Design Principles for Iterated Hash Functions

e-print (September 29, 2004)

Stefan Lucks

University of Mannheim, Germany
http://th.informatik.uni-mannheim.de/people/lucks/

Abstract. This paper deals with the security of iterated hash functions against generic attacks, such as, e.g., Joux' multicollision attacks from Crypto 04 [6]. The core idea is to increase the size of the internal state of an n-bit hash function to $w > n$ bit. Variations of this core idea allow the use of a compression function with n output bits, even if the compression function itself is based on a block cipher.
In a formal model, it is shown that these modifications quantifiably improve the security of iterated hash functions against generic attacks.

Keywords: hash function, Joux attack, provable security, black-box model

1 Introduction

Recently, Joux [6] surprised the cryptographic community with a generic *multi-collision attack* against iterated hash functions, able to find K-Collisions in time $O(\log(K) * 2^{(n/2)})$, instead of time $\Omega(2^{(K-1)n/K})$, as we would expect from an ideal hash function. This and other recent results constitute a great deal of progress in hash function cryptanalysis. As suggested in [11], it may be time for the cryptographic community to design new and more secure hash algorithms. The current paper studies improved hash function design principles.

A hash function $H : \{0,1\}^* \to \{0,1\}^n$ 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, e.g., collision resistance and preimage resistance. 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 $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$.

In their landmark papers, Merkle and Damgård [8, 3] showed 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 abovely mentioned multi-collision attack as a tool, Joux [6] shows that the (parallel) cascade of several hash functions is not as secure as expected. In a similar spirit, Kelsey [7] describes additional attacks against iterated hash functions. All these attacks are generic, i.e., are applicable if we replace the compression function by some abstract oracle.

In the current paper, we propose and analyse modifications of the Merkle-Damgård 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 Notions and Abstractions

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, $L \in \{\lceil |M|/m \rceil, \lceil |M|/m \rceil + 1\}$. In any case, the message expansion is deterministic, and if the first mi bits of two messages M and M' are identical, then $M_1 = M'_1$, ..., $M_i = M'_i$.

Random Oracles. A *fixed-size random oracle* is a function $f : \{0,1\}^a \to \{0,1\}^b$, chosen uniformly at random from the set 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 $x \in \{0,1\}^a$, computes $y = f(x) \in \{0,1\}^b$.

A *variably-sized random oracle* is a random function $g : \{0,1\}^* \to \{0,1\}^b$, accessible by a public oracle. Equivalently, it can be viewed as an infinite set of fixed-size random oracles, one oracle $g_a : \{0,1\}^a \to \{0,1\}^b$ for each $a \in \mathbb{N}_0$.

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

Shannon Cipher (ideal block cipher). A Shannon cipher is the invertible counterpart of a random oracle. Consider a function $E : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, such that for each $M \in \{0,1\}^m$, the function $E(\cdot, M) = E_M(\cdot)$ is a permutation, i.e., an inverse function $E^{-1}(\cdot, M)$ 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 $x = E^{-1}(y, M)$.

Adversary. As usual in the context of the Shannon and random oracle models, we consider a 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 the current paper, 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]

1.2 Types of Attacks for Hash Functions

Informally, a real hash function H should behave like an ideal one (i.e., like a random oracle). This would not be useful for a formal definition, though (see [2]). Instead, one considers somewhat simpler security goals.

Let a hash function $H : \{0,1\}^* \to \{0,1\}^n$ be given. Some "classical" types of attack are

[1] Recall $f = O(g)$, if a constant c exists, such that for all large enough n $f(n) \leq cg(n)$ holds. Similarly, $f = \Omega(g)$, if a constant c exists such that for all large enough n $f(n) \geq cg(n)$.

Collision attack: Find two messages $M \neq M'$ with $H(M) = H(M')$.
Preimage attack: Given a random value $Y \in \{0,1\}^n$, find a message M with $H(M) = Y$.
2nd preimage attack: Given a message M, find a message $M' \neq M$ with $H(M) = H(M')$.

