From: Charles Bouillaguet <charles.bouillaguet@lip6.fr>
Sent: Tuesday, July 18, 2023 2:42 AM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear all,

This messages argues that the parameters chosen by the designers of Biscuit are a bit too small.

Biscuit relies on the hardness of the so-called "PowAff2" problem, namely to solve polynomial systems of a special kind. The secret key contains a secret vector s and the public key contains random polynomials f_1, ..., f_m along with a vector t such that t_i = f_i(s). Signatures are non-interactive proofs of knowledge of s. Solving the corresponding polynomial system reveals the secret key.

To make the scheme more efficient, the polynomials have a special shape f_i = w_i + u_i * v_i, where the w_i, u_i and v_i are affine forms. The point is that evaluating f_i requires a single multiplication (by a non-constant).

These special polynomial systems (product of two affine forms) are easier to solve than arbitrary quadratic systems. For instance, the designers of Biscuit show a simple algorithm that works in q**(0.75*n) operations, where n is the number of variables. No such simple algorithms are known in general.

However, here is another simple "crossbred-style" algorithm that requires only q**(0.5n) operations:
1) perform a linear change of the n variables y == Mx that yields v_i(y) = y_i + cst for (0 <= i < n)
2) guess the first n/2 variables
3) the first n/2 quadratic equations become affine in the remaining n/2 variables.
4) solve the linear system in the remaining n/2 variables
5) rinse, repeat.

This comes down to a loop with q**(n/2) iterations, and each iteration requires O(n^3) operations.

This implies that the secret key can be recovered in 2**174 and 2**236 iterations of this simple procedure for the parameter sets that are claimed to offer 192 and 256 bits of security, respectively.

There is a simple fix: increase the number of variables to 96 and 128.

However, the idea underlying this simple observation ("guess one

variable, get one linear equation for free") has other implications on the security against forgery attacks that are a bit less clear.

More details will appear on the eprint.

Best regards,
--
Charles BOUILLAGUET
Sorbonne Université
Campus Pierre & Marie Curie
LIP6/ALMASTY, Office 24-25/410
4 place Jussieu, 75252 Paris Cedex 05
homepage: https://www-almasty.lip6.fr/~bouillaguet

From: pqc-forum@list.nist.gov on behalf of Charles Bouillaguet <charles.bouillaguet@gmail.com>
Sent: Wednesday, July 19, 2023 3:14 AM
To: pqc-comments
Cc: pqc-forum
Subject: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear all,

This messages argues that the parameters chosen by the designers of Biscuit are a bit too small.

Biscuit relies on the hardness of the so-called "PowAff2" problem, namely to solve polynomial systems of a special kind. The secret key contains a secret vector s and the public key contains random polynomials f_1, ..., f_m along with a vector t such that t_i = f_i(s).
Signatures are non-interactive proofs of knowledge of s. Solving the corresponding polynomial system reveals the secret key.

To make the scheme more efficient, the polynomials have a special shape f_i = w_i + u_i * v_i, where the w_i, u_i and v_i are affine forms. The point is that evaluating f_i requires a single multiplication (by a non-constant).

These special polynomial systems (product of two affine forms) are easier to solve than arbitrary quadratic systems. For instance, the designers of Biscuit show a simple algorithm that works in q**(0.75*n) operations, where n is the number of variables. No such simple algorithms are known in general.

However, here is another simple "crossbred-style" algorithm that requires only q**(0.5n) operations:
1) perform a linear change of the n variables y == Mx that yields v_i(y) = y_i + cst for (0 <= i < n)
2) guess the first n/2 variables
3) the first n/2 quadratic equations become affine in the remaining n/2 variables.
4) solve the linear system in the remaining n/2 variables
5) rinse, repeat.

This comes down to a loop with q**(n/2) iterations, and each iteration requires O(n^3) operations.

This implies that the secret key can be recovered in 2**174 and 2**236 iterations of this simple procedure for the parameter sets that are claimed to offer 192 and 256 bits of security, respectively.

