From: Philip Lafrance <Philip.Lafrance@isara.com>
Sent: Wednesday, January 10, 2018 11:47 AM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: Gravity-SPHINCS

Dear Gravity-SPHINCS authors,

It appears that in your specification, Sections 2.3.3 through 2.3.6 describe the vanilla (plain, original) version of the Winternitz OTS scheme, but the bibliography references Hülsing et. al's WOTS+ paper; which is a major improvement on the vanilla scheme. I have not looked through the provided implementations to see which scheme they employ, but my personal recommendation would be to indeed use WOTS+; I suspect many on the forum would agree with that position.

It is possible that this has already been considered by you good folks, but I thought it worthwhile enough to bring up just in case. My apologies if this has been addressed elsewhere.

Warm regards,

Philip Lafrance

From: Guillaume Endignoux <guillaume.endignoux@gmail.com>
Sent: Thursday, January 11, 2018 6:41 PM
To: Philip Lafrance
Cc: pqc-comments; pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Gravity-SPHINCS

Dear Philip,

Thank you for your comments. We indeed did not give many details about this choice in the specification PDF, in order to keep it small. We give more details in [1, section 3.4], but here is a brief summary.

The rationale for WOTS+ [2] compared to WOTS was mostly to reduce the security to a second-preimage-resistant hash function instead of a collision-resistant function. However, the WOTS+ paper was written in a classical context, as it assumed second-preimage search in O(2^n) [2, section 3.3] when collision search was assumed to be in O(2^{n/2}). As expected, this allows to reduce signature size by 50% [2, section 5], since the hash output size n can be twice smaller than for WOTS. I might have missed something but this is the major selling point of WOTS+.

However, things are quite different against quantum attackers, as Grover's algorithm allows to search second preimages in O(2^{n/2}). We are not aware of a collision search algorithm with a better complexity than O(2^{n/2}), assuming that all costs are taken into account (not only time but also hardware, memory access, communication).

In the post-quantum context, we therefore think that the XOR masks added in WOTS+ compared to WOTS are an unnecessary complexity. This makes the proof as well as the implementation more complex (which means more risks of implementation bugs or potential flaws in the proofs), but without providing a significant security benefit.
The security proof of WOTS+ in [2, section 3.2] is quite long and non-trivial to follow, whereas the security proof of WOTS is simpler. You can find security proofs relevant to Gravity-SPHINCS in [3, chapter 6].

I hope that this helps answering your concern.

Best regards,
Guillaume Endignoux

[1] https://eprint.iacr.org/2017/933
[2] Hülsing, A.: W-OTS+ - shorter signatures for hash-based signature schemes. In: Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings. pp. 173–188 (2013)
[3] https://gendignoux.com/assets/pdf/2017-07-master-thesis-endignoux-report.pdf

Le 11 janv. 2018 16:35, "Philip Lafrance" <philip.lafrance92@gmail.com> a écrit :

> Dear Gravity-SPHINCS authors,
>
> It appears that in your specification, Sections 2.3.3 through 2.3.6 describe the vanilla (plain, original) version of the Winternitz OTS scheme, but the bibliography references Hülsing et. al's WOTS+ paper; which is a major improvement on the vanilla scheme. I have not looked through the provided implementations to see which

From: Thomas Prest <thomas.prest@ens.fr>
Sent: Friday, April 13, 2018 4:28 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: Gravity-SPHINCS

Dear all,

This mail is to inform you of a fault injection attack against SPHINCS+ and Gravity-SPHINCS.
In schemes of the SPHINCS family, the signature essentially consists of reconstructing a bunch of Merkle trees and signing their roots with one-time signatures (OTS).
Introducing one single random fault during the reconstruction of the penultimate Merkle tree has the effect that one OTS signs two different values, which can then be exploited to forge signatures for any message for a computational cost of computing about 2^34 hashes.

More detailed information can be found here:
- Article: https://eprint.iacr.org/2018/102
- Slides: https://tprest.github.io/Slides/grafting-trees-pqcrypto.pdf
We have informed the submitters about this attack.

We want to emphasize that this is a fault injection attack: it does not question the security of these schemes when the signing algorithm is executed normally.

Best,
Laurent, Ange and Thomas

From: Jacob Alperin-Sheriff <jacobmas@gmail.com>
Sent: Friday, April 13, 2018 4:39 PM
To: Thomas Prest
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Gravity-SPHINCS

Is there a good survey of fault injections (and similar) in the wild and how to defend against them?