Additionally, the following natural extensions have been studied:

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

If the adversary is powerful enough, then the attacks are obviously possible. 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 $H : \{0,1\}^* \to \{0,1\}^n$ 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 (i.e., 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 $C : \{0,1\}^{n+m} \to \{0,1\}^n$ 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 $E : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ as a Shannon oracle. Consider a fixed random value $S \in \{0,1\}^n$. Regarding collision and (2nd) preimage attacks, the function $f : \{0,1\}^m \to \{0,1\}^n, f(M) = E_M(S)$ behaves like a random oracle with m input and n output bits.*

2 Weaknesses of Current Iterated Hashes

2.1 Iterated Hashing: the Merkle-Damgård Hash

Recall that we have a fixed-size compression function $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, and our goal is to implement a hash function $H : \{0,1\}^* \to \{0,1\}^n$. Given a (randomly chosen) fixed *initial value* H_0 and a message $M \in \{0,1\}^*$, the Merkle-Damgård (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.[2]
- For i in 1, ..., L: compute $H_i := C(H_{i-1}, M_i)$.
- Finally: set $H(M) = H_L$.

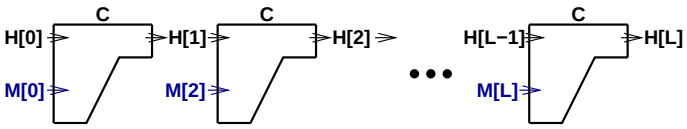

Fig. 1. The Merkle-Damgård Hash

2.2 Length Extension

This is a well-known weakness of the MD hash (see e.g. [4, 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 $H(M||M')$.

2.3 Joux' Attacks

At Crypto 04, Antoine Joux described an attack to find 2^k-Collisions for a MD hash H in time $O(k2^{n/2})$, instead of $\Omega(2^{n(2^k-1)/2^k})$:

[2] Thus, if $|M| \neq |M'|$, then $M_L \neq M'_{L'}$.

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

Note that all messages are of the same (not too large) size of k blocks.

As Joux pointed out, 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).

Joux 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 $M^1, \ldots, M^{2^k}$ with $H_k = H(M^1) = \cdots = H(M^{2^k})$.
– 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 $Y := H(M)$.

2.4 The Davies-Meyer Hash and Kelsey's Attack

Joux' attack 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

$$C(H_{i-1}, M_i) = E_{M_i}(H_{i-1}) + H_{i-1}.$$

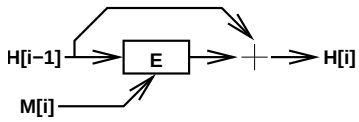

Fig. 2. The Davies-Meyer Construction

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 $H_{i-1} := E_{M_i}^{-1}(0^n)$.

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 [7] 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.

2.5 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 $C(H, M) = E_M(H)+H$ to be a Davies-Meyer compression function. [3] 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

[3] In [7], Kelsey generalises this to some other constructions.

does not assume any specific weakness for E – E can be as strong as a Shannon cipher.

The target of the current paper is a modified MD design for hash functions, *provably secure against all generic attacks*, including, but not limited to Joux' and Kelsey's.

3 The Wide-Pipe Hash: A Modified MD Hash

Since both Joux' and Kelsey's attacks are based on finding internal collisions, it appears to be an obvious idea to "widen" the internal pipe from n bit to $w > n$ bit to improve protection against finding internal collisions.[4] Let $H_0 \in \{0,1\}^w$ be a (randomly chosen) *initial value*. Using *two* compression functions

- $C' : \{0,1\}^w \times \{0,1\}^m \to \{0,1\}^w$ and
- $C'' : \{0,1\}^w \to \{0,1\}^n$,

we compute the wide-pipe iterated hash H:

- For i in $1, \ldots, L$: compute $H_i := C'(H_{i-1}, M_i)$.
- Finally: set $H(M) = C''(H_L)$.

We call H_L the "intermediate hash".

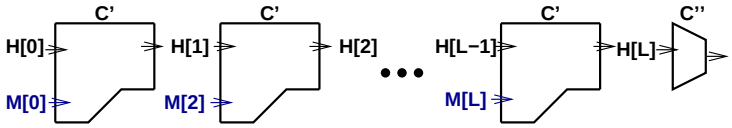

Fig. 3. The Wide-Pipe Hash

3.1 *K*-Collision Attacks

As an upper bound on the security of H, observe that Joux' attack finds 2^k-collisions in time $\min\{k * 2^w, 2^{n(2^k-1)/2^k}\}$. As it turns out, this bound

[4] This has independently been proposed by Finney in a mailing list [5].

