IEICE TRANS. INF. & SYST., VOL.E104–D, NO.6 JUNE 2021

LETTER

New Parameter Sets for SPHINCS$^+$*

Jinwoo LEE[†], **Tae Gu KANG**[†], **Kookrae CHO**[††a)], ***Nonmembers***, ***and*** **Dae Hyun YUM**[†b)], ***Member***

SUMMARY SPHINCS$^+$ is a state-of-the-art post-quantum hash-based signature that is a candidate for the NIST post-quantum cryptography standard. For a target bit security, SPHINCS$^+$ supports many different tradeoffs between the signature size and the signing speed. SPHINCS$^+$ provides 6 parameter sets: 3 parameter sets for size optimization and 3 parameter sets for speed optimization. We propose new parameter sets with better performance. Specifically, SPHINCS$^+$ implementations with our parameter sets are up to 26.5% faster with slightly shorter signature sizes.

key words: *post-quantum cryptography, hash based signatures, SPHINCS, SPHINCS$^+$, parameters*

1. Introduction

Today's popular public-key algorithms are not quantum resistant. Quantum computers can solve the integer factorization problem, the discrete logarithm problem, and the elliptic curve discrete logarithm problem in a polynomial time [1]. Fortunately, most current symmetric cryptographic primitives such as hash functions are considered to be quantum resistant simply by doubling the key size [2].

Unlike most digital signature schemes based on hard mathematical problems, hash-based signature schemes are built solely on hash functions. In 2015, the first practical stateless hash-based signature called SPHINCS was presented [3]. SPHINCS has a hyper-tree structure combining Goldreich's binary certification tree [4, §6.4.2] , WOTS$^+$ (Winternitz One-Time Signature) [5], and HORS (Hash to Obtain Random Subset) [6].

In 2017, SPHINCS$^+$ [7], a revised version of SPHINCS, was submitted to the NIST post-quantum cryptography standardization project. SPHINCS$^+$ employs improved techniques such as multi-target attack protection [8], tree-less WOTS$^+$ public key compression, FORS (Forest Of Random Subsets), and verifiable index selection. SPHINCS$^+$ is a stateless hash-based signature framework rather than a specific signature scheme. Many parameter options offer flexible tradeoffs with respect to the signature size, the signing speed, and the security level. SPHINCS$^+$ provides three instantiations of hash functions (SHAKE256, SHA-256, Haraka) and six parameter sets (128s, 128f, 192s, 192f, 256s, 256f) where 's' stands for "small" and 'f' for "fast" [7]. In 2019, the 2nd round submission of SPHINCS$^+$ was released, where a tweakable hash function can be instantiated in two different ways: SPHINCS$^+$-robust and SPHINCS$^+$-simple [9]. SPHINCS$^+$ is the only hash-based signature moving on to the 2nd round of the NIST post-quantum cryptography standardization [10].

Parameter sets of SPHINCS$^+$ are obtained with the help of a Sage script that is listed in the specification [7], [9], [11]. For a target bit security, the output of the script is a long list of possible parameters achieving this security level together with the signature size and an estimate of the signing speed. The six parameter sets (128s, 128f, 192s, 192f, 256s, 256f) of SPHINCS$^+$ are non-extreme; they are not the smallest (with a very slow speed) or the fastest (with a very long signature size) options. They provide balanced tradeoffs between the signature size and the signing speed.

Because the six parameter sets of SPHINCS$^+$ are non-extreme, a parameter set with shorter signatures and slower signing (or with longer signatures and faster signing) can be found. However, can we find parameter sets with both shorter signatures and faster signing? We answer the question affirmatively by presenting new parameter sets with shorter signatures and faster signing. To search for better parameter sets, we run the Sage script with an improved estimate of the signing speed and a wider range of parameter values. SPHINCS$^+$ implementations with our parameter sets are up to 26.5% faster with slightly shorter signature sizes.

2. SPHINCS$^+$

We briefly explain the parameters of SPHINCS$^+$. Refer to [7], [9], [11] for a more detailed description of SPHINCS$^+$. We consider the robust instantiations because the simple instantiations require the random oracle model and the six parameter sets of SPHINCS$^+$ were chosen only by considering the robust instantiations.

