A Failure-Friendly Design Principle for Hash Functions

Stefan Lucks

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

Abstract. This paper reconsiders the established Merkle-Damgård design principle for iterated hash functions. The internal state size w of an iterated n-bit hash function is treated as a security parameter of its own right. In a formal model, we show that increasing w quantifiably improves security against certain attacks, even if the compression function fails to be collision resistant. We propose the wide-pipe hash, internally using a w-bit compression function, and the double-pipe hash, with $w = 2n$ and an n-bit compression function used twice in parallel.

Keywords: hash function, provable security, multi-collision, failure-friendliness.

1 Introduction

A cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^n$ maps an infinite set of inputs to the finite set of n-bit hash values. While collisions (inputs $X \neq Y$ with $H(X) = H(Y)$) necessarily exist, a hash function should be *collision resistant*: given H, it should be infeasible for an adversary to *actually find* any collisions. But what if a hash function fails to be collision resistant? This paper deals with *failure-friendly* hash functions providing *some* security even if collision resistance has failed. It has been inspired by recent advances in collision finding [25,26,27,28,1].

The design of today's cryptographic hash functions ubiquitously follows the Merkle/Damgård (MD) structure [16,6], iterating some underlying compression function. The hash function is collision resistant, if the compression function is. However, if computing a compression function collision is somehow feasible, the hash function may fail worse than expected. E.g., finding *multiple collisions* should be way more expensive than finding plain (2-)collisions – but Joux [11] disproved this for the MD design. Also, MD hash functions completely fail to defend against *2nd collision* attacks: If $H(M) = H(N)$ for any two messages M, N, then $H(M||S) = H(N||S)$ for all $S \in \{0,1\}^n$. (Technically, this assumes M and N to be "extended messages", see below.) In other words, given a single collision, an adversary can easily construct many more collisions. This has long been known, but recently been exploited to turn "random" collisions (as, e.g., for MD5 [26]) into "meaningful" ones [12,17,14,15]. Even a 2nd preimage like scenario is possible [7]: given any two texts T_1 and T_2, Daum and Lucks presented two corresponding PostScript files with identical MD5 hashes.

B. Roy (Ed.): ASIACRYPT 2005, LNCS 3788, pp. 474–494, 2005.

Our Contributions. This paper describes and analyses *failure-friendly* iterated hash functions. The goal is to defend against certain classes of attacks even if collision resistance fails. We propose and analyse variants of the Merkle-Damgård design, *increasing the internal state to $w > n$ bits*. The *wide-pipe* hash is quite similar to the Merkle-Damgård hash, except for using a "largish" w-bit compression function to finally generate $n < w$ bits of output. The *double-pipe* hash sets $w = 2n$ and employs one single n-bit compression function, used twice in parallel for each message block. In random and standard model settings, we prove the security of our schemes against K-**collision** attacks (for $K \geq w$), and K-way **preimage** and **2nd preimage** attacks (for $K \geq 1$). Additionally, we discuss and semi-formally verify the resistance against **2nd collision** attacks.

Related Proposals. The double-pipe hash may remind the readers of the RIPEMD-family of hash functions [22,8], also calling two compression functions in parallel. The hash functions specified in [22,8] combine both n-bit compression values into a single n-bit state, strictly following the Merkle-Damgård design principle, thus being as failure-unfriendly as any Merkle-Damgård hash function. But [8] also outlines some double-width variants of RIPEMD-128 and -160, which we refer to as RIPEMD-256 and -320. RIPEMD-256 and -320 can almost be viewed as instantiation of our design principle – except for the following:

- By outputting **both** compression values at the end, RIPEMD-256 and -320 use the two n-bit compression functions like a single $2n$-bit compression function – again following the Merkle-Damgård design, thus, e.g., being entirely vulnerable to 2nd collision attacks.
- RIPEMD-256 and -320 were proposed as a a convenience feature for applications requiring a $2n$-bit hash "without needing a larger security level" [8]. On the other hand, our double-pipe construction has been designed *to improve the security* against certain attacks.

We propose a *generic and failure-friendly* design principle providing *provable security* under reasonable assumptions. Assuming a "good" n-bit compression function,[1] our analysis would justify the usage of, say, a failure-friendly variant of RIPEMD-320 with $2n = 320$ internal state bits and $n = 160$ output bits.

Recently, Coron et al. [5] also analysed variants of the Merkle-Damgård design in a fashion similar to the current paper. One of the proposals in [5] is rather similar to our wide-pipe design. However, [5] aims for variably-sized random oracles, based on an (extremely strong) ideal compression function (i.e., a fixed-size random oracle). This is orthogonal to our approach of taking possible compression function weaknesses into account. Nandi et. al. [18] proposed and analysed a rather different "2/3 rate double length compression function". Both [5] and [18] restrict their analysis to the random and Shannon oracle, while the current paper also provides some analyses in the standard model. Also, none of the constructions in [5,18] resemble the current paper's double-pipe hash design.

[1] Note that [8] took great care to ensure that both compression functions behave "differently enough". Somewhat surprisingly, our results indicate that it would even be OK to use the same compression function twice, instead of two different functions.

Road map. We first describe Merkle-Damgård hashing and introduce notations, abstractions, and attacks. Section 2 describes and analyses the wide-pipe hash, a modified Merkle-Damgård design with an extended internal state size. Section 3 modifies the wide-pipe hash, introducing and analysing the double-pipe hash. Section 4 investigates the security of a "weakened" double-pipe hash, based on a common construction for compression functions; see Appendix A for the proofs. Section 5 deals with extension attacks and Section 6 discusses our results and their implications. Appendix B provides examples for our hash constructions.

1.1 The Merkle-Damgård (MD) Principle for Iterated Hashing

A hash function H takes a message $M \in \{0,1\}^*$ to compute $H(M) \in \{0,1\}^n$. (In practice, the length $|M|$ of M may be bounded by some huge constant.) An iterated hash H is based on a compression function C with a fixed number of input bits and splits M into fixed-sized chunks $M_1, M_2, \ldots, M_L \in \{0,1\}^m$. The final chunk M_L may contain additional information, such as $|M|$. $(M_1, \ldots M_L)$ is the "expanded message". Assume a compression function $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ and a fixed *initial value* H_0. Given $M \in \{0,1\}^*$, one computes the MD hash 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. Thus, if $|M| \neq |M'|$, then $M_L \neq M'_{L'}$.)
- For $i \in \{1, \ldots, L\}$: compute $H_i := C(H_{i-1}, M_i)$.
- Finally: output H_L.

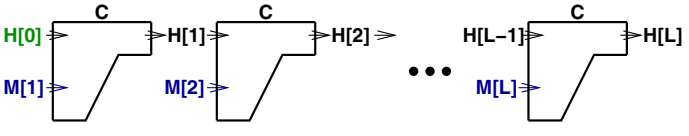

Fig. 1. The Merkle-Damgård (MD) Hash

Note that the MD hash function does not provide any resistance against *2nd collision attacks*: consider messages $M \neq M'$ with expansions $(M[1], \ldots, M[L])$ and $(M'[1], \ldots, M'[L])$. If M and M' collide, then $H[L] = H'[L]$ for $H[L] = C(\cdot, M[L])$ and $H'[L] = C(\cdot, M'[L])$, and therefore *all* expanded messages $(M[1], \ldots, M[L], S[1], \ldots, S[T])$ and $(M'[1], \ldots, M'[L], S[1], \ldots, S[T])$ also collide.

