On the Impossibility of Efficiently Combining Collision Resistant Hash Functions

Dan Boneh[1]* and Xavier Boyen[2]

[1] Computer Science Dept., Stanford University — dabo@cs.stanford.edu
[2] Voltage Inc., Palo Alto — xb@boyen.org

Abstract. Let H_1, H_2 be two hash functions. We wish to construct a new hash function H that is collision resistant if at least one of H_1 or H_2 is collision resistant. Concatenating the output of H_1 and H_2 clearly works, but at the cost of doubling the hash output size. We ask whether a better construction exists, namely, can we hedge our bets without doubling the size of the output? We take a step towards answering this question in the negative — we show that any secure construction that evaluates each hash function once cannot output fewer bits than simply concatenating the given functions.

1 Introduction

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a function. A collision for H is a pair $M, M' \in \{0,1\}^*$ such that $H(M) = H(M')$ and $M \neq M'$. Roughly speaking, we say that H is collision resistant if no efficient algorithm can find collisions for H. Recent attacks [21, 20, 23, 22] on functions previously believed to be collision resistant greatly stimulated the search for new constructions as well as methods to avoid collision resistance altogether [17]. One natural question is how to combine existing hash functions. Suppose we are given two hash functions

$$H_1, H_2 : \{0,1\}^* \to \{0,1\}^v$$

that are currently believed to be collision resistant. We worry that one of these functions will become insecure due to a future attack. Hence, we want to hedge our bet and construct a new hash function $H : \{0,1\}^* \to \{0,1\}^n$ that is collision resistant assuming at least one of H_1 or H_2 is collision resistant. This can be easily achieved by concatenation. Simply define:

$$H(x) := H_1(x) \parallel H_2(x)$$

in which case H outputs digests of length $n = 2v$. It is easy to see that if either H_1 or H_2 is collision resistant then so is H. More precisely, concatenation has the following property:

(*) Any collision on H leads to a collision on **both** H_1 and H_2

* Supported by NSF.

Therefore, if finding collisions for *either* H_1 or H_2 is difficult then finding collisions for H must also be difficult. Note that property $(*)$ holds no matter what hash functions H_1 and H_2 are used. We give precise definitions in the next section.

In this paper we ask whether there are more clever ways to construct H. Specifically, is there a construction that satisfies property $(*)$, but whose output size n is less than $2v$ bits? A positive answer will give a clean way to combine two hash functions, such as SHA-512 [18] and Whirlpool [2], without increasing the output size by too much. Here we only consider constructions for H that evaluate H_1 and H_2 at most once. Concatenation is such an example.

Unfortunately, we show that when $n < 2v$ it is not possible to satisfy property $(*)$. In other words, there is no generic construction that combines arbitrary collision resistant hash functions H_1, H_2 and whose output is any shorter than concatenation. We give a precise statement of the result in the next section. Roughly speaking, our proof shows that for any construction with $n < 2v$ we can construct functions H_1, H_2 for which property $(*)$ fails. It is worth noting that our counterexamples for H_1, H_2 are realistic functions. For instance, they can be very similar to SHA-512 and Whirlpool. Our results apply to an arbitrary number of underlying hash functions and show that we cannot do better than concatenation, as long as each function is only used once.

We begin by precisely defining what it means for a construction combining ℓ hash functions to be secure. Intuitively, the construction should satisfy property $(*)$ — a collision on H should lead to a collision on all ℓ given hash functions. Consequently, if one of the ℓ functions is collision resistant then so is H. We then state and prove our impossibility results.

1.1 Related work

Robust combiners for various cryptographic primitives have received a steady amount of attention in the cryptographic community. Some examples include the early results of Asmuth and Blakely [1] on multiple encryption, Herzberg's work on combiners for commitment and one-way functions [9], Dodis and Katz's on encryption with chosen ciphertext security [7], and Harnik's *et al.* on key agreement [8]. Harnik *et al.* [8] provide lower bounds regarding the existence of "transparent" black-box combiners for oblivious transfer and secure computation [8]. We refer to Herzberg [9] for a survey of robust combiners for various cryptographic primitives as well as formal models. Hohenberger and Lysyanskaya recently investigated a related notion of "outsourcing" cryptographic operations by enlisting potentially malicious helpers to perform some of the calculations [11]. The very notion of increasing the security of encryption by using multiple independent keys dates back to the work of Shannon.

