ERICSSON

Date: September 9, 2022

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on FIPS 180-4, Secure Hash Standard (SHS)

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. FIPS 180-4 is a very important document that should be updated.

Please find below our comments on FIPS 180-4:

- “any change to the message will, with a very high probability, result in a different message digest”

  We suggest reformulating this to better illustrate the avalanche effect [1]. For example: "even a small change to the message will, with a very high probability, result in a message digest that appears uncorrelated with the old message digest (avalanche effect)”.

- “This Standard specifies secure hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256.”

  The standard also specifies SHA-512/t. It would be good to state that SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 are “fixed-length hash functions” to differentiate from more modern “variable-length hash functions”. SHA-512/t is currently defined as a set of 510 fixed-length hash functions. We think it would be much more natural to describe SHA-512/t as a variable-length hash function. We do not think NIST should continue to use the term “eXtendable Output Function (XOF)”. It is very common to describe SHAKE as a variable-length hash function [2] and NIST already describes TupleHash as a variable-length hash function [3]. In addition to SHA-512/t, the introduction should also mention the weak mechanism for producing variable-length message digests described in section 7.

- SHA-1 is not to be considered a cryptographically secure hash algorithm and should not be included in a document titled 'Secure hash algorithms'. We are supportive to deprecate SHA-1 and to establish a strict timeline to as soon as possible disallow all uses of SHA-1. NISTs firm timelines for disallowing 3DES [4] and for requiring TLS 1.3 [5] has had a large impact and greatly benefited society as a whole. We think NIST should continue this habit and publish firm timelines not only for disallowing SHA-1 but also for deprecating SHA-224, and for requiring DTLS 1.3. Without firm timelines from NIST, companies can continue to compete on the market with old obsolete algorithms and still meet NIST requirements. As long as one vendor only support a

legacy algorithms like SHA-1 or 3DES, all other vendors are forced to enable the same legacy algorithm. All security conscious companies and industries started phasing out SHA-1 a long time ago so deprecation should not be a problem. Hopefully there are no planned SHA-1 implementations anywhere, and if there are we think NIST should ignore them. Any fielded implementations should have crypto agility. The only current use of SHA-1 we are aware of in the mobile industry is in IMS media security but new recommended alternatives like SHA-256 and AES-GCM have already been introduced. NIST 800-131A [4] put the deadline to disallow 3DES less than 4 years into the future. As SHA-1 has been known to be broken since 2005 most industries have already largely phased out SHA-1 and a shorter deprecation period should be possible in this case.

- SHA-224 and SHA-512/224 are intended to be used with 3DES for 112-bit security. 3DES is according to NIST disallowed from 2023 [4] and 112-bit security in general can according to NIST [6] requirements only be used until 2031 minus the security life of the protected data. For many use cases, 112-bit security is therefore already forbidden according to NIST requirements and when the update to 180-4 is published the number of use cases will be even smaller. Most industries have already phased out 112-bit symmetric algorithms. RSA-2048 which also have a 112-bit security level is overwhelmingly used with SHA-256. We suggest that NIST not only removes SHA-1, but also SHA-224. SHA-224 and SHA-512/224 have never seen any significant deployment.

- The update to FIPS 180-4 should discuss that SHA-2 can be implemented as a running hash using e.g., an "init(), update(M_i), digest = finalize()" interface. This type of interface is already supported by many libraries such as Java and OpenSSL and is very useful, especially on constrained devices with small amounts of memory. Running hashes are discussed as an implementation option in TLS 1.3 and the lack of running hash interfaces on some platforms is discussed as a problem in the EDHOC protocol specification [5]. The important thing is to describe that data can be provided in several different calls to update(M_i). The actual API is likely better left to implementations. The init() API is e.g., not strictly needed in an implementation and could be called automatically the first time update(M_0) is called.

- NIST should also update FIPS 202 [8] and NIST SP 800-185 [3] to discuss that SHA-3 can be implemented as a running hash. We also strongly think that NIST should update [8] and [3] to include the Keccak Duplex construction [9] and a suitable interface like "init(), digest = update(M_i, d)". The duplex interface can be seen as a generalization of the running hash interface. KMAC in duplex mode is very useful in many applications and would be perfect for implementation of transcript hashes in future security protocols. The security of the duplex construction can be shown to be equivalent to the sponge construction.

- FIPS 180-4 should be updated with a discussion on length extension attacks. The low resistance against length extensions in many of the SHA-2 variants is not very nice and might come as a bad surprise to people using SHA-2. As stated by NIST [10] failure to meet resistance to length-extension attacks is to be "considered a serious attack". It is unfortunate that NIST did not mitigate the length-extension attacks on SHA-2 already when they were pointed out by John Kelsey in the comments to SP 180-2 [13]. NISTs comment at the time "It would be more appropriate for the perceived weaknesses to be addressed in application standards" has not aged well. The fact that most of the SHA-2 functions lack resistance to length-extension attacks and the fact that NIST did not address the weakness is giving both NIST and SHA-2 unnecessary negative attention [12]. Regardless of how likely length-extension attacks are to cause practical

attacks, it points to a fundamental design flaw in the construction. Secure hash algorithms in 2022 should not suffer from length extension attacks.

- FIPS 180-4 should be updated with a table similar to Table 4 in NIST FIPS 202 [8] listing the security against various attacks including length extension attacks.

