE problem were hard, the proposed signature scheme could not be, since, just like in my previous email regarding EagleSign, the distribution of signatures leaks information about the secret key.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

From: pqc-forum@list.nist.gov on behalf of Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Wednesday, July 19, 2023 11:51 PM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

For the sake of completeness again, you can find sample code that implements the attack below in the following GitHub repository:

https://github.com/mti/attack_emle

For now, it targets the "NIST Level-I" parameter set, and recovers a signing key from the corresponding verification key with fairly good probability (>80%) in a few minutes. Larger parameters should be broken by essentially the same attack code, but this hasn't been verified yet.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Tue, Jul 18, 2023 at 06:03:46AM +0200, Mehdi Tibouchi wrote:

> Dear eMLE-Sig 2.0 submitters, dear all,
>
> The eMLE-Sig scheme is a lattice-based scheme that seems to be a
> spiritual successor to the original Round 1 encryption candidate
> Compact LWE. The design principle of this type of schemes is essentially to say:
> “we will use as a trapdoor a lattice vector that isn't particularly
> short, so that there is no hope to recover it by lattice reduction,
> but that has some sort of additional hidden structure that still
> allows it to be used as a secret; moreover, since lattice attacks must
> fail, we will use a ridiculously small lattice dimension”.
>
> The underlying belief is that the additional hidden structure cannot
> be taken advantage of in lattice attacks. This belief, however, proved
> incorrect in the case of Compact LWE (which co-authors and I broke

From: pqc-forum@list.nist.gov on behalf of Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Thursday, July 20, 2023 1:41 AM
To: Mehdi Tibouchi; pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

Thanks for your analysis.

Simply, your attack is NOT correct (at least for your current version), because you set q=256.
Note that q is just used for security analysis in Sage code and it is not a parameter in the specification and in the c-language reference implementation. In Sage code, we test different values of q to do security analysis and determine the concrete security level. A smaller q means smaller amount of noise than designed is added. The hardness of eMLE 2.0 depends on both dimension and the amount of noises, a difference from lattice problem.

As mentioned in the specification, when q=256, the attack code in our submitted package (adapted from Panny's attack) is 100%, better than your 80%. Different values of q is discussed in Section 5.3.1 of the specification. Our attack code actually exhaustively search your "a"; so I think your attack has been covered in our attack analysis and security estimation.

Also note that in Table 2 of our specification , the norm of k1 is usually above 11000 for n=64; not "> |k'_1| \approx 1300" as you said in your first email.

If we use q=2^20 to allow all noises as designed in the specification, the result of your attack on my machine is like this, 0% success rate, run twice:

```
Attack against eMLE-Sig 2.0 keys
--------------------------------
Precompute key-independent reduction:
| ...done
Instance 1/25:
| x1 = (0, 4, 3, -4, 0, -1, 0, 2, 2, 4, 4, -3, -3, -4, -4, ...)
| x2 = (2, 0, -1, 3, -1, 0, 0, 2, 3, -3, 0, -2, 4, 0, -2, ...)
K0:  6975 5096
| x1 recovery failed
K0:  6975 5096
| x2 recovery failed
Instance 2/25:
| x1 = (4, 0, -1, 4, 4, 2, -4, -1, 3, -2, -4, -1, -4, 4, 1, ...)
| x2 = (2, 4, 2, -4, 4, -2, -3, -4, -3, 1, -1, 2, 3, -4, -1, ...)
K0:  6975 5096
| x1 recovery failed
K0:  6975 5096
| x2 recovery failed
Instance 3/25:
| x1 = (-2, -2, 0, -3, -4, -1, 1, -1, -1, -1, -4, -4, -4, -4, -1, ...)
| x2 = (2, -2, -3, 0, -2, 3, -3, 4, 3, 1, 3, -3, -2, 1, 3, ...)
```