is tight, up to the (logarithmic) factor k. If we write T' for the time to find an internal collision, i.e., a collision for C', and $T''(K)$ for the time to find a K-collision for C'', we get the following lower bound on the security of H:

Lemma 4 *Finding a K-collision for the wide-pipe iterated hash H requires at least time $\Omega(\min\{T', T''(K)\})$.*

Proof. Consider a collision $M \neq N$ with $H(M) = H(N)$. M and N are expanded to sequences $(M_1, \ldots, M_L) \neq (N_1, \ldots, N_{L'})$. Denote H_i^M and H_j^N for the internal hash values when computing $H(M)$ and $H(N)$. We distinguish three different types of collisions:

Final collision: $H_L^M \neq H_{L'}^N$, and $C''(H_L^M) = C''(H_{L'}^N)$.
Different length: $L \neq L'$ implies $M_L \neq N_{L'}$ (cf. Footnote 2). Thus, either $H_L^M = H_{L'}^N$ implies an internal collision (see below), or $H_L^M \neq H_{L'}^N$, implies a final collision (see above).
Internal collision: $(H_L^M, M_L) = (H_{L'}^N, N_{L'})$, and thus $L = L'$. Since $(M_1, \ldots, M_L) \neq (N_1, \ldots, N_{L'})$, there exists a collision for C', i.e., values $(H_i^M, M_i) \neq (H_i^N, N_i)$ with $C'(H_i^M, M_i) = C'(H_i^N, N_i)$.

A K-collision for H reduces to either a K-collision for the final compression function C'', or to at least one ("internal") collision for C'. □

As an immediate consequence, we get the following theorem.

Theorem 5. *If we model the compression functions C' and C'' as independent random oracles, finding K-collisions for the wide-pipe iterated hash H takes time $\Omega(\min\{2^{w/2}, 2^{n(K-1)/K}\})$.*

To ensure that H is (asymptotically) as secure against multi-collision attacks as an ideal hash, $w \geq 2n$ is thus sufficient in the random oracle model.

3.2 K-way (2nd) Preimage Attacks

Joux (2nd) preimage attack also works for the wide-pipe hash. Finding 2^k-way (2nd) preimages takes time $O(k * 2^{w/2} + 2^n)$. As will be shown

below, this bound is tight, except for the (logarithmic) factor k. Let T' denote the time to find a collision for C' (as in the previous section) and $P''(K)$ the time to find a K-way preimage for C''. Our lower bound on the security of H is now:

Lemma 6 *Consider the wide-pipe hash H:*

1. *Finding a single preimage for H takes time $\Omega(P''(1))$.*
2. *Finding K-way preimages for H takes time $\Omega(\min\{T', P''(K)\})$.*

Proof. 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 $M^1, \ldots, M^K$ for H either implies finding at least one collision for C', or implies finding K different inputs $H^1_{L^1}, \ldots H^K_{L^K}$ with $C''(H^1_{L^1}) = \cdots = C''(H^K_{L^K}) = Y$, i.e., a K-way preimage for C''. □

Why don't we prove the security of H against (multiple) 2nd preimage attacks, similarly to the second bound? A 2nd preimage attack against C'' means that, given $X \in \{0,1\}^w$, the adversary has to find $X' \in \{0,1\}^w$ with $X' \neq X$ and $C''(X) = C''(X')$. To reproduce the reduction from the 2nd bound of the above proof, we would have to find a message M with $H_L = X$ for the intermediate hash H_L of M. This is (or should be) hard. In the random oracle model, a little trick allows us to show that finding 2nd preimages is as infeasible as finding plain preimages.

Theorem 7. *Consider the wide-pipe hash function H. If we model the compression functions C' and C'' as independent random oracles, then*

- *finding a single preimage takes time $\Omega(2^n)$,*
- *finding a K-way preimage takes time $\Omega(\min\{2^{w/2}, K2^n\})$, and*
- *finding a K-way 2nd preimage takes time $\Omega(\min\{2^{w/2}, K2^n\})$, as well.*

Proof. The first two bounds are direct consequences of Lemma 6 For the thrird bound, we choose an arbitrary message M with the expansion M_1, $\ldots$, M_L, query the C'-oracle for the internal hash values H_1, $\ldots$, H_L, and define

$$C''' : \{0,1\}^w \to \{0,1\}^n : \quad \begin{cases} C'''(H_L) = C''(X), \\ C'''(X) \;\; = C''(H_L), \\ C'''(Z) \;\; = C''(Z) \quad \text{if } Z \notin \{X, H_L\}. \end{cases}$$

