From: 'Thomas Pornin' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Tuesday, February 13, 2024 2:46 PM
To: pqc-forum
Subject: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

The Falcon team, after discussing with NIST, envisions to modify the key pair generation implementation method by leveraging the techniques described in https://eprint.iacr.org/2023/290 and used in the HAWK scheme (HAWK is described on https://hawk-sign.info/ and was submitted to the additional round for post-quantum signatures). In a nutshell, the Falcon key pair generation process uses the following:

- The two polynomials f and g are generated with a fixed Gaussian distribution centred on zero. If the vector (f,g) has too large a norm, the process starts over.

- Polynomials F and G that fulfill the NTRU equation (fG - gF = q) are computed and reduced (with Babai's round-off algorithm).

- If suitable F and G cannot be found then the process starts over. A solution can be easily verified to fulfill the equation, so there is no risk of accepting "wrong" (F,G).

In general, for given (f,g), there can be several reduced (F,G) solutions and neither is "more correct" than any other. The initial Falcon proposal uses floating-point operations in the reduction process, which raises some implementation issues with various hardware platforms, and slightly impairs test vector reproducibility since different platforms might apply distinct rounding and fall into a different (F,G) solution. The implementation used in HAWK uses only integer computations, making it much easier to reproduce exactly on many software platforms; it is also faster and uses less RAM. As a consequence, though, it can reject some (f,g) pairs for which suitable (F,G) solutions mathematically exist but are not found by the implementation, leading to regeneration of a new (f,g) pair. The rejection rate is less than 29% (about 8.2% for Falcon-512, 28.5% for Falcon-1024), and thus cannot have an impact on security worse than log_2(1 - 0.29) bits, i.e. we may lose, theoretically, at worst 0.49 bits of security or so. No specific structure is known for key pairs that get rejected, but even if there were, it would not matter much for security.

The current implementation of the HAWK keygen uses the library code available at https://github.com/pornin/ntrugen , which also supports Falcon key pair generation directly. That library implements a second change, which is that the Gaussian distribution table used in the generation of (f,g) uses only 16 bits per entry, instead of 64 bits for the current Falcon code. This also slightly reduces the overall key pair generation time, by using fewer bits from the RNG. It should have no non-negligible impact on security.

The main effect of these changes is that for a given source seed, a different key pair is obtained; however, nothing in the rest of Falcon is modified, and the key pair generation methods are interoperable.

Note that we talk here about producing the (F,G) pair from (f,g); the generation of the Falcon tree is another process which can be considered to be a part of key pair generation, or to be instead a component of signature generation whose cost can be shared among distinct uses of the same key, since the tree depends only on the private key. The Falcon tree generation still uses floating-point operations.

While the proposed change is objectively good from a software engineering perspective (smaller, faster, more reproducible...) and seems innocuous for security, we would welcome any feedback from the community on this subject.

--Thomas Pornin, for the Falcon team

From: pqc-forum@list.nist.gov on behalf of Blumenthal, Uri - 0553 - MITLL <uri@ll.mit.edu>
Sent: Tuesday, February 13, 2024 3:06 PM
To: Thomas Pornin; pqc-forum
Subject: Re: [EXT] [pqc-forum] [FALCON OFFICIAL] Keygen implementation

I welcome this change, and would like to see all the floating-point operations gone, making FALCON more similar to HAWK in that sense.

Thanks!
--
V/R,
Uri

There are two ways to design a system. One is to make it so simple there are obviously no deficiencies.
The other is to make it so complex there are no obvious deficiencies.
- C. A. R. Hoare

From: 'Thomas Pornin' via pqc-forum <pqc-forum@list.nist.gov>
Reply-To: Thomas Pornin <pornin@bolet.org>
Date: Tuesday, February 13, 2024 at 14:46
To: pqc-forum <pqc-forum@list.nist.gov>
Subject: [EXT] [pqc-forum] [FALCON OFFICIAL] Keygen implementation

The Falcon team, after discussing with NIST, envisions to modify the key pair generation implementation method by leveraging the techniques described in https://eprint.iacr.org/2023/290 and used in the HAWK scheme (HAWK is described on https://hawk-sign.info/ and was submitted to the additional round for post-quantum signatures). In a nutshell, the Falcon key pair generation process uses the following:

- The two polynomials f and g are generated with a fixed Gaussian distribution centred on zero. If the vector (f,g) has too large a norm, the process starts over.

