Changes in Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard

Quynh Dang

Abstract This paper describes the changes between FIPS 180-3 and FIPS 180-4. FIPS 180-4 specifies two new secure cryptographic hash algorithms: SHA-512/224 and SHA-512/256; it also includes a method for determining initial value(s) for any future SHA-512-based hash algorithm(s). FIPS 180-4 also removes a requirement for the execution of the message length encoding operation.

Keywords FIPS 180, hash algorithm, SHA-512, SHA-512/224, SHA-512/256, hash function

1. Background Information in FIPS 180-3

Cryptographic hash functions are very critical elements in building security components for computer systems. They are used in widely-deployed cryptographic applications, such as digital signature applications [1], Keyed-hash message authentication codes (HMACs) [2], hash-based key derivation functions [3] and Deterministic Random Bit Generators [4].

Federal Information Processing Standard (FIPS) 180-3 specified five cryptographic hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512. Among these five hash algorithms, there are three "base" algorithms: SHA-1, SHA-256 and SHA-512. However, only the use of SHA-256 and SHA-512 is discussed further in this paper.

SHA-224 and SHA-384 are based on SHA-256 and SHA-512, respectively, using different initial values (IVs) and truncating to the left-most 224 or 384 bits of the outputs from the functions for the final hash values.

Generally, within the same class of cryptographic algorithms, the algorithms with stronger security have poorer performance than those that provide weaker security. For example, a 2048-bit key for RSA provides much stronger security than a 1024-bit key and takes a lot more computation in signing and verifying a message than a 1024-bit key; this means that 2048-bit RSA has poorer performance than 1024-bit RSA. As another example, consider the SHA-256 and SHA-512 hash algorithms. SHA-512 is expected to provide much greater security than SHA-256, and SHA-512 is slower (poorer in performance) than SHA-256 on many computing platforms.

However, SHA-512 consumes roughly 10-45% fewer clock cycles per byte than SHA-256 as shown from performance-comparison data for SHA-256 and SHA-512 on many different 64-bit platforms by [5]. This means that SHA-512 runs roughly 10-80% faster than SHA-256 and SHA-224 on these 64-bit machines, which are becoming more

prevalent. Also, [6] provides performance comparison data for SHA-256 and SHA-512 on a specific 2010 Intel architecture, the Xeon X5670 processor. The data shows that SHA-512 consumes roughly 37% fewer clock cycles per byte than SHA-256. Put another way, SHA-512 is roughly 60% faster (more efficient) than SHA-256 on this machine.

Hash values of 224 and 256 bits are commonly used in many cryptographic applications, such as digital signature applications. Therefore, alternatives to SHA-224 and SHA-256 with the performance of SHA-512 to generate 224 and 256-bit hash values would be useful. Clearly, generating 224 and 256-bit hash values faster (than generating them by using SHA-224 and SHA-256, respectively) is possible when using SHA-512 with an additional truncation operation at the end; the specification of the truncation method is provided in NIST SP 800-107 [7].

To improve interoperability in applications or protocols, hash functions with specific hash value lengths are specified in object identifiers (OIDs). If only the hash function were identified in the OID, the length of the hash value (including any required truncation) would be ambiguous. For example, if SHA-512 were identified by its OID (meaning the length of hash values is 512 bits), but the hash value was intended to be truncated to 224, 256, or some other number of bits, there would be ambiguity in the application execution. Of course, this problem could be taken care of by specifying a specific length for the truncated hash value to be used in the protocol specification. However, introducing a new parameter into existing protocols would create a lot of unnecessary complications and incompatibility issues. Even with new protocols/applications, having a simple design is desirable. Thus, it makes sense to specify truncated variants of SHA-512 as new hash algorithms with their own OIDs.

Beside the need for new hash algorithms, another issue in FIPS 180-3 that needed to be dealt with concerned the requirement that the preprocessing, which has two operations: padding and parsing the message, must be completed before hash computation begins. In addition to ensuring that the padded message is a multiple of 512 or 1024 bits (depending on the algorithm), the padding operation strengthens the security of the hash algorithms [8]. Therefore, it was specified as a required operation for the hash algorithms in FIPS 180-3. However, requiring the preprocessing to be completed before hash computation begins is unnecessary in implementing the hash algorithms. This mandate limited flexibility for implementing hash algorithms in many computer network applications where a message to be hashed could have multiple message blocks, and each message block may be received at different times. With the mandate, the hashing operation can’t start until all of the message blocks are received and the preprocessing is completed, which slows down the application. Limiting algorithm implementation in this manner was unintended.

2. IV Generation Framework for SHA-512-based Hash Algorithms

As discussed above, two new hash algorithms based on SHA-512 are needed as alternatives to SHA-224 and SHA-256. Therefore, SHA-512/224 and SHA-512/256 were specified in FIPS 180-4 [9].

