Comments Received on SP 800-185

Comments Received #1

From: "Willumsen, Kraig" <Kraig.Willumsen@ttiinc.com>
Date: Friday, August 5, 2016 at 12:10 PM

My name is Kraig Willumsen, Product Technologist for TTI Inc, Fort Worth TX, acting as Engineering Support for BAE SYSTEMS. My responsibilities include to an extent ensuring TTI cryptography compliance when sending secure BAE documentation. My concerns center around what appears to be a *reverse* in cryptographic (encryption) standards.

Back in 2010, there was a significant push (supported by the Government) to adopt the **SHA Std** of Encryption. Currently, many are now in process of becoming **FIPS 140-2 Compliant** to **AES-256** encryption level, with a deadline Of Midnight, December 31, 2017.

Today I have read the draft of SP800-185, and my questions are below:

- Will these new measures *undo* what will be done bringing AES-256 on board, or will these SHA features be seamless while operating in an AES-256 environment?
- Is the SHA standard destined to *replace* AES-256?
- If so, what time frame would this take place, and become a new cybersecurity requirement?

Thank you for your time and consideration.

Kraig Willumsen
Product Technologist, TTI Inc, Fort Worth Texas
Engineering Support, BAE SYSTEMS.

NIST: Hash functions are not encryption functions, so the SHA Standards that specify NIST-approved hash functions have nothing to do with an encryption standard like AES.

Comments Received #2

From: "Chang, Kou-Chuan" <kou-chuan.chang@boeing.com>
Date: Friday, August 5, 2016 at 3:48 PM

Sp800-185 page 19, line 632 and table 1, on approved MAC algorithms CMAC[5] and HMAC[6],
Question:
Do HMAC-SHA256(K, text) and HMAC-SHA512(K, text) include SHA3-256, and SHA3-512, or only refer to SHA2-256 and SHA2-512?
If SHA3 224/256/384/512 are approved HASH functions, does
"National Institute of Standards and Technology, *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards (FIPS) Publication 198-1, July 2008, 13" include SHA3 or only include SHA2? (SHA3 is approved many years after the HMAC standard).

Thanks much
KC Chang

NIST: HMAC is approved for use with the SHA-3 hash functions, and the security properties will be very similar to those of HMAC with the equivalent SHA-2 hash functions.

That said, the text you cited only refers to the SHA-2 hash functions. If we were referring to SHA-3, we would write it as HMAC-SHA3-256, HMAC-SHA3-512. The examples in the table are merely meant to be illustrative, not exhaustive. Again, the SHA-3 hash functions are approved for use in any application where the equivalent SHA-2 functions are approved, and that includes HMAC.

Comments Received #3

From: martin andrew <martin_andrew_n@hotmail.com>
Date: Friday, August 5, 2016 at 6:14 PM
Subject: comments ParallelHash

Assuming strings & artificial intelligence are now incorporated in blocks.....
Microsoft xml && v3 spark are capable of these ParallelHash.....
Xerox peptide counters may be applied to hex/dna strings .....

Applications ?

Crysler citroen (jack nicholson hydrogen powered cadillac)) plants?

Thales ratp train synchronisation of doors with security cameras ?

& of course

Traffic control by sensor captor sequence ?

Martin Andrew l n

PSie --paris

No comment from NIST.

Comments Received #4

From: Chang, Kou-Chuan [mailto:kou-chuan.chang@boeing.com]
Date: Friday, August 05, 2016 at 5:16 PM

Appreciate your fast reply.

What are the advantages of KMAC256 compared to HMAC-SHA3-512?
Variable output length?
S:customization bit string?

NIST: In addition to the factors you mentioned, KMAC256 is somewhat faster than HMAC-SHA3-512, especially for short messages.

Comments Received #5

From: NSA
Date: Wednesday, September 14, 2016 at 4:15 PM
Comments on Draft of NIST SP 800-185, dated August 2016

Abstract:
p. ii (line 100): "...four types of SHA-3-derived functions: cSHAKE,..." NIST: Done.