There is a simple fix: increase the number of variables to 96 and 128.
Some extra margin is probably required as well, because the simple technique outlined above can be improved.

In addition, the idea underlying this simple observation ("guess one variable, get one linear equation for free") has other implications on the security against forgery attacks that are a bit less clear.

More details will appear on the eprint.

Best regards,
--
Charles BOUILLAGUET
Sorbonne Université
Campus Pierre & Marie Curie

From: Ludovic Perret <ludovic.perret@lip6.fr>
Sent: Sunday, July 23, 2023 3:49 PM
To: Charles Bouillaguet; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear Charles,

On behalf of the Biscuit team, we would like to thank you for your comments. We acknowledge that the approach described below is correct
and yields an algorithm whose complexity is O((n/2)^3q^(n/2)). In your computations, you did not included the cost of the linear algebra part.
Taking into account this additional factor:

- Cat I still reach the security level
- Cat III is only1 bit off (206.32 instead of 207)
- Cat V is 3 bits off (269.6 instead of 272)

Your approach improves our previous structured algorithm for solving the problem underlying Biscuit, PowAff2, which has a complexity of q**(0.75*n).
We will update the specification document to take into account your new algorithm and modify the parameters accordingly. As you mentioned, the fix is
rather minor and has a minimal impact on the signature sizes.

The parameters of Biscuit are derived taking into account two approaches : "ad-hoc" algorithms dedicated to Pow2 that use the structure of the equations and general-purpose algorithms
(such as Groebner bases, polynomial approximation, etc …). The latter type of algorithms were faster and used to derive the parameters.

We disagree on you statement that PowAff2 systems are computationally easier to solve than quadratic algorithms. It is of course correct to say that simple algorithms exist for PowAff2,
but this does not necessarily implies that such algorithms are faster than general-purpose algorithms. At this stage, your new ad-hoc algorithm has — in fact — roughly the same asymptotical
complexity as general-purpose algorithms.

Best Regards,

The Biscuit team,

------ Message d'origine ------
De "Charles Bouillaguet" <charles.bouillaguet@gmail.com>
À pqc-comments@nist.gov
Cc pqc-forum@list.nist.gov
Date 19/07/2023 09:13:37
Objet [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

> Dear all,
>
> This messages argues that the parameters chosen by the designers of Biscuit are a bit too small.
>
> Biscuit relies on the hardness of the so-called "PowAff2" problem, namely to solve polynomial systems of a special kind. The secret key contains a secret vector s and the public key contains random polynomials f_1, ..., f_m along

From: pqc-forum@list.nist.gov on behalf of Charles Bouillaguet <charles.bouillaguet@gmail.com>
Sent: Monday, July 24, 2023 6:19 PM
To: pqc-forum
Cc: Ludovic Perret; pqc-comments; Julia Sauvage
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear Ludovic, designers of the Biscuit teams, other members of the mailing-list,

> We disagree on you statement that PowAff2 systems are
> computationally easier to solve than quadratic
> algorithms. [...] At this stage, your new ad-hoc algorithm has
> — in fact — roughly the same asymptotical complexity as
> general-purpose algorithms.

Let us look at what happens over the field with 2 elements.

For arbitrary Boolean quadratic systems, the best algorithm known to mankind at this time, which is due to Dinur [1], requires 2^(0.6943*n) operations. The simple algorithm outlined in my previous message, which is specific to the "PowAff2" problem, requires 2^(0.5*n), and thus is exponentially better.

So, at least over the field with two elements, PowAff2 is much easier than the problem of solving random quadratic systems.