To distinguish SHA-512/224 and SHA-512/256 from SHA-512 and SHA-384 (the previously-approved, truncated SHA-512-based hash function), two new initial values (IVs) are needed. FIPS 180-4 specifies a standard method to determine the IVs for SHA-512/224 and SHA-512/256, as well as any other possible SHA-512-based hash algorithms that may be approved in the future. The method is described below.

Let SHA-512/*t* be the general name for a *t*-bit hash function based on SHA-512 whose output is truncated to *t* bits, where *t* is any positive integer such that $t < 512$, and *t* is not 384 (since SHA-384 is already a SHA-512-based hash algorithm).

Denote $H^{(0)'}$ to be the initial value (IV) of SHA-512 as specified in Section 5.3.5 of FIPS 180-4.

Denote $H^{(0)}$ to be the IV for SHA-512/*t*.

Denote $H^{(0)''}$ to be an IV used during the generation of SHA-512/*t*.

Denote $H_i^{(0)''}$ to be the ith 64-bit word of $H^{(0)''}$, where *i* is defined from 0 to 7, counting from left to right. $H_0^{(0)''}$ is the left-most 64-bit word of $H^{(0)''}$.

$H^{(0)''}$ is computed as follows:

For $i = 0$ to 7
{
$H_i^{(0)''} = H_i^{(0)'} \oplus$ a5a5a5a5a5a5a5a5(in hex).
}

$H^{(0)}$ = SHA-512 ("SHA-512/*t*") using $H^{(0)''}$ as the IV, where *t* is the specific truncation value: an integer without any leading zero. For example, *t* is 192, but not 0192.

As an example, for SHA-512/256, $H^{(0)}$ = SHA-512 ("SHA-512/256"), where "SHA-512/256" is an 11-byte ASCII character string that is equivalent to 53 48 41 2D 35 31 32 2F 32 35 36 (in hexadecimal).

NIST has verified that the method above will generate a different IV for each valid value of *t*. The idea of using the compression function of a family of hash algorithms to generate a unique IV for each specific hash algorithm (with specific hash output length) has been used in the specification of Skein: a final candidate in the SHA-3 hash algorithm design competition [10].

The IV, $H^{(0)''}$, used in the execution of SHA-512 in the above method is different from SHA-512's IV to avoid the property that the IV of a new SHA-512/*t* is an output from another hash algorithm (SHA-512, in this case). This property does not seem to have any

security issue. However, it also does not seem to be a good or desired property for any new hash algorithm. Therefore, $H^{(0)''}$ was constructed to avoid this property.

IVs for SHA-512/224 and SHA-512/256 are defined in FIPS 180-4.

3. Hash Outputs of SHA-512/224 and SHA-512/256

After hashing a whole message, $H^{(0)}$ now becomes $H^{(N)}$ (see FIPS 180-4 for details), which is 512 bits long; the output hash values of SHA-512/224 and SHA-512/256 are the 224 and 256 left most bits of $H^{(N)}$, respectively.

4. Removal of the requirement for a message length encoding operation

As discussed above, the requirement for preprocessing before hash value computation begins creates inefficiency and inflexibility. FIPS 180-4 removes the preprocessing requirement. Without the requirement, the first message block or any other message block except the last block (which can be a partial block) can be hashed (processed) as soon as it is received. The padding operation can be performed when the last (full or partial) block is received, because the length of the message can be determined at this point. After the padding is completed, the final computations needed to generate the hash value can be performed.

5. Conclusion

Specifying SHA-512/224 and SHA-512/256 in FIPS 180-4 provides alternatives to SHA-224 and SHA-256. SHA-512/224 and SHA-512/256 are roughly 30-100% faster than SHA-224 and SHA-256 on computing platforms optimized for 64-bit computing operations, which are becoming major computing platforms for high volume computational devices, such as banking servers. By specifying the method in FIPS 180-4 for determining IV(s), any SHA-512-based hash algorithm that may be approved in the future will be able to be quickly specified. Removing the requirement for the message encoding operation before hash computation begins will allow flexibility in implementing hash algorithms that may speed up application execution.

References

1. Federal Information Processing Standard (FIPS) 186-3, Digital Signature Standard (DSS), National Institute of Standards and Technology, June 2009.

2. Federal Information Processing Standard (FIPS) 198-1, The Keyed-Hash Message Authentication Code (HMAC), National Institute of Standards and Technology, July 2008.

3. NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2007.

4. NIST SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, (Draft) May 2011.

5. http://bench.cr.yp.to/xweb-hash/long-sha256.html

6. Shay Gueron, Simon Johnson and Jesse Walker, *SHA-512/256*, 2011 Eighth International Conference on Information Technology: New Generation.

7. NIST SP 800-107, Recommendation for Applications Using Approved Hash Algorithms, February 2009.

8. I. Damgård, *A Design Principle for Hash Functions*. In Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard, ed, Springer-Verlag, 1989, pp. 416-427.

9. Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard, National Institute of Standards and Technology, Date?.

10. http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html