Journal of the Nigerian Association of Mathematical Physics
***Volume* 13 (November, 2008), 273 - 280**
© J. of NAMP

Design of provably secure cryptographic hash functions

Tola John Odule,
Department of Mathematical Sciences
Olabisi Onabanjo University, Ago-Iwoye, Nigeria

Abstract

***It was shown in this paper that the size w of the internal hash values is a security parameter of its own right, with w ≥ n but otherwise independent from the final hash size n. Given "good" compression functions, this paper shows how to compose "good" hashes. Though the random oracle model is quite useless to define what it means to be a "good" compression function* [6]*, the given lemmas provide some specific requirements for the compression functions*.**

***Keywords*:** *Hash function, Adversarial attack, provable security, Ideal model*

1.0 Introduction

A hash function $H:\{0,1\}^*\rightarrow\{0,1\}^n$ is used to compute an *n*-bit fingerprint from an arbitrarily-sized input. Informally, cryptographers require a good hash function to behave like a *random oracle*. More formal security requirements are, for example, collision resistance and preimage resistance [1]. In practice, cryptographic hash functions for inputs of (almost) arbitrary input sizes are realised by splitting the message into m-bit chunks and iterating a compression function $H:\{0,1\}^n \times \{0,1\}^m\rightarrow\{0,1\}^n$.

Merkle and Damgard [2, 3] showed in their papers that a collision resistant compression function implies a collision resistant iterated hash function. On the other hand, if the adversary is powerful enough to find collisions (this takes time ($\Omega 2^{n/2}$) for a random oracle), many interesting attacks against iterated hash functions become possible, far beyond plain collision-finding.

Using the above-mentioned multi-collision attack as a tool, Joux [4] shows that the (parallel) cascade of several hash functions is not as secure as expected. Similarly, Kelsey [5] describes additional attacks against iterated hash functions. All these attacks are generic. In this paper, we propose and analyse modifications of the Merkle-Damgard design for iterated *n*-bit hash functions. The core idea is to use more than *n*-bit for the internal hash values. We formally prove that these modifications improve security against generic attacks.

1.1 Definitions and abstractions

1.1.1 Iterated hash functions

Cryptographic hash functions take a message $M \in \{0,1\}$ of any length, to compute an *n*-bit output $H(M)$. (In practice, "any length" may be actually be bounded by some huge constant, larger than any message we ever would want to hash.) For an iterated hash, we split the message M into fixed-sized chunks M_1, M_2, …, $M_L \in \{0,1\}^m$, which gives the expanded message $(M_1,\ldots,M_L)$. An iterated hash H iterates an underlying\compression function" C, and the final hash depends on $C(C(\ldots C(C(H_0,M_1),M_2)\ldots),M_L)$, where H_0, is some constant "initial value".

The one or two last chunks of the expanded message are padded, and the last chunk M_L may contain additional information, such as the length $|M|$ of the non-expanded message M. Thus, $L \in \{[|M|/m],[|M|/m] + 1\}$. In any case, the message expansion is deterministic, and if the first m_i bits of two messages M and M' are identical, then $M_1 = M'_1,\ldots,M_i = M_i'$.

1.1.2 Random oracles

A fixed-size random oracle is a function $f:\{0,1\}^a\rightarrow\{0,1\}^b$, chosen uniformly at random from the set

e-mail: tee_johnny@yahoo.com

of all such functions. For interesting sizes a and b, it is infeasible to implement such a function, or to store its truth table. Thus, we assume a public oracle which, given $x \in \{0,1\}^n$, computes $y = f(x) \in \{0,1\}^b$.

A variably-sized random oracle is a random function $g:\{0,1\}^a \rightarrow \{0,1\}^b$, accessible by a public oracle. Equivalently, it can be viewed as an infinite set of fixed-size random oracles, one oracle $g_a:\{0,1\}^a \rightarrow \{0,1\}^b$ for each $a \in N_0$.