Regarding concrete security levels, I agree that the naive algorithm I outlined in my previous message was not enough to go beyond the pain threshold. However, it is easy to improve upon. Combining it with the (self-proclaimed) dumbest possible algorithm that beats brute force [2] yields the following:

1) perform a linear change of the n variables y == Mx that yields v_i(y) = y_i + cst for (0 <= i < n)
2') Set k := 0.5 * (n - sqrt(2*m - n + 1) + 1). Guess the first k variables
3') The first k quadratic equations become affine in the remaining n - k variables. Use these to eliminate k variables in the last m-k quadratic polynomials
[at this stage, m-k quadratic polynomials in n-2k variables remain]
4') Solve the resulting quadratic system by linearization (it has less than m-k monomials of degree <= 2).
5') Go back to step 2' until a solution is found

Step 4' costs O((m-k)^3) operations, which is 8 times more than before.
However, the number of iterations is decreased compared to my first message.
This results in:

instance	n	m	k	#monomials (1)	log2(#operations) (2)
biscuit128	64	67	29	27	130.2
biscuit192	87	90	40	35	175.4
biscuit256	118	121	54	65	234.0

(1) There are 1/2*(n-2*k + 3)*(n-2*k) non-constant monomials of degree <= 2 in n-2*k variables
(2) This is (#monomials)^3 * q^k

This beats the claimed bit-security levels given in Table 11 of the submission document by 30, 34 and 42 bits for biscuit128, biscuit192 and biscuit256, respecticely, if I'm not mistaken.

Note that these results are obtained using the simplest possible methods, that require no memory and just combine exhaustive search with small, dense linear algebra. I consider this to be an indication that the "PowAff2" problem is easier than arbitrary quadratic systems.

In any case, it seems likely that even better results will be obtained using more sophisticated techniques. As such, if I were you, I would be conservative in revising the parameters.

Best regards,

Charles Bouillaguet

[1] Itai Dinur. Improved algorithms for solving polynomial systems over GF(2) by multiple parity- counting. In Dániel Marx, editor, Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021, pages 2550–2564. SIAM, 2021.

[2] Charles Bouillaguet, Claire Delaplace, and Monika Trimoska. A simple deterministic algorithm for systems of quadratic polynomials over GF(2) . In Karl Bringmann and Timothy Chan, editors, 5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022, Virtual Conference, January 10-11, 2022, pages 285–296. SIAM, 2022.

On Sunday, July 23, 2023 at 9:49:34 PM UTC+2 Ludovic Perret wrote:

> Dear Charles,
>
> On behalf of the Biscuit team, we would like to thank you for your comments. We acknowledge that the approach described below is correct
> and yields an algorithm whose complexity is O((n/2)^3q^(n/2)). In your computations, you did not included the cost of the linear algebra part.
> Taking into account this additional factor:
>
> - Cat I still reach the security level
> - Cat III is only1 bit off (206.32 instead of 207)
> - Cat V is 3 bits off (269.6 instead of 272)
>
> Your approach improves our previous structured algorithm for solving the problem underlying Biscuit, PowAff2, which has a complexity of q**(0.75*n).
> We will update the specification document to take into account your new algorithm and modify the parameters accordingly. As you mentioned, the fix is
> rather minor and has a minimal impact on the signature sizes.
>
> The parameters of Biscuit are derived taking into account two approaches : "ad-hoc" algorithms dedicated to Pow2 that use the structure of the equations and general-purpose algorithms

From: pqc-forum@list.nist.gov on behalf of Ludovic Perret <ludovic.perret@lip6.fr>
Sent: Thursday, July 27, 2023 11:15 AM
To: pqc-comments; pqc-forum; Charles Bouillaguet
Cc: Julia Sauvage
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Charles,

Again, many thanks for the effort to analyze the security of Biscuit. The approach seems correct and we like the idea of using the hybrid approach.

Still, the complexities that you computed below are field operations. In our documents (Table 11), we provide bit operations (using the fact that a field operations cost (log(q))^2 bit operations). If you use the same unit for the comparison, we obtain the following complexities.

Set	k	num_mon	num_polys	n_bit_op
I	29	27	38	146.2646625064904
III	40	35	50	191.3878490508349
V	54	65	67	250.06710343908537

So, still Cat I parameters are above the security margin. Of course, we were planning to take into careful consideration of your comments and adapt the parameters accordingly.