Let $\mathbb{B}$ be the set of bytes. SPHINCS$^+$ uses several instantiations of tweakable hash functions of the form $\mathbf{T}_\ell : \mathbb{B}^n \times \mathbb{B}^{32} \times \mathbb{B}^{\ell n} \to \mathbb{B}^n$. Hash functions $\mathbf{F} \stackrel{\text{def}}{=} \mathbf{T}_1$ and $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{T}_2$

Manuscript received December 19, 2019.
Manuscript revised May 25, 2020.
Manuscript publicized March 2, 2021.

[†]The authors are with the Department of Information and Communication Engineering, Myongji University, Yongin, Gyeonggido, 17058, Republic of Korea.
[††]The author is with the Division of Electronics and Information System, DGIST, Daegu, 42988, Republic of Korea.

*This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03031413) and the DGIST R&D Program of the Ministry of Science, ICT and Future Planning (20-BT-05).

a) E-mail: kookrae@dgist.ac.kr
b) E-mail: dhyum@mju.ac.kr

DOI: 10.1587/transinf.2019EDL8223

Table 1 Signature size

	Sig
Size	$(h + k(\log t + 1) + d \cdot \mathsf{len} + 1)n$

Table 2 The number of function calls required for signing. The single calls to $\mathbf{H}_{\mathbf{msg}}$, $\mathbf{PRF}_{\mathbf{msg}}$, and $\mathbf{T}_k$ are omitted as they are negligible when estimating speed.

	sign
F	$kt + d(2^{h/d})w{\cdot}\mathsf{len}$
H	$k(t-1) + d(2^{h/d} - 1)$
PRF	$kt + d(2^{h/d})\mathsf{len}$
$\mathbf{T}_{\mathsf{len}}$	$d2^{h/d}$

are two special cases of $\mathbf{T}_\ell$. SPHINCS$^+$ uses pseudorandom functions **PRF** and $\mathbf{PRF}_{\mathbf{msg}}$ and an additional keyed hash function $\mathbf{H}_{\mathbf{msg}}$.

SPHINCS$^+$ is a hyper-tree of height h that consists of d layers of XMSS (eXtended Merkle Signature Scheme) trees where each leaf of XMSS is the public key of a WOTS$^+$ key pair. Each WOTS$^+$ key of the 2^h leaves in the bottom layer signs a FORS public key, which is then used to sign the message. The public key of SPHINCS$^+$ is the root of the hyper-tree and the private key is a secret seed value that can generate all WOTS$^+$ and FORS keys pseudorandomly. A WOTS$^+$ key pair defines a structure that consists of len hash chains of length w. FORS consists of k trees of height a where the leaves of each tree are the hashes of the $t = 2^a$ private key elements and the public key is computed by compressing the concatenation of all the k root nodes with the tweakable hash $\mathbf{T}_k$.

The theoretical formulas for the size and the speed of SPHINCS$^+$ are given in Table 1 and Table 2 [9].

3. New Parameter Sets

3.1 Sage Script

For a target security level, the Sage script searches through a large space of possible parameter values to select the hyper-tree parameters h and d, the FORS parameters $\log t$ and k, and the WOTS$^+$ parameter w. The original search range is as follows [7].

$$\begin{aligned} h &\in \{60, 62, 64, \ldots, 72\} \\ \log t &\in \{4, 5, 6, \ldots, 16\} \\ k &\in \{5, 6, 7, \ldots, 39\} \\ d &\in \{4, 5, 6, \ldots, h-1\} \\ w &\in \{16,\ 256\} \end{aligned} \tag{1}$$

The signing speed of a parameter set is estimated as follows.

$$\begin{aligned} &\text{speed} = (\text{num. of calls to } \mathbf{F}) + (\text{num. of calls to } \mathbf{H}) \\ &\quad = (kt + d(2^{h/d})w{\cdot}\mathsf{len}) + (k(t-1) + d(2^{h/d}-1)) \\ &\quad \approx (kt + d(2^{h/d})w{\cdot}\mathsf{len}) + (kt + d2^{h/d}) \\ &\quad = 2kt + d(2^{h/d}(w{\cdot}\mathsf{len} + 1)) \\ &\quad = k2^{\log t+1} + d(2^{h/d}(\mathsf{len}{\cdot}w + 1)) \end{aligned} \tag{2}$$