We view a fixed-size random oracle as an *ideal compression function*, and a variably-sized random oracle as an *ideal hash function*.

1.1.3 Shannon Cipher (ideal block cipher)

A Shannon cipher is the invertible counterpart of a random oracle. Consider a function $E:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$, such that for each $M \in \{0,1\}^m$, the function $E(.,M) = E_M(.)$ is a permutation, i.e., an inverse function $E^{-1}(.,M)$ exists. A Shannon (block) cipher E is uniformly chosen at random from all such functions. Again, we can't implement a Shannon cipher, but we assume a "Shannon oracle": Given x and M, one can ask the oracle for $y = E(x,M)$, and, given y and M, one can ask the oracle for $x = E^{-1}(y,M)$.

1.1.4 Adversary

As usual in the context of the Shannon and random oracle models, we consider a computationally unbounded adversary with access to some Shannon or random oracle. The adversaries "running time" is determined by her number of oracle queries.

In this article, adversaries are probabilistic algorithms, and we concentrate on the expected running time (i.e., the expected number of oracle queries). We will describe the running time asymptotically, but omit asymptotic notation when possible. In a formal context, though, we are using the symbols O ("big-Oh", for "the expected running time is asymptotically *at most*") and Ω ("big-Omega", "the expected running time is asymptotically *not less than*").

1.2 Typical hash functions attacks

Informally, a real hash function H should behave like an ideal one, a random oracle. This would not be useful for a formal definition, though (see [6]). Instead, one considers somewhat simpler security goals. Let a hash function $H:\{0,1\}^* \rightarrow \{0,1\}^n$ be given. Some "classical" types of attack are:

1.2.1 Collision attack

Find two messages $M \neq M'$ with $H(\mathrm{M}) \neq H(M')$.

1.2.2 Preimage attack

Given a random value $\mathrm{Y} \in \{0,1\}^n$, find a message M with $H(M) = Y$.

1.2.3 2nd preimage attack

Given a message M, find a message $M \neq M'$ with $H(\mathrm{M}) \neq H(M')$.

The following natural extensions have also been studied and added:

K-collision attack: for $K \geq 2$. Find K different messages M^i, with $H(\mathrm{M}^1) = \ldots = H(M^K)$.

K-way (2nd) preimage attack, for $\mathrm{K} \geq 1$:: Given Y (or M with $H(M) = Y$.), find K different messages M^i, with $H(M^i) = Y$ (and $M^i \neq M$)

The aforementioned attacks are obviously possible with a sufficiently skilled adversary. To measure the security of a hash function H, one compares the resistance of H against these attacks, with the amount of resistance a random oracle would provide:

***Fact* 1**

Model $H:\{0,1\}^* \rightarrow \{0,1\}^n$ as a random oracle. Finding a K-collision for H takes time $\Omega(2^{(K-1)n/K})$, and finding a K-way preimage or a K-way 2nd preimage for H takes time $\Omega(K2^n)$.

A part of our security analysis depends on idealised building blocks for iterated hash functions. The above attacks against hash functions--variably-sized random oracles--generalise for compression functions--fixed-size random oracles. The following two facts describe the basic security properties of fixed-size random oracles against multiple collision and (2nd) preimage attacks, and the security of an idealised block cipher, with fixed plaintexts.

***Fact* 2**

Model $C:\{0,1\}^{n+m} \rightarrow \{0,1\}^n$ as a random oracle. Finding a K-collision for C takes time $\Omega(2^{(K-1)n/K})$, and finding a K-way preimage or a K-way 2nd preimage for C takes time $\Omega(K2^n)$..

***Fact* 3**

Model $E:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ as a Shannon oracle. Consider a fixed random value $S \in \{0,1\}^n$. Regarding collision and (2nd) preimage attacks, the function $f:\{0,1\}^m \rightarrow \{0,1\}^n$, $f(M) = E_M(S)$ behaves like a random oracle with m input and n output bits.