K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 4/25:
| x1 = (1, -3, 2, 4, -2, 0, 1, 3, 2, -4, -2, 1, 4, 3, -2, ...)
| x2 = (0, -2, 1, -4, -3, 0, -3, 2, -3, -2, 0, 1, -4, 3, -3, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 5/25:
| x1 = (4, -2, 4, -1, -2, -3, -3, 2, -1, 0, 3, -2, 0, 0, 3, ...)
| x2 = (0, -2, 4, -2, -2, -3, -2, 1, -2, -3, -3, -2, 0, 3, 1, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 6/25:
| x1 = (2, -3, 0, 4, 1, -4, 1, 3, 2, 3, -4, 0, -1, -1, 3, ...)
| x2 = (-3, 0, 4, -4, -4, -1, -1, 3, 2, 0, 4, -4, -4, -4, 0, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 7/25:
| x1 = (-3, -2, 1, -2, 0, 0, 2, 1, 4, 2, -4, -2, -2, 2, 2, ...)
| x2 = (2, -1, 4, -1, -1, -4, 3, 3, -4, 0, 4, 3, -1, -1, -4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 8/25:
| x1 = (-4, 1, -3, 4, 3, 0, -4, 3, 1, 1, 0, 2, -2, -1, 3, ...)
| x2 = (4, 2, -4, 3, 4, 4, 2, -3, 2, -2, 1, -1, -1, 0, -2, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 9/25:
| x1 = (-2, -1, 2, -2, -1, 1, 2, -1, -2, -2, 0, 4, 4, 1, 4, ...)
| x2 = (4, -1, -4, -4, -2, -3, 3, 1, -1, -3, 4, 3, -4, -2, -4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 10/25:
| x1 = (2, -4, 0, 0, 1, 3, 3, 1, 4, -4, 3, 2, 2, 1, -3, ...)
| x2 = (-1, -4, -1, 3, -2, -4, -3, 4, -1, -3, 2, -2, -4, 3, 1, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096

| x2 recovery failed
Instance 11/25:
| x1 = (3, -4, 4, -4, -1, 1, 3, 0, 4, -4, -4, 3, -1, 2, -4, ...)
| x2 = (0, -4, -2, -4, 1, 2, -1, -3, -4, 3, 3, 0, 3, -3, 4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 12/25:
| x1 = (-3, 1, 4, 2, 4, -4, 1, -4, -2, 4, 0, -2, -1, -3, 1, ...)
| x2 = (-1, 4, -1, 0, 3, -4, -2, -4, 3, -2, -3, 1, -1, 3, 0, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 13/25:
| x1 = (-4, -4, 2, -3, -1, 1, 0, -1, 3, 1, -3, -3, -2, -3, -4, ...)
| x2 = (1, 4, 1, 4, -2, -3, -2, -3, -3, -1, 2, 3, -1, 3, -1, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 14/25:
| x1 = (4, 1, -1, 1, 3, 1, 2, 4, -2, -4, -2, -1, 4, -1, 4, ...)
| x2 = (1, -3, -1, -3, 1, -4, 2, 3, -2, -4, -2, -1, 0, -2, -1, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 15/25:
| x1 = (3, 3, -2, 4, 2, -1, 3, 2, -1, -4, 0, -3, -2, 0, 4, ...)
| x2 = (-1, 0, -1, 4, -1, 3, -4, 3, -4, -3, 1, 2, -2, -1, -4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 16/25:
| x1 = (-4, 4, 2, 3, -3, 1, -1, -3, 4, -2, -2, -1, 4, 3, -4, ...)
| x2 = (1, 0, 1, -4, 4, 2, -1, -2, -1, 2, 3, -1, -4, 3, 2, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 17/25:
| x1 = (-3, 0, 1, 2, 2, -2, 1, -1, 1, -1, -1, 4, -1, -4, -2, ...)
| x2 = (-4, 4, 2, 1, -4, -1, -4, 0, -1, 1, -4, 4, -1, 2, 4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 18/25:
| x1 = (-2, 1, -1, -4, 3, -4, -2, 3, 3, 1, -4, -2, 0, 2, 2, ...)