Note that if $X = H_L$, then $C'' = C'''$. Now we run the adversary to find single or multiple 2nd preimages for M, replacing C'' by C'''. Observe that X is a random value, and, since C' is a random oracle, H_L is random, too. Thus, C''' is uniformly distributed random function, just like C'' – the adversary can't distinguish between C'' and C'''. Our little manipulation (replacing C'' by C''' for the adversary) does not affect her probability of success or running time. We write H''' for the wide-pipe hash function using C' and C'''.

If the adversary succeeds, she finds 2nd preimage(s) M^i with $H'''(M) = H'''(M^i)$. Consider the corresponding inputs $H^i_{L^i}$ for C'''. If $H^i_{L^i} = H_L$, we have found a collision for C'. Else, $H^i_{L^i}$ is a 2nd preimage for C''. □

Note that increasing w improves the security of H against multiple (2nd) preimage attacks, but an unlimited adversary can always benefit from the structure of any iterated hash by applying Joux' multiple preimage attack.

4 The Double-Pipe Hash (Two Twined Pipes)

The wide-pipe design in Section 3 suffers from one serious drawback: To achieve the amount of security an n-bit hash function *should* have, we need an internal building block with an extremely high level of security. Namely, any collision attack for the w-bit compression function C'' has to take at least time 2^n ($w \geq 2n$ is necessary, but not sufficient).

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? [5]

Using *one single narrow-pipe* compression function

– $C : \{0,1\}^n \times \{0,1\}^{n+m} \rightarrow \{0,1\}^n$,

with $m \geq n$ and three distinct (random) *initial values* $H'_0, H''_0, H^* \in \{0,1\}^w$, we compute the double-pipe hash H:

– For i in 1, ..., L: compute

[5] E.g., if we assume the internal compression function of SHA1 to be as secure as we would expect from a 160-bit compression function, can we show that some "double-pipe" SHA1 significantly improves on the security of "normal" SHA1?

- $H'_i := C(H'_{i-1}, H''_{i-1}||M_i)$ and
- $H''_i := C(H''_{i-1}, H'_{i-1}||M_i)$

– Finally: set $H(M) = C(H^*, H'_L||H''_L||0^{m-n})$.

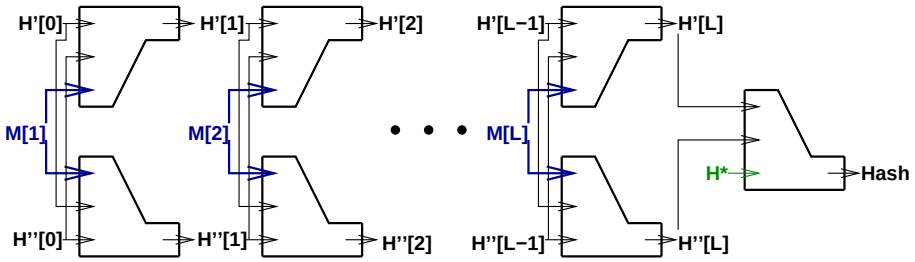

Fig. 4. The Double-Pipe Hash

4.1 *K*-Collision Attacks

Similarly to the wide-pipe design, we distinguish internal collisions (corresponding to collisions for C') and final collisions (corresponding to C''):

Final collision: $(H', H'') \neq (G', G'')$ with

$$C(H^*, H'||H''||0^{m-n}) = C(H^*, G'||G''||0^{m-n})).$$

Internal collision: $(H', H'', M) \neq (G', G'', N)$ with

$$C(H'', H'||M) = C(G'', G'||N) \quad \text{and} \quad C(H', H''||M) = C(G', G''||N).$$

The improved security of the wide-pipe hash over the plain MD hash depends on internal collision resistance being much stronger than final collision resistance. Unfortunately, this reasoning does not hold for the double-pipe construction. Finding internal collisions with $H' = H''$ and $G' = G''$ may be as "easy" as finding collisions for C, i.e., as finding final collisions. To deal with this, we define two special cases of internal collisions:

Strict internal collision: internal collision with

$$H' \neq H'' \quad \text{and} \quad G' \neq G''.$$

Internal cross collision: $H'_{i-1} \neq H''_{i-1}$, M_i with

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

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.

