From: Smith-Tone, Daniel C. (Fed)
To: Dang, Quynh H. (Fed); Cooper, David (Fed); Kelsey, John M. (Fed); internal-pqc
Subject: RE: SPHINCS+ write-up
Date: Wednesday, June 10, 2020 9:44:10 AM
Attachments: image001.png
image002.png

This discussion makes me nervous in the same way that standardizing stateful HBS and stating that we can validate hybrid signatures made me nervous: I just don't think that it is a good idea. Anyway, on stateful HBS and hybrid signatures, that check's been cashed. For the number of signatures requirement, the check has not been written and I hope we don't.

To me, this situation is comparable to the memory issue with a few twists. As David mentioned, I don't think that it is likely that we could really limit the use cases of SPHINCS+ even if we wanted to (especially if you consider the popularity and influence of the proponents). I guess that I agree that 2^48 signatures is a big number, and it is hard to imagine a threat to a realistic application because of that number of signatures, but I don't think that we need to go very much below that bound for there to be problems. If we are asking for astronomical figures for other aspects of security, I think that it makes sense to have a similar perspective here. Hedging our bets against not only new technology (relevant for the memory discussion) but new applications we can't consider yet (relevant here) and the fact that we can't really control where these things are used would make me more comfortable with keeping a ridiculously high bound on the number of required signatures, something more in line with our stance on AES-operations and on memory issues. (The situation is not exactly the same as that for memory, though. Asking a server for 2^48 signatures seems more like a DoS attack than a cryptographic one.)

Cheers,
Daniel

From: Dang, Quynh H. (Fed) <quynh.dang@nist.gov>
Sent: Wednesday, June 10, 2020 9:14 AM
To: Cooper, David A. (Fed) <david.cooper@nist.gov>; Kelsey, John M. (Fed) <john.kelsey@nist.gov>; Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: Re: SPHINCS+ write-up

Hi Dave, John and the rest of our team,

There are a few problems with SPING+.

1) If the other standardized sig algorithm(s) are not secure, SPING+ probably won't be their replacement, especially with new options which have much smaller max. numbers of sigs.

The reason is that standard bodies would want algorithms which would work pretty much for all situations without the risk of using too many sigs. So, they would have to adopt another algorithm beside SPING+. When they adopt this algorithm, there would be likely no need for SPING+ unless there are no other secure algorithms that they can adopt: This is a very unlikely situation. They would likely prefer rainbow over SPING+ because rainbow does not have a small max. number of sigs problem. Rainbow's big public key issue can be improved by any of the methods that I mentioned before. In addition, sigs can't be reused, but public keys can be sent just once and there are more sigs in TLS than public keys.

2) It is extremely complicated and risky that all peers in all connections must keep a counter of sigs they generate: when a machine is restarted, the counter is lost. To deal with this, basically, people would have to use similar costly techniques in handling state for stateful hbs.

Quynh.

From: David A. Cooper <david.cooper@nist.gov>
Sent: Wednesday, June 10, 2020 8:37 AM
To: Kelsey, John M. (Fed) <john.kelsey@nist.gov>; Smith-Tone, Daniel C. (Fed) <daniel.smith@nist.gov>; internal-pqc <internal-pqc@nist.gov>
Subject: Re: SPHINCS+ write-up

I understand the motivation and why signatures can be smaller and faster if the maximum number of signatures is reduced. The SAGE script allows us to better quantify the impact of allowing such parameter sets. Given the numbers in the table below, is it fair to say that any of the parameter sets with fewer max. sigs. are both much smaller and faster?

I am confused about how the parameter sets were chosen for the submission. According to the SAGE script, the chosen "fast" parameter set takes about 120,480 hashes to create a signature of size 16,976. However, the SAGE script says there is a parameter set that would create a signature of size 16,112 in about 116,114 hashes. What was the reason this smaller and faster option wasn't chosen? Is this an indication that not all of the parameter set options listed by the SAGE script are usable?

I agree that the number 2^64 is somewhat arbitrary and it would be difficult for an application to generate more than 2^48 signatures. But how much of a performance improvement would there need to be for us to consider allowing an exception for SPHINCS+?

Allowing parameter sets with max. sigs. of 2^32 or even 2^20 is a much different situation, as there will definitely be applications that exceed these numbers of signatures. While we can say that the parameter sets are intended for special environments how do we limit their use? This is an issue we've discuss a lot with respect to stateful hash-based signatures (and I would guess it is also an issue in the lightweight cryptography project). The SP for stateful HBS says that they are only intended for certain types of applications, but there is no way we can enforce that, and there are indications that they will be used more widely than we would like. For stateful HBS we decided to impose limitations, such as that only hardware cryptographic modules that do not allow export of private keying material could be validated. This was an unpopular limitation with commenters and I don't think we could get away with it for a specialized SPHINCS+ parameter set. For parameter sets with smaller max. sigs. (e.g., 2^32), would we just include a warning in the documentation not to generate too many signatures with any one key (the way we do with 3DES) and hope that's enough?