1.2 Notation, Abstractions, and Attacks

Random Oracles. A *fixed-size random oracle* is a function $f : \{0,1\}^a \to \{0,1\}^b$, chosen uniformly at random. For interesting sizes a and b, it is infeasible to implement f, 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, g is an infinite set of fixed-size random oracles $g_a : \{0,1\}^a \to \{0,1\}^b$ for $a \in \{0,1,2\ldots\}$. We view a fixed-size random oracle as an *ideal compression function*, and a variably-sized random oracle as an *ideal hash function*.

Shannon Oracle. An *ideal block cipher* is some invertible random oracle $E : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, such that for each $M \in \{0,1\}^m$, for the function $E(\cdot, M) = E_M(\cdot)$ an inverse $E^{-1}(\cdot, M)$ exists. Apart from that, E is uniformly chosen at random. Given x and M, one can ask a Shannon 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 computationally unbounded adversary with access to either a Shannon or a random oracle. The adversary's "running time" is determined by her number of oracle queries. Our 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. When necessary for clarity, we use the symbols O ("big-Oh", for "the expected running time is asymptotically *at most*") and Ω ("big-Omega", for "... *at least*"). [2]

Classes of Attacks. 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 [4]). Instead, one considers somewhat simpler security goals for $H : \{0,1\}^* \to \{0,1\}^n$. We consider the following classes of attacks:

K-collision for $K \geq 2$**:** Find K different M^i, with $H(M^1) = \cdots = H(M^K)$.
K-way (2nd) preimage for $K \geq 1$**:** Given Y (or M with $H(M) = Y$), find K different messages M^i, with $H(M^i) = Y$ (and $M^i \neq M$).
2nd collision: Given any collision $A \neq B$ with $H(A) = H(B)$, find C, D with $C \notin \{A, B, D\}$ and $H(C) = H(D)$.

The first two classes include "traditional" 2-collisions, 1-way preimages and 1-way 2nd preimages. Some applications need protection against the large-K-variants, e.g., [10,23,3]. The third class deals with a very natural assumption for "good" hash functions: even if the adversary somehow – with a great deal of luck, by doing much computational work, or by a mixture of both – has found *one* collision, it should still be *hard* to find *another one*. The poor defence of established hash functions against such attacks has been elaborated above.

Facts. Our analysis uses the following facts:

1. **Fact:** *Finding a K-collision for a fixed size random oracle $C : \{0,1\}^{n+m} \to \{0,1\}^n$ or for a variably-sized random oracle Model $H : \{0,1\}^* \to \{0,1\}^n$ takes time $\Omega(2^{(K-1)n/K})$, and finding a K-way preimage or a K-way 2nd preimage for H or C takes time $\Omega(K2^n)$.*
2. **Fact:** *Given a collision $A \neq B$ with $C(A) = C(B)$ for a fixed size random oracle $C\{0,1\}^{n+m} \to \{0,1\}^n$ (or $H(A) = H(B)$ for a variably-sized random oracle $H\{0,1\}^* \to \{0,1\}^n$), finding a 2nd collision $C \neq D$, $C \notin \{A, B\}$ for C (or H) takes time $\Omega(2^{n/2})$.*

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

Initial Values. Like the MD hash, our hash functions depend on the compression function(s) and an initial value (IV). One can set the IV to some fixed ("random") constant. But for our analysis, we will even allow the adversary to actually choose the IV.[3] This makes our results all the more meaningful.

Standard Model Formalism. For a fixed hash function $H : \{0,1\}^* \to \{0,1\}^n$, trivial algorithms to "find" collisions exist: given any $M \neq M'$ with $H(M) = H(M')$, output M and M'. Collision resistance implies the non-existence of algorithms to "find" collisions. Thus, for a standard model proof of collision resistance, we must refine our formalism. Instead of a fixed hash function, we actually consider a *hash function family* $H : \mathcal{I} \times \{0,1\}^* \to \{0,1\}^n$. Here, $\mathcal{I}$ is a finite nonempty set of indices (or "keys"). We assume an index $i^* \in \mathcal{I}$ being chosen uniformly at random, write $H(\cdot)$ instead of $H(i^*, \cdot)$ and consider the fixed hash function $H : \{0,1\}^* \to \{0,1\}^n$ as a *random member* of its family.

Fix some RAM model of computation. In any attack game, the adversary is given i^* as its first input. We measure the adversary's *expected running time* over uniformly distributed random i^* (and the adversary's internal coin flips, if applicable). To capture a trivial adversary using huge tables, the running time of any program is assumed to be at least linear in the program size.

We formalise compression functions C exactly like hash functions: assume a family $C : \mathcal{I}_C \times \{0,1\}^\alpha \to \{0,1\}^\beta$ and an index $i^C \in \mathcal{I}_C$ chosen uniformly at random, write $C(\cdot)$ instead of $C(i^C, \cdot)$, and consider the fixed compression function $C : \{0,1\}^\alpha \to \{0,1\}^\beta$ as a random member of its family. An adversary's running time is taken over random i^C. If H is defined by iterating C, a random member of the hash family H is defined by i^C and some random initial value H_0, i.e., $i^* = (i^C, H_0)$. Similarly, if H is constructed by applying C' and C'', then $i^* = (i^{C'}, i^{C''}, H_0)$. Recall that in our attacks we even allow the adversary to choose H_0. The adversary can make this choice *after* being given i^C or $(i^{C'}, i^{C''})$.

2 The Wide-Pipe Hash: A Modified MD Hash

Constructing a collision-resistant compression function with $w > n$ output bits may be simpler than constructing an n-bit compression function with the same level of collision resistance. The *wide-pipe hash* uses such a w-bit compression function to generate an n-bit hash value at the end. [4] This approach defeats Joux' attack – and even provides security against *all* generic K-collision attacks (which treat the compression function as a random oracle). Let $H_0 \in \{0,1\}^w$ be a (random) *initial value*. Using **two compression functions**

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

we compute the wide-pipe hash H:

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

[3] This is similar to the "aSec" and "aPre" notions of hash function security from [24].
[4] This idea has independently been proposed by Finney in a mailing list [9].

