From: Cooper, David A. (Fed)
To: Kelsey, John M. (Fed)
Cc: internal-pqc
Subject: Re: SPHINCS+ and message bound signatures
Date: Tuesday, October 26, 2021 2:16:56 PM

Hi John,

I'm not very good a reading security proofs. However, looking at the definition of message-bound signatures (Definition 3.2 in https://eprint.iacr.org/2020/1525.pdf) along with the definition of a *negligible function*, I think that SPHINCS+ does, at least technically, have this property. However, I believe that the attack you describe is correct.

This is basically the same as the issue that you raised with stateful hash-based signatures (SP 800-208) at the 192-bit security level. At you suggestion, Section 9.2 of SP 800-208 notes that the security strength against a collision generated by the private key holder is half the number of bits as the security strength provided against all other attacks. In particular, the 192-bit parameter sets generally provide 192 bits of security strength, but only 96 bits of security against a private key holder trying to create a collision.

In the case of SPHINCS+, the loss is security is actually much less than it is for LMS or XMSS. So, we can certainly raise the issue, but I think our suggestion should be that it is sufficient to simply note the issue in the final standard, if SPHINCS+ is selected.

On 10/14/21 6:12 PM, Kelsey, John M. (Fed) wrote:

> Eveyone,
>
> The BUFFing paper talks about one of its properties as having message-bound signatures. That's defined as having signatures only valid for a unique message. You can think of this as having a signature on a message be a binding commitment to the message—once I've seen S = sign(SK,M), you can't produce another M' such that verify(PK,S,M') = 1. They claim that SPHINCS+ meets this requirement, but I think there's a counterexample. I'm hoping I can get a couple of you to check me on this, in case I'm missing something.
>
> In the current version of SPHINCS+, we end up computing the message hash like this:
>
> R = the randomness for this signature
>
> idx,md = message_hash(PK, R , M)
>
> The idx is the path through the Hypertree—it's basically selecting which of the 2^{64} or so FORS keys will be used to sign the message. Then, md is the message digest—it's the thing that is given to FORS to sign.
>
> The size of idx is determined by the height of the hypertree, h. The size of md is the number of FORS trees (k) times the log of the number of leaves per tree (a). So the

total size of everything the message does to affect the final signature is

h + k*a

As long as this value is less than twice the security level, the signer can find pairs of messages that have the same idx,md just by brute force collision search.

Here's the table from the SPHINCS+ spec, on page 38:

	n	h	a	k	**idx+md**	collision work
SPHINCS -128s	16	63	12	14	**231**	115.5
SPHINCS -128f	16	66	6	33	**264**	132
SPHINCS -192s	24	63	14	17	**301**	150.5
SPHINCS -192f	24	66	8	33	**330**	165
SPHINCS -256s	32	64	14	22	**372**	186
SPHINCS -256f	32	68	9	35	**383**	191.5

So it looks to me like the signer can produce a pair of messages (M,M') that will have an identical signature under his private key. Since idx,md will be identical for M and M', the signature has to be the same.

Anyone see something I'm doing wrong here?

Thanks,

--John