I understand that the proposed text doesn't say NIST will allow such parameter sets, only that NIST may consider it. However, I think we should be careful about including such text in the report, just as we were careful in responding to the question about modeling the attacks that require very large amount of memory.

max sig	h	d	b	k	w	sec	sigsize	speed
2^64	64	8	15	10	16	-133	8,080	1,804,288
2^64	60	20	9	30	16	-130	16,976	120,480
2^64	66	22	9	17	16	-129	16,112	116,144
2^48	48	16	6	40	16	-129	14,224	76,928
2^48	48	12	8	24	16	-134	10,960	120,000
2^32	32	16	6	41	16	-131	14,080	41,152
2^32	32	8	9	19	16	-128	8,048	91,264

On 6/9/20 10:57 PM, Kelsey, John M. (Fed) wrote:

> I'm not sure whether this makes sense, which is why I phrased it in the conditional tense. We might consider it, we're not promising to do so. We can omit that, but it seems worthwhile to float the idea that we might allow smaller-numbers-of-keys versions of SPHINCS+ at some point in the future. The two places I'm thinking for this are:
>
> 1. If the other PQ signatures look too shaky, we might want SPHINCS+ as a fallback. In that case, the requirement for 2^{64} signatures per key seems like something we'd want to revisit, given that it makes the signatures much bigger and slower, and that the specific number is very much pulled out of the air. (If we'd said 2^{48} signatures per key, how many applications would have become questionable as a result?) For all the other signature schemes, I think this number has very little performance impact; for SPHINCS+, it matters a lot becauser it says how big the hypertree has to be.
> 2. If we wanted an alternative to stateful hbs, we might allow a much smaller SPHINCS+ (say, with a hypertree of only 2^{20} FORS keys). That would be longer/slower than a stateful scheme, but would only require limiting the number of signatures, not maintaining a monotonic counter for the lifetime of the key.
>
> --John
>
> ---
>
> **From:** "Smith-Tone, Daniel C. (Fed)" <daniel.smith@nist.gov>
> **Date:** Tuesday, June 9, 2020 at 17:12
> **To:** "Cooper, David A. (Fed)" <david.cooper@nist.gov>, internal-pqc <internal-pqc@nist.gov>
> **Subject:** RE: SPHINCS+ write-up
>
> I think that (at least for me) I would need to know what the intended application is to have an idea of how reasonable it is to allow this tradeoff. What applications are there for which we would prefer SPHINCS+ to other candidates AND be okay with such low numbers of allowed signatures? I can't think of anything at the moment.
>
> ---

From: David A. Cooper <david.cooper@nist.gov>
Sent: Tuesday, June 9, 2020 4:30 PM
To: internal-pqc <internal-pqc@nist.gov>
Subject: SPHINCS+ write-up

I'd like to discuss the text that is proposed for SPHINCS+:

> One interesting tradeoff made in SPHINCS+ involves the number of signatures expected per key. The size of the signature is quite sensitive to this number, and going from a limit of 2^{64} to 2^{48} or 2^{32} signatures per key would allow for a much smaller and faster signature scheme. In the future, NIST may consider variants of SPHINCS+ with fewer signatures allowed per key for use in some special environments, for example as an alternative to stateful hash-based signatures.

Based on John's earlier suggestion, I tried the SPHINCS+ parameter exploration SAGE script available at https://sphincs.org/software.html for various values of the maximum number of signatures at the 128-bit security level. The SAGE script output tens of thousands of parameter set options, but below are a few of them for maximum signature values of 2^{64}, 2^{30}, and 2^{20}. The two parameter sets shown for 2^{64} are the ones that are specified in the submission. The "speed" is an estimate of the number of hashes that need to be computed.

There are options that result in both faster signing and smaller signatures (with 2^{30} max signatures, one-third the signing time with 20% smaller signatures, or 56% smaller signatures with same signing time). However, introducing options like this would come with risk. Are the speed and/or signature size savings worth the risk?

max sig	h	d	b	k	w	sec	sigsize	speed
2^64	64	8	15	10	16	-133	8,080	1,804,288
2^64	60	20	9	30	16	-130	16,976	120,480
2^30	36	9	8	17	16	-128	8,080	89,488
2^30	28	7	10	21	16	-141	8,080	105,840
2^30	38	19	6	51	16	-233	16,976	49,164
2^30	30	6	8	25	16	-138	7,456	120,512
2^30	30	15	6	40	16	-129	13,376	38,780
2^20	20	10	6	40	16	-129	10,416	27,560
2^20	24	8	7	25	16	-143	8,080	42,304
2^20	38	19	6	51	16	-281	16,976	49,164
2^20	25	5	10	15	16	-142	5,856	120,480