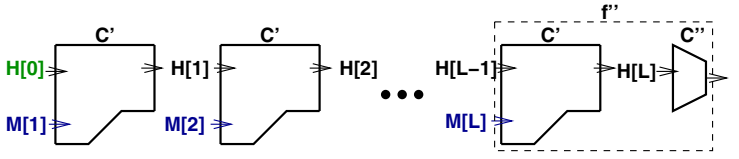

Fig. 2. The Wide-Pipe Hash

For technical reasons, we need to distinguish between different kinds of collisions. Consider $M \neq N$ 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 define

Final collisions: $H_L^M \neq H_{L'}^N$ but $C''(H_L^M) = C''(H_{L'}^N)$.
Internal collisions: $H_L^M = H_{L'}^N$. (Note that an internal collision implies a collision for C', i.e., $(H_i^M, M_i) \neq (H_i^N, N_i)$ with $C'(H_i^M, M_i) = C'(H_i^N, N_i)$.)
Final K-collisions: Any K-collision $M^1, \ldots, M^K$ (with $H(M^1) = \cdots = H(M^K)$) is *final*, if all 2-collisions (M^i, M^j) (with $i \neq j$) are final.

2.1 Resistance Against K-Collision Attacks

Observe that Joux finds 2^k-collisions in time $\min\{k * 2^{w/2}, 2^{n(2^k-1)/2^k}\}$. This tightly describes the security of H, up to the (logarithmic) factor k. Define the composition $f'' : \{0,1\}^w \times \{0,1\}^m \to \{0,1\}^n$ of C' and C'' by $f''(H, M) = C''(C'(H, M))$, as indicated in Figure 2. Make the following two assumptions:

1. C' is collision resistant, and **2.** f'' is K-collision resistant.

Under these assumptions, we prove the K-collision resistance of H.[5] For the concrete security analysis, we assume that finding a collision for C' takes at least time T', and finding a K-collision for f'' at least time $T''(K)$.

Lemma 1. *An adversary needs $\Omega(\min\{T', T''(K)\})$ units of time to find a K-collision for the wide-pipe Hash H, even if she can choose H_0.*

Proof. Any final K-collision is equivalent to a K-collision for f''. On the other hand, if a K-collision for H is not a final K-collision, then an internal collision has been found. For all H_0, finding an internal collision is equivalent to finding a collision for C'. Thus, finding a K-collision for H is at least as hard as finding either a K-collision for f'', or a collision for C. □

In the random oracle model, H **is as secure against multi-collision attacks as an ideal hash** for $w \geq 2n$.

[5] It would seem natural to assume the K-collision resistance of C''. Indeed, f'' is K-collision resistant if C' is collision resistant and C'' is K-collision resistant. But even if C'' is K-collision vulnerable, f'' can still be K-collision resistant. E.g., model C' as a random oracle and set C'' to be the plain truncation of w-bit inputs to n-bit outputs. For $\log_2(K) \leq w - n$, C'' is trivially K-collision weak, but f'' is not.

Theorem 2. *Consider the wide-pipe hash H. Allow the adversary to choose H_0.*

1. *Model C' and C'' as independent random oracles. The adversary needs time $\Omega(\min\{2^{w/2}, 2^{n(K-1)/K}\})$ to find a K-collision for H.*
2. *Define $C'' : \{0,1\}^w \to \{0,1\}^n, C''(x_1, \ldots, x_w) = (x_1, \ldots, x_n)$ as the n-bit truncation of its w-bit input. Model C' as a random oracle. The adversary needs time $\Omega(\min\{2^{w/2}, 2^{n(K-1)/K}\})$ to find a K-collision for H.*

Proof. Due to Lemma 1, finding a K-collision takes time $\Omega(\min\{T', T''(K)\})$. By Fact 1, $T' = \Omega(2^{w/2})$. If C'' is an independent random oracle, then $T''(K) = \Omega(2^{n(K-1)/K})$. If C'' just truncates, then f'' can be viewed as a random oracle with n output bits. Again, this gives $T''(K) = \Omega(2^{n(K-1)/K})$. □

2.2 Resistance Against K-Way (2nd) Preimage Attacks

Joux' (2nd) preimage attack also works for the wide-pipe hash. Its time $O(k * 2^{w/2} + 2^n)$ tightly bounds the security of H, up to the (logarithmic) k. Let T' be a lower bound for finding collisions for C' (as before) and assume that finding K-way preimages for f'' takes at least time $P''(K)$.

Lemma 3. *Consider the wide-pipe hash H. Allow the adversary to choose H_0.*

1. *The adversary needs time $\Omega(P''(1))$ to find a single preimage for H.*
2. *She needs time $\Omega(\min\{T', P''(K)\})$ to find a K-way preimage for H.*

Proof. Finding a preimage for H implies finding a preimage for f''. Finding a K-way preimage for H either implies finding at least one internal collision – and thus a collision for C' – or a K-way preimage for f''. □

In the random oracle model, we also consider 2nd preimage attacks.

Theorem 4. *Consider the wide-pipe hash H. Model C' and C'' as independent random oracles. An adversary allowed to choose H_0 needs*

1. *time $\Omega(2^n)$ to find a single preimage for H,*
2. *time $\Omega(\min\{2^{w/2}\})$ to find a K-way preimage for H, and*
3. *time $\Omega(\min\{2^{w/2}, K2^n\})$ to find a K-way 2nd preimage for H.*

Proof. The first two bounds are direct consequences of Lemma 3 and Fact 1. Now consider 2nd preimages: given a random $X \in \{0,1\}^w$, we are searching for one or more different $X^i \in \{0,1\}^w$ with $C''(X) = C''(X^i)$. 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 with overwhelming probability $X \neq H_L$. 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 a 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 the adversary's 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 $H'''(M) = H'''(M^i)$. We write L^i for the length of the expansion of M^i (in chunks). Consider the 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''. □

Increasing w improves the security of H against multiple (2nd) preimage attacks. But an adversary whose running time exceeds $2^{w/2}$ can still run Joux' attack and benefit from the iterated structure of H. In fact, no hash function with some fixed internal state size w can be as secure against multiple (2nd) preimage attacks as an ideal hash.

3 The Double-Pipe Hash

There is one drawback for the wide-pipe design: its compression function C' needs a larger output and finding collisions for C' must be much harder than finding collisions for the hash function itself. It would be interesting to use a compression function which only has to satisfy essentially the same security requirements as the hash. For instance, if we assume the internal compression function of, SHA-1, RIPEMD-160, or SHA-256 to be as secure as an ideal 160-bit (256-bit for SHA-256) compression function, can we construct some variant to improve security? Note that the SHA-1 and RIPEMD-160 compression functions can be written as $C : \{0,1\}^{160} \times \{0,1\}^{512} \to \{0,1\}^{160}$, their SHA-256 counterpart as $C : \{0,1\}^{256} \times \{0,1\}^{512} \to \{0,1\}^{256}$. Thus, the following construction would be applicable to all of them: Using *one single narrow-pipe* compression function $C : \{0,1\}^n \times \{0,1\}^{n+m} \to \{0,1\}^n$, with $m \geq n$ and two distinct (random) *initial values* $H'_0 \neq H''_0 \in \{0,1\}^n$, we compute the double-pipe hash H^{D}:

- For $i \in \{1, \ldots, L-1\}$: compute
  - $H'_i := C(H'_{i-1}, H''_{i-1}||M_i)$ and
  - $H''_i := C(H''_{i-1}, H'_{i-1}||M_i)$
- Finally: $H^{\mathrm{D}}(M) := C(H'_{L-1}, H''_{L-1}||M_L)$

So in $H^{\mathrm{D}}(M)$, we have replaced the wide-pipe chaining values $H_{i-1} \in \{0,1\}^w$ by pairs $(H'_{i-1}, H''_{i-1}) \in (\{0,1\}^n)^2$. In each iteration, the value $H'_i = C(H'_{i-1}, H''_{i-1}||M_i)$ – one half of the new chaining value – functionally depends on both halfs H'_{i-1} and H''_{i-1} of the old chaining value (similarly for H''_i). This is vital for the security of the double-pipe hash. Otherwise, $H^{\mathrm{D}}(M)$ would degenerate into the cascade of two hash functions, thus being vulnerable to Joux' attack.