- Polynomials F and G that fulfill the NTRU equation (fG - gF = q) are computed and reduced (with Babai's round-off algorithm).

- If suitable F and G cannot be found then the process starts over. A solution can be easily verified to fulfill the equation, so there is no risk of accepting "wrong" (F,G).

In general, for given (f,g), there can be several reduced (F,G) solutions and neither is "more correct" than any other. The initial Falcon proposal uses floating-point operations in the reduction process, which raises some implementation issues with various hardware platforms, and slightly impairs test vector reproducibility since different platforms might apply distinct rounding and fall into a different (F,G) solution. The implementation used in HAWK uses only integer computations, making it much easier to reproduce exactly on many software platforms; it is also faster and uses less RAM. As a consequence, though, it can reject some (f,g) pairs for which suitable (F,G) solutions mathematically exist but are not found by the implementation, leading to regeneration of a new (f,g) pair. The rejection rate is less than 29% (about 8.2% for Falcon-512, 28.5% for Falcon-1024), and thus cannot have an impact on security worse than log_2(1 - 0.29) bits, i.e. we may lose, theoretically, at worst 0.49 bits of security or so. No specific structure is known for key pairs that get rejected, but even if there were, it

From: pqc-forum@list.nist.gov on behalf of Simon Hoerder <simon@hoerder.net>
Sent: Tuesday, February 13, 2024 3:52 PM
To: pqc-forum
Subject: Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

Hi,

if you can eliminate all floating point operations from Falcon, I'd be very interested in the proposal.

If only some but not all of the floating points are removed, I struggle to see appeal though. You'd still be in a situation where no established leakage analysis and side-channel countermeasures exist for a key part of the algorithm and deployment options would be as limited as for the current Falcon spec.

Please correct me if I'm wrong but the changes do sound fairly extensive. I would very much encourage to produce a full spec and reference implementation for the proposed changes and then allow a generous amount of time (1 to 2 years?) for study and analysis. Ideally, I'd like to see a 3rd party security review, a HW implementation and research into side-channel security (at least one constant time implementation, preferably also a masked implementation) and fault resistance before a final FIPS document is released.

Even with Dilithium the hint unpacking issue was discovered by Mike Hamburg only after the end of the official comment period. Since we
already* have Dilithium and Sphincs+ I believe there's no need to rush; better make sure we get this 100% right and provide developers with the knowledge to roll out high quality implementations.

Best,
Simon

*) The final FIPS documents are not published yet but close enough. For me that counts as already having the algorithms.

From: pqc-forum@list.nist.gov on behalf of Christine Cloostermans <cvvrede@gmail.com>
Sent: Friday, February 16, 2024 2:31 AM
To: Simon Hoerder
Cc: pqc-forum
Subject: Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

Hi all,

Let me start by saying that we do agree that removing FP operations is a good general direction for improvement on embedded platforms.

However, we agree with Simon here: we do not see a (significant) benefit from removing the (partial) FP operations of the key generation of Falcon.

Key generation is in many cases not even performed on an embedded device.

In many cases, keys are provisioned beforehand and therefore this change would not help with easing Falcon implementation.

Even in the cases where key generation would be performed on chip, the FP operations that remain are still a question mark for side-channel protection.

Without removing FP operations from signing (that does not impact performance as grossly as FP to Integer conversion does), we do not see a clear benefit for embedded context.