Section 1: In the 2nd bullet of the 4th paragraph
Remove "correctly" from "providing functions that hash tuples of input strings correctly and unambiguously". This gives the reader the impression that there is an already-defined "correct" way to hash tuples of input strings. (Also see section 5.1 – the same change is suggested).
NIST: Both done.

Section 1: In the 4^{th} bullet of the 5^{th} paragraph the claim that any change in the requested output length completely changes the function and yields unrelated outputs is not true for cSHAKE. Note that the fact that it is not true for cSHAKE is also mentioned later in Section 4.1.
NIST: Revised.

Section 2.2: Use the notation "mod(a,b)" or notation "a mod b" consistently between this section, section 2.3.3 and Appendix B. Suggest, citing the restriction of a and b as integers for added specificity. NIST: Done.

Section 2.3.1: The first step of right_encode(x) and left_encode(x) should stipulate that n is the smallest positive integer for which $2^{8n} > x$. Otherwise, if x = 0 (which is allowed by the validity conditions) then n does not exist. NIST: Done.

Section 2.3.1: The font of On+1 in step 5 of right_encode(x) does not match the font of O1, O2, etc. Similarly, The font of On in step 5 of left_encode(x) does not match the font of O0, O1, etc.
NIST: Corrected.

Section 3.2 Parameters:
p. 7 (line 307): Move the reference to the footnote from the end of the 3^{rd} bullet to the end of the 4^{th} bullet (as otherwise the N referred to in the footnote will not yet have been defined).
NIST: Revised. NIST also switched the order of parameters *S* and *N*. Because NIST defines *N*, thus, can always make *N* a byte-oriented string. The order change can ease piecing strings together if *S* is not byte-oriented.
p. 7 (footnote 2): "...implement this function would be to set the default values..." NIST: Done.

Section 3.3 Definition:
p. 8 (line 344): "...sponge functions, respectively; ~~and~~ the characters..." NIST: Done.

Section 3.4: The third paragraph, replace “it will never be possible.” with “it will be very difficult.” While it does appear that this would be very difficult to do, it does not seem impossible. Even if the two inputs, public_key and email_contents are equal but different S values are used, it is still possible for the two 256-bit outputs to collide. While there will be a difference in the 1600-bit outputs after the final application of the permutation, the first 256 bits of these two outputs could be the same. NIST: Done.

Section 4.1: Remove the last word “below” of the paragraph. NIST: Done.

Section 4.3.1 KMAC with Arbitrary-Length Output:
p. 11 (lines 408-409): It is not clear what is meant by “...setting the encoded output length L to 0” and “...when called with an encoded length of zero”. Is this equivalent to setting the requested output length, namely L, to 0? That is, is KMAC called with L=0? (Also see section 5.3.1 and 6.3.1 where the same comment can be made)
NIST: Revised with additional pseudocode provided for all three Sections.

Section 4.2 Parameters:
p. 10 (line 380): “...of any length (up to 2^{2040}-1 bits), including zero.”
p. 10 (line 383): “...of any length (up to 2^{2040}-1 bits), including zero.”
NIST: Revised with inserted footnote (#2) on the maximum length allowed for all bit strings specified in this document.

Section 5 TupleHash:
p. 12 (line 417): “...in parentheses like (“a”,”b”,”c”,...,”z”) in this document.” (That is, put double quotes around the character strings, include a comma before the last character string “z”, and remove the italics font from the character strings.) NIST: Done.
p. 12 (line 429): “...of any length (up to 2^{2040}-1 bits), including zero.” NIST: See Footnote #2.

