From: Sydney Antonov <ska84@protonmail.com>
To: pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Wednesday, April 20, 2022 07:20:57 PM ET

Dear all,

SPHINCS+ relies on the distinct-function, multi-target second-preimage resistance (DM-SPR) of the underlying keyed hash function. This property can be broken for SHA-256(key||message) (which is used by SPHINCS+-SHA-256) using around (t-1)2**128 + 2^256/t compression function calls when attacking t targets using the following attack:

In this attack keys are initial hash values instead of message prefixes, without loss of generality.

Let C: {0,1}^256 x {0,1}^512 → {0,1}^256 be SHA-256's compression function.

1. If there are multiple keys:
1.1. For each pair of keys (k, l):
1.1.1. Find an (x_k, x_l) such that y = C(k, x_k) = C(l, x_l).
       This can be done using around 2^128 compression function calls.
1.1.2. Replace the pair with the single key y.
1.2. If there's a remaining key k because there was an odd number of keys replace it with C(k, 0).
1.3. Repeat step 1.
2. Find a SHA-256 preimage of one of the targets using the final key as an IV. This can be done using around 2^256/t compression function calls.
3. Concatenate the sequence of compression function blocks used in step 1 to derive the final key from the key corresponding to the target for which a second-preimage was found by step 2.
4. Concatenate the results of step 3 and step 2.

The result of step 4 is a preimage for one of the targets and an attacker

can ensure it's almost certainly not the original one by randomizing step 2.

This attack means the use of unique prefixes/IVs with SHA-256 provides very little protection against multi-target second-preimage attacks when t < 2^64, but it's less effective on truncated SHA-256.

If "internal claws" could be found with cost c, this attack would cost (t-1)c + 2^256/t, so if they could be found more efficiently it could also threaten SHAKE256 and truncated SHA-256. Also claw-freeness seems almost as strong an assumption as collision-resistance so I don't think it's a desirable assumption for a collision-resilient signature scheme.

Sydney

From: Andreas Hülsing <ietf@huelsing.net> via pqc-forum@list.nist.gov
To: Sydney Antonov <ska84@protonmail.com>, pqc-comments@nist.gov
CC: pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Thursday, April 21, 2022 03:51:48 AM ET

Dear Sydney,

Thanks for looking at SPHINCS+. This is an interesting attack that does demonstrate that our real hash functions do not perfectly behave like random oracles (and that the chaining value of SHA2 may be a bit too short for some use cases). Luckily its impact against SPHINCS+ is limited (that is of course if I did not miss anything). The reason is that SPHINCS+ only needs DM-SPR for SHA2 with fixed length messages. The longest message length used is 67*n (+n for the key) bytes which occurs in the compression of the WOTS public keys for w=16 and uses 34 compression function calls. This means you can at most gain a 2^33 speed-up / 33 bits of security for the n=32 variant, classically.

As you mentioned, the impact is less for the truncated variants. To be precise, for n = 16 there will be no advantage as the internal state is twice the length of the hash values, so finding collisions on that state takes as much time as finding preimages. For n=24 the maximum message length processed by SHA2 in the relevant setting is bounded by 51, meaning 26 compression function calls, and hence at most a 2^26 speed-up.

For quantum attacks, the difference between collision search and preimage search shrinks down even if we do not take access times for quantum accessible memory into account. E.g., for n = 24 the attack cost would be (t-1)2**64 + 2^96/t in the ideal case where memory access is free. In this case one can at most gain 16 bits anyway and using reasonable estimates for memory access, even less.

Best wishes,

Andreas

On 21-04-2022 01:20, 'Sydney Antonov' via pqc-forum wrote:

> Dear all,
>
> SPHINCS+ relies on the distinct-function, multi-target second-preimage
> resistance (DM-SPR) of the underlying keyed hash function. This
> property can be broken for SHA-256(key||message) (which is used by
> SPHINCS+-SHA-256) using around (t-1)2**128 + 2^256/t compression function
> calls when attacking t targets using the following attack:
>
> In this attack keys are initial hash values instead of message prefixes,
> without loss of generality.
>
> Let C: {0,1}^256 x {0,1}^512 → {0,1}^256 be SHA-256's compression
> function.
>
> 1. If there are multiple keys:
> 1.1. For each pair of keys (k, l):
> 1.1.1. Find an (x_k, x_l) such that y = C(k, x_k) = C(l, x_l).
>   This can be done using around 2^128 compression function calls.
> 1.1.2. Replace the pair with the single key y.
> 1.2. If there's a remaining key k because there was an odd number of
> keys replace it with C(k, 0).
> 1.3. Repeat step 1.
> 2. Find a SHA-256 preimage of one of the targets using the final key as
>   an IV. This can be done using around 2^256/t compression function
>   calls.
> 3. Concatenate the sequence of compression function blocks used in step
>   1 to derive the final key from the key corresponding to the target
>   for which a second-preimage was found by step 2.
> 4. Concatenate the results of step 3 and step 2.
>
> The result of step 4 is a preimage for one of the targets and an attacker
> can ensure it's almost certainly not the original one by randomizing
> step 2.
>