Cheers, on behalf of the NXP PQC team,

Christine

Op di 13 feb 2024 om 21:52 schreef Simon Hoerder <simon@hoerder.net>:

> Hi,
>
> if you can eliminate all floating point operations from Falcon, I'd be
> very interested in the proposal.
>
> If only some but not all of the floating points are removed, I struggle
> to see appeal though. You'd still be in a situation where no established
> leakage analysis and side-channel countermeasures exist for a key part

From: pqc-forum@list.nist.gov on behalf of Blumenthal, Uri - 0553 - MITLL <uri@ll.mit.edu>
Sent: Friday, February 16, 2024 12:09 PM
To: Christine Cloostermans; Simon Hoerder
Cc: pqc-forum
Subject: Re: [EXT] Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

One significant benefit of removing all the FP operation from FALCON would be simplification of its validation process – the fact that validation difficulties are explicitly called out in the NIST reports should tell something.

--
V/R,
Uri

From: <pqc-forum@list.nist.gov> on behalf of Christine Cloostermans <cvvrede@gmail.com>
Date: Friday, February 16, 2024 at 02:31
To: Simon Hoerder <simon@hoerder.net>
Cc: "pqc-forum@list.nist.gov" <pqc-forum@list.nist.gov>
Subject: [EXT] Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

Hi all,

Let me start by saying that we do agree that removing FP operations is a good general direction for improvement on embedded platforms.

However, we agree with Simon here: we do not see a (significant) benefit from removing the (partial) FP operations of the key generation of Falcon.

Key generation is in many cases not even performed on an embedded device.

In many cases, keys are provisioned beforehand and therefore this change would not help with easing Falcon implementation.

Even in the cases where key generation would be performed on chip, the FP operations that remain are still a question mark for side-channel protection.

From: 'Maxime Bros' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Friday, April 5, 2024 8:50 PM
To: pqc-forum
Cc: Blumenthal, Uri - 0553 - MITLL; pqc-forum; Christine Cloostermans; Simon Hoerder
Subject: Re: [EXT] Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

Dear all,

I hope you're doing well.

First, we would like to thank the community for the feedback on the potential change in the keygen of Falcon (FN-DSA). At this moment, NIST's opinion on this change is not final, but we do lean a little bit towards accepting the modification of replacing Falcon Round 3 NTRUGen procedure by Hawk's one as mentioned by Thomas' Pornin in this thread. I will try to explain why in this email.

This change would bring some improvements:

1) Keygen without floating point (provided that the FFT part is done in the signature algorithm, which does not slow signature a lot according to Falcon's team experiments)

2) Easier to get a constant time keygen

3) Keygen would use less RAM

4) Numerically stable keygen (given a seed, there is no ambiguity on the key a correct implementation will generate)

It is worth noting that these improvements come without impacting the security (as far as we are aware of), and they do not change the signature nor the verification process. In fact, only a small portion of the keygen as specified in the 3rd Round specs would be modified, namely the generation of F and G.

Item 4) above addresses validation's concerns, as raised by Uri, and it is worth noting that besides NIST's CAVP, testability is a priority for a lot of people / companies / institutions.

The use of floating point in FN-DSA and the challenges it raised could delay by a great deal its adoption, and so even if this tweak only solves the problem partially, we do agree on that, it is still an improvement that one gets almost "for free".

Overall, NIST agrees with Falcon's Team that this idea sounds reasonable, but we also want to ensure that the community supports the idea!

Thus, we encourage more people to answer on this thread to make sure that we've heard as much of the community's opinion before we make a decision.

Feel free to reach out to us during the NIST 5th Standardization Conference as well.

Sincerely,
Dr. Maxime BROS
(on behalf of the NIST FIPS 206 - FN-DSA Team)

From: 'Thomas Pornin' via pqc-forum <pqc-forum@list.nist.gov>
Sent: Thursday, April 11, 2024 8:49 AM
To: pqc-forum
Cc: Bros, Maxime P. (IntlAssoc); Blumenthal, Uri - 0553 - MITLL; pqc-forum; Christine Cloostermans; Simon Hoerder
Subject: Re: [EXT] Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