| x2 = (-4, 3, -3, 3, 0, 4, 3, 3, -4, 2, 4, 0, -3, 4, -3, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 19/25:
| x1 = (-2, 4, -3, -1, 2, -1, -4, 1, 2, 1, 4, 4, 2, -3, -4, ...)
| x2 = (0, -1, -2, 0, -1, 0, 4, -4, 1, 1, 2, -2, -3, 3, 0, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 20/25:
| x1 = (0, 4, 0, -3, -3, 2, 2, -1, -3, -2, -3, -4, 2, 4, 0, ...)
| x2 = (-2, -4, 3, 2, 2, -4, -4, -1, 3, 3, 3, 2, -1, 2, -4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 21/25:
| x1 = (4, 2, -3, 3, -2, 2, -3, 2, -4, -1, 4, 3, 1, 0, 2, ...)
| x2 = (4, -2, 0, 1, -2, 2, -1, 3, 3, 0, 3, 0, -4, 4, 2, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 22/25:
| x1 = (-1, 1, -1, 4, -3, -1, -2, -3, 3, -2, 3, 2, 2, -3, 0, ...)
| x2 = (-1, -1, 3, -2, 1, 2, -2, -3, 0, -3, -3, 2, -1, 2, -4, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 23/25:
| x1 = (-3, 1, -4, -3, 0, -3, 3, 3, -2, 1, 4, 3, 4, 4, 2, ...)
| x2 = (2, 0, -4, -4, -4, 0, 3, 1, -1, -3, -2, 4, -3, -4, 2, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 24/25:
| x1 = (2, -4, 4, -3, -4, -1, 2, -2, 3, 0, 3, 1, -2, 0, -4, ...)
| x2 = (-1, 4, -3, 3, 4, -4, 3, -3, 4, -1, 1, 1, 4, -2, -1, ...)
K0: 6975 5096
| x1 recovery failed
K0: 6975 5096
| x2 recovery failed
Instance 25/25:
| x1 = (3, 3, -1, -3, -3, 3, -2, 4, 2, 4, -1, -1, -3, 1, 1, ...)
| x2 = (4, -2, 4, 4, -1, 1, -2, 2, 1, 1, 3, 2, -1, -3, -4, ...)
K0: 6975 5096
| x1 recovery failed

K0: 6975 5096
| x2 recovery failed
0/50 correct recoveries (0.0% success rate)

Regards,
Dongxi Liu

From: pqc-forum@list.nist.gov <pqc-forum@list.nist.gov> on behalf of Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Thursday, 20 July 2023 at 1:51 pm
To: pqc-comments@nist.gov <pqc-comments@nist.gov>
Cc: pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

For the sake of completeness again, you can find sample code that
implements the attack below in the following GitHub repository:

https://github.com/mti/attack_emle

For now, it targets the "NIST Level-I" parameter set, and recovers a
signing key from the corresponding verification key with fairly good
probability (>80%) in a few minutes. Larger parameters should be broken
by essentially the same attack code, but this hasn't been verified yet.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Tue, Jul 18, 2023 at 06:03:46AM +0200, Mehdi Tibouchi wrote:
> Dear eMLE-Sig 2.0 submitters, dear all,
>
> The eMLE-Sig scheme is a lattice-based scheme that seems to be a
> spiritual successor to the original Round 1 encryption candidate Compact
> LWE. The design principle of this type of schemes is essentially to say:
> “we will use as a trapdoor a lattice vector that isn't particularly
> short, so that there is no hope to recover it by lattice reduction, but
> that has some sort of additional hidden structure that still allows it to
> be used as a secret; moreover, since lattice attacks must fail, we will
> use a ridiculously small lattice dimension”.
>
> The underlying belief is that the additional hidden structure cannot be

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Thursday, July 20, 2023 6:22 AM
To: Liu, Dongxi (Data61, Marsfield)
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

I acknowledge that I had misunderstood the provided SageMath code, and the attack does not work as proposed. My apologies.

It might in fact be the case that the eMLE problem (in the sense of recovering x from h) is hard. If so, however, I am fairly convinced that you will not be able to build actual cryptographic schemes that are as hard to break as the problem itself.

In the case of eMLE-Sig, as I mentioned earlier, the scheme suffers from clear secret key leakage in signatures. Example code demonstrating this has now replaced the previous, incorrect attack on the following repository:

https://github.com/mti/attack_emle

It basically recovers the full signing key in the "NIST Level-I" parameter set with around 2,500,000 signature samples.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Thu, Jul 20, 2023 at 05:41:29AM +0000, Liu, Dongxi (Data61, Marsfield) wrote:

> Hi M. Tibouchi,
>
> Thanks for your analysis.
>
> Simply, your attack is NOT correct (at least for your current version), because you set q=256.
> Note that q is just used for security analysis in Sage code and it is not a parameter in the specification and in the c-language reference implementation. In Sage code, we test different values of q to do security analysis and determine the concrete security level. A smaller q means smaller amount of noise than designed is added. The hardness of eMLE 2.0 depends on both dimension and the amount of noises, a difference from lattice problem.