Our focus in this paper is on robust combiners for collision resistant hash functions. Many cryptographic hash functions today are based on the Merkle-Damgård paradigm [15, 6], which repeatedly applies a given compression function on successive message blocks in order to hash messages of arbitrary size. Joux [12] showed that the concatenation of two Merkle-Damgård functions is not much

more secure than the individual functions. In particular, let H_1, H_2 be Merkle-Damgård functions that output v bits each. Joux showed that a collision for $H = H_1 \parallel H_2$ can be found in expected time $O(v\, 2^{v/2})$. This is far less than the time for finding a collision on a random function outputting $2v$ bits. In other words, concatenation provides a good hedge, but does not increase security for Merkle-Damgård functions. Generalizations of Joux's attack were given by Nandi and Stinson [16] and Hoch and Shamir [10].

Kelsey [13] observed that truncating a collision resistant hash function need not be collision resistant. A special case of our results (the case $\ell = 1$ in Theorem 1) implies that *any* construction that evaluates a collision resistant function H and outputs fewer bits than the output of H need not be collision resistant.

For constructing hash functions, Lucks [14] studied the idea of increasing the internal state of the iterated hash beyond the size of the final output, specifically to defend it against Joux's attack. Along the same lines, Coron *et al.* [5] showed how to increase the security of an iterated hash function in which the compression function is viewed as a random oracle. They gave several techniques for building an economical Merkle-Damgård hash that becomes indistinguishable from a random oracle provided we stay below the birthday bound, by truncating the output or by encoding the input using a prefix-free code.

Generic constructions that attempt to build secure compression functions from ideal block ciphers have also been investigated. Preneel *et al.* [19] identified 12 constructions that were potentially secure. Black *et al.* [4] provided formal security proofs for all 12 constructions in the ideal cipher model. Some impossibility results have also been shown. Notably, Black *et al.* [3] proved that secure hash functions based on block ciphers cannot be "highly efficient", *i.e.*, require only one block cipher call per message block on a fixed set of keys.

2 Secure combination of collision resistant hashing

Suppose we wish to combine ℓ collision resistant hash functions $H_1, \ldots, H_\ell$ into a single function that is collision resistant if any of the H_i is collision resistant. We first define precisely what it means for such a construction to be secure. Throughout the section we let H_i be a function from $\{0,1\}^*$ to $\{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$.

A **collision resistant combination** of $H_1, \ldots, H_\ell$ is a pair (C, P) where C is a boolean circuit and P is a probabilistic algorithm such that:

- The circuit C is a boolean circuit that includes special "oracle" gates for evaluating the hash functions $H_1, \ldots, H_\ell$. We refer to $C^{H_1,\ldots,H_\ell}$ as the hash function constructed from $H_1, \ldots, H_\ell$. The circuit takes as input a string in $\{0,1\}^{\leq m}$, namely a string of length at most m, and outputs a digest in $\{0,1\}^n$. The upper bound on the input size merely reflects the finiteness of the circuit. This upper bound only strengthens our results since it shows that efficient combiners are not possible even when the input space for C is finite. We assume C is compressing so that m is (much) larger than n. The

output of C on input M and using hash functions $H_1, \ldots, H_\ell$ is denoted by $C^{H_1,\ldots,H_\ell}(M)$.

- Algorithm P is an oracle Turing machine. It provides the proof of security for C. The algorithm takes as input a pair of messages (M, M'). It then repeatedly queries the oracles $H_1, \ldots, H_\ell$ on inputs of its choice and finally outputs two vectors,

$$\boldsymbol{W} = (W_1, \ldots, W_\ell) \quad \text{and} \quad \boldsymbol{W}' = (W'_1, \ldots, W'_\ell)$$

  The output of P on input (M, M') is denoted by $P^{H_1,\ldots,H_\ell}(M, M')$.

The purpose of algorithm P is to convert any collision on $C^{H_1,\ldots,H_\ell}$ into collisions on all $H_1, \ldots, H_\ell$. Thus, we define security of a pair (C, P) as follows.

- Let $H_1, \ldots, H_\ell$ be functions where $H_i : \{0,1\}^* \to \{0,1\}^{v_i}$.
- Let $M, M' \in \{0,1\}^{\leq m}$ be a collision for the resulting function $C^{H_1,\ldots,H_\ell}$.

We say that P **succeeds** on this $(H_1, \ldots, H_\ell)$ and (M, M') if the output of $P^{H_1,\ldots,H_\ell}(M, M')$ is two vectors $\boldsymbol{W} = (W_1, \ldots, W_\ell)$ and $\boldsymbol{W}' = (W'_1, \ldots, W'_\ell)$ such that

$$\text{For all} \quad i = 1, \ldots, \ell : \qquad (W_i,\ W'_i) \quad \text{is a collision for } H_i.$$

We define the "advantage" $\mathrm{Adv}_P[(H_1, \ldots, H_\ell), (M, M')]$ as the probability that P succeeds on input (M, M') relative to the oracles $(H_1, \ldots, H_\ell)$. The probability is over the random bits used by algorithm P.