Best Regards,

The Biscuit team

Le 25 juil. 2023 à 00:18, Charles Bouillaguet <charles.bouillaguet@gmail.com> a écrit :

Dear Ludovic, designers of the Biscuit teams, other members of the mailing-list,

> We disagree on you statement that PowAff2 systems are
> computationally easier to solve than quadratic
> algorithms. [...] At this stage, your new ad-hoc algorithm has
> — in fact — roughly the same asymptotical complexity as
> general-purpose algorithms.

Let us look at what happens over the field with 2 elements.

For arbitrary Boolean quadratic systems, the best algorithm known to mankind at this time, which is due to Dinur [1], requires 2^(0.6943*n) operations. The simple algorithm outlined in my previous message, which is specific to the "PowAff2" problem, requires 2^(0.5*n), and thus is exponentially better.

So, at least over the field with two elements, PowAff2 is much easier than the problem of solving random quadratic systems.

From: pqc-forum@list.nist.gov on behalf of Charles Bouillaguet <charles.bouillaguet@gmail.com>
Sent: Friday, July 28, 2023 2:40 PM
To: pqc-forum
Cc: Ludovic Perret; Julia Sauvage; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear Ludovic,

> Still, the complexities that you computed below are field operations. In our
> documents (Table 11), we provide bit operations (using the fact that a field
> operations cost (log(q))^2 bit operations). If you use the same unit for the
> comparison, we obtain the following complexities.

The numbers you provide are taken by adding 16 to what I came up with. Indeed, each field operation requires O(log2(16)**2) == O(16) bit operations in some simplified abstract model... assuming that this has any meaning at all.

However, in the table I provided, the complexities are given as the base-2 logarithms of the number of operations. So, if you want to include the fact that a multiplication over GF(16) requires about 16 bit operations, then you should have added log2(16) == 4, and not 16.

In fact, the structure of PowAff2 instances has the following consequence (already stated in my first message): after a suitable change of variable, if you guess one variable, you get one extra linear equation for free, and this allows you to eliminate another variable. The point is that this shifts the tradeoff of the "hybrid method" in a more favorable way (or unfavorable way, depending on the point of view).

Starting with n variables and m polynomials, hereafter denoted as (n, m), the usual hybrid method consists in guessing k variables; this goes from (n, m) to (n-k, m).

With PowAff2, the strategy I described above goes from (n, m) to (n-2k, m-k). This is almost certainly more efficient asymptotically, even though I have not yet worked out the precise details yet.

In the following, I use the MQEstimator, as suggested in the submission document. I checked that it computes "bit complexities": it considers that a finite-field multiplication requires 2 * log2(q)**2 + log2(q) "gates".

Here is a table that shows the difference in efficiency between the "classic" hybrid method (the only one that is applicable to an arbitrary system of quadratic polynomials) and the "improved" hybrid method (applicable to PowAff2 systems). k and k' are the number of variables that are guessed in both case. The complexity is obtained by assuming that the subsequent system is solved with the F5 algorithm, and using MQEstimator to get its number of bit operations. The "claimed bit-security" is taken from table 11 of the submission document.

instance	claimed	n	m		k	Classic hybrid method	k'	Improved hybrid method	bit-security (*)				(log2 bit complexity)				(log2 bit complexity)
biscuit128	160	64	67		11	151.3	17	124.2
biscuit192	210	87	90		17	200.6	26	163.4
biscuit256	276	118	121		21	266.5	31	214.9

The bit complexity of the improved hybrid method (using the F5 algorithm) are always below what they should be, including for biscuit128, although only by a small margin (4 bits).