> This attack means the use of unique prefixes/IVs with SHA-256 provides
> very little protection against multi-target second-preimage attacks when
> t < 2^64, but it's less effective on truncated SHA-256.
>
> If "internal claws" could be found with cost c, this attack would cost
> (t-1)c + 2^256/t, so if they could be found more efficiently it could
> also threaten SHAKE256 and truncated SHA-256. Also claw-freeness seems
> almost as strong an assumption as collision-resistance so I don't think
> it's a desirable assumption for a collision-resilient signature scheme.
>
> Sydney
>

From: msg <msg260@gmail.com> via pqc-forum@list.nist.gov
To: pqc-forum <pqc-forum@list.nist.gov>
CC: Andreas Hülsing <ietf@huelsing.net>, pqc-...@list.nist.gov <pqc-forum@list.nist.gov>, Sydney Antonov <ska84@protonmail.com>, pqc-co...@nist.gov <pqc-comments@nist.gov>
Subject: Re: [pqc-forum] ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Thursday, April 21, 2022 04:24:39 AM ET

Thanks for the discussion.

What do you think using KMAC or HMAC instead of Hash(Key||Message). I think these structures resist Sydney's attack.
Best.

21 Nisan 2022 Perşembe tarihinde saat 10:51:44 UTC+3 itibarıyla Andreas Hülsing şunları yazdı:

> Dear Sydney,
>
> Thanks for looking at SPHINCS+. This is an interesting attack that does demonstrate that our real hash functions do not perfectly behave like random oracles (and that the chaining value of SHA2 may be a bit too short for some use cases). Luckily its impact against SPHINCS+ is limited (that is of course if I did not miss anything). The reason is that SPHINCS+ only needs DM-SPR for SHA2 with fixed length messages. The longest message length used is 67*n (+n for the key) bytes which occurs in the compression of the WOTS public keys for w=16 and uses 34 compression function calls. This means you can at most gain a 2^33 speed-up / 33 bits of security for the n=32 variant, classically.
>
> As you mentioned, the impact is less for the truncated variants. To be precise, for n = 16 there will be no advantage as the internal state is twice the length of the hash values, so finding collisions on that state takes as much time as finding preimages. For n=24 the maximum message length processed by SHA2 in the relevant setting is bounded by 51, meaning 26 compression function calls, and hence at most a 2^26 speed-up.
>
> For quantum attacks, the difference between collision search and

preimage search shrinks down even if we do not take access times for quantum accessible memory into account. E.g., for n = 24 the attack cost would be (t-1)2**64 + 2^96/t in the ideal case where memory access is free. In this case one can at most gain 16 bits anyway and using reasonable estimates for memory access, even less.

Best wishes,

Andreas

On 21-04-2022 01:20, 'Sydney Antonov' via pqc-forum wrote:

> Dear all,
>
> SPHINCS+ relies on the distinct-function, multi-target second-preimage
> resistance (DM-SPR) of the underlying keyed hash function. This
> property can be broken for SHA-256(key||message) (which is used by
> SPHINCS+-SHA-256) using around (t-1)2**128 + 2^256/t compression function
> calls when attacking t targets using the following attack:
>
> In this attack keys are initial hash values instead of message prefixes,
> without loss of generality.
>
> Let C: {0,1}^256 x {0,1}^512 -> {0,1}^256 be SHA-256's compression
> function.
>
> 1. If there are multiple keys:
> 1.1. For each pair of keys (k, l):
> 1.1.1. Find an (x_k, x_l) such that y = C(k, x_k) = C(l, x_l).
> This can be done using around 2^128 compression function calls.
> 1.1.2. Replace the pair with the single key y.
> 1.2. If there's a remaining key k because there was an odd number of
> keys replace it with C(k, 0).
> 1.3. Repeat step 1.
> 2. Find a SHA-256 preimage of one of the targets using the final key as
> an IV. This can be done using around 2^256/t compression function

> calls.
> 3. Concatenate the sequence of compression function blocks used in step
> 1 to derive the final key from the key corresponding to the target
> for which a second-preimage was found by step 2.
> 4. Concatenate the results of step 3 and step 2.
>
> The result of step 4 is a preimage for one of the targets and an attacker
> can ensure it's almost certainly not the original one by randomizing
> step 2.
>
> This attack means the use of unique prefixes/IVs with SHA-256 provides
> very little protection against multi-target second-preimage attacks when
> t < 2^64, but it's less effective on truncated SHA-256.
>
> If "internal claws" could be found with cost c, this attack would cost
> (t-1)c + 2^256/t, so if they could be found more efficiently it could
> also threaten SHAKE256 and truncated SHA-256. Also claw-freeness seems
> almost as strong an assumption as collision-resistance so I don't think
> it's a desirable assumption for a collision-resilient signature scheme.
>
> Sydney
>

--

From: Sydney Antonov <ska84@protonmail.com>
To: msg <msg260@gmail.com>
CC: pqc-...@list.nist.gov <pqc-forum@list.nist.gov>, Andreas Hülsing <ietf@huelsing.net>, pqc-co...@nist.gov <pqc-comments@nist.gov>
Subject: Re: [pqc-forum] ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Thursday, April 21, 2022 06:11:33 AM ET

> For quantum attacks, the difference between collision search and
> preimage search shrinks down even if we do not take access times for
> quantum accessible memory into account. E.g., for n = 24 the attack cost
> would be (t-1)2**64 + 2^96/t in the ideal case where memory access is
> free. In this case one can at most gain 16 bits anyway and using
> reasonable estimates for memory access, even less.

