From: Ward Beullens <WBE@zurich.ibm.com>
Sent: Tuesday, July 18, 2023 10:14 AM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC

Hi,

If I understand correctly, the verification algorithm of HPPC accepts if F(signature,public key) = Hash(message), for some function F, and for some hash function with only 128,192, or 256 bits output for security levels 1,3,or 5 respectively.

This means that one can do a chosen-message attack with roughly 2^64, 2^96 or 2^128 amount of work by finding a collision for H, or one can do a key-only universal forgery attack with roughly the same amount of work by finding claws between F and H.

Ward

From: ward@beullens.com
Sent: Tuesday, July 18, 2023 10:17 AM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC

Hi,

It looks like the verification algorithm of HPPC accepts if F(signature,public key) = Hash(message), for some function F, and for some hash function with only 128,192, or 256 bits of output for security levels 1,3,or 5 respectively.

This means that one can do a chosen-message attack with roughly 2^64, 2^96 or 2^128 of work by finding a collision for H, or one can do a key-only universal forgery attack with roughly the same amount of work by finding claws between F and H.

Ward

From: pqc-forum@list.nist.gov on behalf of Borja Gomez <borgomez026@gmail.com>
Sent: Wednesday, July 19, 2023 9:09 PM
To: pqc-forum
Cc: wa...@beullens.com; pqc-forum; pqc-comments
Subject: [pqc-forum] Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC

Hello Ward,

Thanks for commenting. As you state, an adversary can sign 2^64, 2^96 and 2^128 messages to obtain a collision P(x, v) = H(m_1) = H(m_2) (by the birthday paradox). The output of the hash function (SHA256) is truncated to 128 and 192 bits for HPPC128 - HPPC192.

In literature we can find other schemes that take collision resistance into consideration like: QUARTZ, Gui, GeMSS and Ultra-Short Signatures by Patarin et al.

From here multiple aspects to strengthen the scheme against collision attacks are devised:

1. Restrict message signing to no more than 2^64, 2^96 and 2^128 inputs. Once the threshold is passed, the collision attack is probable. Similar to the usage of symmetric algorithms under CBC mode.
2. Include a mode of operation as seen in literature (i.e Feistel-Patarin mode of operation).
3. Increase signature size, impacting performance which might require optimizing/rearming code.
4. Increase signing time by using a slower inversion procedure.

In my opinion, restricting message signing under the same key is enough to resist against collision attacks.

I'll be glad to hear your comments.

Borja

El martes, 18 de julio de 2023 a las 16:17:34 UTC+2, wa...@beullens.com escribió:

> Hi,
>
> It looks like the verification algorithm of HPPC accepts if F(signature,public key) = Hash(message), for some function F, and for some hash function with only 128,192, or 256 bits of output for security levels 1,3,or 5 respectively.
>
> This means that one can do a chosen-message attack with roughly 2^64, 2^96 or 2^128 of work by finding a collision for H, or one can do a key-only universal forgery attack with roughly the same amount of work by finding claws between F and H.
>
> Ward

From: Borja Gomez <borgomez026@gmail.com>
Sent: Thursday, July 20, 2023 9:28 AM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC
Attachments: hppc-fullspeed.nb

HI all,

I want to share with you a Mathematica code that generates the public key and a random message evaluation so the equations P(X) - Y = 0 are saved to a .txt file where the script gb2.py attemps to find the image vector X=(x_1,...,x_n). For that n equations x_i^2 - x_i = 0 are added to the system (semi-regular sequence) so we have a polynomial system of 2n quadratic equations on n variables.

Follow these steps to test the algorithm and to attempt to solve via Gröbner basis:

1. In Mathematica call GenSys(), Eval() and WriteEqs(). Last call will write eqs.txt file in the path you specify.
2. Then invoke gb2.py in the same folder where your eqs.txt lies.

Hope this helps to anyone interested on HPPC.

Regards,
Borja

From: Borja Gomez <borgomez026@gmail.com>
Sent: Thursday, July 20, 2023 10:04 AM
To: pqc-forum
Cc: Borja Gomez; pqc-forum; pqc-comments
Subject: Re: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC

Hi Ward,

Hoping you're doing great.