Definition 1. *We say that (C, P) is an ϵ-***secure collision resistant combination** *if for all $H_1, \ldots, H_\ell$ and all collisions (M, M') on $C^{H_1,\ldots,H_\ell}$ we have that*

$$Adv_P[(H_1, \ldots, H_\ell), (M, M')] > 1 - \epsilon$$

For example, for the concatenation construction discussed in the introduction, the pair (C, P) is as follows:

- $C^{H_1,H_2}(M) = H_1(M) \parallel H_2(M)$, and
- $P^{H_1,H_2}(M, M')$ simply outputs the two vectors (M, M) and (M', M').

This pair (C, P) is a secure collision resistant combination since for any H_1, H_2 and any collision (M, M') on C^{H_1,H_2} we have that

$$\mathrm{Adv}_P[(H_1, H_2), (M, M')] = 1$$

Our goal is to show that for any construction (C, P) where C outputs fewer bits than concatenation, we have that $\mathrm{Adv}_P[(H_1, \ldots, H_\ell), (M, M')]$ is negligible, for some $(H_1, \ldots, H_\ell)$ and some collision (M, M'). This will show that there is no provably secure way of combining generic hash functions that is better than concatenation. In this paper we focus on constructions (C, P) where C makes at most one call to each of $H_1, \ldots, H_\ell$. We do not discuss constructions where C uses some hash function multiple times.

3 Black-box impossibility results

Our main result states that, given any pair (C, P) where C evaluates each function once, it will always be possible to find a collision for $C^{H_1,\ldots,H_\ell}$ while preventing P from finding collisions for all the H_i. The only trivial exceptions are (1) when the output size of C is at least as large as the combined output sizes of all the H_i, which is to say that C does no better than concatenation, and (2) when the output of C is as large as its input, which is to say that C is non-compressing. Formally, we have the following theorem.

Theorem 1. *Let (C, P) be a collision resistant combination of oracles $H_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$, where $C : \{0,1\}^{\leq m} \to \{0,1\}^n$. Let $w = v_1 + \ldots + v_\ell$. Suppose that $n < w \leq m - \log_2 \ell$. Suppose further that C calls each of the H_i at most once. Then, there exist inputs $M, M' \in \{0,1\}^{\leq m}$ and functions $\hat{H}_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$, relative to which:*

$$Adv_P[(\hat{H}_1, \ldots, \hat{H}_\ell), (M, M')] \leq q^2/2^{v+1},$$

where $v = \min_{i=1}^{\ell}\{v_i\}$ is the smallest oracle output size, and q is the maximum number of oracle queries made by P during each execution.

The theorem shows that if C outputs fewer than $w = v_1 + \cdots + v_\ell$ bits then the probability of defeating P is overwhelming, provided that $q \ll \sqrt{2^{v+1}}$, namely, $q \ll \sqrt{2^{v_i+1}}$ for all $i = 1, \ldots, \ell$. The condition $q \ll \sqrt{2^{v+1}}$ must hold since otherwise one of the H_i is trivially not collision resistant and should not be used to begin with.

We prove Theorem 1 in two steps. First, in Section 3.1 we assume the existence of $H_1, \ldots, H_\ell$ and a pair $M \neq M'$ that is a collision for $C^{H_1,\ldots,H_\ell}$ satisfying a certain property. We use a randomization argument to build $\hat{H}_1, \ldots, \hat{H}_\ell$ that defeat P. Then, in Section 3.2 we show that for any C satisfying the conditions of Theorem 1 there must exist functions $H_1, \ldots, H_\ell$ and a collision $M \neq M'$ that satisfies the required property. This second step is the heart of the proof. These two results taken together will immediately prove Theorem 1.

Proof. Theorem 1 follows from Theorems 2 and 3. □

3.1 Randomization

Suppose there exist functions $H_1, \ldots, H_\ell$ and two messages M, M' such that:

- M, M' are a collision for $C^{H_1,\ldots,H_\ell}$, and
- there is some $j \in \{1, \ldots, \ell\}$ so that the process of evaluating $C^{H_1,\ldots,H_\ell}(M)$ and $C^{H_1,\ldots,H_\ell}(M')$ does not present a collision for H_j.

