ERICSSON

Date: October 8, 2024

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on FIPS 202 and SP 800-185 Decision Proposal

Dear NIST,

Thanks for your continuous efforts to produce well-written, user-friendly, and open-access security documents.

We welcome NIST's plans to update FIPS 202, revise SP 800-185, and to specify and approve additional SHA-3 derived functions, including those for authenticated encryption with associated data (AEAD). Below, we offer our feedback on the FIPS 202 and SP 800-185 decision proposal:

- The decision proposal states that NIST intends to revise SP 800-185 to provide "streaming" specifications of SHAKE128 and SHAKE256. While we strongly support improved APIs in implementations, our understanding of FIPS 202 is that such "streaming" APIs are already allowed. This understanding is based on the following clause:

  > *"For every computational procedure that is specified in this Standard, a conforming implementation may replace the given set of steps with any mathematically equivalent set of steps. In other words, different procedures that produce the correct output for every input are permitted."*

  We interpret APIs such as SHAKE(M, d) as mathematical steps in the computation process. For example, a single call to multiply() could be replaced by multiple calls to add(). The update to FIPS 202 should make this more explicit and reference that alternative APIs are discussed in SP 800-185. The announcement about the related upcoming change to FIPS 180-4 [1] stated that the revision will "discuss" alternative APIs, making it clear that APIs supporting streaming are already allowed and can be made compatible with FIPS 180-4. Hardware and software implementations with limiting APIs are detrimental for innovation.

- We support NIST's plans to specify and approve additional SHA-3 derived functions, including those for authenticated encryption with associated data.

    - Cryptographic agility is very important. Mobile systems have always been designed with this principle in mind and strive to have at least two different cryptographic algorithms implemented for each cryptographic functionality. In case vulnerabilities (theoretical or implementation) are found in one algorithm, it should be possible to quickly switch to another algorithm. If AES is broken, NIST does not have any approved backup AEAD for general use.

    - Having a single cryptographic primitive like Keccak that efficiently provides all the necessary symmetric cryptographic functions (KDF, MAC, AEAD, etc.) is a desire for many developers and hardware implementors. IETF has already standardized the use of Keccak for encryption [2] and more uses have been discussed. In addition to AEAD, the new special publication should specify a length preserving mode of encryption, which could just be the AEAD with a tag length of zero. A reason that IETF specified the use of Keccak for encryption was that current AEAD APIs could not be used for length-preserving encryption, which was a very strong requirement.

    - Current encryption algorithms exhibit significant security and usability limitations. The security of GCM and CCM is limited by the narrow 128-bit block size of AES and the 128-bit digest size of GHASH. Neither provides commitment security, both are vulnerable to release of unverified plaintext, and except for CCM with short tags, they do not behave like ideal MACs and lack reforgeability resistance. The confidentiality advantage in GCM and CCM is quadratic in the number of queries. The integrity advantage in GCM is linear in the plaintext length and the expected number of forgeries is quadratic in the number of queries. The integrity advantage in CCM is linear in the plaintext length and quadratic in the number of queries, and the expected number of forgeries is cubic in the number of queries. Furthermore, the limited nonce length and absence of nonce-resistance make both GCM and CCM unsuitable for use with random nonces.

      We think it is important with NIST approved encryption algorithms with better confidentiality advantages, fewer opportunities for users and developers to shoot themselves in the foot, and that behaves like ideal MACs with tag lengths between 32 bits and 128 bits. A Keccak-based AEAD mode could address several of these limitations. Even if it does not completely resolve them, it would be a step toward developing a Keccak-based accordion that could. Looking ahead, we would like to see Robust Authenticated Encryption (RAE) with succinct commitment and very good security bounds. A standardized encryption algorithm capable of handling many keys, many invocations per key, and long plaintexts without users and developers having to care too much about various limits and implementation pitfalls would be highly beneficial.

      We do not have any specific suggestions for non-accordion Keccak-based AEAD modes, but we think (docked) double-decker is a promising approach for VIL-SPRPs. Our understanding is that the upcoming special publication is likely to specify a duplex mode of (Turbo)SHAKE. Duplex mode has applications beyond encryption and there does not appear to be any reasons to disallow duplex mode for key derivation. The multi-sponge / multi-extract-expand pattern is widely employed in security protocols for key derivation. In line with NIST's

principles that cryptographic standards should be designed to "minimize the demands on users and implementers, as well as the adverse consequences of human mistakes and equipment failures," we believe that keyed hash APIs should be designed to prevent leakage of the secret state. The compromise of the secret state has more serious consequences than compromise of a single key.

- Several specifications of Keccak-based AEAD specify the use of sessions that takes care of nonce handling and replay protection. We strongly agree with this design principle and suggest that the new special publication specifies the use of sessions. Users, application developers, and protocol designers should ideally not have to deal with nonces and replay protection. Replay protection should be a mandatory requirement unless careful analysis shows that replay can be tolerated in some limited part of the system, such as 0-RTT in TLS 1.3. Our expectation is that the sessions in the new special publication would be limited to use cases with reliable and ordered delivery. Sessions applicable for unreliable and unordered delivery would likely require a Keccak-based accordion and an AERO-like derived function.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

[1] NIST Requests Public Comments on FIPS 180-4, Secure Hash Standard (SHS)
https://csrc.nist.gov/news/2022/public-comments-requested-on-fips-180-4-shs

[2] IETF RFC 9528 Ephemeral Diffie-Hellman Over COSE (EDHOC)
https://datatracker.ietf.org/doc/rfc9528/