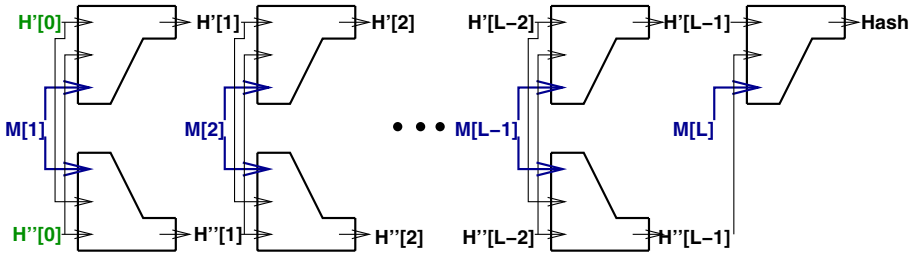

Fig. 3. The Double-Pipe Hash

3.1 Security Against Multiple Collision Attacks

In principle, the double-pipe hash is a special case of the wide-pipe hash with $w = 2n$ and $C'(H', H''||M) = (\, C(H'_{i-1}, H''_{i-1}||M_i),\ C(H''_{i-1}, H'_{i-1}||M_i)\,)$, where $C''(H', H'') = H'$ simply truncates $2n$ input bits to n output bits. (Thus, we do not need to compute the value $H''_L := C(H'_{L-1}, H''_{L-1}||M_L)$, as indicated in Figure 3.) Similarly to our analysis of the wide-pipe design, we distinguish internal collisions from final ones. 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, in addition to considering K-collisions, and make the following three assumptions:

1. *It is infeasible to find a* ***strict (internal) collision*** *for C, i.e., two triples $(H', H'', M) \neq (G', G'', N)$ with $H' \neq H''$ and $G' \neq G''$, but $C(H'', H'||M) = C(G'', G'||N)$ and $C(H', H''||M) = C(G', G''||N)$.*
2. *It is infeasible to find an* ***(internal) cross collision*** *for C: a triple (H', H'', M), with $H' \neq H''$ but $C(H', H''||M) = C(H'', H'||M_i)$.*
3. *It is infeasible to find K-collisions for C.*

We will prove H^{D} to be secure under the above three assumptions. While dealing with strict or cross collisions is unusual in cryptography, these assumptions appear to be natural and reasonable. We analyse the feasibility of finding strict or cross collisions for a random oracle C. For the concrete security analysis, we assume that finding strict collisions takes at least time T_{S}, finding cross collisions at least time T_{X}, and finding K-collisions at least time $T(K)$.

Theorem 5. *If we model the compression function C as a random oracle, then finding cross collisions for C needs time $T_{\mathrm{X}} = \Omega(2^n)$, and finding strict collisions for C needs time $T_{\mathrm{S}} = \Omega(2^n)$,*

Proof. First, consider T_{X}. Any triple (H', H'', M) can only be part of a cross collision, if $H' \neq H''$ and $C(H', H''||M) = C(H'', H'||M)$, i.e., with a probability of 2^{-n} (for $H' \neq H''$). Thus, we expect to make $T_{\mathrm{X}} = \Omega(2^n)$ oracle queries to find a cross collision.

Now consider T_{S}. For any triple (G', G'', M) with $G' \neq G''$, the pair $(H', H'') \in \{0,1\}^{2n}$ with $H' = C(G', G''||M)$ and $H'' = C(G'', G'||M)$ is a uniformly distributed $2n$-bit random value, chosen independently from all the other $C(\cdot,\cdot||\cdot)$-values. If the adversary chooses q different triples (G', G'', M) and makes q queries to the C-oracle, then her probability to succeed is $\sum_{0 \leq j < q} j/2^{2n} = \Omega(q^2/2^{2n})$. Thus, we expect to make $T_{\mathrm{S}} = q = \Omega(2^n)$ oracle queries to find a strict collision. □

Lemma 6. *Consider H^{D}. Allow the adversary to choose $H'_0 \neq H''_0$.*

1. *Any internal collision for H^{D} reduces to a strict or to a cross collision.*
2. *The adversary needs time $\Omega(\min\{T_{\mathrm{S}}, T_{\mathrm{X}}, T(K)\})$ to find a K-collision.*

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 $H'_{j+1} = C(H'_j, H''_j||M_{j+1}) = C(H''_j, H'_j||M_{j+1}) = H''_{j+1}$. Thus, any non-strict internal collision implies a cross collision.

For claim 2, we argue as in the proof of Lemma 1. A K-collision for H^{D} either reduces to a final K-collision (taking time $T(K)$), or to an internal collision. By the first claim, an internal collision is either strict (taking time T_{S}), or is a cross collision (taking time T_{X}). □

Theorem 7. *Consider H^{D}, and model C as a random oracle. An adversary who can choose $H'_0 \neq H''_0$ needs time $\Omega(2^{n(K-1)/K})$ to find K-collisions.*

Proof. The result follows from Theorem 5, Lemma 6, and Fact 1. □

3.2 Resistance Against K-Way (2nd) Preimage Attacks

Our treatment of K-way (2nd) preimage attacks is quite similar to Section 2.2. Let T_{S} and T_{X} be defined as above and assume finding preimages for C to take at least time $P(1)$.

Lemma 8. *Consider H^{D}. Allow the adversary to choose $H'_0 \neq H''_0$.*

1. *To find a single preimage, the adversary needs time $\Omega(P(1))$.*
2. *To find K-way preimages, the adversary needs time $\Omega(\min\{T_{\mathrm{S}}, T_{\mathrm{X}}, T(K)\})$.*

Proof. Claim 1: See proof of Lemma 3 with $f''(\cdot,\cdot||\cdot) := C(\cdot,\cdot||\cdot)$. Claim 2 follows from claim 1 of Lemma 6. Note that a K-way preimage also is a K-collision. □

Theorem 9. *Consider the double-pipe hash H^{D}. Model the compression function C as a random oracle. An adversary who can choose H_0 needs time $\Omega(2^n)$ for finding a single or K-way preimage or a single or K-way 2nd preimage.*

The proof of Theorem 9 is quite similar to the proof of Theorem 12 below.