In relation to the new Falcon keygen implementation, I have also written a Python equivalent, which outputs the exact same key pairs as the C code, for the same input seed. This is a one-file implementation meant to support tests and debug, in particular printing out intermediate values. It is available there:
https://github.com/pornin/ntrugen/blob/main/python/falcon.py

Thomas

Le vendredi 5 avril 2024 à 20:49:31 UTC-4, Maxime Bros a écrit :

> Dear all,
>
> I hope you're doing well.
>
> First, we would like to thank the community for the feedback on the potential change in the keygen of Falcon (FN-DSA). At this moment, NIST's opinion on this change is not final, but we do lean a little bit towards accepting the modification of replacing Falcon Round 3 NTRUGen procedure by Hawk's one as mentioned by Thomas' Pornin in this thread. I will try to explain why in this email.
>
> This change would bring some improvements:
>
> 1) Keygen without floating point (provided that the FFT part is done in the signature algorithm, which does not slow signature a lot according to Falcon's team experiments)
>
> 2) Easier to get a constant time keygen
>
> 3) Keygen would use less RAM
>
> 4) Numerically stable keygen (given a seed, there is no ambiguity on the key a correct implementation will generate)
>
> It is worth noting that these improvements come without impacting the security (as far as we are aware of), and they do not change the signature nor the verification process. In fact, only a small portion of the keygen as specified in the 3rd Round specs would be modified, namely the generation of F and G.
>
> Item 4) above addresses validation's concerns, as raised by Uri, and it is worth noting that besides NIST's CAVP, testability is a priority for a lot of people / companies / institutions.
>
> The use of floating point in FN-DSA and the challenges it raised could delay by a great deal its adoption, and so even if this tweak only solves the problem partially, we do agree on that, it is still an improvement that one gets almost "for free".
>
> Overall, NIST agrees with Falcon's Team that this idea sounds reasonable, but we also want to ensure that the

From: pqc-forum@list.nist.gov on behalf of Anjan Roy <anjanroy708@gmail.com>
Sent: Thursday, April 11, 2024 1:23 PM
To: Thomas Pornin
Cc: pqc-forum; Bros, Maxime P. (IntlAssoc); Blumenthal, Uri - 0553 - MITLL; Christine Cloostermans; Simon Hoerder
Subject: Re: [EXT] Re: [pqc-forum] [FALCON OFFICIAL] Keygen implementation

Dear Thomas,

Thank you very much for making it available. It'll be very useful.

Regards,
Anjan Roy

On Thu, 11 Apr 2024, 16:49 'Thomas Pornin' via pqc-forum, <pqc-forum@list.nist.gov> wrote:

> In relation to the new Falcon keygen implementation, I have also written a Python equivalent, which outputs the exact same key pairs as the C code, for the same input seed. This is a one-file implementation meant to support tests and debug, in particular printing out intermediate values. It is available there:
> https://github.com/pornin/ntrugen/blob/main/python/falcon.py
>
> Thomas
>
> Le vendredi 5 avril 2024 à 20:49:31 UTC-4, Maxime Bros a écrit :
>
> > Dear all,
> >
> > I hope you're doing well.
> >
> > First, we would like to thank the community for the feedback on the potential change in the keygen of Falcon (FN-DSA).
> > At this moment, NIST's opinion on this change is not final, but we do lean a little bit towards accepting the modification of replacing Falcon Round 3 NTRUGen procedure by Hawk's one as mentioned by Thomas' Pornin in this thread. I will try to explain why in this email.
> >
> > This change would bring some improvements:
> >
> > 1) Keygen without floating point (provided that the FFT part is done in the signature algorithm, which does not slow signature a lot according to Falcon's team experiments)
> >
> > 2) Easier to get a constant time keygen
> >
> > 3) Keygen would use less RAM
> >
> > 4) Numerically stable keygen (given a seed, there is no ambiguity on the key a correct implementation will generate)
> >
> > It is worth noting that these improvements come without impacting the security (as far as we are aware of), and they do not change the signature nor the verification process. In fact, only a small portion of the keygen as specified in the 3rd Round specs would be modified, namely the generation of F and G.