2.0 Review of current ierated hashes

2.1.1 Merkle-Damgard hash

Recall that we have a fixed-size compression function $C:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$, and our goal is to

implement a hash function $H:\{0,1\}^* \rightarrow \{0,1\}^n$. Given a (randomly chosen) fixed initial value H_0 and a message $M \in \{0,1\}^*$ the Merkle-Damgard (MD) hash $H(M)$ is computed as follows:

- Expand M to $(M_1, \ldots, M_L) \in \{0,1\}^{mL}$.
  MD strengthening
  The last block M_L takes the length $|M|$ in bits
- For i in 1,…,L: compute $H_i = C(H_{i-1}, M_i)$.

Finally: set $H(M) = H_L$.

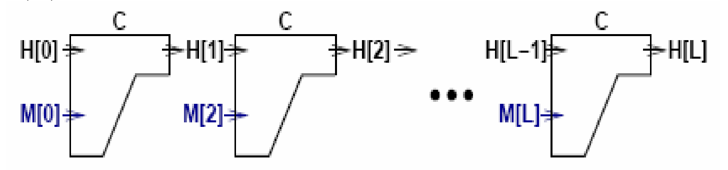

Figure 2.1: The Merkle-Damgard hash

2.1.2 Length extension

This is a well-known weakness of the MD hash (see e.g. [7, Section 6.3.1]):
given $H = H(M)$, it is straightforward to compute M' and H', such that $H' = H(M||M')$ - even for unknown M (but for known length $|M|$). The attack is based on using $H(M)$ as an internal hash for computing $H(M||M')$.

2.1.3 Joux' Attacks

Antoine Joux described an attack to find 2^k-Collisions for MD hash H in time $O(k^{2n/2})$, instead of $\Omega(2^{n(k-1)n/2^k})$.

- For i in 1,…,k find a local collision $M^0_i \neq M^1_i$ with $H_i = C(H_{i-1,} M_i^0) = C(H_{i-1,} M_i^1)$.. All the 2^k messages $(M_1^0, K, M_k^0), (M_1^0, K, M_{k-1}^0, M_k^1), K, (M_1^1, K, M_k^1),$ hash to the same value H_k.

We hereby note that all messages are of the same, not too large, size of k blocks. Joux pointed out that this technique can be used to attack cascaded hash functions: Let a hash $H:\{0,1\}^* \rightarrow \{0,1\}^n$ be defined as $H(M) = H(H^1(M)||H^2 M)$ with two independent n-bit hashes H^1 and H^2. If both H^1 and H^2 are independently defined as random oracles, then finding collisions for H takes time 2^n. If, however, either is constructed as a MD hash, finding a collision for H only takes time $O(n/2 * 2^{n/2})$. W.l.o.g., let H^1 be the MD hash:

- Find $2^{n/2}$-collisions for H^1 (in $n/2*2^{n/2}$ units of time).

Statistically, one such collision also collides for H^2 (and thus H).
He also demonstrated the applicability of the multi-collision attack as a tool to find multiple (2nd) preimages very efficiently. Given a target $Y \in \{0,1\}^n$, the attack proceeds as follows:

- Generate 2^k colliding k-block messages $(M^1, \ldots, M^{2k})$ with $H_k = H(M^1) = \ldots = H(M^{2k})$.
- Find a message chunk M_{k+1}, such that $C(H_k, M_{k+1}) = Y$.

This provides a 2^k -way preimage. The first step takes time $k*2^{(n/2)}$, which is marginal, compared to the second step. This takes about the time for a single preimage attack, i.e., $O(2^n)$. For a 2nd preimage message attack with the target message M, just set $Y = H(M)$.

