X-Sieve: CMU Sieve 2.2
X-F2-Envelope-From: tromer@csail.mit.edu
X-F2-Envelope-To: hash-function@nist.gov
Date: Fri, 27 Apr 2007 17:52:43 -0400
From: Eran Tromer <tromer@csail.mit.edu>
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.10) Gecko/20070302
Fedora/1.5.0.10-1.fc6 pango-text Thunderbird/1.5.0.10 Mnenhy/0.7.5.0
To: hash-function@nist.gov
CC: Ron Rivest <rivest@mit.edu>, Ran Canetti <canetti@csail.mit.edu>
Subject: Hash Algorithm Requirements and Evaluation Criteria
X-Proofpoint-Virus-Version: vendor=fsecure engine=4.65.5502:2.3.11,1.2.37,4.0.164
definitions=2007-04-27_06:2007-04-27,2007-04-27,2007-04-27 signatures=0
X-PP-SpamDetails: rule=spampolicy2_notspam policy=spampolicy2 score=0 spamscore=0
ipscore=0 phishscore=0 adultscore=0 classifier=spam adjust=0 reason=mlx engine=3.1.0-
0703060001 definitions=main-0704270150
X-PP-SpamScore: 0
X-NIST-MailScanner: Found to be clean
X-NIST-MailScanner-From: tromer@csail.mit.edu

We wish to submit the attached comments in response to the NIST notice
on the development of new hash algorithms and the revision of FIPS 180-2
(docket no. 061213336-6336-01).

Please acknowledge receipt of this file.

Sincerely,
Ran Canetti
Ron Rivest
Eran Tromer

Comments follow below.

Comments on NIST Draft Requirements and Criteria for Hash Algorithm

Ran Canetti [*†], Ron Rivest [*], Eran Tromer [*]

April 27, 2007

The National Institute of Standards and Technology has recently published draft minimum acceptability requirements, submission requirements, and evaluation criteria for candidate algorithms to the revised secure hash standard, and solicited comments upon this draft [6]. We strongly concur with the need for a revised standard, and find the proposed competition-based methodology to be sound and effective means for the standardization process to reflect and advance the state of the art. Addressing the details and scope of the draft, we wish to submit the following comments for consideration.

Henceforth, "Requirement" shall refer to an item of the Proposed Draft Submission Requirements, and "Criterion" shall refer to an item of the Proposed Draft Evaluation Criteria of Candidate Algorithms.

1 Keyed modes of operation

The present draft addresses only the basic hash function functionality. However, hash functions are often used to obtain strongly related functionality, most notably:

- Message Authentication Codes (MAC)
- Pseudorandom Functions (RPF)
- Extractors (motivated, e.g., by key derivation [8][4])

Henceforth we shall refer to the above as "keyed modes".[1] The prevalence of such use justifies consideration during the submission and evaluation process. In particular, the choice of hash algorithm should be affected by the performance and plausible security of keyed modes based

[*] {canetti,rivest,tromer}@csail.mit.edu. Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, {G666,G692,G682}, Cambridge, MA 02139, USA

[†] IBM Research, 19 Skyline Dr., Hawthorne, NY 10532 ,USA

[1] Note that in the case of extractors, the key (i.e., seed) is public.

on the hash algorithm. Furthermore, we suggest that the new Secure Hash Standard explicitly specifies algorithms for the above keyed modes, in order to promote interoperability, security and modularity of implementations using these common functionalities.

It does not suffice to rely solely on the existence of generic reductions from keyed modes to basic hash functionality, such as the HMAC construction for MACs. Indeed, some hash functions can directly offer a "native" keyed mode functionality via an internal degree of freedom (e.g., choice of IV), without the overhead of further padding and transformation imposed, e.g., by HMAC. Moreover, the performance of a hash function embedded in, e.g., HMAC, may vary due to initialization and cache effects. Accordingly, we suggest the following:

1. Add Requirements for explicit specification and analysis of algorithms for the above three keyed modes, analogously to the Requirements for a hash algorithm (i.e., including a reference implementation, performance estimates, etc.). These algorithms may be an instance of a generic construction (e.g., HMAC) applied to the proposed hash function, or a specialized algorithm related to the proposed hash function.