Our results indicate that in the random oracle model, **the double-pipe hash H^{D} is asymptotically as secure as the wide-pipe hash with $w = 2n$.**

4 Davies-Meyer (DM) Compression Functions

If we trust an existing MD-hash to meet its security goal, it seems reasonable to use its compression function as the building-block C for the double-pipe hash. But most practical hash (or rather, compression) functions (including the SHA-family of hash functions, see Table 1) suffer from a specific structural weakness: They use a block cipher like function $E : \{0,1\}^{n+m} \times \{0,1\}^n \to \{0,1\}^n$, i.e., that for each "key" $K \in \{0,1\}^{n+m}$ the function $E(K,\cdot)$ permutates over $\{0,1\}^n$, and both $E(M,\cdot)$ and its inverse can efficiently be computed. A **DM compression function** $C : \{0,1\}^n \times \{0,1\}^{n+m} \to \{0,1\}^n$ is defined as follows:

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

(Here "+" is any group operation over $\{0,1\}^n$.) The ability to efficiently compute $E_M^{-1}(\cdot)$ can be useful for the adversary, see e.g. Kelsey and Schneier [13] for examples. Thus, we have to extend our formalism for the security proofs accordingly – by considering a Shannon oracle, instead of a random oracle.

4.1 Double-Pipe Hash with DM Compression Function

Some generic attacks against hash functions don't apply in the random oracle model, but are feasible in the Shannon model [13]. Fortunately, this does not pose a problem for the double-pipe hash. Those parts of our analysis of the double-pipe hash which do not assume random oracles are still relevant and applicable.[6] However, trusting those parts of our analysis which treat C as a random oracle would be risky. For this reason, we additionally analyse the double-pipe hash in the Shannon-model. See Appendix A for the proofs of the Theorems below.

Theorem 10. *Consider a DM compression function C. If we model E by a Shannon oracle, then $T_{\mathrm{X}} = \Omega(2^n)$ and $T_{\mathrm{S}} = \Omega(2^n)$.*

Theorem 11. *Consider H^{D} with a DM compression function C. If we model E by a Shannon oracle, then finding K-collisions takes time $\Omega(2^{(n-1)(K-1)/K})$.*

Theorem 12. *Consider H^{D} with a DM compression function C. 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 $P(1) = \Omega(2^n)$.*

5 Resistance Against 2nd Collision Attacks

Note that our definition of a 2nd collision attack assumes the adversary to be given the first collision essentially "for free". This is difficult to handle in the standard model. Thus, we concentrate on the random oracle model.

[6] Observe that the "DM compression function" is the function C with some specific non-random property. Given such C, the definition of H^{D} is the same.

In general, our hash designs do not protect against 2nd collision attacks: given an *internal collision*, attacking the wide-pipe or double-pipe hash is as easy as attacking the MD hash. Our design rationale, however, has been to defend against internal collisions, leaving final collisions as the "dotted line", where the hash function is likely to break (if it breaks at all). This is the foundation for the security proofs in the previous sections. In the remainder of this section, we thus focus on the specific case that the adversary is only given a *final collision*.

5.1 Wide-Pipe Hash: 2nd Collision Resistance

Consider the following attack: *fix* H_0, *choose* two incomplete expanded messages $(M_1, \ldots, M_{L-1})$ and $(N_1, \ldots, N_{L'-1})$, defining some pre-final internal states H^M_{L-1} and $H^N_{L'-1}$, *receive* a first collision and finally *provide* a 2nd collision. The first collision is defined by $M_L, N_{L'}$ such that the hash collides, but C' does not:

$$f''(H^M_{L-1}, M_L) = f''(H^N_{L'-1}, N_{L'-1}) \quad \text{but} \quad C'(H^M_{L-1}, M_L) = C'(H^N_{L'-1}, N_{L'-1})$$

In this section, we consider an attack game giving the adversary even more freedom: *choose* any H^M_{L-1} and $H^N_{L'-1}$, *receive* $M_L, N_{L'}$ for a first collision as above, and *provide* any four messages $A, B, C, D \in \{0,1\}^*$, $A \neq B$, $C \neq D$, $H(A) = H(B)$, $H(C) = H(D)$, with $C \notin \{A, B, D\}$.

Theorem 13. *Consider the wide-pipe hash H. Model C' as a random oracle. If C'' either is an independent random oracle, or the n-bit truncation of its w-bit input, the adversary needs time $\Omega(2^{n/2})$ to win the 2nd collision game for H.*

Proof (Sketch). Recall that we have got a first collision for f'', but no collision for C'. Finding messages $A, B, C, D \in \{0,1\}^*$ as required implies finding

- an internal collision (a collision for C'), taking time $\Omega(2^{w/2}) > \Omega(2^{n/2})$,
- or a 2nd collision for f'', namely intermediate hashes H_A, H_B, H_C, $H_D \in \{0,1\}^w$, and final message blocks M_A, M_B, M_C, $M_D \in \{0,1\}^m$ with
  $(H_A, M_A) \neq (H_B, M_B)$,
  $(H_C, M_C) \notin \{(H_A, M_A), (H_B, M_B), (H_D, M_D)\}$,
  $f''(H_A, M_A) = f''(H_B, M_B)$, and $f''(H_C, M_C) = f''(H_D, M_D)$.

We argue that finding a 2nd collision for f'' would take time $\Omega(2^n)$. If the 2nd collision for f'' includes a collision for C', then we need time time $\Omega(2^{w/2})$ to find it. Else, the 2nd collision is still as hard to find as a 2nd collision for any n-bit random oracle – both when C'' is an independent random oracle and when C'' plainly truncates –, thus taking time $\Omega(2^{n/2})$, see Fact 2. □

5.2 The Double-Pipe Hash: 2nd Collision Resistance

We adapt the attack game from above to the double-pipe hash: *choose* four arbitrary pairs $G' \neq G'', H' \neq H'' \in \{0,1\}^n$, *receive* $M, N \in \{0,1\}^m$ with $C(G', G''||M) = C(H', H''||N)$, and *provide* $A, B, C, D \in \{0,1\}^*$, with $A \neq B$, $C \neq D$, $H^{\mathrm{D}}(A) = H^{\mathrm{D}}(B)$, $H^{\mathrm{D}}(C) = H^{\mathrm{D}}(D)$, and $C \notin \{A, B, D\}$.