On Fri, Apr 13, 2018, 4:27 PM Thomas Prest <thomas.prest@ens.fr> wrote:

> Dear all,
>
> This mail is to inform you of a fault injection attack against SPHINCS+
> and Gravity-SPHINCS.
> In schemes of the SPHINCS family, the signature essentially consists of
> reconstructing a bunch of Merkle trees and signing their roots with
> one-time signatures (OTS).
> Introducing one single random fault during the reconstruction of the
> penultimate Merkle tree has the effect that one OTS signs two different
> values, which can then be exploited to forge signatures for any message
> for a computational cost of computing about 2^34 hashes.
>
> More detailed information can be found here:
> - Article: https://ia.cr/2018/102
> - Slides: https://tprest.github.io/Slides/grafting-trees-pqcrypto.pdf
> We have informed the submitters about this attack.
>
> We want to emphasize that this is a fault injection attack: it does not
> question the security of these schemes when the signing algorithm is
> executed normally.
>
> Best,
> Laurent, Ange and Thomas
>
> --
>

From: Reza Azarderakhsh <azarderakhsh@gmail.com>
Sent: Friday, April 13, 2018 5:23 PM
To: Jacob Alperin-Sheriff
Cc: Thomas Prest; pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Gravity-SPHINCS

Here it is: https://dl.acm.org/citation.cfm?doid=2930664

On Fri, Apr 13, 2018 at 4:38 PM Jacob Alperin-Sheriff <jacobmas@gmail.com> wrote:

> Is there a good survey of fault injections (and similar) in the wild and how to defend against them?
>
> On Fri, Apr 13, 2018, 4:27 PM Thomas Prest <thomas.prest@ens.fr> wrote:
>
>> Dear all,
>>
>> This mail is to inform you of a fault injection attack against SPHINCS+
>> and Gravity-SPHINCS.
>> In schemes of the SPHINCS family, the signature essentially consists of
>> reconstructing a bunch of Merkle trees and signing their roots with
>> one-time signatures (OTS).
>> Introducing one single random fault during the reconstruction of the
>> penultimate Merkle tree has the effect that one OTS signs two different
>> values, which can then be exploited to forge signatures for any message
>> for a computational cost of computing about 2^34 hashes.
>>
>> More detailed information can be found here:
>> - Article: https://ia.cr/2018/102
>> - Slides: https://tprest.github.io/Slides/grafting-trees-pqcrypto.pdf
>> We have informed the submitters about this attack.
>>
>> We want to emphasize that this is a fault injection attack: it does not
>> question the security of these schemes when the signing algorithm is
>> executed normally.
>>
>> Best,
>> Laurent, Ange and Thomas
>>
>>
>
>

From: Alessandro Barenghi <alessandro.barenghi@polimi.it>
Sent: Friday, April 13, 2018 6:59 PM
To: Jacob Alperin-Sheriff
Cc: pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Gravity-SPHINCS

____
From: Jacob Alperin-Sheriff <jacobmas@gmail.com>
Sent: 13 April 2018 22:38
To: Thomas Prest
Cc: pqc-comments; pqc-forum
Subject: Re: [pqc-forum] OFFICIAL COMMENT: Gravity-SPHINCS

Is there a good survey of fault injections (and similar) in the wild and how to defend against them?

Although it's growing a bit old, a rather comprehensive one is:

https://doi.org/10.1109/JPROC.2012.2188769

Kind regards,

-- Alessandro

From: Aymeric Genêt <aymeric.gen@gmail.com>
Sent: Tuesday, April 17, 2018 9:38 AM
To: pqc-forum
Cc: pqc-comments; thomas.prest@ens.fr
Subject: Re: OFFICIAL COMMENT: Gravity-SPHINCS

Hello,

I evaluated SPHINCS-256 with respect to side-channel and fault attacks last year as part of my Master thesis.

If I may make a comment based on the conclusion of my analysis, the reason why the fault attack works in the case of SPHINCS is because the scheme inherited of the Goldreich's scheme structure which "reuses" a one-time signature in a clever way.

What I mean is that, during a SPHINCS signature, there are instances of OTS used to sign messages that are not supposed to change from one execution to another. In the case of CMSS (the ancestor of the XMSS structure used within SPHINCS) those messages are the roots of the intermediate Merkle trees.