Section 6.2 Parameters:
p. 14 (line 475): The statement that the main input string X may be of any length is not quite true, since $\lceil len(X)/B \rceil < 2^{2040}$. NIST: Footnote (#10) inserted.
p. 14 (line 476): “It may be any integer ~~> 0~~ between 0 and 2^{2040}.” NIST: Done.
p. 14 (line 478): “...of any length (up to 2^{2040}-1 bits), including zero.” NIST: No change, see Footnote #2.

Section 6.3 Definition:
p. 14 (line 495): Remove step 3 (since i will be set to 0 in the next step). NIST: Done.
p. 15 (line 507): Remove step 3 (since i will be set to 0 in the next step). NIST: Done.

Section 7.1: In the 1st paragraph it is stated that cSHAKE is defined to fill one entire call to Keccak-f with the padded S and N strings. However, depending on the length of S and N, these

padded strings could fill multiple calls to Keccak-f. Perhaps it is better to say that cSHAKE is defined so that all the calls to Keccak-f to accommodate S and N (of which there may be more than one) will process exactly r bits, where r is the rate parameter. NIST: Revised.

Section 7.2 Limited Implementations:
p. 16 (line 536): "...and inputs including involving fractional bytes)..." NIST: Done.

Section 8.1.1 Equivalent Security to SHAKE for Any Legal S and N:
p. 18 (line 566): "...as SHAKE128(X,L); ~~and~~ cSHAKE256(X,L,S,N) has..." NIST: Done.

Section 8.1.2: In the title of the section change "Functions" to "Outputs". Similarly, change "functions" to "outputs" in the 3rd sentence of the 1st paragraph and the last sentence of the section. NIST: Revised.

Section 8.1.2 Different S and N Give Unrelated Functions:
p. 18 (line 569): "...two customization and name string~~s~~ pairs, and..."
NIST: Revised; the order of the customization string and name string is also reversed. The order switch was described in the comment resolution of "Section 3.2 Parameters".
p. 18 (line 570): "...suppose ~~x1and~~ x1 and x2 are..." NIST: Revised.
p. 18 (line 582): "There is, for example, no relationship between KMAC..." NIST: Done.

Section 8.3: In the 2nd sentence of the 2nd paragraph, change "makes" to "may make". NIST: Revised.

Section 8.3: In the last sentence of the 2nd paragraph, it is more accurate to say that a collision attack will require min($2^{L/2}$, 2^{128}) work and a preimage attack will require at least min(2^{L}, 2^{128}) work. Note also that this would be consistent with the security strength for SHAKE128 listed in Table 4 of FIPS 202. NIST: Done.

Section 8.4 Guidance for Using KMAC Securely:
p. 19 (line 613): "...and key lengths (up to 2^{2040}-1 bits). However, not all..." NIST: Done.

Section 8.4.2: In Table 1 CMAC(K, text) only provides a security strength that is equivalent to KMAC128(K, text, 128, S) if the underlying approved block cipher is AES (so that 128-bit blocks are used) and no truncation of the 128-bit tag is employed (truncation is allowed in NIST SP 800-38B). Otherwise, guessing the tag for CMAC(K, text) would be easier than guessing the tag for KMAC128(K, text, 128, S). This distinction should be noted. NIST: Revised.

Section 8.4.2: In Table 1 is there a reason only KMAC128(K, text, 128, S) is listed as having a security strength equivalent to CMAC(K, text)? KMAC256(K, text, 128, S) would appear to also give the same security strength (2^{64} work for collisions and 2^{128} work for preimages), but it is not listed. NIST: Revised.

Section 8.4.2: In Table 1 HMAC-SHA256(K, text) only provides a security strength that is equivalent to KMAC256(K, text, 256, S) if no truncation of the 256-bit tag is employed (truncation is allowed in FIPS PUB 198-1). Otherwise, guessing the tag for HMAC-SHA256(K, text) would be easier than guessing the tag for KMAC256(K, text, 256, S). Similarly, HMAC-SHA512(K, text) only provides a security strength that is equivalent to KMAC256(K, text, 512, S) if no truncation of the 512-bit tag is employed. These distinctions should be noted. NIST: Revised.