Theorem 14. *Consider the double-pipe hash H^{D}. Model C as a random oracle. The adversary needs time $\Omega(2^{n/2})$ to win the 2nd collision game for H^{D}.*

Proof (Sketch). As above, finding such $A, B, C, D \in \{0,1\}^*$ with $A \neq B$ and $C \notin \{A, B, D\}$, implies finding

- either an **internal collision**, taking time $\Omega(2^n)$ ($\rightarrow$ Lemma 6, Theorem 5)
- or **intermediate hashes** H'_A, H''_A, H'_B, H''_B, H'_C, H''_C, H'_D, $H''_D \in \{0,1\}^n$ and **final message blocks** $M_A, M_B, M_C, M_D \in \{0,1\}^m$ with

$$
\begin{aligned}
(H'_A, H''_A||M_A) &\neq (H'_B, H''_B||M_B),\\
(H'_C, H''_C||M_C) &\notin \{\ (H'_A, H''_A||M_A), (H'_B, H''_B||M_B), (H'_D, H''_D||M_D)\ \},\\
C(H'_A, H''_A||M_A) &= C(H'_B, H''_B||M_B), \quad \text{and}\\
C(H'_C, H''_C||M_C) &= C(H'_D, H''_D||M_D).
\end{aligned}
$$

The intermediate hashes and message blocks constitute a 2nd preimage for C. According to Fact 2, finding such a 2nd preimage takes time $\Omega(2^{n/2})$. □

Theorem 15. *Consider H^{D} with a DM compression function C. Model E by a Shannon oracle. Winning the 2nd collision game takes time $\Omega(2^{n/2})$.*

See Appendix A for a sketch of the proof.

6 Discussion

A Variant of the double-pipe hash. To reduce the set of cryptographic assumptions, Preneel [21] proposed to use $C : \{0,1\} \times \{0,1\}^n \times \{0,1\}^{n+m} \rightarrow \{0,1\}^n$ with one extra bit of input. Set $H'_i := C(0, H'_{i-1}, H''_{i-1}||M_i)$, $H''_i := C(1, H''_{i-1}, H'_{i-1}||M_i)$, and finally $\mathrm{Hash}(M) := C(0, H'_{L-1}, H''_{L-1}||M_L)$. Proofs of security for this variant of the double-pipe hash are very similar to the proofs for H^{D} itself, but without the need to assume finding cross collisions to be infeasible.

Two Independent Security Parameters. The main lesson from [11,13] and the current paper is that *the internal state size w* of an iterated hash function should be seen as *a security parameter of its own right.*

Any security architect choosing parameters for a cryptographic hash should choose both w and n according to her specific security requirements. For an application where even a single hash collision would be the ultimate disaster, $w = n$ suffices. If, on the other hand, additional multi-collisions or (multiple or single) preimages or 2nd preimages or feasible 2nd collisions would turn things from bad to worse, $w \gg n$ is recommendable, due to an improved failure mode.

2nd Collision Resistance. For applications such as digital signatures, 2nd collision resistance can have a *huge impact on practical security.* Our constructions are reasonably 2nd collision resistant. E.g., a double-pipe hash using the MD5 compression function would fail collision resistance due to [26], but for the double-pipe hash, this attack could only be used to generate final collisions. Accordingly, this double-pipe hash still defeats known exploits that make collisions "meaningful" [12,17,14,15,7].

Cascading. The idea to improve the security of hash functions by cascading has been discussed for a long time, see, e.g., [20]. Cascading looks like an obvious technique to improve the security of hash functions – but due to Joux' attack, cascading iterated hash functions is not that useful. On the other hand, the double-pipe construction can be seen as a cascade of compression functions. To this end, our double-pipe construction provides a *theoretically sound* technique to *cascade compression functions instead of the complete hash functions.*

Summary. This paper takes an abstract and proof-centric look at the design of hash functions. Similarly to [2], we consider our work a "feasible and useful step for understanding the security" of iterated hash functions, thereby complementing the attack-centric approach [11,13]. In the spirit of Merkle [16] and Damgård [6], this paper shows how to compose "good" hash functions, given "good" compression functions. We provide standard model explanations, what it means for the compression function to be "good". Additionally, we analyse the security of our constructions in the random oracle and Shannon model.

Acknowledgement. The author thanks Frederik Armknecht, John Kelsey, Ulrich Kühn, Arjen Lenstra, Bart Preneel, and the anonymous reviewers for their suggestions, discussions, and inspirations.

References

1. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, W. Jalby. Collisions of SHA-0 and reduced SHA-1. Eurocrypt 2005, LNCS 3494, 36–57.
2. J. Black, P. Rogaway, T. Shrimpton. Black-box analysis of the block-cipher based hash-function construction from PGV. Crypto 02, LNCS 2442, 320-335.
3. E. Brickell, D. Pointcheval, S. Vaudenay, M. Yung. Design validation for discrete logarithm based signature schemes. PKC 2000, LNCS 1751, 276–292
4. R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. 30th STOC 1998, 209–218.
5. J. Coron, Y. Dodis, C. Malinaud, P. Punyia. Merkle-Damgård revisited: how to construct a hash function. Accepted for Crypto 2005.
6. I. Damgård. A design principle for hash functions. Crypto 89, LNCS 435, 416–427.
7. M. Daum, S. Lucks. The story of Alice and her boss. Eurocrypt 05 rump session. `http://th.informatik.uni-mannheim.de/people/lucks/HashCollisions/`
8. H. Dobbertin, A. Bosselaers, B. Preneel. RIPEMD-160, a strengthened version of RIPEMD. FSE 1996, LNCS 1039, 71-82. See also `http://www.esat.kuleuven.ac.be/bosselae/ripemd160.html`
9. H. Finney. More problems with hash functions. The cryptography mailing list. 24 Aug 2004. `http://lists.virus.org/cryptography-0408/msg00124.html`
10. M. Girault, J. Stern. On the length of cryptographic hash-values used in identification schemes. Crypto 94, LNCS 839, 202–215
11. A. Joux. Multicollisions in iterated hash functions, application to cascaded constructions. Crypto 04, LNCS 3152, 306–316.
12. D. Kaminski. MD5 to be considered harmful someday. Dec. 2004. `http://www.doxpara.com/md5_someday.pdf`

13. J. Kelsey. B. Schneier. Second preimages on n-bit hash functions in much less than 2^n work. Eurocrypt 2005, LNCS 3494, 474–490.
14. A. Lenstra, B. de Weger. On the possibility of constructing meaningful hash collisions for public keys. To appear at ACISP 2005.
15. A. Lenstra, B. de Weger, X. Wang. Colliding X.509 certificates. Cryptology eprint archive report 2005/067. `http://eprint.iacr.org/2005/067/`
16. R. Merkle. One-way hash functions and DES. Crypto 89, LNCS 435, 428–446.
17. O. Mikle. Practical attacks on digital signatures using MD5 message digest. Cryptology eprint archive report 2004/356. `http://eprint.iacr.org/2004/356/`
18. M. Nandi, W. Lee, K. Sakurai, S. Lee. Security analysis of a 2/3-rate double length compression function in the black box model. FSE 2005, LNCS 3557, 243–254.
19. National Institute of Standards and Technology (NIST). Secure hash standard. FIPS 180-2. August 2002.
20. B. Preneel. Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit Leuven, 1993.
21. B. Preneel. Private communication. Hash Function Workshop, Krakow, 2005.
22. RIPE integrity primitives for secure information systems. Final report of RACE integrity primitives evaluation (RIPE RACE 1040). LNCS 1007, 1995.
23. R. Rivest, A. Shamir. PayWord and MicroMint – two simple micropayment schemes. CryptoBytes, Vol. 2, # 1 (1996), 7–11.
24. P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision-Resistance. FSE 2004, LNCS 3017, 371-388.
25. X. Wang, X. Lai, D. Feng, H. Cheng, X. Yu. Cryptoanalyisis of the hash functions MD4 and RIPEMD. Eurocrypt 2005, LNCS 3494, 1–18.
26. X. Wang, H. Yu. How to break MD5 and other hash functions. Eurocrypt 2005, LNCS 3494, 19–35.
27. X. Wang, H. Yu, Y. L. Yin. Efficient collision search attacks on SHA0. Accepted for Crypto 2005.
28. X. Wang, Y. L. Yin, H. Yu. Finding collisions in the full SHA1. Accepted for Crypto 2005.