2.1.4 The Davies-Meyer hash and Kelsey's Attack

The attack described above is applicable for any compression function C. Often, compression functions are designed according to the "Davies-Meyer" principle: given a block cipher like E, the function C is defined by

$$C(H_{i-1,} M_i) = E_M(H_{i-1}) + H_{i-1,} \cdot$$

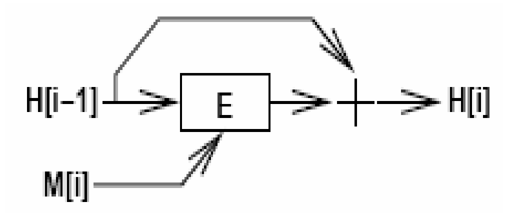

Figure 2.2: The Davies-Meyer construction

Here "+" is any group operation over $\{0,1\}^n$, and we write 0^n for the neutral element. E_{M_i} is invertible for all M_i (like any n-bit block cipher). This allows the adversary to compute (random) fixed points for C:

- Select a message M_i and compute $H_{i-1} = E^{-1}_{M_i}(0^n)$.

This is a fixed point, since $H_i = C(H_{i-1,} M_i) = E_{M_i}(H_{i-1}) + H_{i-1} = 0^n + H_{i-1}$. Finding such a fixed point takes one "decryption" E^{-1}. Note that the fixed point $H_{i-1} = H_i$ depends on the choice of M_i, but for any M_i such a fixed point exists.

Let a message M be given, and let the expansion $(M_1,\ldots,M_L)$ of M be L chunks long. Using the fixed point finder as a tool, Kelsey [5] describes an algorithm to compute a 2nd preimage for M in time $O(\max\{2^{n/2}, 2^n/L\})$. In an extreme case, i.e., for $T \approx 2^{n/2}$ the entire attack asymptotically takes time $2^{n/2}$ to compute a 2nd preimage--instead of time 2^n, as would be expected for a random oracle.

2.1 Security against Generic Attacks

The above attacks are generically applicable against a wide class of hash functions. Joux' attack is applicable against all MD hashes, and the compression function C can be realised by a random oracle. Further, the attack can be made to work even if the adversary only has oracle access to the hash function H, but not to the compression function C. So Joux' attack is generic in a very strong sense.

Kelsey's attack requires the compression function $C(H,M) + EM(H) + H$ to be a Davies-Meyer compression function. In contrast to Joux' attack, Kelsey's would not work with oracle access to H only--the adversary needs oracle access to E^{-1}. But Kelsey's attack is still generic, since it does not assume any specific weakness for E - E can be as strong as a Shannon cipher.

In consequence of the weaknesses of iterated hash functions noted above, can we design iterated hashes and prove their security without making the assumption that some internal building block is much stronger than the hash function itself?

The focus of this article is to propose a modified MD design for hash functions, provably secure against all generic attacks, including, but not limited to Joux' and Kelsey's.

3.0 Two twined-pipes hash with Davies-Meyer (DM)

In technical discussions, the compression function is treated like a random oracle with fixed input size. However, for most practical hash functions, the compression function is, by itself, based on some block cipher-like building block, often according to the DM construction. This provides the adversary with some additional handles. If we use such a compression function for the Two Twined-Pipe Hash, we must re-examine the security of the double-pipe hash.

In this section, we consider the double-pipe hash H, using a DM-based compression function $C:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$, $C(H_{i-1,} M_i) = E_M(H_{i-1}) + H_{i-1,}$

For each M Є $\{0,1\}^m$, the function E_M is a permutation over $\{0,1\}^n$.

3.1 Notations

For our formal treatment, we consider an adversary A with access to a Shannon oracle for E and E^{-1}. Similarly to [1], we assume:

- The adversary A never asks a query in which the response is already known. Namely, if A asks for $E_k(x)$ and receives y, she neither asks for $E^{-1}{}_k(y)$, nor for $E_k(x)$ again. Similarly, if she has asked for $E^{-1}{}_k(y)$ and received x.
- Recall that for the type of attacks we consider, a successful adversary always outputs one or more messages M^i, which either collide or constitute some (2nd) preimages. Before finishing, the adversary makes all the oracle calls to compute all hash values $H(M^i)$.
- We define a *simulator*, to respond to A's oracle queries:
    - Initially:

```
        * set $i = 0$; clear the logbook;
        * for all $(k,x)$: mark $E_k(x)$ as undefined;
    ▪ Responding to an oracle query $E_k(x)$:
        * set $i + i + 1$
        * randomly choose $y$ from $\overline{RANGE}$ $(E_K)$
        * append $(x_i, k_i, y_i) = (x, k, y)$ to the logbook;
        * respond $y$;
    ▪ Responding to an oracle query $E^{-1}{}_k(y)$:
        * set $i + i + 1$
        * randomly choose $x$ from $\overline{DOMAIN}$ $(E_K)$
        * append $(x_i, k_i, y_i) = (x, k, y$ to the logbook;
        * respond $x$;
```

Here, $\overline{DOMAIN}$ (E_K) is the set of points x where $E_k(x)$ is still undefined. Similarly, $\overline{RANGE}$ (E_K) is the set of points y where $E^{-1}{}_k(y)$ is still undefined. For our proofs, we will discuss the logbook entries (x_i, k_i, y_i). This is without loss of generality: (w.l.o.g) any adversary not following the first two conventions can easily be transformed into an equivalent one following them. And an adversary following the first two conventions can't distinguish the simulator from a "true" random oracle.

3.2 Internal collisions

Lemma 3.2.1

Consider the double-pipe iterated hash H:

(i). Any internal collision either reduces to a strict or to a cross collision.

(ii). Finding a K-collision requires time $\Omega(min(T_S, T_X, T(K))$.

Proof

For the first claim, observe that the initial values H'_0 and H''_0 are different. Any non-strict internal collision implies a triple $(H'_{i-1}, H''_{i-1}, M_i)$ with $H'_{i-1} = H''_{i-1}$. This implies the existence of a cross-colliding triple (H'_j, H''_j, M_{j+1}), with $j \le i-2$, $H'_j \neq H''_j$ and $H'_{j+1} = C(H'_j, H''_j \| M_{j+1}) = C(H''_j, H'_j \| M_{j+1}) = H''_{j+1}$.

For the second claim, we observe that a K-collision for H reduces to either a final K-collision (which takes time $T(K)$, or to an internal collision. Due to the first claim, an internal collision is either strict (and needs time T_S), or is a cross collision (time T_X).

Here we write T_S for the time to find a strict internal collision, T_X for an internal cross collision, and $T(K)$ for the time to find a final K-collision.

***Theorem* 3.2.2**

Consider the DM-based double-pipe hash H. If we model E by a Shannon oracle, then $T_X = \Omega(2^n)$ *and* $T_S .= \Omega(2^n)$

Proof

For the proof, we assume that the adversary does not make more than $q \le 2^{n-1}$ queries. This is technically correct, since $2^{n-1} .= \Omega(2^n)$.

3.2.1 Time T_X to find internal cross collisions

A cross collision is described by $H'_{i-1} \neq H''_{i-1}, M_i$ with

$$C(H'_{i-1}, H''_{i-1} \| M_i) = H'_i = H''_i = (H''_{i-1}, H'_{i-1} \| M_i) \qquad (3.1)$$

In time q, we can check at most $q/2$ such triples $(H'_{i-1}, H''_{i-1}, M_i)$ for cross collisions. Now we argue that for $q \le 2^{n-1}$, for each such triple the probability p_x to satisfy Equation (3.1) is at most $1/2^{n-1}$. This implies that the expected number of oracle queries we need to make before we get the first cross collision is $T_X = \Omega(2^n)$, as claimed.

We still have to show $p_x \le 2^{n-1}$. Observe that if the adversary's answer involves a cross collision, then, by the above conventions, the simulator's logbook contains two triples (x_a, k_a, y_a) and (x_b, k_b, y_b) with $a \neq b$,

$x_a = H'_{i-1},\ k_a = (H''_{i-1} \| M_i), y_a = E_{k_a}(x_a)$,

$x_b = H''_{i-1},\ k_b = (H'_{i-1} \| M_i)$ and $y_b = E_{kb}(x_b)$.