Table 3 The original parameter sets of the SPHINCS$^+$ specification where the column labeled "bitsec" gives the bit security [7], [9].

	n	h	d	$\log t$	k	w	bitsec	sig bytes
SPHINCS$^+$-128s	16	64	8	15	10	16	133	8080
SPHINCS$^+$-128f	16	60	20	9	30	16	128	16976
SPHINCS$^+$-192s	24	64	8	16	14	16	196	17064
SPHINCS$^+$-192f	24	66	22	8	33	16	194	35664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	29792
SPHINCS$^+$-256f	32	68	17	10	30	16	254	49216

where the last equation is used in the Sage script of the SPHINCS$^+$ specification [7], [9], [11]. Table 3 shows the six parameter sets of the SPHINCS$^+$ specification [7], [9] that are obtained with Eq. (1) and Eq. (2).

To find new parameter sets, we use a more precise estimate of the signing speed. Whereas Eq. (2) counts the calls to the tweakable hash functions **F** and **H**, we count the calls to the underlying hash function SHAKE256. A call to the pseudorandom function **PRF** invokes SHAKE256 once and a call to the robust instantiation of the tweakable hash function $\mathbf{T}_\ell$ (of which **F**, **H**, and $\mathbf{T}_{\mathsf{len}}$ are special cases) invokes SHAKE256 twice [9, §7.2.1] :

$$\begin{aligned} &\mathbf{PRF}(\mathbf{SEED}, \mathbf{ADRS}) \\ &= \text{SHAKE256}(\mathbf{SEED}\|\mathbf{ADRS}, 8n) \\ &\mathbf{T}_\ell(\mathbf{PK}.\text{seed}, \mathbf{ADRS}, M) \\ &= \text{SHAKE256}(\mathbf{PK}.\text{seed}\|\mathbf{ADRS}\|M^{\oplus}, 8n) \\ &M^{\oplus} = M \oplus \text{SHAKE256}(\mathbf{PK}.\text{seed}\|\mathbf{ADRS}, l) \end{aligned}$$

Based on the function calls of Table 2, we compute the number of calls to the underlying hash function as follows.

$$\begin{aligned} &\text{speed}' \\ &= 2{\cdot}(\text{num. of calls to } \mathbf{F}) + 2{\cdot}(\text{num. of calls to } \mathbf{H}) \\ &\quad + (\text{num. of calls to } \mathbf{PRF}) + 2{\cdot}(\text{num. of calls to } \mathbf{T}_{\mathsf{len}}) \\ &= 2(kt + d(2^{h/d})w{\cdot}\mathsf{len}) + 2(k(t-1) + d(2^{h/d} - 1)) \\ &\quad + (kt + d(2^{h/d})\mathsf{len}) + 2(d2^{h/d}) \\ &= d2^{h/d}(2w{\cdot}\mathsf{len} + \mathsf{len} + 4) + 5kt - 2(k + d) \end{aligned} \tag{3}$$

Finally, we search through a wider range of parameter values as suggested in the most recent version of the SPHINCS$^+$ Sage script [11] as follows.

$$\begin{aligned} h &\in \{56, 57, 58, \ldots, 83\} \\ \log t &\in \{3, 4, 5, \ldots, 23\} \\ k &\in \{1, 2, 3, \ldots, 63\} \\ d &\in \{4, 5, 6, \ldots, h-1\} \\ w &\in \{16,\ 256\} \end{aligned} \tag{4}$$

Table 4 Comparison of parameter sets

	parameters						bitsec	signature size		signing speed			
	n	h	d	$\log t$	k	w		bytes	ratio (%)	function calls	ratio (%)	runtime (sec)	ratio (%)
SPHINCS$^+$-128f	16	60	20	9	30	16	130	16976	100	262140	100	0.174840	100
SPHINCS$^+$-128f-A1	16	63	21	9	21	16	131	16144	95.1	248388	94.8	0.165918	94.9
SPHINCS$^+$-192s	24	64	8	16	14	16	196	17064	100	8042452	100	5.304968	100
SPHINCS$^+$-192s-A1	24	65	13	13	18	256	201	15744	92.3	6287490	78.2	4.232028	79.8
SPHINCS$^+$-192s-A2	24	65	13	11	22	256	198	16032	94.0	5775482	71.8	3.898911	73.5
SPHINCS$^+$-256f	32	68	17	10	30	16	254	49216	100	755986	100	0.500191	100
SPHINCS$^+$-256f-A1	32	64	16	10	36	16	258	49056	99.7	751256	99.4	0.497667	99.5