Appendix

A Security of Double-Pipe Hash with Davies-Meyer

A.1 Conventions

In this section, we analyse the security of the double-pipe hash with a Davies-Meyer compression function. The adversary A has access to a Shannon oracle for E and E^{-1}. Similarly to [2], we assume:

- A never asks a query for 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 $E_k^{-1}(y)$ 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;
  - At any time, $\overline{\text{DOMAIN}}(E_k)$ denotes the set of points x where $E_k(x)$ is still undefined. Similarly we write $\overline{\text{RANGE}}(E_k)$, for the set of points y where $E_k^{-1}(y)$ is still 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;

  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 cannot distinguish the simulator from a "true" Shannon oracle.

A.2 Internal Collisions

Theorem 10. *Consider a DM compression function C. If we model E by a Shannon oracle, then $T_{\text{X}} = \Omega(2^n)$ and $T_{\text{S}} = \Omega(2^n)$.*

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 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 $T_{\text{X}} = \Omega(2^n)$, as claimed.

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

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

Thus, we can rewrite Equation 1 as

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

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 collisions: for triples (G', G'', M) with $G' \neq G''$, 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 collision consists of such a triple (G', G'', M) and another triple $(F', F'', N) \neq (G', G'', M)$ with

$$C(F', F''||N) = H' \quad \text{and} \quad C(F'', F'||N) = H''. \tag{4}$$

After q oracle queries, there are $\Omega(q^2)$ pairs $((G', G'', M),(F', F'', N))$ of triples. We claim that for $q \leq 2^{n-1}$, the probability p_s to satisfy Equation 4 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 another triple (F', F'', N) to satisfy Equation 4. This corresponds to two more triples (x_c, k_c, y_c) and (x_d, k_d, y_d) on the server's logfile with

$$H' = y_a + x_a = y_c + x_c \tag{5}$$

$$H'' = y_b + x_b = y_d + x_d. \tag{6}$$

Both equations are 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. 5 to hold is no more than $1/2^{n-1}$; similarly for Eq. 6. More importantly, the conditional probability to satisfy Eq. 6, assuming Eq. 5 is at most $1/2^{n-1}$. Thus, the joint probability p_s for *both* Eq. 5 *and* Eq. 6 is $p_s \leq 1/2^{2(n-1)}$. □

A.3 Resistance Against K-Collision Attacks

Theorem 11. *Consider H^{D} with a DM compression function C. If we model E by a Shannon oracle, then finding K-collisions takes time $\Omega(2^{(n-1)(K-1)/K})$.*

Proof. Due to the first claim of Lemma 6 and Theorem 10, we know that an internal collision would take time $\Omega(2^n)$. Thus, in time $\Omega(2^{(n-1)(K-1)/K})$ we cannot expect to find any internal collision. The only chance to find a K-way collision for H is finding a final K-collision, which takes time $T(K)$. In the remainder of this proof, we show $T(K) = \Omega(2^{(n-1)(K-1)/K})$. As in the proof of Theorem 10, we assume $q \leq 2^{n-1} = \Omega(2^n)$.

A final K-collision consists of K different triples with (G^i, H^i, M^i) with

$$C(G^1, H^1||M^1) = \cdots = C(G^K, H^K||M^K).$$

By the above conventions, this implies that there are K triples $(x_1, k_1, y_1), \ldots, (x_K, k_K, y_K)$ in the simulator's logbook with

$$\overbrace{E_{k_1}(x_1)}^{y_1} + x_1 = \cdots = \overbrace{E_{k_K}(x_K)}^{y_K} + x_K.$$

These are K sums $x_i + y_i$, and similarly to the proof of Theorem 10, for each such sum either x_i or y_i has been chosen from a huge subset $\{0,1\}^n$. Since $q \leq 2^{n-1}$, the size of this subset exceeds $2^n - q \geq 2^{n-1}$. For this reason, we expect to make $T(K) = \Omega(2^{(n-1)(K-1)/K})$ Shannon oracle queries for a K-collision. □

A.4 Resistance Against K-way (2nd) Preimage Attacks

Theorem 12. *Consider H^{D} with a DM compression function C. 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 $P(1) = \Omega(2^n)$.*

Proof. As in some of the proofs above, we assume $q \leq 2^{n-1}$.

Finding K-way (2nd) preimages isn't faster than finding single (2nd) preimages. Thus, we concentrate on single ones.

First, we start with singe preimages. Due to Lemma 8, finding a single preimage for the hash H^{D} takes time $\Omega(P(1))$, i.e., is asymptotically not faster than finding a preimage for the compression function $C(K, X) = E_K(X) + X$. Let a target Z be given, and an adversary is trying to find K and X with $C(K, X) = E_K(X) + X = Z$. By the above conventions, this corresponds to an entry (x_i, k_i, y_i) in the simulator's logbook with $x_i + y_i = Z$, and either x_i or y_i has been chosen from a huge subset of $\{0,1\}^n$ of size $> 2^n - q \geq 2^{n-1}$. Thus, for each query to the Shannon oracle, the probability to find a preimage for Z is at most 2^{n-1}, and we expect to make $P(1) = \Omega(2^n)$ such queries to find such a preimage.

Now consider 2nd preimages: assume an algorithm to find 2nd preimages for H^{D}. Consider we are given (K, N) and searching for some 2nd preimage $(K', N') \neq (K, N)$ with

$$C(K', N') = E_{K'}(N') + N' = E_K(N) + N = C(K, N).$$

The following technique resembles the proof of Theorem 4. We choose some message M, expand it to $(M_1, \ldots, M_L)$ and accordingly compute the internal hashes H'_1, H''_1, $\ldots$, H'_{L-1}, H''_{L-1}. Assume
$(K, N) \notin \{(H'_i, H''_i || M_i), (H''_i || H'_i || M_i) \mid 1 \leq i < L\}$ (this holds with overwhelming probability).

Set $N^{-1} := E_K^{-1}(Z - N)$ and define the function $E' : \{0,1\}^n \times \{0,1\}^{n+m} \to \{0,1\}^n$:

$$\begin{aligned} E'_K(N) &= Z - N \\ E'_K(N^{-1}) &= E_K(N) \\ E'_Q(R) &= E_Q(R) \text{ for } (Q, R) \notin \{(K, N), (K, N^{-1})\}. \end{aligned}$$

Now we run the adversary, replacing the (Shannon-) oracle for E and E^{-1} by an oracle for E' and its inverse. Observe that for the adversary $H^{\mathrm{D}}(M) = Z$ holds. Further, both E and E' are random permutations over $\{0,1\}^n$, so the adversary's chances of success are not affected by the change from E to E'.