Lemma 8 *Consider the double-pipe iterated hash H:*

1. *Any internal collision either reduces to a strict or to a cross collision.*
2. *Finding a K-collision requires time $\Omega(\min\{T_{\mathrm{S}}, T_{\mathrm{X}}, T(K)\})$.*

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 $j \leq i-2$, $H'_j \neq H''_j$, 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 argue similarly to the proof of Lemma 4. 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}). □

Theorem 9. *Consider the double-pipe hash H. If we model the compression function C as a random oracle, then*

1. *$T_{\mathrm{X}} = \Omega(2^n)$, $T_{\mathrm{S}} = \Omega(2^n)$, and*
2. *finding K-collisions for H takes time $\Omega(2^{n(K-1)/K})$.*

Proof. Consider finding internal cross collisions. Each time we choose $H' \neq H''$ and M, there is a 2^{-n}-chance for a collision $C(H', H''||M) = C(H'', H'||M)$. Thus, a cross collision needs $\Omega(2^n)$ oracle queries, i.e., $T_{\mathrm{X}} = \Omega(2^n)$.

Now consider finding strict internal collisions. For any triple (G', G'', M) with $G' \neq G''$, the pair $(H', H'') \in \{0,1\}^{2n}$ with

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

is a uniformly distributed $2n$-bit random value, independently from all the other $C(\cdot)$-values. If the adversary chooses q such triples (G', G'', M) and makes q queries to the C-oracle, then her probability of success is $\sum_{0 \le j < q} j/2^{2n} = \Omega(q^2/2^{2n})$. Again, we expect to make $q = \Omega(2^n)$ oracle queries, before the first strict internal collision. Hence, $T_{\mathrm{S}} = \Omega(2^n)$.

The second claim follows from the first claim, Lemma 8, and Fact 2. □

4.2 *K*-way (2nd) Preimage Attacks

Our treatment of K-way preimage and 2nd preimage attacks is quite similar to Section 3.2. Recall the notions of strict internal collisions and internal cross collisions. Finding such collisions requires time T_{S} for strict and time T_{X} for cross collisions. Write $P(K)$ for the time to find a preimage for C. Very similar to Section 4.2 we get the following results:

Lemma 10 *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 $\Omega(\min\{T_{\mathrm{S}}, T_{\mathrm{X}}, P(K)\})$.*

Proof. First claim: See proof of Lemma 6. Second claim: Follows from claim 1 of Lemma 8. □

Theorem 11. *Consider the double-pipe hash function H. If we model the compression functions C as a random oracle, then finding a single or K-way preimage or a single or K-way 2nd preimage takes time $\Omega(2^n)$.*

The proof of Theorem 11 is straightforward and omitted here. The result may appear rather unimpressive – but it is tight, except for the factor $k = \log(K)$. Joux' preimage attack allows to find 2^k-way preimages in time $\Omega(k2^n)$.

5 Double-Pipe Hash with Davies-Meyer (DM)

So far, we did treat the compression function like a random oracle (with fixed input size). 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 Double-Pipe Hash (as motivated in Footnote 5), 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

$$C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n, \quad C(H_{i-1}, E_i) = E_{M_i}(H_{i-1}) + H_{i-1}.$$

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

5.1 Conventions

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 $E_k^{-1}(y)$, nor for $E_k(x)$ again. Similarly, if she has asked for for $E_k^{-1}(y)$ 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 $\overline{\text{RANGE}}(E_k)$
    - append $(x_i, k_i, y_i) := (x, k, y)$ to the logbook;
    - respond y;
  - Responding to an oracle query $E_k^{-1}(y)$:
    - set $i := i + 1$
    - randomly choose x from $\overline{\text{DOMAIN}}(E_k)$
    - append $(x_i, k_i, y_i) := (x, k, y)$ to the logbook;
    - respond x;

  Here, $\overline{\text{DOMAIN}}(E_k)$ is the set of points x where $E_k(x)$ is still undefined. Similarly, $\overline{\text{RANGE}}(E_k)$ is the set of points y where $E_k^{-1}(y)$ is still undefined.

  For our proofs, we will discuss the logbook entries (x_i, k_i, y_i).