Best regards,
--
Charles Bouillaguet
On Thursday, July 27, 2023 at 5:15:08 PM UTC+2 Ludovic Perret wrote:

> Charles,
>
> Again, many thanks for the effort to analyze the security of Biscuit. The approach seems correct and we like the idea of using the hybrid approach.
>
> Still, the complexities that you computed below are field operations. In our documents (Table 11), we provide bit operations (using the fact that a field operations cost (log(q))^2 bit operations). If you use the same unit for the comparison, we obtain the following complexities.
>
> | Set | k | num_mon | num_polys | n_bit_op |
> |---|---|---|---|---|
> | I | 29 | 27 | 38 | 146.2646625064904 |
> | III | 40 | 35 | 50 | 191.3878490508349 |
> | V | 54 | 65 | 67 | 250.06710343908537 |
>
> So, still Cat I parameters are above the security margin. Of course, we were planning to take into careful consideration of your comments and adapt the parameters accordingly.
>
> Best Regards,
>
> The Biscuit team
>
> Le 25 juil. 2023 à 00:18, Charles Bouillaguet <charles.b...@gmail.com> a écrit :
>
> Dear Ludovic, designers of the Biscuit teams, other members of the mailing-list,

From: Fukang Liu <liufukangs@gmail.com>
Sent: Thursday, February 29, 2024 4:56 AM
To: pqc-comments
Cc: pqc-forum; Willi Meier; m.mahzoun@tue.nl
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit
Attachments: analysis_of_Biscuit.pdf

Dear All,

We would like to post our recent findings on the cryptanalysis of Biscuit, a candidate in NIST PQC competition.

The details of our attacks can be found in the attached file.

We also briefly summarize the main idea here.

The security of Biscuit relies on the hardness to solve a structured system of quadratic equations over F_{2^t}.

Specifically, given m quadratic equations f_i(x_1,...,x_n)=0 in n variables (x_1,...,x_n) over F_{2^t}, where m=n+k and k \in {2,3}, find the solution of this system.

Especially, each f_i is of the following form. For simplicity, we directly use f rather than f_i to make the description simpler.

f = d + \sum_{i=1}^{n} a_ix_i + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_ix_i).

Indeed, according to the property of the arithmetic operation over F_{2^t}, we always have

\forall x,y \in F_{2^t}, \forall i \in [0,t-1]: (x+y)^{2^i} = x^{2^i} + y^{2^i}.

Hence, we can have the following 4 equivalent equations for f:

0 = f = d + \sum_{i=1}^{n} a_ix_i + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_ix_i).

0 = f * (\sum_{i=1}^{n} b_ix_i) = (d + \sum_{i=1}^{n} a_ix_i) * (\sum_{i=1}^{n} b_ix_i) + (\sum_{i=1}^{n} b_i^2x_i^2) * (\sum_{i=1}^{n} c_ix_i).

0 = f * (\sum_{i=1}^{n} c_ix_i) = (d + \sum_{i=1}^{n} a_ix_i) * (\sum_{i=1}^{n} c_ix_i) + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_i^2x_i^2).

0 = f^2 =d^2 + \sum_{i=1}^{n} a_i^2x_i^2 + (\sum_{i=1}^{n} b_i^2x_i^2) * (\sum_{i=1}^{n} c_i^2x_i^2).

By introducing n variables (x_{n+1},...,x_{2n}) as follows:

\forall i \in [1,n]: x_{n+i} = x_i^2 ..... (*****)

we can have the following 4 quadratic equations in (x_1,....,x_{2n}) derived from f:

0 = f = d + \sum_{i=1}^{n} a_ix_i + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_ix_i).

0 = f * (\sum_{i=1}^{n} b_ix_i) = (d + \sum_{i=1}^{n} a_ix_i) * (\sum_{i=1}^{n} b_ix_i) + (\sum_{i=1}^{n} b_i^2x_{i+n}) * (\sum_{i=1}^{n} c_ix_i).

0 = f * (\sum_{i=1}^{n} c_ix_i) = (d + \sum_{i=1}^{n} a_ix_i) * (\sum_{i=1}^{n} c_ix_i) + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_i^2x_{i+n}).

0 = f^2 =d^2 + \sum_{i=1}^{n} a_i^2x_{i+n} + (\sum_{i=1}^{n} b_i^2x_{i+n}) * (\sum_{i=1}^{n} c_i^2x_{i+n}).