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Friday, July 21, 2023 3:12 AM
To: Mehdi Tibouchi
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

Your leakage attack works. Thanks for further analysis.

Regards,
Dongxi Liu

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Thursday, 20 July 2023 at 8:22 pm
To: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Cc: pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

I acknowledge that I had misunderstood the provided SageMath code, and the attack does not work as proposed. My apologies.

It might in fact be the case that the eMLE problem (in the sense of recovering x from h) is hard. If so, however, I am fairly convinced that you will not be able to build actual cryptographic schemes that are as hard to break as the problem itself.

In the case of eMLE-Sig, as I mentioned earlier, the scheme suffers from clear secret key leakage in signatures. Example code demonstrating this has now replaced the previous, incorrect attack on the following repository:

https://github.com/mti/attack_emle

It basically recovers the full signing key in the "NIST Level-I" parameter set with around 2,500,000 signature samples.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

From: 'Lorenz Panny' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Sunday, July 23, 2023 4:26 AM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear all,

here's software that computes eMLE-Sig 2.0 secret keys from public keys within a few (single-core) hours per keypair at level 1:

https://yx7.cc/files/emle2- attack.tar.gz

The software implements the attack outlined by Mehdi Tibouchi in his comment from 2023-07-18 on this mailing list:

https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/zas5PLiBe6A/m/EVmNzzglBQAJ

To find the short target vector, the code runs progressive BKZ tours up to block size 90 using the G6K library. On average, a block size around 80 seems to suffice.

I've tested the software against the 100 public keys from the KATs.
It successfully recovered 83 secret keys, which have been confirmed capable of forging signatures, and failed in 17 cases for various non-fundamental reasons (higher precision required for fplll not to crash, or stronger reduction required to find the target vector).

To run the attack, a working setup of SageMath version 10.0 with the G6K library is required. For convenience, a Dockerfile to run such an environment is included. Thus, a given level-1 public key can be attacked by simply running

```
./run.sh A821C296...BF8A0ABB
```

where A821C296...BF8A0ABB is the public key (given in the format of the KATs). After it has finished, the resulting secret key can be tested for validity by running

```
./check.py A821C296...BF8A0ABB 000201FC...2760238D
```

where 000201FC...2760238D is the secret key found by the attack.

Best,
Lorenz

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Tuesday, August 1, 2023 12:49 AM
To: Lorenz Panny; pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi Lorenz, Hi M. Tibouchi,

Thanks for your attacks.
eMLE-Sig 2.0 has been revised to address the attacks.

The major revision is summarized below. Welcome further analysis (from anyone).

==========

1. Leakage attack

[current] s = x1*c1 + x2*c2 + y
[revised] s = x1*c1 + x2*c2 + (c1+c2)*y

The leakage attack now finds x1-x2 (checked with Tibouchi's code), no longer individual x1 and x2. So one of x1 or x2 has to be guessed for being able to forge signatures. Together with the revision below, there are about 380 bits to guess for level 1.

2. Lattice reduction attack

The revision is to aggregate public keys and share a new secret z across two eMLE values in a public key.

—Private key
[current] x1, x2
[revised] (x1, x1'), (x2, x2'), z

—Public key
[current] h1 = eMLE(G, x1, G[1]), h2 = eMLE(G, x2, G[1])
[revised] h1 = eMLE(G, x1, G[1]+z)+ eMLE(G', x1', 0), h2 = eMLE(G, x2, G[1]+z) + eMLE(G', x2', 0)

G[1]+z is embedded into all layers. Elements of z are from 0 to p[2]-1.
The shared z can be removed by h1-h2. But by this, at the best case the attack recovers x1-x2, and x1'-x2' (the same result as leakage attack). One of (x1, x1') or (x2, x2') needs to be guessed.

