Public Comments on the Draft Federal Information Processing Standard (FIPS) Draft FIPS 180-2, *Secure Hash Standard (SHS)*

[in response to a notice in the May 30, 2001 Federal Register
(Volume 66, No. 104; p. 29287)]

---

1. Jakob Jonsson *(08/21/2001)*

From: "Jakob Jonsson" <jjonsson@rsasecurity.com>
To: <Proposed180-2@nist.gov>
Subject: Comments regarding the Draft FIPS 180-2
Date: Tue, 21 Aug 2001 18:07:10 +0200
Organization: RSA Laboratories Europe

Dear NIST SHS team,

We congratulate NIST on what appears to be a clear description of the hash algorithms SHA-1, SHA-256, SHA-384, and SHA-512. We have a few comments related to the document:

1. We would like to encourage NIST to publish a security evaluation of SHA-256, SHA-384, and SHA-512 giving rationale for the different design choices. Such an evaluation would increase people's confidence in the algorithms and also facilitate external security analysis. The new algorithms are not straightforward "extrapolations" of SHA-1, so existing analysis of SHA-1 does not necessarily apply to the new algorithms directly. Recall also that SHA-1 is an ostensibly innocent adaptation of the flawed SHA-0; hence seemingly arbitrary design choices might have a serious impact on the security of a hash function.

2. In some constrained environments it might be desirable to have just one hash function implemented; still, the application may require support for different output lengths, e.g. 160 and 256 bits. We suggest including a note in the standard indicating whether SHA-256 truncated to 160 bits (possibly with a different initial hash value) would be an appropriate alternative to SHA-1 in such environments.

3. An attractive feature of SHA-256 and SHA-512 is that they are very similar. The SHA-256 constants are easily extracted from the SHA-512 constants, and the specifications are basically equivalent except for the structure of the sigma functions and a few other very minor differences. This allows for compact code size, which might be worth mentioning.

Best regards,
Jakob Jonsson
RSA Laboratories

2. Carol Widmayer *(08/22/2001)*

From: Carol.Widmayer@do.treas.gov
Date: Wed, 22 Aug 2001 07:18:51 -0400
Subject: Comments on Draft FIPS 180-2, Secure Hash Standard
To: Proposed180-2@nist.gov
Cc: Melanie.Leschnik@do.treas.gov, Michelle.Moldenhauer@do.treas.gov,
Kim.Phalen@do.treas.gov, Tom.Wiesner@do.treas.gov

The Department of the Treasury has no comments on Draft FIPS 180-2, Secure Hash Standard.

Carol Ann Widmayer
Department of the Treasury
CIO
Office of Information Systems Security
(202) 622-1110

3. John Kelsey *(08/28/2001)*

From: "John Kelsey" <jkelsey@certicom.com>
To: Proposed180-2@nist.gov
Date: Tue, 28 Aug 2001 20:21:19 -0700

To: Proposed180-2@nist.gov

A Comment on Draft FIPS 180-2
John Kelsey, August 2001

The FIPS-180-2 document describes one currently fielded hash function, SHA1, and three newly-proposed hash functions, of 256, 384, and 512 bit output size, respectively. For convenience, I will call these hash functions SHA-256, SHA-384, and SHA-512, respectively, and will call the three new hash functions together SHA-(256,384,512).

There is an unintuitive and potentially dangerous property in the new hash functions, which they have inherited from SHA1 (and indirectly, from MD4): Knowledge of hash(X) allows an attacker who doesn't know X to compute hash(X+P+Y), for P determined only by the size of X, and Y arbitrarily chosen by the attacker. I'll call this the length-extension property.

The length-extension property is the reason why we can't use the simple MAC definition MAC_K(X) = hash(K+X). It does not allow an attacker to find collisions on the hash function, and so isn't directly relevant for attacks against digital signature schemes or message/file fingerprinting schemes. However, the way hash functions are commonly used in applications and protocols to bind values together, generate pseudorandom outputs, etc., makes the length-extension attack potentially dangerous. It is also apparently very cheap and easy to fix, in a way that doesn't alter the security of the hash function in other ways.

Niels Ferguson suggested the following simple fix to me, some time ago: Choose some nonzero constant C0, of the same size as the hash function chaining variable. Hash messages normally, until we come to the last block in the padded message. XOR C0 into the chaining variable input into that last compression function computation. The resulting compression function output is used as the hash result. For concreteness, I propose C0 = 0xa5a5...a5, with the 0xa5 repeated until every byte is filled in. This should be interpreted in little-endian bit ordering.

Security Impact

I claim that this change will have no practical security impact on the new hash functions' collision resistance or one-wayness. The specific construction of the new hash functions allows a reduction proof, showing that collisions in the whole hash function are no easier to find than collisions in the compression function. The same proof works for these hash functions with my recommended change--the modified hash functions are no easier to find collisions for than is the underlying compression function. Preimage resistance follows from collision resistance.

The change very cleanly blocks the length-extension property.

Let F(H,M) be the compression function; for SHA-256, H is 256 bits wide and M is 512 bits wide. An attacker who tries to compute hash(X+P+Y) from hash(X) needs to have a way of computing F(C0 xor H,M) from F(H,M) and either M or H. While the ability to do this apparently isn't related to collision-resistance, there appears to be no way to do this for the existing compression functions, and the ability to do so implies a high-probability differential through the compression function. Proving that this isn't possible requires some kind of pseudorandomness assumption about the compression function, using either M or H as the key.

Performance Impact

This proposed change doesn't appear to have any important performance impact. I note that:

a. Implementations of the hash functions must already know when they're dealing with the last block of the message, in order to apply the padding. This means that there is already code or logic to handle this special case of the last block, and this won't have to be added to support this proposed fix.

b. The compression function is unchanged, so there is no need for additional hardware or code to implement a variant compression function for the last message block.

c. The same number of compression function calls are needed for the original and "fixed" hash versions, regardless of the message length or contents.

d. The total work added is a single XOR of a constant value into a 256, 384, or 512 bit value.

Alternatives

If for some reason this proposed fix is unacceptable, there are a number of others which will also eliminate the length-extension property, at relatively low cost and with little or no possible security impact in other areas.

a. The simplest solution is to add one more final compression function computation, in which the output from the original hash function is included as data. This can be done in such a way that it's provably no weaker than the original hash function with respect to collision-finding attacks. However, it adds considerable work when hashing short messages.

b. Another simple solution is to alter the table of constants used in the final compression function. Intuitively, it is hard to see how most such alterations would weaken the hash, but it seems to be almost impossible to prove anything along these lines, and having two different compression functions may seriously impact the gate count of hardware implementations of the hash function.

c. Still another simple solution is to apply a cheap output transformation to the output of the hash function, which is one-way, and requires an unacceptably large guess (half the output) to recover the final compression-function output. This adds work to the hash, and seems to be hard to prove much about, but holds some promise.

Conclusions

The length-extension property in the new hashes defined in FIPS-180-2 is both unintuitive and potentially dangerous. I have proposed a very cheap way to fix it, without altering the fundamental security properties of the hash functions that are fixed, and without imposing any noticeable performance impact. I hope this fix or some similar fix is applied, so that the new hash functions will not have this unintuitive property.

Patent Statement

To my knowledge, nothing in this proposed fix is patented.

--John Kelsey, jkelsey@certicom.com / kelsey.j@ix.netcom.com