Hence, we can construct in total 4m quadratic equations from the m equations f_i=0 in 2n variables (x_1,...,x_{2n}).

In addition, by definition (*****) of the new variables, we have n extra quadratic equations.

Therefore, recovering the key of Biscuit is reduced to solving 4m+n quadratic equations in 2n variables.

With Gröbner basis, we could break all the parameters of the original and revised versions of Biscuit.

As a result, it seems that Biscuit has to be carefully revised again.

Best regards,
Fukang, Mairon and Willi

From: pqc-forum@list.nist.gov on behalf of Javier Verbel <javh10@gmail.com>
Sent: Thursday, February 29, 2024 9:56 AM
To: Mairon
Cc: pqc-forum; willimeier48@gmail.com; liufukangs@gmail.com
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit
Attachments: test_LMM_paper.py.zip

Dear Fukang, Mairon and Willi,

Thanks for showing interest in Biscuit's security.
I believe your analysis of the solving degree and the degree of regularity is mistaken.
It assumes that the resulting 4 * m + n equations are semiregular.
These equations are clearly not.

Find attached a code computing the Hilbert Series (HS) of your (homogenized) 4 * m + n equations from your modeling and the HS of semiregular sequence of the same dimensions.
For instance, for n=5, m = 7 and q= 256

The HS of semi-regular homogenized sequences of 4 * m + n equations and 2 *n variables is 1 + 11*t + 33*t^2 - 77*t^3 - 649*t^4 +O(t^5) so D_sol = 3,
while the HS in practice of your modeling is 1 + 11*t + 54*t^2 + 154*t^3 + 325*t^4 + 651*t^5 + 1176*t^6 + 1968*t^7 + 3105*t^8 + 4675*t^9 + 6776*t^10 + 9516*t^11 + 13013*t^12 + 17395*t^13 + 22800*t^14 so the system is far from been solved at degree 3.

You can reproduce this particular example by just executing sage test_lmm_paper.py

Best,
Javier Verbel (Speaking by my self)

El jue, 29 feb 2024 a las 14:30, Mairon (<mail@mahzoun.me>) escribió:

> Dear Biscuit team,
>
> We would like to post our recent findings on the cryptanalysis of Biscuit, a candidate in the NIST PQC competition.
>
> The details of our attacks can be found in the attached file.
>
> We also briefly summarize the main idea here.
>
> The security of Biscuit relies on the hardness to solve a structured system of quadratic equations over F_{2^t}.
>
> Specifically, given m quadratic equations f_i(x_1,...,x_n)=0 in n variables (x_1,...,x_n) over F_{2^t},
> where m=n+k and k \in {2,3}, find the solution of this system.
>
> Especially, each f_i is of the following form. For simplicity, we directly use f rather than f_i to make the description simpler.
>
> f = d + \sum_{i=1}^{n} a_ix_i + (\sum_{i=1}^{n} b_ix_i) * (\sum_{i=1}^{n} c_ix_i).

From: pqc-forum@list.nist.gov on behalf of Mairon <mail@mahzoun.me>
Sent: Thursday, February 29, 2024 12:27 PM
To: Javier Verbel
Cc: pqc-forum; willimeier48@gmail.com; liufukangs@gmail.com
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear Javier,

Thank you for your email. Indeed the resulting polynomial system is not semi-regular. We are using the degree of regularity as an upper bound for the solving degree of our polynomial system. Our experimental results (Table 3 on page 7) show that computing the Grobner basis of the polynomial system we define with 4m +n equations in 2n variables has a smaller solving degree than random quadratic equations with 4m + n equations in 2n variables, and can be solved in practical time for n up to 15. Therefore, solving the polynomial systems derived from PowAff2 is generally faster than a random (and empirically semi-regular) sequence.

For a sequence that is not semi-regular, the Hilbert series is not very meaningful to compute the solving degree. Yet, we can still find meaningful upper bounds, supported by experimental results, that the Hilbert series gives an upper bound for the solving degree of the system.