Thus, we can rewrite Equation 1 by

$\overbrace{E_{k_a}(x_a)}^{y_a} + x_a = \overbrace{E_{k_b}(x_b)}^{y_b} + x_b$ which corresponds to

$$y_a + x_a = y_b + x_b \tag{3.2}$$

If (w.l.o.g.) $a < b$, then either y_b or x_b is a uniformly distributed random value from a huge subset of $\{0, 1\}^n$:

- If the b^{th} oracle query has been $E_{k_b}(x_b)$, then y_b is a random value from $\overline{RANGE}\ (E_{k_b})$.
- Else x_b is a random value from $\overline{DOMAIN}\ (E_{k_b})$.

Since $\left|\overline{RANGE}\ (E_{k_b})\right| = \left|\overline{DOMAIN}\ (E_{k_b})\right| = 2^n - b + 1 \geq 2^n - q$, and due to $q \leq 2^{n-1}$, we get $p_x \leq 2^{n-1}$, as claimed.

3.2.2 Time T_S to find strict internal collisions

(G', G'', M) with $H' \neq H''$, we consider pairs (H'. H'') Є $\{0,1\}^{2n}$, where

$$H' = C(G', G''||M) \text{ and } H'' = C(, G'', G'||M) \tag{3.3}$$

A strict internal collision are two different triples, where the corresponding H' and H'' values both collide. When making q oracle queries, there are $\Omega(q^2)$ such pairs. We claim that for $q \leq 2^{n-1}$, the probability p_s to satisfy equation (3.3) is $p_x \leq 1/2^{n-1}$. Hence, the expected number of oracle queries to get a strict collision is $T_S .= \Omega(2^n)$.

It remains to prove $p_x \leq 1/2^{n-1}$. Consider a triple (x_a, k_a, y_a) with $x_a = G',\ k_a = (G'' \| M)$ and $y_a = E_{k_a}(x_a)$ from the simulator's logfile. We only have a chance for a strict collision, if the logfile contains another triple (x_b, k_b, y_b) with $x_b = G'',\ k_b = (G' \| M)$, and $y_b = E_{k_b}(x_b)$. Note that x_b and k_b are uniquely determined by x_a and k_a, and vice versa. Equation 3 can then be rewritten as

$$H' = E_{k_a}(x_a) + x_a = y_a + x_a \text{ and } H'' = E_{k_b}(x_b) + x_b = y_b + x_b.$$

A strict collision implies the adversary to handle a colliding triple (F', F'', N), that is, $H' = C(F', F'' \| N)$ and $H'' = C(F'', F' \| N)$. This information corresponds to two more triples (x_c, k_c, y_c) and (x_d, k_d, y_d) on the server's logfile with

$$H' = y_a + x_a = y_c + x_c \tag{3.4}$$

$$H'' = y_b + x_b = y_d + x_d \tag{3.5}$$

Each of these two equations is of the same type as Equation 3.2. As in that context, we argue that due to $q \leq 2^{n-1}$ the probability for (3.4) to hold is no more than $1/2^{n-1}$; similarly for (3.5). More importantly, the conditional probability to satisfy (3.5), assuming (3.4) is at most $1/2^{n-1}$. Thus, the joint probability p_s for both (3.4) and (3.5) is $P_s \leq 1/2^{2(n-1)}$

3.3 *K*-Collisions

***Theorem* 3.3.1**

Consider the DM-based double-pipe hash H. If we model E by a Shannon oracle, then finding K-collisions for H takes time $\Omega(2^{n(K-1)/K})$.

Proof

Due to the first claim of *Lemma* 3.2.1 and *Theorem* 3.2.2, we know that an internal collision would take time $\Omega(2^n)$. Thus, in time $\Omega(2^{n(K-1)/K})$. we don't find any such collision. In order to find a *K*-collision faster than in time $\Omega(2^n)$, we must find a final K-collision. In the remainder of this proof, we will show that finding a final K-collision takes time $\Omega(2^{n(K-1)/K})$.