While this feature is a core improvement that allows most of current hash-based schemes to be practical, I believe that OTSs were not meant to be used that way, which is what caused—in my opinion—the fault attack.

Even though I agree that any unprotected implementation of a cryptographic primitive can't be secure against fault attacks, here the defect comes from the design itself of the algorithm and not an implementation of it. Unlike, for instance, RSA where a glitch injection was possible due to the Chinese remainder theorem speed-up, there are no "algorithmic fixes" or ways to do it so the fault does not impact the security. Therefore, SPHINCS can be deployed only with a robustness enhancement that will most likely be circumvented given the adversary model.

Anyway, those were my two cents. If you're interested in my work on this subject, you can find it here:
https://infoscience.epfl.ch/record/253317

Aymeric

On Friday, April 13, 2018 at 10:27:59 PM UTC+2, Thomas Prest wrote:

> Dear all,
>
> This mail is to inform you of a fault injection attack against SPHINCS+
> and Gravity-SPHINCS.
> In schemes of the SPHINCS family, the signature essentially consists of
> reconstructing a bunch of Merkle trees and signing their roots with
> one-time signatures (OTS).
> Introducing one single random fault during the reconstruction of the
> penultimate Merkle tree has the effect that one OTS signs two different
> values, which can then be exploited to forge signatures for any message
> for a computational cost of computing about 2^34 hashes.
>
> More detailed information can be found here:
> - Article: https://ia.cr/2018/102
> - Slides: https://tprest.github.io/Slides/grafting-trees-pqcrypto.pdf
> We have informed the submitters about this attack.
>
> We want to emphasize that this is a fault injection attack: it does not

From: Santi Vives <santi.x256@gmail.com>
Sent: Friday, November 02, 2018 6:29 PM
To: pqc-comments
Subject: OFFICIAL COMMENT: SPHINCS

Hi all. Bellow is a possible optimization. It is a small change, and it can save some wastefull hash computations. I've already mailed the Gravity-Sphincs team, and it seems to apply to SPHINCS+ as well.

As you know, when using WOTS we pick a compression parameter w that determines the size.

We can have more efficient signatures if we allow two different compression parameters: one for the main part of the signature, and other for the checksum. I like to call this Unbalanced Winternitz since it results in chains of uneven heights.

Consider this:

Instead of defining a compression parameter (w) and computing the size of the signature from that, let's start by defining the final size, and compute the optimal compression parameters allowing different values for the main part and the checksum.

Let's start by defining a size k:
k = ka + kc
Here ka is the size of the main part, and kc is the size of the checksum.

The message is a random value in the range 0...2^256-1.

Since the main part has length ka, we need to encode the message as a number with ka digits. The smallest possible base for that number is:
ba = ceil ((2^256) ^ (1/ka)) = ceil (2^(256/ka))

Now, each chain can take a value in the range 0...ba-1. Then the checksum encodes a value in the range 0...(ba-1)*ka. That's (ba-1)*ka+1 possible values.
So the smallest base for the checksum is:
bc = ceil (((ba - 1) * ka + 1) ^ (1/kc))

Notice that from this we can compute the full cost of the ots (the total number of computed hashes from the bottom of the chains to the top):
ka * (ba - 1) + kc (bc - 1)

So, ka and kc are all we need to know the cost.

We can split a size k into a pair (ka, kc) in any of these ways:
(1, k-1), (2, k-2), ..., (k-1, 1)

So, just by having the size k we can compute the cost for each possible (ka, kc) pair and choose the one that's optimal.

Now, the numbers:

For WOTS+ and Unbalanced WOTS+ with size k=67 we have:
WOTS

w=16, k=67, cost=1005
UWOTS
ka=64, kc=3, ba=16, bc=10, cost=987
(we save 18 hashes per ots)

For WOTS+ and Unbalanced WOTS+ with size k=34 we have:
WOTS
w=256, k=34, cost=8670
UWOTS
ka=32, kc=2, ba=256, bc=91, cost=8340
(we save 330 hashes per ots)

Computing the checksum only takes 1 or 2 divmod extra computations (depending on the case) for the ots in use only, which is negligible compared to the hashes.

Below is some Python code. It uses 4 bases instead (2 for the main part, 2 for the checksum) which is better when the size of main part is not a power of 2.

https://jota.tuxfamily.org
https://jotasapiens.com/research

Best regards,
Santi J. (@jotasapiens)