I think the cost of 192-bit Grover search is at least 2^128 because
attackers don't have time for 2^96 iterations of Grover's algorithm so
instead would have to do something like build 2^64 quantum computers
which each do 2^64 iterations in parallel.

> What do you think using KMAC or HMAC instead of Hash(Key||Message).
> I think these structures resist Sydney's attack.

KMAC is a totally different construction to HMAC. It adsorbs the key
once at the start so it's as vulnerable as to my attack SHAKE. That
is, it could become vulnerable if a breakthrough is made in Keccak
cryptanalysis. Internal collisions in SHAKE256 require around 2^256
classical queries to the Keccak permutation if it's modeled as a random
oracle. The best known attacks are no better than generic attacks.

Sydney

From: Sydney Antonov <ska84@protonmail.com>
To: Andreas Hülsing <ietf@huelsing.net>
CC: pqc-comments@nist.gov, pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Thursday, April 21, 2022 03:19:11 PM ET

Dear Andreas,

> Thanks for looking at SPHINCS+. This is an interesting attack that does
> demonstrate that our real hash functions do not perfectly behave like
> random oracles (and that the chaining value of SHA2 may be a bit too
> short for some use cases). Luckily its impact against SPHINCS+ is
> limited (that is of course if I did not miss anything). The reason is
> that SPHINCS+ only needs DM-SPR for SHA2 with fixed length messages. The
> longest message length used is 67*n (+n for the key) bytes which occurs
> in the compression of the WOTS public keys for w=16 and uses 34
> compression function calls. This means you can at most gain a 2^33
> speed-up / 33 bits of security for the n=32 variant, classically.

This can be improved upon using a ternary tree of 3-claws with that
message block limitation.

The number of 3-claws required is just under half the number of targets
(1/3 + 1/9 + ... = 1/2) and finding a 3-claw uses around 3*2**((2/3)256)
compression function calls so the total cost is around 1.5t2^((2/3)256) +
(2^256)/t compression function calls.

The optimal number of targets for that formula is sqrt((2^(256/3))/1.5)
= 2^42.374... = 3^26.735..., resulting in a 2^214.63 attack.

P.S. Maybe this attack can be optimized further by mixing 2-claws and 3-claws.

Sydney

From: Sydney Antonov <ska84@protonmail.com>
To: Sydney Antonov <ska84@protonmail.com>
CC: Andreas Hülsing <ietf@huelsing.net>, pqc-comments@nist.gov, pqc-forum@list.nist.gov
Subject: Re: [pqc-forum] ROUND 3 OFFICIAL COMMENT: SPHINCS+
Date: Friday, April 22, 2022 05:49:39 AM ET

> P.S. Maybe this attack can be optimized further by mixing 2-claws and 3-claws.

It can. The following is a 2^209.91 attack against 2^10*3^23 =
2^46.454 ... targets:

In this attack keys are initial hash values instead of message prefixes,
without loss of generality.

Let C: {0,1}^256 x {0,1}^512 → {0,1}^256 be SHA-256's compression
function.

1. Repeat 10 times:
1.1. For each pair of keys (k, l):
1.1.1. Find an (x_k, x_l) such that y = C(k, x_k) = C(l, x_l).
This can be done using around 2*2^(256/2) compression function calls.
1.1.2. Replace the pair with the single key y.
2. Repeat 23 times:
2.1. For each trio of keys (k, l, m):
2.1.1. Find an (x_k, x_l, x_m) such that y = C(k, x_k) = C(l, x_l) = C(m, x_m).
This can be done using around (3/2)2^((2/3)256) compression function calls.
2.1.2. Replace the trio with the single key y.
3. Find a SHA-256 preimage of one of the targets using the final key
as an IV. This can be done using around 2^256/2^10/3^23 compression
function calls.
4. Concatenate the sequence of compression function blocks used in steps
1 and 2 to derive the final key from the key corresponding to the target
for which a preimage was found by step 3.
5. Concatenate the results of step 4 and step 3.

The result of step 5 is a preimage for one of the targets and an attacker
can ensure it's almost certainly not the original one by randomizing

step 3.

In total this attack makes around 2^10*3^23*2*2^(256/2) + 3^23*(3/2)*2^((2/3)256) + 2^256/2^10/3^23 or 2^209.91 compression function calls.

Sydney