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

Comment period: June 9, 2022 – September 9, 2022

On June 9, 2022, NIST's Crypto Publication Review Board announced a proposal to revise Federal Information Processing Standards (FIPS) 180-4 Secure Hash Standard (SHS).

The comments that NIST received during the comment period from June 9, 2022, to September 9, 2022, are collected below.

More information about this review is available from NIST's Crypto Publication Review Project site.

LIST OF COMMENTS

1. Comments from Gary Peterson (NOAA Affiliate), June 9, 2022

Until you dump SHA1 formally and completely, it will be the go-to encryption allowed on every connection. Sure, we have better, but we also offer weak ciphers involving SHA1. Just say SHA1 is dead so we can move on to secure things.

2. Comments from Stiepan A. Kovac (QRCrypto SA), June 16, 2022

Please consider disallowing SHA1 without delay being that it is clearly established as broken, with few exceptions:

https://www.din.de/resource/blob/236540/5b946078899f420e2b15fb64e3ca3e17/20170425-sc-27-statement-of-sha-1-data.pdf

3. Comments from Danny Niu (NJF), June 23, 2022

This comment focuses on 1 point with regard to the possible removal of SHA-1 from a future revision of FIPS-180.

I'd oppose removing SHA-1 in its entirety from the FIPS, for the following reasons:

1. SHA-1 has been sed in some applications extensively, most notably the Git SCM.

2. SHA-1 can be secure when not relying on it for collision resistance, and it's considered still secure for randomness derivation.

I think the IT industry would still prefer to see an authoritative specification of SHA-1. But we can, and I would prefer to "Move" it, to a dedicated "obsolete algorithms" section.

4. Comments from Jeff Mears (Blizzard), August 22, 2022

I have some minor comments on non-normative parts of the Secure Hash Standard specification (FIPS 180-4).

In the current published version (August 2015), in section 5.3.2, SHA-224's initial hash value, there is no mention of the origin of these constants, whereas 256, 384, 512 and 512/t have origins. I feel that this sentence or similar should be added to 5.3.2:

"These words were obtained by taking the thirty-third through sixty-fourth bits of the fractional parts of the square roots of the ninth through sixteenth prime numbers."

And another clarification sentence, if desired; something like this:

"That is, these are the thirty-two least significant bits of each of the SHA-384 initial hash value words, shown below."

SHA-1's initial hash value is also not described, but it's obvious that they're just simple hexadecimal nibble patterns, and NIST plans to remove SHA-1 anyway.

5. Comments from Aleksandr V Tereschenko (Intel), August 29, 2022

Intel Product Assurance and Security (IPAS) Cryptology team's comments in response to NIST Request for Public Comments on FIPS 180-4

1. We support the SHA1 deprecation idea. Due to known practical attacks, like the one mentioned by NIST in the call for comments, any use of this function in the collision-resistance context is clearly broken and even though the preimage resistance is not formally compromised yet, the trust is already lost. At the same time, we don't have any specific input as to the deprecation timelines, other than suggesting to deprecate it sooner rather than later, and leave it to other comments to help NIST decide.
2. As far as the "Init, Update, Final" interface is concerned, we don't think adding this layer into FIPS 180-4 would be helpful. While this interface is frequently used, its use and indeed existence heavily depend on the particular solution context, so on the one hand it will probably be hard to meaningfully standardize it other than at a high level (begging the question of what value that would bring). On the other hand, there's usually no requirement on the cross-solution interoperability at that level, which is usually one of the main drivers for standardization. If NIST decides to pursue this direction though, one suggestion we have is to generalize it to include not only hash functions, but also MAC and relevant encryption algorithms (e.g. GCM, CTR mode), which allow "on-line" processing approach reflected by the "Init, Update, Final" high-level interface.

6. Comments from Sophie Schmieg (Google), Stefan Kölbl (Google), September 7, 2022

NIST has put out a request for public comment for FIPS 180-4, in particular concerning the deprecation of SHA-1 and discussing an Init/Update/Finalize API to the standard.

Deprecation of SHA-1

SHA-1 has long been broken and is no longer collision resistant. While it still provides preimage and second preimage resistance, we think it is time to start the deprecation of this hash function. Special care should be taken when it comes to applications that only rely on the preimage/second preimage resistance, like HMAC-SHA1. While this usage should also be phased out, the fact that this primitive is not broken, and still widely used in legacy applications. In particular, this means that removing SHA-1 from all security relevant contexts without consideration of cases where it is used as hash function for HMAC-SHA1 would be fairly expensive without necessarily improving security. It might make sense to have an incremental approach, phasing out HMAC-SHA1 over, for example, 5 years.

Adding an Init/Update/Finalize API

As opposed to authenticated encryption, where an Init/Update/Finalize API is usually quite dangerous, as it can leak part of the key material if misused, we are not aware of any reason to not use Init/Update/Finalize for hash functions or HMAC, and have been using this API, available in OpenSSL and BoringSSL widely, with many applications otherwise being infeasible.

Truncated Versions

The standard allows multiple ways to get a hash function with e.g. 256-bit output from SHA-512. There is the concrete definition with SHA-512/256, but Section 7 would allow us to use SHA-512 and truncate the output to 256-bit. Add some guidance text that clarifies that while this is not an issue by itself, discuss how to deal with this ambiguity by pointing out use cases which require domain-separation to use SHA-512/256 over truncating SHA-512 directly.

7. Comments from John Mattsson (Ericsson), September 9, 2022

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 d = 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.

[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

8. Comments from Panos Kampanakis (Amazon), September 9, 2022

Regarding SHA-1, we welcome disallowing SHA-1. One concern is legacy systems. There is additional compatibility complexity around using SHA-2 with RSA-OAEP. Not all implementations do the same thing. For example, Java (by default) will use SHA-2 for the message digest but will continue using SHA-1 for the MGF. This isn't a serious security issue, but we should probably ensure that SHA-1 for the MGF remains allowed for the foreseeable future. What's more, although not widely used any more, SHA-1 is still OK when used in HMAC. Additionally, we know NIST is aware that TLS 1.2 and SSH allow SHA-1 as the digest and for key based authentication which practically cannot be exploited on the fly. NIST has made provisions for TLS in SP 800-52r2 and SP.800-131Ar2 calls out that there are protocol exceptions.

Thus, overall we welcome disallowing SHA-1, but would like to see more adoption of SHA-3 and a little more time (no more than 5 years) for hopefully phasing out these old implementations.

More suggestions:

We are sure it will be included, but we would encourage the **inclusion of SHA-3 and XOFs (SHAKES)** in the document or just a reference the SHA-3 and SHAKE options from FIPS PUB 202.

Regarding the **Init, Update, Final Interface**, we would welcome the discussion of the interface in the SP. It would clarify how libraries also implement these algorithms.

We also want to propose the discussion of **length extension attacks** in Merkle-Damgard constructions, how they can be exploited, how practical they are (not at all) in common standards where SHA-2 is used, and how these types of attacks do not apply to Sponge constructions like SHA-3.