Section 8.4.2 KMAC Output Length:
p. 19 (lines 626-627): "...total number of invalid (message, MAC) pairs that can be submitted for verification failures under a given key." NIST: Revised.

Appendix B: In the last sentence of the 1st paragraph change "uniformly" to "uniform". Also change "over that range" to "over that range, assuming the above functions approximate a uniform random variable". NIST: Done.

Appendix B: At the end of step 3 of the procedure to create an X between 0 and R-1, insert "where the bits_to_integer function is defined below." NIST: Done.

Appendix B: It is not clear how the claim $Prob(t) - 1/R \leq 2^{-128}$ was derived. It only indicates how close Prob(t) could be to 1/R from above. Since no lower bound is given, it does not rule out something like $Prob(t) = 2^{-1000}$, which might not be close to 1/R at all. Instead, consider the following bound. Recall that k = ceiling(lg(R)) + 128. If R is a power of 2, then Prob(t) = 1/R. If R is not a power of 2, then Prob(t) is either $floor(2^k/R)/2^k$ or $(floor(2^k/R) + 1)/2^k$. Since $floor(2^k/R) \leq 2^k/R$, it follows that $Prob(t) \leq (2^k/R + 1)/2^k = 1/R + 1/2^k$. Similarly, since $floor(2^k/R) > 2^k/R - 1$, it follows that $Prob(t) > (2^k/R - 1)/2^k = 1/R - 1/2^k$. Thus, we have the bound

$$1/R - 1/2^k < Prob(t) \leq 1/R + 1/2^k.$$

NIST: Revised, and footnote inserted for the bound.

Appendix B: Is there a reason step 2 of the bits_to_integer function is boldface? NIST: Fixed.

General:
To improve clarity in this SP, provide reasoning behind the upper limit for inputs and outputs 2^{2040}. Additionally, for the same reason, provide test vectors for the functions in the draft.
NIST: Reasoning for the upper limit is provided in Footnote #2.
Test vectors will be provided on NIST's Cryptographic Toolkit page when they become available.

Comments Received #6

From: "Jenkins, Maria O. (Contractor)" <Maria.Jenkins.ctr@dcma.mil>
Date: Tuesday, September 20, 2016 at 8:59 AM

Good morning,

The Defense Contract Management Agency (DCMA) IT-K has reviewed NIST SP 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash (Draft). We have no comments to submit but have attached a completed comment form as an artifact of our review.

Thank you for your support.

Respectfully,
Maria Jenkins
Policy Analyst
DCMA Information Assurance Directorate (IT-K)
Network Security Systems Plus

Organization	Commenter	Type (add/del)	Line #	Section	Comment (include rationale for comment)
DCMA	Gilliam				Document reviewed. No comment.
DCMA	McIntosh				Document reviewed. No comment.
DCMA	Jenkins				Document reviewed. No comment.

Thank you for your comments!

Comments Received #7

From: "Harris, Michael W. (CDC/OCOO/OCIO)" <fnb0@cdc.gov>
Date: Monday, September 26, 2016 at 6:16 AM

CDC has no comments to provide on the *Draft NIST Special Publication (SP) 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash*.

Thank you for the opportunity to review and comment.

Michael Harris, CISSP Information Technology Specialist (Information Security)
Centers for Disease Control and Prevention

Comments Received #8

From: Ronny VAN KEER <ronny.vankeer@st.com>
Date: Thursday, September 29, 2016 at 3:26 AM

Dear Hash team,

This draft looks fine to us. We have no further comments on it.
We thank you for taking our suggestion of a XOF mode into account.
Will you also publish test vectors for the derived functions?

Kind regards,
Gilles, Guido, Joan, Michaël and Ronny
The Keccak & co. team

NIST: Test vectors will be provided on NIST's Cryptographic Toolkit page when they become available.