—Signing
[current] (u, s), where u = eMLE(G, y, c1'+c2'), s = x1*c1 + x2*c2 + y
[revised] (u, s, s'), where u = eMLE(G, y, c1'+c2'-z) + eMLE(G', y', 0),
s = x1*c1 + x2*c2 + (c1+c2)*y,
s' = x1'*c1 + x2'*c2 + (c1+c2)*y'

3. Sizes of public key and signature for the revised version

p[2] changes from 2^26 to 2^22. New sizes are listed below (bigger signature, but smaller pk).

[level 1] pk: 2*22 * 64/8 = 352 bytes, sig: (22 + 2*9)*64/8 = 320 bytes
[level 3] pk: 2*24 * 96/8 = 576 bytes, sig: (24 + 2*10)*96/8 = 528 bytes
[level 5] pk: 2*26 * 128/8 = 832 bytes, sig: (26 + 2*10)*128/8 = 736 bytes

The revised sage implementation is available at:
https://gitlab.com/raykzhao/emle-sig/-/blob/main/Extra/sage-code/impl-gen.sage
https://gitlab.com/raykzhao/emle-sig/-/blob/main/Extra/sage-code/correctness.sage

==========

Regards
Dongxi Liu

From: 'Lorenz Panny' via pqc-forum <pqc-forum@list.nist.gov>
Date: Sunday, 23 July 2023 at 6:25 pm
To: pqc-comments@nist.gov <pqc-comments@nist.gov>
Cc: pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear all,

here's software that computes eMLE-Sig 2.0 secret keys from public
keys within a few (single-core) hours per keypair at level 1:

https://yx7.cc/files/emle2-attack.tar.gz

The software implements the attack outlined by Mehdi Tibouchi in his
comment from 2023-07-18 on this mailing list:

https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/zas5PLiBe6A/m/EVmNzzglBQAJ

To find the short target vector, the code runs progressive BKZ tours
up to block size 90 using the G6K library. On average, a block size
around 80 seems to suffice.

I've tested the software against the 100 public keys from the KATs.
It successfully recovered 83 secret keys, which have been confirmed
capable of forging signatures, and failed in 17 cases for various
non-fundamental reasons (higher precision required for fplll not to
crash, or stronger reduction required to find the target vector).

To run the attack, a working setup of SageMath version 10.0 with the
G6K library is required. For convenience, a Dockerfile to run such
an environment is included. Thus, a given level-1 public key can be
attacked by simply running

```
./run.sh A821C296...BF8A0ABB
```

where A821C296...BF8A0ABB is the public key (given in the format of
the KATs). After it has finished, the resulting secret key can be
tested for validity by running

```
./check.py A821C296...BF8A0ABB 000201FC...2760238D
```

where 000201FC...2760238D is the secret key found by the attack.

From: pqc-forum@list.nist.gov on behalf of Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Wednesday, August 2, 2023 2:13 AM
To: Liu, Dongxi (Data61, Marsfield)
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

I find the description of the revision below a bit difficult to follow, so pardon me if I misunderstand, but I fail to see how the changes address the leakage attack at all.

I hadn't explained the attack in much details before, so let me describe it on the original scheme. Signatures in the original submission contain an element of the form:

$$s = x_1 \cdot c_1 + x_2 \cdot c_2 + y,$$

where c_1 and c_2 are known outputs of a random oracle, y is a random mask (which is unknown), and x_1, x_2 are the secret key elements. All variables are elements of the "convolution" ring R = Z[x]/(x^n-1).

The coefficients of c_1 and c_2 (in the polynomial basis of R) are uniformly and independently distributed in {0, 1, 2, 3}. The coefficients of y have a somewhat more complicated distribution, but they are i.i.d.
(and independent of c_1, c_2 thanks to the random oracle).

By the law of large numbers, if we generate many signatures and average everything out, the resulting value will converge towards the
expectation:

$$E[s] = x_1 \cdot E[c_1] + x_2 \cdot E[c_2] + E[y].$$

In this equation, the expectations of E[c_1] and E[c_2] are the same and easy to compute: they're the ring element with all coefficients equal to the mean of the uniform distribution on {0, 1, 2, 3}, namely (3/2)w where w = (1,1,...,1) = 1 + x + ⋯ + x^{n-1}. Similarly, since the coefficients of y are i.i.d., E[y] is some multiple mu·w for some real number mu (which we can estimate, but it doesn't matter).