A final K-collision consists of K different pairs $(G_i,H_i) \in (\{0,1\}^n)^2$ with $C\left(H^*, G^1 \| H^1 \| 0^{m-n}\right)$ $= \Lambda = C\left(H^*, G^K \| H^K \| 0^{m-n}\right)$. Hence, after a possible permutation of triples, we have to find K triples $(H^*, k_1, y_1), \ldots, (H^*, k_k, y_k)$ in the simulator's logbook with different k_i but

$$\overbrace{E_{k_1}(H^*) + H^*}^{y_1} = \Lambda = \overbrace{E_{k_K}(H^*) + H^*}^{y_K},$$

or equivalently
$$\overbrace{E_{k_1}(H^*)}^{\tilde{y}_1} = \Lambda = \overbrace{E_{k_K}(H^*)}^{\tilde{y}_K}$$

By fixing the input H^* for E, we turn the Shannon-oracle into an ordinary random oracle, see Fact 2. According to Fact 2, finding a K-collision takes time $\Omega(2^{n(K-1)/K})$.

3.4 K-Way (2nd) preimages

***Theorem* 3.4.1**

Consider the DM-based double-pipe hash H. If we model E by a Shannon oracle, then finding a single or K-way preimage or a single or K-way 2nd preimage takes time $\Omega(2^n)$.

Proof

Finding K-way (2nd) preimages isn't faster than finding single (2nd) preimages. Thus, we concentrate on single ones. Due to *Lemma* 3.2.2, (see appendix) finding a single preimage for K takes time $\Omega(P(1))$.$P(1) = \Omega(2^n)$ follows from Facts 3 and 2.

Now assume an algorithm exists to find 2nd preimages for H. Consider we are given $X \in \{0,1\}^{n+m}$, and searching for some 2nd preimage key $Y \neq X$ with $E_Y(H^*) = E_X(H^*)$ for E. The proof is as follows:

We choose some message M and compute the internal hashes $H'_1, H''_1, \ldots, H'_L, \ldots, H''_L$. Assume

$$X \notin \{(H'_i \| H''_i \| M_i), (H''_i \| H'_i \| M_i) \mid 1 \le i \le L\}$$

(this holds with overwhelming probability). Set $H_L = \left(H'_L \| H''_L \| 0^{n-m}\right)$. We define the function

$$E' : \{0,1\}^n \to \{0,1\}^{n+m} \to \{0,1\}^n : \begin{cases} E'_X(\cdot) = E_{HL}(\cdot) \\ E'_{HL}(\cdot) = (\cdot) \\ E'_Z = E_Z(\cdot) \text{ for } Z \notin \{X, H_L\} \end{cases}$$

Now we run the adversary, replacing the (Shannon-) oracle for E and E^{-1} by an oracle for E' and its inverse. Both E and E^{-1} are random permutations over $\{0, 1\}^n$. If the adversary succeeds in finding a 2nd preimage for M, she either has found an internal collision (which would take time $\Omega(2^n)$, or $Y := H_L \neq X$ is a solution to the 2nd preimage problem for E. By Facts 2 and 3, this would take time $\Omega(2^n)$,. In any case, finding a 2nd preimage for M reduces to solving a problem we know to take time $\Omega(2^n)$,.

4.0 Discussion

4.1 Main contributions

The major inference from [4, 5, 9] and in this treatise is that the size w of the internal hash values is a security parameter of its own right, with $w \geq n$, but otherwise independent from the final hash size n.

Any cryptographer, choosing a cryptographic hash, should choose both w and n according to her specific security requirements and also considering, of course, efficiency concerns, compatibility issues, etc.). For some applications, the Merkle-Damgard setting with $w > n$. may be appropriate, while others may require $w > n$.

