SPHINCS$^+$

Modifications for Round 2 submission to the NIST post-quantum project

Jean-Philippe Aumasson, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe

March 14, 2019

1 SPHINCS$^+$-'simple' and SPHINCS$^+$-'robust'

The updated Round 2 submission of SPHINCS$^+$ introduces instantiations of the tweakable hash functions similar to those of the LMS proposal for stateful hash-based signatures [1]. These instantiations are called 'simple' (compared to the established instantiations which we now call 'robust'). The 'simple' instantiations omit the use of bitmasks, i.e., no bitmasks have to be generated and XORed with the message input of the tweakable hash functions **F**, **H** or **T**. This has the advantage of better speed since the calls to the underlying hash function (needed in order to generate the bitmasks for each tweakable hash calculation) are saved. However, the resulting drawback is a security argument which in its entirety only applies in the random oracle model.

Another reason to propose these simple instantiations is the possibility to align the construction with the stateful scheme [1] such that clients can easily implement the verification procedure for both with a small code-base, as for the robust instantiations and XMSS. However, the simple instantiations of SPHINCS$^+$ are so far not compatible with the LMS signature scheme as described in [1]. The simple instantiations of SPHINCS$^+$ uses **PK**.`seed` and **ADRS** to distinguish hash calls. LMS uses a specially crafted security string which has the same purpose, is similar, but differs in the details.

2 Performance Optimizations for SHA-256 instantiations

Most of the time in SPHINCS$^+$ is spent on **F** computations. In the SPHINCS$^+$-SHA-256 instantiation we reduce the cost of **F** computations by carefully optimizing the length and format of the SHA-256 input. For this purpose we compress the used hash addresses and ensure that key-pair-dependent inputs fill a full input block. This allows to precompute the internal state of SHA-256 after absorbing the key-pair-dependent data and reduces the necessary online computations by a factor of two. For the simple instantiations the online part of evaluating **F** is a single compression function call.

For the SPHINCS$^+$-SHAKE256 and SPHINCS$^+$-Haraka instantiations such an optimization is of no effect as one **F** computation already takes only a single call to the inner function.

3 Security Evaluation

The 'Security Evaluation' section presented a security reduction for SPHINCS$^+$ based on a statistical assumption that should not hold for a good cryptographic hash function. At the time of writing, we believe to have found an alternative approach to extend the proof to other functions – especially to functions that behave like random functions. This work will be made available via the NIST pqc-forum mailing list, as well as via the SPHINCS$^+$ website `https://sphincs.org`, upon publication. The existing proof is also limited to the robust instantiations and does not cover the simple instantiations. We will also publish an extended proof covering the simple instantiations and make it available via the above channels.

4 New Team Members

We are happy to announce that Jean-Philippe Aumasson from the Gravity-SPHINCS team joins the SPHINCS$^+$ team.

References

[1] David McGrew, Michael Curcio, and Scott Fluhrer. Hash-based signatures. Internet Draft, IETF Crypto Forum Research Group, 2019. 2