This is without loss of generality: 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.

5.2 Internal Collisions

Note that Lemma 8 is still valid in the current context. Recall the definitions of T_{S} and T_{X} .

Theorem 12. *Consider the DM-based double-pipe hash H. If we model E by a Shannon oracle, then $T_{\mathrm{X}} = \Omega(2^n)$ and $T_{\mathrm{S}} = \Omega(2^n)$.*

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

Time T_{X} to find internal cross collisions: a cross collision is described by $H'_{i-1} \neq H''_{i-1}$, M_i with

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

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 $q \leq 2^{n-1}$, for each such triple the probability p_x to satisfy Equation 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 $T_{\mathrm{X}} = \Omega(2^n)$, as claimed.

We still have to show $p_x \leq 2^{n-1}$. 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 $a \neq b$,

$$\begin{array}{lll} x_a = H'_{i-1},\ k_a = (H''_{i-1}||M_i), & & y_a = E_{k_a}(x_a), \\ x_b = H''_{i-1},\ k_b = (H'_{i-1}||M_i), & \text{and} & y_b = E_{k_b}(x_b). \end{array}$$

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

$$y_a + x_a = y_b + x_b. \tag{2}$$

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 $\overline{\text{RANGE}}(E_{k_b})$.
– Else x_b is a random value from $\overline{\text{DOMAIN}}(E_{k_b})$.

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

Time T_{S} to find strict internal collisions: for triples (G', G'', M) with $H' \neq H''$, we consider pairs $(H', H'') \in \{0,1\}^{2n}$, where

$$H' = C(G', G''||M) \quad \text{and} \quad H'' = C(G'', G'||M). \tag{3}$$

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 is $p_s \leq 1/2^{2(n-1)}$. Hence, the expected number of oracle queries to get a strict collision is $T_{\text{S}} = \Omega(2^n)$.

It remains to prove $p_s \leq 1/2^{2(n-1)}$. 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

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

A strict collision implies the adversary to handle a colliding triple (F', F'', N), i.e., $H' = C(F', F''||N)$ 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

$$H' = y_a + x_a = y_c + x_c \tag{4}$$

$$H'' = y_b + x_b = y_d + x_d. \tag{5}$$

Each of these two equations is of the same type as Equation 2. As in that context, we argue that due to $q \leq 2^{n-1}$ the probability for Eq. 4 to hold is no more than $1/2^{n-1}$; similarly for Eq. 5. More importantly, the conditional probability to satisfy Eq. 5, assuming Eq. 4 is at most $1/2^{n-1}$. Thus, the joint probability p_s for *both* Eq. 4 *and* Eq. 5 is $p_s \leq 1/2^{2(n-1)}$. □

5.3 K-Collisions

Theorem 13. *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 8 and Theorem 12, we know that an internal collision would take time $\Omega(2^n)$. Thus, in time $\Omega(2^{(n-1)(K-1)/K})$ 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

$$C(H^*, G^1||H^1||0^{m-n}) = \cdots = C(H^*, G^K||H^K||0^{m-n}).$$

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

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

or equivalently

$$\overbrace{E_{k_1}(H^*)}^{y_1} = \cdots = \overbrace{E_{k_K}(H^*)}^{y_K}.$$

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 $\Omega(2^{(K-1)n/K})$. □

5.4 K-way (2nd) Preimages

Theorem 14. *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 10, finding a single preimage for H 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 quite similarly to the proof of Theorem 7. We choose some message M and compute the internal hashes H'_1, H''_1, ..., H'_L, ..., H''_L. Assume $X \notin \{(H'_i||H''_i||M_i), (H''_i||H'_i||M_i) \mid 1 \leq i \leq L\}$ (this holds with overwhelming probability). Set $H_L := (H'_L||H''_L||0^{n-m})$. We define the function

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

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' 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 $\Omega(2^n)$. □

6 Discussion

6.1 Lessons to be Learned

The main lecture from [6, 7] and the current paper 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 security architect, choosing a cryptographic hash, should choose both w and n according to her specific security requirements (also considering, of course, efficiency concerns, compatibility issues, ...). For some applications, the Merkle-Damgård 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 $w = n$ (i.e., 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 to feasibly find large multi-collisions for the plain MD hash. I.e., finding 2^k-collisions takes time $k * \text{time}(\mathcal{A})$. 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 double-pipe hash (in comparison to an ideal hash) would be $2^{n/2}/\text{time}(\mathcal{A})$. 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.