Assume the adversary succeeds in finding a 2nd preimage $\overline{M}$ for M. Write $(\overline{M_1}, \ldots, \overline{M_L})$ for the expansion of $\overline{M}$ and $\overline{H'_1}$, $\overline{H''_1}$, $\ldots$, $\overline{H'_{L-1}}$, $\overline{H''_{L-1}}$ for the internal hashes.

- If $(\overline{H'_{L-1}}, \overline{H''_{L-1}}, \overline{M_L}) = (H'_{L-1}, H''_{L-1}, M_L)$, then the adversary has found an internal collision. From above, we know that this needs time $\min\{T_{\mathrm{X}}, T_{\mathrm{S}}\} = \Omega(2^n)$.
- Otherwise, $(\overline{H'_{L-1}}, \overline{H''_{L-1}}, \overline{M_L})$ is a preimage for Z. From above, we know that this takes time $P(1) = \Omega(2^n)$.

Thus, in order to find a 2nd preimage for H, the adversary either has to find an internal collision, or a 2nd preimage for C, and solving either problem takes time $\Omega(2^n)$. □

A.5 2nd Collision Resistance

Theorem 15. *Consider H^{D} with a DM compression function C. Model E by a Shannon oracle. Winning the 2nd collision game takes time $\Omega(2^{n/2})$.*

Proof (Sketch). Recall the proof of Theorem 14. A 2nd collision for H^{D} either implies an internal collision or a 2nd preimage for C. Finding an internal collision reduces to strict or internal collisions, thus taking time $\Omega(2^n)$ ($\to$ Theorem 10).

We still have to show that finding 2nd collisions for C takes time $\Omega(2^{n/2})$. From Theorem 11, we know that finding (first) collisions (i.e., K-collisions with

$K = 2$) takes time $\Omega(2^{n/2})$. In the proof of Theorem 11, finding such collisions for C is shown equivalent to the following task:

$$\begin{array}{llll} \text{find} & x_1, k_1, x_2, k_2 & \text{with} & E_{k_1}(x_1) + x_1 = E_{k_2}(x_2) + x_2, \\ & & \text{and} & (x_1, k_1) \neq (x_2, k_2). \end{array}$$

Similarly, finding 2nd collisions for C is equivalent to the task:

$$\begin{array}{llll} \text{given} & x_a, k_a, x_b, k_b & \text{with} & E_{k_a}(x_a) + x_a = E_{k_b}(x_b) + x_b \\ & & \text{with} & (x_1, k_1) \neq (x_2, k_2), \\ \text{find} & x_c, k_c, x_d, k_d & \text{with} & E_{k_c}(x_c) + x_c = E_{k_d}(x_d) + x_d, \\ & & \text{and} & (x_a, k_a) \neq (x_b, k_b), \\ & & \text{and} & (x_c, k_c) \notin \{ (x_a, k_a), (x_b, k_b), (x_c, k_c) \}. \end{array}$$

Regarding the second task, we replace the family E of permutations by a modified family E':

- Randomly choose x_a, k_a, x_b, k_b. Assume $k_a \neq k_b$ (this is overwhelmingly probable).
- Compute $y^* := E_{k_b}(x_b) + x_b - x_a$ and $x^* := E_{k_a}(y^*)$.
- Set $E'_{k_a}(x_a) := y^*$ and $E'_{k_a}(x^*) := E_{k_a}(x_a)$. Otherwise, E' behaves identical to E.
- Observe $E'_{k_a}(x_a) + x_a = E'_{k_b}(y_b)$. Given such x_a, k_a, x_b, k_b, solve the second collision task for E' instead of E. The solution is x_c, k_c, x_d, k_d as above.

With significant probability, we have $\{(x_c, k_c), (x_d, k_d)\} \cap \{(x_a, k_a), (x^*, k_a)\} = \{\}$. In this case, our 2nd collision for E' is a first collision for E. Thus, our proof reveals a technique to efficiently find collisions for C, if one can find 2nd collisions. Due to Theorem 11, finding such collisions takes time $\Omega(2^{n/2})$. □

B Examples

The SHA standard. Two of the five SHA-$*$ hash functions [19], namely SHA-224 and -384, have already been designed according to this paper's "wide-pipe" paradigm, see Table 1. Of course, the authors of SHA-224 and -384 where to reuse existing compression functions, but they could have done so – improving the hash function's performance – by truncating the internal hash values to 224 or 348 bit and extending the message chunk size by 256-224=32 or 512-348=128 bit. Our results provides some formal ("after the fact") justification for the design of SHA-224 and -348.

A natural choice for the parameters w and n would, however, be $w = 2n$. As an *example for the wide-pipe hash*, we could set $C' :=$ (SHA-512 compression function) and $C'' :=$ (SHA-256 compression function) to define a 256-bit hash with an internal hash size of $w = 512$. For large messages, this 256-bit hash would be about as fast as SHA-512. As an *example for a 256-bit double-pipe hash*, consider $C :=$ (SHA-256 compression function). The size of a SHA-256

Table 1. SHA standard hash functions and their parameters [19]

	final hash size n [bit]	internal hash size w [bit]	message chunk size [bit]	uses compression function from
SHA-1	160	160	512	(own)
SHA-224	224	256	512	SHA-256
SHA-256	256	256	512	(own)
SHA-384	384	512	1024	SHA-512
SHA-512	512	512	1024	(own)

message chunk is $m+n = 512$, so the size of a double-pipe message chunk would be $m = 512 - n = 256$ bit. For large messages, double-piped SHA-256 would be about four times slower than plain SHA-256. Similarly, a double-piped SHA-1 hash would be about three times slower than plain SHA-1.[7]

AES-based example for the double-pipe hash. Consider an AES-based MD hash $H^{\mathrm{MD}}_{\mathrm{AES}}$, using the AES block cipher in Davies-Meyer mode. The block size of $H^{\mathrm{MD}}_{\mathrm{AES}}$ is the AES block size: 128 bit. 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 the Joux and the Kelsey-Schneier attacks, we know that $H^{\mathrm{MD}}_{\mathrm{AES}}$ is much less resistant against these attacks than we would expect from a 128-bit hash. For a well funded and motivated adversary, it is possible to find, say, a 2^{16}-collision for $H^{\mathrm{MD}}_{\mathrm{AES}}$. This weakness does not much depend on the AES key size (either 128 bit, 192 bit, or 256 bit).

In contrast to $H^{\mathrm{MD}}_{\mathrm{AES}}$, its double-pipe counterpart (only defined for the AES key size of 256 bit) provides much better protection against these attacks, assuming the AES itself 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 therefore seems to be infeasible today. The price for the improved security is a performance penalty by a factor of four, similarly to double-piped SHA-256.

[7] Note that sharing initial values between different hash functions is never recommendable. Thus, H'_0 and H''_0 should not be taken from [19].