2. Add Criteria for consideration of the aforementioned algorithm for keyed modes, analogously to the Criteria for the hash algorithms (i.e., in terms of security, performance, etc.).

3. Add a Criterion: The security relations (and in particular, reductions) between the proposed algorithms for keyed modes and hashing. Well-related algorithms are preferable.

4. Add a Criterion: The cost (e.g., circuit size) of jointly implementing the hash and keyed modes algorithms in one device.

5. Consider including the keyed modes algorithm specification in the new Secure Hash Standard, and as a primary goal of the competition.

We stress that the consideration of the keyed modes algorithms (items 1–4) is well-motivated even if the keyed modes algorithm are not subsequently standardized (item 5).

2 Attack types

The draft Requirements and draft Criteria mention some specific cryptanalytic attacks, but the lists differ. Requirement B.1 mentions "collision-finding" and "second-preimage-finding", whereas Criterion C.1 mentions the "first and second preimage resistance, collision resistance, and resistance to generic attacks (e.g., length extension)". While both lists are defined as non-exclusive, it appears prudent to unify and extend these lists.

Lastly, security risks due to side-channel attacks, as well as presently envisioned technological or algorithmic progress, should be elucidated. Accordingly, we suggest the following:

1. Unify the lists of attacks referenced in the Requirements and Criteria.

2. Add additional security properties to this list:

   (a) Resistance to near-collisions.

   (b) For keyed modes (see Comment 1), fulfilling the properties in the standard definitions of these modes.

   Additional properties are given in Comment 3.

3. Add explicit consideration of specific attack techniques:

   (a) Side-channel attacks, based the specifications from AES process and updated to reflect recent results such as efficient cache-based side-channel attacks.[2] This includes all practical attacks in the contexts of servers, personal computers, embedded systems, smartcards and RFID tags.

   (b) Susceptibility to attack under technological or algorithmic progress that is presently unavailable but plausibly envisioned, such as quantum computers or efficient integer factorization.[3]

3 Indistinguishability from a random oracle

Criterion C.1 includes the following item: "The extent to which the algorithm output is indistinguishable from a random oracle". This appears technically ill-defined, and by a strict interpretation impossible: any fixed algorithms is trivially distinguishable from a random oracle. To the extent possible this requirement should be replaced or augmented by more precise formulations. One notable case is where the data processed by a hash algorithm is secret, e.g., when employed for MAC, key derivation, or pseudorandom generation; pertinent attacks should be explicitly included in the consideration.

We thus suggest adding the following to the list of security properties (see Comment 2):

1. Extent to which, when the hash function is applied to any natural source of sufficient entropy, the digest is indistinguishable from a uniform random distribution. One way to argue for high compliance with this Criterion, for the case where the hash function is sampled from a set of functions (see Comment 9), is by claiming that this sets forms a family of pseudorandom functions (in the sense of [2][5]).

2. Extent to which the digest and input are not correlated by any natural statistical test.

3. Extent to which the digest preserves the computational secrecy of the input, when used with salting or on natural, high-entropy sources.

[2] See [1, 7, 9, 10, 12] and subsequent works.

[3] For example, the VSH hash function [3] is susceptible to both.

Here, the notion of "natural source" is technically informal; it is intended to exclude the inevitable pathological artificial distributions and tests whose definition depend on the choice of algorithm.[4] Stated otherwise, we assume that nature does not depend on FIPS standards.

4 Side-channel protections and their efficiency

Susceptibility to side-channel attacks (see Comment 2) is a property not merely of the algorithm, but primarily of its concrete implementation. Side-channel attacks can often be mitigated by a suitable implementation — but usually, at a significant cost in resources. Reflecting this tradeoff, we suggest:

1. Allow submission of multiple implementations of the proposed algorithms, with differing levels of side-channel protection. This may be restricted, for example, to a "fastest" implementation and a "safest" implementation.

2. Evaluate the security and cost of each such implementation, with preference given to algorithms that are competitive in both the "fastest" and "safest" variants.