6.2 Examples

As a concrete example, consider an AES-based MD hash $H^{\text{MD}}_{\text{AES}}$, using the AES block cipher in Davies-Meyer mode. Since the AES block size is 128 bit, $H^{\text{MD}}_{\text{AES}}$ is a 128-bit hash. For applications which do not require collision resistance, it may be fine to use a 128-bit hash. But resistance against multi-collision attacks or 2nd preimage attacks could be a concern for these applications – and from Joux' and Kelsey's attacks, we know that $H^{\text{MD}}_{\text{AES}}$ is much less resistant against these attacks than we would expect from a 128-bit hash. For a reasonably funded and motivated adversary, it is possible to find, say, a 2^{16}-collision for $H^{\text{MD}}_{\text{AES}}$.

In contrast to $H^{\text{MD}}_{\text{AES}}$, its double-pipe counterpart provides a greatly improved protection against these attacks (assuming the AES does not suffer from some still unknown cryptanalytic weaknesses). Even finding a 3-collision for a double-pipe 128-bit hash would take more than 2^{80} units of running time and therefor seems to be infeasible, today.

Interestingly, two of the five hash functions from the SHA standard [9], namely SHA-224 and SHA-348, have already been designed according to this paper's "wide-pipe" paradigm, see Table 1. This may have been motivated by the intention to re-use compression functions,[6] but one could as well imagine the immediate truncation of the internal hash values after each iteration. In the light of this paper, the designers of SHA-224 and SHA-348 did choose well.

[6] SHA-224 uses the compression function from SHA-256, and SHA-384 uses the compression function from SHA-512.

	n	w
SHA-1	160	160
SHA-224	224	256
SHA-256	256	256
SHA-384	384	512
SHA-512	512	512

Table 1. SHA standard hash functions: final hash size n and internal hash size w [9].

6.3 Cascading

The idea to improve the security of hash functions by cascading has been discussed for a long time, see, e.g., [10]. Cascading looks like an obvious technique to improve the security of hash functions – but due to Joux' attack, cascading iterated hash functions is not such useful. On the other hand, the double-pipe construction can be seen as *a cascade of compression functions*. As our results indicate, cascading compression functions can greatly improve the security. Indeed, one could extend the double-pipe hash and define some "t-tuple hash", to provide improved resistance against K-way (2nd) preimage attacks.

Thus, in the context of cascading and iterated hash functions, we argue that cascading compression function(s) is more desirable than cascading hash function(s).

6.4 Summary

In the current paper, we took a rather abstract and proof-centric look at the design of hash functions. Similarly to others, the current author considers this style a "feasible and useful step for understanding the security" [1] of iterated hash functions, thereby complementing the attack-centric approach [6, 7], 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 [2], our lemmas provide some specific requirements for the compression functions.

Acknowledgement

The author thanks Frederik Armknecht and John Kelsey.

References

1. Black, Rogaway, Shrimpton. Black-box analysis of the block-cipher based hash-function construction from PGV. Crypto 02.
2. R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. 30th STOC 1998, pp. 209–218.
3. I. Damgård. A design principle for hash functions. Crypto 89, LNCS 435, pp. 416–427.
4. N. Ferguson, B. Schneier. Practical Cryptography. Wiley Publishing, 2003.
5. H. Finney. More problems with hash functions. The cryptography mailing list. 24 Aug 2004. http://lists.virus.org/cryptography-0408/msg00124.html
6. A. Joux. Multicollisions in iterated hash functions, application to cascaded constructions. Crypto 04, LNCS 3152, pp. 306–316.
7. J. Kelsey. A long-message attack on SHAx, MDx, Tiger, N-Hash, Whirlpool, and Snefru. Draft. Unpublished Manuscript.
8. R. Merkle. One-way hash functions and DES. Crypto 89, LNCS 435, pp. 428–446.
9. National Institute of Standards and Technology (NIST). Secure hash standard. FIPS 180-2. August 2002.
10. B. Preneel. Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit Leuven, 1993.
11. B. Schneier. Cryptanalysis of MD5 and SHA. Crypto-Gram Newsletter, September 2004. http://www.schneier.com/crypto-gram-0409.html#3