If instead I restrict attention to signatures such that the first (i.e.,
constant) coefficient of c_1 is, say, 3, everything remains the same, except that E[c_1] in now (3, 3/2, 3/2, ..., 3/2) = 3/2(1+w). Thus, averaging many such values of s yields a value close to:

$$e_3 = E[s \mid c_{1,0} = 3] = (3/2)\, x_1 + w \cdot ((3/2)x_1 + (3/2)x_2 + mu).$$

Now, w is a zero divisor in R, with for example z·w = 0 for z = 1 - x = (1,-1,0,...,0). Thus, we can compute:

$$z \cdot e_3 = (3/2)\, z \cdot x_1 + z \cdot w \cdot ((3/2)x_1 + (3/2)x_2 + mu) = (3/2)\, z \cdot x_1,$$

and from that, recovering x_1 is easy since it suffices to guess a single coefficient in {-4,...,4} to recover the whole vector.

Ok, so now, how does the proposed change, namely:

> s = x1*c1 + x2*c2 + (c1+c2)*y,

affect this attack? Not much as far as I can tell from the description alone. By independence, we now have:

E[s] = x_1·E[c_1] + x_2·E[c_2] + (E[c_1] + E[c_2]) · E[y],

and similarly for conditional expectations, so the same trick should reveal x_1, and not just x_1-x_2 as claimed below. Therefore, I am curious about the following statement:

> The leakage attack now finds x1-x2 (checked with Tibouchi's code), no
> longer individual x1 and x2.

Please share the code used to arrive at this conclusion. (It is not practical to test anything, especially not statistical attacks requiring millions of signatures, based on the toy SageMath implementation alone, which is slow and clunky).

Regarding the lattice attack, an updated version of the specification would be appreciated, as well as a clarification of the security claims.
To be clear, are you still claiming hardness for some underlying "eMLE problem", or are you just saying that recovering the three unknowns x1, x1' and z from the sole datum of h1 is difficult? (Depending on the parameters, the latter might conceivably be information theoretically hard, but in that case, again, there is no hope of building cryptographic schemes from it).

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Tue, Aug 01, 2023 at 04:48:40AM +0000, Liu, Dongxi (Data61, Marsfield) wrote:

> Hi Lorenz, Hi M. Tibouchi,
>
> Thanks for your attacks.
> eMLE-Sig 2.0 has been revised to address the attacks.
>
> The major revision is summarized below. Welcome further analysis (from anyone).
>
> ==========
>
> 1. Leakage attack
>

From: pqc-forum@list.nist.gov on behalf of Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Wednesday, August 2, 2023 4:56 PM
To: Mehdi Tibouchi
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

> Please share the code used to arrive at this conclusion.

Here are the files:
https://gitlab.com/raykzhao/emle-sig/-/blob/main/Extra/sage-code/leakage/impl.c
https://gitlab.com/raykzhao/emle-sig/-/blob/main/Extra/sage-code/leakage/test_attack.c

[impl.c] the sign function is revised to implement "x1*c1 + x2*c2 + (c1+c2)*y", with y's elements limited from -4 to 4.

[test_attack.c] your file, with modification to recover also x2zrec, and then print x1z, x2z, x1zrec, x2zrec, x1z-x2z, x1zrec – x2zrec.

You just put these two files into your package; all other steps are the same.
Thanks for your checking. I will answer your other questions after this.

Regards,
Dongxi

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Wednesday, 2 August 2023 at 4:12 pm
To: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear eMLE-Sig 2.0 submitters, dear all,

I find the description of the revision below a bit difficult to follow,
so pardon me if I misunderstand, but I fail to see how the changes
address the leakage attack at all.

I hadn't explained the attack in much details before, so let me describe
it on the original scheme. Signatures in the original submission contain
an element of the form:

$$s = x_1 \cdot c_1 + x_2 \cdot c_2 + y,$$

where c_1 and c_2 are known outputs of a random oracle, y is a random
mask (which is unknown), and x_1, x_2 are the secret key elements. All

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Wednesday, August 2, 2023 10:08 PM
To: Liu, Dongxi (Data61, Marsfield)
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Liu,

Thanks for the modified implementation. A corresponding attack has been pushed to the following branch of my repository:

https://github.com/mti/attack_emle/tree/revisedscheme20230822

As you can see, and in accordance with the analysis I posted yesterday, the attack is exactly the same as before, except that it requires fewer signatures for full recovery: about 350,000 now vs. 2,500,000 originally (having a centered y makes convergence faster).