5 Efficiency for various input sizes

It is hard to model the performance of an hash algorithm on various input sizes, due to such effects as setup cost and cache effects. For example, short messages often require many more cycles per byte than long messages, and this overhead is not necessarily a constant factor. We thus suggest the following:

1. Extend Requirement B.3 and Criteria C.2 to consider the costs of a variety of representative message sizes (e.g., 2^i bytes for $i = 4, \ldots, 18$).

2. Furthermore, in the above, distinguish between the case where the message size is known in advance to the case where it isn't.

6 Short digests

Some applications rely on very short digests, e.g., for protocols involving manual comparison of strings by humans. Such short digest often suffice, especially when accompanied by proper randomization. Accordingly, we suggest:

1. Add a Requirement: Specify an algorithm for shortening the digest to arbitrary lengths (e.g., by truncation).

[4] For example, consider the uniform distribution of messages whose digest (under the fixed algorithm) ends with ten zeros.

2. Add a Criterion: Whether security of digests shortened to arbitrary lengths is worse than implied by generic attacks.

7 Reduced variants and their conjectured security

Well-designed cryptographic primitives with adequate parameters are often very difficult to analyze and compare, due to the magnitude of the computational problems involved. Examples can be seen in the last stages of the AES process (where all remaining candidates had essentially the same perceived security), and in algebraic attacks (whose applicability to large ciphers has remained controversial and unverifiable for years). We propose the following means to facilitate effective cryptanalytic evaluation, and provide clear indicators of violated assumptions:

1. Add a Requirement: Full specification of a series of hash algorithms that are reduced variants of the full algorithm. The weakest variant in the series shall be easily broken by brute force on a typical desktop computer, while the strongest variant shall be identical to the proposed algorithms with 224-bit digests. Each variant shall be accompanied by a security estimate: a conjectured lower bound on the amount of tangible resources (e.g., CPU operations or equipment cost and time) needed to find a collision[5] in that variant. There shall be at least 5 such variants, and the corresponding security estimates shall be roughly evenly spaced on logarithmic scale.

2. Add a Criterion: Extent to which the conjectured security levels of reduced variants, as submitted, support the conjectured security of the full algorithm. In particular, there should be minimal discontinuity in the sequence of variants and in their conjectured security.

3. Add a Criterion: Extent to which the conjectured security levels of reduced variants have proven consistent with subsequent independent analysis.

While this suggestion places a non-negligible toll on the submitter, it confers several significant benefits:

- It encourages submission of small-yet-indicative reduced variants. These will ease study and enable the cryptographic community to effectively test potential cryptanalytic techniques within the allotted review periods.

- It favors algorithms with a flexible and easily-scalable structure. We consider this advantageous, as historically such algorithms have proven more conductive to systematic analysis.

[5] This may be relaxed to include other non-trivial attacks.

- It provides means for testing the soundness of the assumptions, design principles and analysis underlying the candidate algorithm. Violations of conjectured security levels for reduced variants would constitute a clear indication that some of the applied assumptions, design guidelines or analysis are inadequate, and would thus reflect badly on the full algorithm. If done during the review process, this can affect the choice of algorithm. If done after standardization, it provides significant and clear-cut early warning, possibly many years before the full algorithm is compromised.
- Assuming the submitters of the proposal are initially best-equipped to evaluate its security, it is valuable to learn their honest security estimate at an early stage. The suggested Requirement and Criteria motivate the submitters to honestly portray their conjectured security for the reduced and full variants: overly confident security conjectures for reduced variants stand a high risk of being contradicted by subsequent independent analysis, and given honest portrayal of those reduced variants, overly confident security conjectures for the full algorithm would entail a large, unjustified discontinuity in the sequence of conjectured security levels.