3.2 Results

We propose four new parameter sets: SPHINCS$^+$-128f-A1, SPHINCS$^+$-192s-A1, SPHINCS$^+$-192s-A2, and SPHINCS$^+$-256s-A1, where 'A' stands for "additional" or "alternative." The comparison of our parameter sets with the corresponding original parameter sets is given in Table 4.

parameters: Numerical values of SPHINCS$^+$ variables.

bitsec: Each version of the Sage script in [7], [9], [11] can sometimes output a slightly different bit security value. We used the most recent version of the Sage script [11].

signature size: Exact values of signature sizes are computed from Table 1. The column of "ratio (%)" sets the signature size of the original parameters as 100%.

signing speed: The column of "function calls" is the output of the Sage script with Eq. (3). The column of "runtime (sec)" is the benchmark result showing the median of 10,000 runs on 3.2GHz Intel(R) Core(TM) i5-6550 with the C reference implementation compiled with gcc-7.4.0. The column of "ratio (%)" sets the signing speed of the original parameters as 100%.

SPHINCS$^+$-128f-A1 improves both the signature size and the signing speed by approximately 5%. We propose two parameter sets for 192s; SPHINCS$^+$-192s-A1 provides 7.7% shorter signature and SPHINCS$^+$-192s-A2 provides 26.5% faster signing. Finally, the improvement of SPHINCS$^+$-256s-A1 is less than 1%. We could not find better parameter sets for 128s, 192f and 256s.

4. Conclusion

SPHINCS$^+$ is a state-of-the-art post-quantum hash-based signature framework that supports various tradeoffs by selecting different parameter sets. With a more precise estimate of the signing speed and a wider range of parameter values, we could find new parameter sets that provide faster signing and shorter signature sizes. For a given security level, most signature schemes do not allow performance tradeoffs by parameter selection and thus speeding up requires algorithmic techniques (e.g., [12]). Flexible parameter selection is one of the distinguishing and nice characteristics of SPHINCS$^+$.

References

[1] P.W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM J. Comput., vol.26, no.5, pp.1484–1509, 1997.

[2] D.J. Bernstein, "Grover vs. McEliece," Proc. Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010, pp.73–80, 2010.

[3] D.J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: practical stateless hash-based signatures," Advances in Cryptology - EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, vol.9056, pp.368–397, 2015.

[4] O. Goldreich, The Foundations of Cryptography, Volume 2, Basic Applications, Cambridge University Press, 2004.

[5] A. Hülsing, "W-OTS+ – shorter signatures for hash-based signature schemes," Progress in Cryptology - AFRICACRYPT 2013, Cairo, Egypt, June 22-24, 2013. Proceedings, vol.7918, pp.173–188, 2013.

[6] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," Information Security and Privacy, 7th Australian Conference, ACISP 2002, Melbourne, Australia, July 3-5, 2002, Proceedings, vol.2384, pp.144–153, 2002.

[7] D.J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M.M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, and P. Schwabe, "SPHINCS$^+$ – Submission to the NIST post-quantum project," November 2017. Specification document of the 1st round NIST submission package.

[8] A. Hülsing, J. Rijneveld, and F. Song, "Mitigating multi-target attacks in hash-based signatures," Public-Key Cryptography - PKC 2016, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I, vol.9614, pp.387–416, 2016.

[9] J.P. Aumasson, D.J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M.M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, and P. Schwabe, "SPHINCS$^+$ – Submission to the NIST post-quantum project," March 2019. Specification document of the 2nd round NIST submission package.

[10] NIST, "Post-quantum cryptography standardization," https://csrc.nist.gov/projects/post-quantum-cryptography, accessed Dec. 13, 2019.

[11] D.J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS$^+$ signature framework," Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, Nov. 11-15, 2019, pp.2129–2146, 2019.

[12] G. Horng, "Accelerating DSA signature generation," Cryptologia, vol.39, no.2, pp.121–125, 2015.