Based on my previous comment about the collision attack I can see two cases for signature forgery via collision.

1. P(x_1, v_1) = P(x_2, v_2) = H(m_1)
2. P(x_1,v_1) = H(m_1) = H(m_2)

In 1. we try to find a collision in the signing function to obtain two pairs (x_1, v_1) and (x_2, v_2) that verify to the same message m_1.
In 2. we try to find a collision in the hash function (SHA256) to obtain two messages m_1 and m_2 with the same digest, so the initial signature is valid for both messages.

It appears that I approached the case 1. in my comment where limiting message signing would suffice to stop collision in the signing function.
However case 2. has not been approached, yet. Using a slow hash function should suffice to guarantee resistance against "offline" collision attacks.

I'd like to know what you think about the options I gave on case 1. and 2.

Regards,
Borja

El jueves, 20 de julio de 2023 a las 15:27:58 UTC+2, Borja Gomez escribió:

> HI all,
>
> I want to share with you a Mathematica code that generates the public key and a random message evaluation so the equations P(X) - Y = 0 are saved to a .txt file where the script gb2.py attemps to find the image vector X=(x_1,...,x_n). For that n equations x_i^2 - x_i = 0 are added to the system (semi-regular sequence) so we have a polynomial system of 2n quadratic equations on n variables.
>
> Follow these steps to test the algorithm and to attempt to solve via Gröbner basis:
>
> 1. In Mathematica call GenSys(), Eval() and WriteEqs(). Last call will write eqs.txt file in the path you specify.
> 2. Then invoke gb2.py in the same folder where your eqs.txt lies.
>
> Hope this helps to anyone interested on HPPC.

From: Perlner, Ray A. (Fed)
Sent: Friday, July 21, 2023 3:44 PM
To: pqc-comments
Cc: pqc-forum
Subject: Round 1 (Additional Signatures) OFFICIAL COMMENT: HPPC
Attachments: HPPC_code.magma; HPPC_n_30.txt; HPPC_n_50.txt

Dear all,

We would like to report weaknesses in the polynomial system underlying HPPC.

This system was assumed to behave as a semi-regular one over GF(2) in the specification. However, our MAGMA experiments show a rather different behavior. More precisely, we observe 2n degree fall polynomials from degree 3 to degree 2 regardless of the value of d.

We believe that they are caused by the central map having 2-rank 2. This map is given by a univariate polynomial F(X) = l_1(X) \times l_2(l_1(X)), where l_1 and l_2 are linearized permutation polynomials. The number of degree falls may come from the fact that for the linearized polynomials A = l_1 and B = l_2 ° l_1, A*F and B*F have 2-rank 2. The conclusion follows from the same standard arguments as in [1], Equation (5).

The presence of degree falls at degree 3 does not necessarily mark the end of the Gröbner basis computation but it does show a weakness in the scheme:

- For instance, there could be the possibility of stopping the algorithm at this step and then exploiting these polynomials in a more astute way (this was for example the case in [1]).

- This may also pave the way for practical MinRank attacks. In addition to the Minors modeling mentioned in the specification, we can think of applying methods similar to the one against GeMSS in [2] (with target rank 2). More precisely, for the category 5 parameters, if we use b=1, the first step should involve solving (5 choose 3)*256 linear equations in (5 choose 2)*256 variables (concretely this should cost something like (2560)^2.8 = 2^32 bit operations). For the GeMSS parameters attacked in [2], the first step was always the most expensive, but even if the second step turns out to be the most expensive step, we are confident this attack will still be practical.

Please find attached our MAGMA code and two traces of its execution on n=30 and n=50 parameters where d=10 (note that d=10 and n=128 corresponds to security level I).

Sincerely,

Pierre Briaud, Maxime Bros, and Ray Perlner

[1] https://eprint.iacr.org/2020/1442.pdf

[2] https://eprint.iacr.org/2021/1677.pdf

(apologies if this is a double post. We tried posting through the website but it didn’t work immediately.)