8 Determinism

Many applications of hash functions assume that the hash function is deterministic (i.e., implement a fixed mapping and does not employ randomness), and indeed the present Secure Hash Standard has this property; this should be maintained by the new standard. Accordingly, we suggest:

1. Revise the definition of "hash function" to require determinism.

9 Trapdoors

Most concrete constructions of hash functions implicitly describe a set of functions parametrized by a seed (e.g., an IV), and prescribe a specific member of this set by fixing the seed to some arbitrary value. This raises concerns about the possibility of trapdoors that would offer some advantage to the party who chose the seed. For example, in the VSH algorithm [3] the seed is a large integer and knowledge of its factorization facilitates efficient collision-finding. Accordingly, we suggest:

1. Add a Requirement: A statement on the possibility of trapdoors in the proposed algorithm, and a corresponding Evaluation Criterion. In particular, arbitrary-looking values in the description of the algorithm should be justified.
2. Add a Requirement: A proposed procedure for selecting the parameters in the algorithm in a way that prevents introduction of trapdoors (if relevant). The procedure can consist, for example, of a fully-specified and well-motivated deterministic algorithm that selects the pertinent parameters, or a feasible multiparty computation protocol.

3. Add a Criterion: Extent to which the algorithm appears free of trapdoors, or can be made so via a feasible procedure.

10 Patents and adaptation of components

Patents should not form a barrier to adaptation of components from one candidate to another algorithm, during the submission or revision stages. This is specially important since the design of a hash function typically involves several semi-independent components (a compression function, a mode of operation for hashing and, if our Comment 1 is accepted, a mode of operation for MAC), and it is plausible that different candidates will show strengths in different components. Adaptation of components may be prevented by patent licenses which apply to the algorithm in whole, or that are conditioned upon winning the competition. We thus suggest the following:

1. Amend minimum acceptability requirement A.1 to refer "the algorithm *and any part thereof*", and append "*starting at the time of submission*".

2. Amend Requirement B.5 to refer to "this algorithm *or any part thereof*".

Acknowledgments. We thank Chris Crutchfield and Mayank Varia for their valuable comments.

References

[1] Onur Acıiçmez, Çetin Kaya Koç, Jean-Pierre Seifert, *Predicting secret keys via branch prediction*, proc. RSA Conference Cryptographers Track (CT-RSA) 2007, LNCS 4377, 225–242, Springer, 2007

[2] Shafi Goldwasser, Mihir Bellare, *Lecture Notes on Cryptography*, lecture notes, 2001, `http://www-cse.ucsd.edu/~mihir/papers/gb.html`

[3] Scott Contini, Arjen K. Lenstra, Ron Steinfeld, *VSH, an efficient and provable collision-resistant hash function*, proc. Eurocrypt 2006, LNCS 4004, 165–182, Springer, 2006

[4] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, Tal Rabin, *Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes*, proc. CRYPTO 2004, 494–510, 2004

[5] Oded Goldreich, *Foundations of Cryptography - Volume 1*, Cambridge University Press, 2001

[6] National Institute of Standards and Technology, *Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180–2, Secure Hash Standard*, Federal Register, vol. 72 no. 14, 2861–2863, 2007

[7] Daniel Page, *Theoretical use of cache memory as a cryptanalytic side-channel*, technical report CSTR-02-003, Department of Computer Science, University of Bristol, 2002

[8] Tim Polk,Quynh Dang, *Hash-Based Key Derivation (HKD)*, Internet Draft draft-dang-nistkdf-01.txt, 2006, `draft-dang-nistkdf-01.txt`

[9] Colin Percival, *Cache missing for fun and profit*, BSDCan 2005, Ottawa, 2005

[10] Adi Shamir, Dag Arne Osvik, Eran Tromer, *Cache attacks and countermeasures: the case of AES*, proc. RSA Conference Cryptographers Track (CT-RSA) 2006, LNCS 3860, 1–20, Springer, 2006

[11] Marc Stevens, Arjen K. Lenstra, Benne de Weger, *Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities*, Cryptology ePrint Archive, Report 2006/360; to appear in proc. Eurocrypt'07

[12] Yukiyasu Tsunoo, Etsuko Tsujihara, Kazuhiko Minematsu, Hiroshi Miyauchi, *Cryptanalysis of block ciphers implemented on computers with cache*, proc. International Symposium on Information Theory and its Applications 2002, 803–806, 2002