We show how to tweak the H_i into a new set of oracles, $\hat{H}_1, \ldots, \hat{H}_\ell$, such that P will fail to reduce a collision for $C^{\hat{H}_1,\ldots,\hat{H}_\ell}$ to a collision on all the $\hat{H}_i$. In Section 3.2 we show that such $H_1, \ldots, H_\ell$ and M, M' must exist.

To state the result more precisely, we first introduce some notation. As before, we let $H_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$ be a set of functions, and denote by $C^{H_1,\ldots,H_\ell} : \{0,1\}^m \to \{0,1\}^n$ the function evaluated by C when it is given oracle access to the H_i. Consider an input message $M \in \{0,1\}^m$. For each $i = 1, \ldots, \ell$, we define:

- $\mathbf{W}_i(M)$ to be the set of oracle queries to H_i made by C while evaluating $C^{H_1,\ldots,H_\ell}(M)$,
- $\mathbf{V}_i(M) = \{H_i(W) : W \in \mathbf{W}_i(M)\}$ to be the set of corresponding values output by H_i.

These sets are taken without multiplicity, and hence, we have a collision for H_j upon evaluating $C(M)$ if and only if $|\mathbf{W}_j(M)| > |\mathbf{V}_j(M)|$. If $|\mathbf{W}_j(M)| = |\mathbf{V}_j(M)|$ then no collision occurred for H_j. We can now state the following theorem.

Theorem 2. *Let (C, P) be a collision resistant combination of ℓ oracles as previously defined. Assume that there exist oracles $H_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$ and a pair of messages M, M', such that:*

- *$M \neq M'$ and $C^{H_1,\ldots,H_\ell}(M) = C^{H_1,\ldots,H_\ell}(M')$, and*
- *$|\mathbf{V}_j(M) \cup \mathbf{V}_j(M')| = |\mathbf{W}_j(M) \cup \mathbf{W}_j(M')|$ for some $j \in \{1, \ldots, \ell\}$.*

Let $v = \min_{i=1}^{\ell}\{v_i\}$. Suppose that P never makes more than q oracle queries upon each execution. Then, there exist oracles $\hat{H}_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$, relative to which:

$$Adv_P[(\hat{H}_1, \ldots, \hat{H}_\ell), (M, M')] \leq q^2/2^{v+1}$$

Proof. Let thus M, M', and H_i for $i = 1, \ldots, \ell$, be as stated in the theorem. We construct the $\hat{H}_i$ by patching H_i in a way that breaks P without affecting the output of $C(M)$ and $C(M')$. Basically, we instruct $\hat{H}_i$ to emulate H_i on all queries that appear in the course of evaluating one or both of $C^{H_1,\ldots,H_\ell}(M)$ and $C^{H_1,\ldots,H_\ell}(M')$. On all other inputs, the output of $\hat{H}_i$ is set to a fresh random string, so that $\hat{H}_i$ behaves as a random function on these inputs.

We now give an explicit construction of the $\hat{H}_i$ based on the H_i and the circuit C. For each $i \in \{1, \ldots, \ell\}$ we pick an independent random function $R_i : \{0,1\}^* \to \{0,1\}^{v_i}$. Then, for $i \in \{1, \ldots, \ell\}$, we define $\hat{H}_i$ as follows:

$$\hat{H}_i(W) := \begin{cases} H_i(W) & \text{if } W \in \mathbf{W}_i(M) \cup \mathbf{W}_i(M') \\ R_i(W) & \text{otherwise.} \end{cases}$$

Notice that the random function R_i can be selected and evaluated efficiently using a lazy algorithm that would pick and store a randomly drawn $R_i(W) \in \{0,1\}^{v_i}$ upon each novel query for $R_i(W)$.

It is easy to see that the messages M and M' still collide under $C^{\hat{H}_1,\ldots,\hat{H}_\ell}$ for the new oracle functions, since C is a deterministic function, and its evaluation on inputs M and M' is unaffected by the change of oracles.

Now, recall that j is the index of (one of) the hash oracle for which no collision occurred during the evaluations of $C^{H_1,\ldots,H_\ell}(M)$ and $C^{H_1,\ldots,H_\ell}(M')$, or, equivalently, during the evaluations of $C^{\hat{H}_1,\ldots,\hat{H}_\ell}(M)$ and $C^{\hat{H}_1,\ldots,\hat{H}_\ell}(M')$. At least one such $j \in \{1,\ldots,\ell\}$ exists by our assumptions.