The design of hash functions is not only about appropriate choices of the security parameters w and n, though. If n is sufficiently large to prohibit all attacks with $2^{n/2}$ running time, then $w = n$, as in the plain *MD* design, appears to be fine. But assume a feasible collision attack. This implies a cryptanalytic weakness in the compression function, namely a feasible attack $\mathcal{A}$ against the underlying compression function. Assume there is no variant of $\mathcal{A}$ to feasibly find multi-collisions. Nevertheless, Joux' attack allows one to feasibly find large multi-collisions for the plain *MD* hash. Thus, finding 2^k-collisions takes time k*time ($\mathcal{A}$). Observe that the speed-up over attacking an ideal hash quickly grows with *k*. If we use the same compression for a double-pipe hash, the failure of the compression function would be less catastrophic. The speed-up for finding K-collisions for the two twine-pipe hash, compared to an ideal hash, would be $2^{n/2}$/time ($\mathcal{A}$). This does not depend on *K* at all.

Note that the hash functions proposed here do not suffer from the straightforward length extension attack, in contrast to the plain MD hash.

5.0 Conclusion

As suggested in [10], it may be time for the cryptographic community to design new and more secure hash algorithms. In this treatise, we took a rather abstract and proof-centric look at the design of hash functions. Similarly to others, we consider this style a "feasible and useful step for understanding the security" [11] of iterated hash functions, thereby complementing the attack-centric approach [4, 5], though not replacing it.

Given "good" compression functions, this paper shows how to compose "good" hashes. Though the random oracle model is quite useless to define what it means to be a "good" compression function [6], the given lemmas provide some specific requirements for the compression functions.

Appendix

***Lemma* 3.2.2** Consider the double-pipe hash *H*:

1. Finding a single preimage for *H* takes time $\Omega(P(1))$
2. Finding *K*-way preimages for *H* takes time $\Omega(\min\{T_S, T_X, P(K)\})$.

Proof

First claim: Consider the wide-pipe hash *H:*

First bound: observe that finding a preimage for *H* (some *M* with *H*(*M*) = *Y*) implies finding a preimage H_L for C", since C"(H_L) = Y.

Second bound: finding *K* different preimages $M^1, ..., M^K$ for *H* either implies finding at least one collision for ', or implies finding *K* different inputs $H^1_{L^1}, \Lambda, H^K_{L^K}$ with $C''(H^1_{L^1}) = \Lambda = C''(H^K_{L^K}) = Y$, i.e., a *K*-way preimage for C″. Second claim: Follows from *claim 1* of *Lemma* 3.2.1.

References

[1] Odule, T.J. "Incremental Cryptography and Security of Public Hash Functions." Journal of Nigerian Association of Mathematical Physics, vol. 11 pp.467-474; 2007.

[2] R. Merkle. One-way hash functions and DES. Crypto 89, LNCS 435, pp. 428{446.

[3] I. Damgard. A design principle for hash functions. Crypto 89, LNCS 435, pp. 416-427.

[4] A. Joux. Multicollisions in iterated hash functions, application to cascaded constructions. Crypto 04, LNCS 3152, pp. 306{316.

[5] J. Kelsey. A long-message attack on SHAx, MDx, Tiger, N-Hash, Whirlpool, and Snefru. Draft. Unpublished Manuscript.

[6] R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. 30th STOC 1998, pp. 209{218.

[7] N. Ferguson, B. Schneier. Practical Cryptography. Wiley Publishing, 2003.

[8] National Institute of Standards and Technology (NIST). Secure hash standard. FIPS 180-2. August 2002.

[9] B. Preneel. Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit Leuven, 1993.

[10] B. Schneier. Cryptanalysis of MD5 and SHA. Crypto-Gram Newsletter, September 2004. http://www.schneier.com/crypto-gram-0409.html#3

[11] Black, Rogaway, Shrimpton. Black-box analysis of the block-cipher based hash function construction from PGV. Crypto 02.