- FIPS 180-4 should be updated with a discussion on the security weaknesses with the truncation mechanism to produce variable-length message digests described in section 7. This mechanism produces hashes with worse properties that the SHA-512/t mechanism described in section 5.3.6 and the KMAC construction in [3]. While it is infeasible to find any relation between a SHA-512 hash and a SHA-512/256 hash of the same message, there is a trivial relation between a SHA-256 hash and a SHA-256 hash truncated to 64 bits. Secure hash algorithms in 2022 should not use simple truncation as a mechanism to produce variable-length digests.

- FIPS 180-4 states that SHA-512/t is not an approved hash algorithm except for t = 224 or t = 256. FIPS 180-4 does not state anything similar for SHA-512 truncated to t bits using the weak mechanism in section 7 giving the impression that such use is approved. This does not make sense as SHA-512/t is a much better construction with better security properties. NIST should make it clear that SHA-512/t is preferred over SHA-512 truncated to t bits.

- FIPS 180-4 describes how the initial hash values for SHA-256, SHA-384, SHA-512, and SHA-512/t were obtained. NIST should also explain how the initial hash values for SHA-1 and SHA-224 were chosen, if the hashes are not altogether removed from the document.

- FIPS 180-4 describes how the constants in SHA-224, SHA-256, SHA-384, SHA-512, and SHA-512/t were obtained. NIST should also explain how the constants in SHA-1 were chosen, if the hash is not altogether removed from the document.

- The fixed-length hash functions SHA3-224, SHA3-256 SHA3-384, SHA3-512 [8] were designed as drop-in replacements for all uses of SHA-224, SHA-256, SHA-384, SHA-512 including HMAC. The fixed-length SHA-3 hash functions have however seen little or no practical use. Instead the variable-length functions such as SHAKE and KMAC have seen significant practical use in implementations as well as in published and upcoming standards such as EdDSA (RFC 8032), XMSS (RFC 8391), LMS (NIST SP 800-208), CMS (RFC 8702), RSASSA-PSS and ECDSA (FIPS 186-5 (Draft), RFC 8692), COSE (draft-ietf-cose-hash-algs), EDHOC (draft-ietf-lake-edhoc), CPace (draft-irtf-cfrg-cpace), FROST (draft-irtf-cfrg-frost), OPRF (draft-irtf-cfrg-voprf), CRYSTALS-Kyber, CRYSTALS-Dilithium, Falcon, and SPHINCS+. For companies and industries striving for crypto agility it is an increasing problem that there are no NIST approved drop-in replacements for SHAKE128 and SHAKE256 and derived functions such as cSHAKE and KMAC. We hope that the LWC project will standardize such drop-in replacements. NIST could also specify variable-length hash functions based on SHA-256 and SHA-512 following the SHA-512/t construction but with some additional small tweak to mitigate length-extension attacks [11].

- NISTs use of the /t notation in SP 180-4 and SP 800-208 [11] is very unfortunate and affects usability. The current situation is that /t as way to create variable-length digests means three different things for SHA-256, SHA-512, and SHAKE256. For SHA-256, /t just means truncation, which is weak, for SHA-512, /t means choosing function t in the set SHA-512/t of 510 fixed-length hash functions, and for SHAKE256, /t means setting $\mathrm{d} = \mathrm{t}$ in the variable-length hash function SHAKE256(M, d). NIST seems to have been hesitant to introduce variable-length variants of SHA-

2, but the fact is that both the weak truncation mechanism in section 7 and the strong truncation mechanism in 5.3.6 can be seen as variable-length hash functions.

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

[1] Al-Kuwari, Davenport, Bradford, “Cryptographic Hash Functions: Recent Design Trends and Security Notions”
https://eprint.iacr.org/2011/565.pdf

[2] IETF RFC 8692 “Internet X.509 Public Key Infrastructure: Additional Algorithm Identifiers for RSASSA-PSS and ECDSA Using SHAKEs”
https://eprint.iacr.org/2011/565.pdf

[3] NIST SP 800-185 “SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash”
https://doi.org/10.6028/NIST.SP.800-185

[4] NIST SP 800-131A Rev.2 “Transitioning the Use of Cryptographic Algorithms and Key Lengths”
https://doi.org/10.6028/NIST.SP.800-131Ar2

[5] NIST SP 800-52 Rev. 2 “Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations”
https://doi.org/10.6028/NIST.SP.800-52r2

[6] NIST SP 800-57 Part 1 Rev. 5 “Recommendation for Key Management: Part 1 – General”
https://doi.org/10.6028/NIST.SP.800-57pt1r5

[7] Ephemeral Diffie-Hellman Over COSE (EDHOC)
https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/

[8] NIST FIPS 202 “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions”
https://doi.org/10.6028/NIST.FIPS.202

[9] Bertoni, Daemen, Peeters, Van Assche, “Duplexing the sponge: single-pass authenticated encryption and other applications”
https://keccak.team/files/SpongeDuplex.pdf

[10] NIST “Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family”
https://www.govinfo.gov/content/pkg/FR-2007-11-02/pdf/FR-2007-11-02.pdf

[11] NIST SP 800-208 “Recommendation for Stateful Hash-Based Signature Schemes”

https://doi.org/10.6028/NIST.SP.800-208

[12] David Wong, “How did length extension attacks made it into SHA-2?”
https://www.cryptologie.net/article/417/how-did-length-extension-attacks-made-it-into-sha-2/

[13] Kelsey “Public Comments on the Draft Federal Information Processing Standard (FIPS) Draft FIPS 180-2, Secure Hash Standard (SHS) (2001)”
http://csrc.nist.gov/encryption/shs/dfips-180-2-comments1.pdf