I suspect that the convergence failure you were observing was due to not updating the random seed between successive calls to the sign() function, which was presumably causing repeated randomness in generated signatures.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Wed, Aug 02, 2023 at 08:56:14PM +0000, Liu, Dongxi (Data61, Marsfield) wrote:

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Wednesday, August 2, 2023 11:25 PM
To: Mehdi Tibouchi
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

Your attack works. Thanks for checking.

Regards,
Dongxi Liu

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Thursday, 3 August 2023 at 12:08 pm
To: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Liu,

Thanks for the modified implementation. A corresponding attack has been
pushed to the following branch of my repository:

https://github.com/mti/attack_emle/tree/revisedscheme20230822

As you can see, and in accordance with the analysis I posted yesterday,
the attack is exactly the same as before, except that it requires fewer
signatures for full recovery: about 350,000 now vs. 2,500,000 originally
(having a centered y makes convergence faster).

I suspect that the convergence failure you were observing was due to not
updating the random seed between successive calls to the sign() function,
which was presumably causing repeated randomness in generated signatures.

Best regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Wed, Aug 02, 2023 at 08:56:14PM +0000, Liu, Dongxi (Data61, Marsfield) wrote:

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Thursday, August 3, 2023 8:46 AM
To: Mehdi Tibouchi
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

Based on your observation "not updating the random seed", the following solution passes leakage attack.

In the attacked version, we have s = c1*x1+c2*x2 + (c1+c2)*y, where y is randomly sampled. Now we let y = y0 + y1, where y0 is randomly sampled, but y1 is selected from a small number of cases (4 in the test), which can be sampled as part of the private key or derived from private key. y1 simulates the effect of "not updating the random seed".

Here is the updated implementation file:
https://gitlab.com/raykzhao/emle-sig/-/blob/main/Extra/sage-code/leakage-v1/impl.c

Attack file is the same.
Thanks for your observation and further checking.

Regards,
Dongxi Liu

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Date: Thursday, 3 August 2023 at 1:25 pm
To: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Hi M. Tibouchi,

Your attack works. Thanks for checking.

Regards,
Dongxi Liu

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Thursday, 3 August 2023 at 12:08 pm
To: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Liu,

Thanks for the modified implementation. A corresponding attack has been
pushed to the following branch of my repository:

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Sent: Friday, August 4, 2023 2:59 AM
To: Liu, Dongxi (Data61, Marsfield)
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Liu,

I don't think the proposal here is sufficiently fleshed out to evaluate (and I suspect a detailed analysis might not be a productive use of anyone's time), but I can still make a few quick remarks.

The same leakage attacks exactly recovers the following elements:

x1 + y* and x2 + y*

where y* = E[y] is the average of the choices for y1. Now, I strongly suspect that those values are already sufficient to forge signatures, but since the provided verification algorithm does not match the signature generation, it is difficult to test for sure. In any case, however, one can provide an upper bound on how hard it would be to recover y*, and hence the key, given the above.

Since x1, x2 has integer coefficients, and y* coefficients in (1/4)Z, one exactly learns the fractional part of y* from the above, and the conditional entropy of each coefficient knowing the fractional part is about 1.5 bits. The conditional entropy knowing x1+y* and x2+y* is again significantly lower (for example, if one coefficient of either is equal to 6, one learns for sure that the corresponding coefficient of y* is 2); I haven't done the exact computation, but it might be around 1 bit per coefficient or less. Overall, a trivial enumeration attack on the n=64 parameter should therefore take time 2^64.

Moreover, one should be able to carry out a meet-in-the-middle attack on the average of u to further cut the bit complexity in half.

So I am fairly confident that the proposal remains insecure.