Some comments on the code you send: How you generate the system does not guarantee a solution, and Gröbner basis is {1}. If you fix that, you can verify that the system is indeed solvable at degree 3.

Best regards,

Mairon

On Thu, Feb 29, 2024 at 3:55 PM Javier Verbel <javh10@gmail.com> wrote:

> Dear Fukang, Mairon and Willi,
>
> Thanks for showing interest in Biscuit's security.
> I believe your analysis of the solving degree and the degree of regularity is mistaken.
> It assumes that the resulting 4 * m + n equations are semiregular.
> These equations are clearly not.
>
> Find attached a code computing the Hilbert Series (HS) of your (homogenized) 4 * m + n equations from your modeling and the HS of semiregular sequence of the same dimensions.
> For instance, for n=5, m = 7 and q= 256
>
> The HS of semi-regular homogenized sequences of 4 * m + n equations and 2 *n variables is 1 + 11*t + 33*t^2 - 77*t^3 - 649*t^4 +O(t^5) so D_sol = 3,
> while the HS in practice of your modeling is 1 + 11*t + 54*t^2 + 154*t^3 + 325*t^4 + 651*t^5 + 1176*t^6 + 1968*t^7 + 3105*t^8 + 4675*t^9 + 6776*t^10 + 9516*t^11 + 13013*t^12 + 17395*t^13 + 22800*t^14 so the system is far from been solved at degree 3.
>
> You can reproduce this particular example by just executing sage test_lmm_paper.py
>
> Best,

From: pqc-forum@list.nist.gov on behalf of Javier Verbel <javh10@gmail.com>
Sent: Saturday, March 2, 2024 10:08 PM
To: Mairon
Cc: pqc-forum; willimeier48@gmail.com; liufukangs@gmail.com
Subject: Re: [pqc-forum] Round 1 (Additional Signatures) OFFICIAL COMMENT: Biscuit

Dear Marion, Dear all

We disagree with the note's complexity statement, particularly the claimed degree of regularity/solving.

The Hilbert Series example in the previous email was to show that the resulting system is not semi-regula and that its degree of regularity is far from the one of a semi-regular sequence. Hence, it is unclear why the solving degree of your system (not semi-regular) is expected to be smaller than the degree of the regularity of a semi-regular sequence.

In order to minimize the traffic on the mailing list, we are happy to continue the discussion off-line and inform the list a second time.

Best regards,
Javier, on behalf of the Biscuit Team

El jue, 29 feb 2024 a las 21:26, Mairon (<mail@mahzoun.me>) escribió:

> Dear Javier,
>
> Thank you for your email. Indeed the resulting polynomial system is not semi-regular. We are using the degree of regularity as an upper bound for the solving degree of our polynomial system. Our experimental results (Table 3 on page 7) show that computing the Grobner basis of the polynomial system we define with 4m +n equations in 2n variables has a smaller solving degree than random quadratic equations with 4m + n equations in 2n variables, and can be solved in practical time for n up to 15. Therefore, solving the polynomial systems derived from PowAff2 is generally faster than a random (and empirically semi-regular) sequence.
>
> For a sequence that is not semi-regular, the Hilbert series is not very meaningful to compute the solving degree. Yet, we can still find meaningful upper bounds, supported by experimental results, that the Hilbert series gives an upper bound for the solving degree of the system.
>
> Some comments on the code you send: How you generate the system does not guarantee a solution, and Gröbner basis is {1}. If you fix that, you can verify that the system is indeed solvable at degree 3.
>
> Best regards,
>
> Mairon
>
> On Thu, Feb 29, 2024 at 3:55 PM Javier Verbel <javh10@gmail.com> wrote:
>> Dear Fukang, Mairon and Willi,
>>
>> Thanks for showing interest in Biscuit's security.
>> I believe your analysis of the solving degree and the degree of regularity is mistaken.
>> It assumes that the resulting 4 * m + n equations are semiregular.
>> These equations are clearly not.