Regards,

--
M. Tibouchi
<mehdi.tibouchi@normalesup.org>
http://www.normalesup.org/~tibouchi/
****
NTT Social Informatics Laboratories
Abe Research Laboratory
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan.
****

On Thu, Aug 03, 2023 at 12:46:28PM +0000, Liu, Dongxi (Data61, Marsfield) wrote:

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Saturday, February 17, 2024 2:26 AM
To: Mehdi Tibouchi
Cc: Lorenz Panny; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Tibouchi,

Thanks for your time and insight to analyse eMLE-Sig last year. Your analysis is valid.

1. eMLE-Sig has been thoroughly updated to address all attacks. The updated document, sage implementation, and tests are included in the following folder. No c implementation is available yet.

   https://gitlab.com/raykzhao/emle-sig/-/tree/main/Extra/sage-code/Updates2024Feb

2. Addressing leakage attack
   The updated signature still incudes the component s = c1*x1 + c2*x2 + ... But in the updated scheme, c1[i] and c2[i] are entangled.
   c1[i] takes the value either 0 or 1. If c1[i] =0, then c2[i] must be 1. If c1[i] =1, then c2[i] = -1.
   More detailed analysis is in the Section 4.1 of the PDF document in the above folder, with leakage.sage to confirm the analysis there.

3. Addressing lattice-based attack from the following aspects
   The dimension parameter n is increased from 64 to 128 for the first security level; a new secret z shared between h1 and h2 requires h1 and h2 must be embedded in one lattice to attack, leading to a bigger lattice to reduce than before even for the same n.
   The modulus p[d-1] decreases from 2^26 in the submitted version to 6083 in the new version to make a more dense system. More details are in the PDF document.

4. Sizes of signatures and public keys
   The signature sizes become bigger than the submitted version.
   Level 1: pk size = 416, sig size = 576 bytes
   Level 3: pk size = 672, sig size = 936 bytes
   Level 5: pk size = 896, sig size = 1280 bytes

Welcome any comments and questions!

Regards,
Dongxi Liu

From: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Date: Friday, 4 August 2023 at 4:59 pm
To: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Liu,

From: Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Sent: Tuesday, April 2, 2024 8:12 PM
To: pqc-comments
Cc: pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear All,

The c implementation of updated eMLE-Sig 2.0 is available at: https://gitlab.com/raykzhao/emle-sig
The performance evaluation is included in the updated document here:
https://gitlab.com/raykzhao/emle-sig/-/tree/main/Extra/sage-code/Updates2024Feb

Thanks for comments.

Regards,
Dongxi Liu

From: pqc-forum@list.nist.gov <pqc-forum@list.nist.gov> on behalf of Liu, Dongxi (Data61, Marsfield) <Dongxi.Liu@data61.csiro.au>
Date: Saturday, 17 February 2024 at 6:26 pm
To: Mehdi Tibouchi <mehdi.tibouchi@normalesup.org>
Cc: Lorenz Panny <l.s.panny@tue.nl>, pqc-comments@nist.gov <pqc-comments@nist.gov>, pqc-forum@list.nist.gov <pqc-forum@list.nist.gov>
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: eMLE-Sig 2.0

Dear Dr. Tibouchi,

Thanks for your time and insight to analyse eMLE-Sig last year. Your analysis is valid.

1. eMLE-Sig has been thoroughly updated to address all attacks. The updated document, sage implementation, and tests are included in the following folder. No c implementation is available yet.

https://gitlab.com/raykzhao/emle-sig/-/tree/main/Extra/sage-code/Updates2024Feb

2. Addressing leakage attack

The updated signature still incudes the component s = c1*x1 + c2*x2 + ... But in the updated scheme, c1[i] and c2[i] are entangled.
c1[i] takes the value either 0 or 1. If c1[i] =0, then c2[i] must be 1. If c1[i] =1, then c2[i] = -1.
More detailed analysis is in the Section 4.1 of the PDF document in the above folder, with leakage.sage to confirm the analysis there.

3. Addressing lattice-based attack from the following aspects

The dimension parameter n is increased from 64 to 128 for the first security level; a new secret z shared between h1 and h2 requires h1 and h2 must be embedded in one lattice to attack, leading to a bigger lattice to reduce than before even for the same n.
The modulus p[d-1] decreases from 2^26 in the submitted version to 6083 in the new version to make a more dense system. More details are in the PDF document.