Next, we consider the set of *all* distinct queries to $\hat{H}_j$ that are made during an execution of P on input (M, M'). Each output value $\hat{H}_j(x)$ returned by $\hat{H}_j$ on query x is either:

1. "unpatched", *i.e.*, so that $\hat{H}_j(x) = H_j(x)$, corresponding to a query in $\mathbf{W}_j(M) \cup \mathbf{W}_j(M')$;
2. "patched", *i.e.*, such that $\hat{H}_j(x) = R_j(x)$, in response to any other query.

We know that the few unpatched outputs of $\hat{H}_j$ are all distinct, otherwise the evaluations of $C(M)$ and $C(M')$ would have caused a collision on H_j and thus $\hat{H}_j$, contradicting our assumption. As for the patched outputs, by construction they are random binary strings of v_j bits.

Therefore, using the union bound we obtain the following bound on the probability that an arbitrary set of queries to $\hat{H}_j$ will result in a collision on $\hat{H}_j$:

$$\Pr\left[\begin{array}{c} q \text{ queries cause} \\ \text{a collision on } \hat{H}_j \end{array}\right] \leq \sum_{\substack{a \neq b \in \\ \{0,\ldots,q-1\}}} \Pr\left[\begin{array}{c} \text{query } a \text{ collides} \\ \text{with query } b \end{array}\right] \leq$$

$$\leq \sum_{\substack{a \neq b \in \\ \{0,\ldots,q-1\}}} \frac{1}{2^{v_j}} \leq \frac{q^2}{2^{v_j+1}} \leq \frac{q^2}{2^{v+1}}$$

We conclude that the probability that $P^{\hat{H}_1,\ldots,\hat{H}_\ell}(M, M')$ outputs a collision on $\hat{H}_j$ is at most $\frac{q^2}{2^{v+1}}$. The probability is over the random choice of $R_1,\ldots,R_\ell$ and the random bits used by P. But then by Markov there must exist some *fixed* setting of $R_1,\ldots,R_\ell$ so that for the corresponding $\hat{H}_1,\ldots,\hat{H}_\ell$, algorithm $P^{\hat{H}_1,\ldots,\hat{H}_\ell}(M, M')$ outputs a collision on $\hat{H}_j$ with probability at most $\frac{q^2}{2^{v+1}}$. This time the probability is only over the random bits of P. Therefore, the advantage $\mathrm{Adv}_P[(\hat{H}_1,\ldots,\hat{H}_\ell),(M, M')]$ is at most $q^2/2^{v+1}$ as required. □

3.2 Existence argument

We now turn to the second step of the proof of Theorem 1, which is the main part of the proof. Let C be a circuit that outputs elements in $\{0,1\}^n$ and uses hash functions (i.e. oracles) $H_1,\ldots,H_\ell : \{0,1\}^* \to \{0,1\}^{v_i}$. As in Section 3, let $w = v_1 + \ldots + v_\ell$. We assume that $n < w$ so that the output of C is at least one bit less than the concatenation of all the outputs of $H_1,\ldots,H_\ell$. In this section we will occasionally use the standard notation $A \to B$ to denote $(\neg A) \vee B$ where A and B are boolean expressions.

Suppose C uses each function H_i for $i = 1,\ldots,\ell$ at most once. We show that there exists $H_1,\ldots,H_\ell$ and $M, M' \in \{0,1\}^m$ such that: M, M' are a collision

for $C^{H_1,\ldots,H_\ell}$, but for some H_j $(1 \le j \le \ell)$ evaluating $C(M)$ and $C(M')$ causes no internal collision on H_j. The argument of Section 3.1 then shows that this M, M' pair cannot be used to find collisions on H_j proving that C cannot be a secure combination of $H_1, \ldots, H_\ell$.

More precisely, we state the main result of this section in the following theorem. We use the notation introduced at the beginning of Section 3.1.

Theorem 3. *Let C be a circuit computing a function from $\{0,1\}^m$ to $\{0,1\}^n$ using a set of oracles $H_i : \{0,1\}^* \to \{0,1\}^{v_i}$ for $i = 1, \ldots, \ell$. Furthermore, we assume that each H_i is used at most once in C. Let $w = v_1 + \ldots + v_\ell$. Then, whenever $n < w \le m - \log_2 \ell$, there exist functions $H_1, \ldots, H_\ell$ and messages M, M' such that:*

- $M \neq M'$ *and* $C^{H_1,\ldots,H_\ell}(M) = C^{H_1,\ldots,H_\ell}(M')$, *and*
- $|\mathbf{V}_j(M) \cup \mathbf{V}_j(M')| = |\mathbf{W}_j(M) \cup \mathbf{W}_j(M')|$ *for some* $j \in \{1, \ldots, \ell\}$.

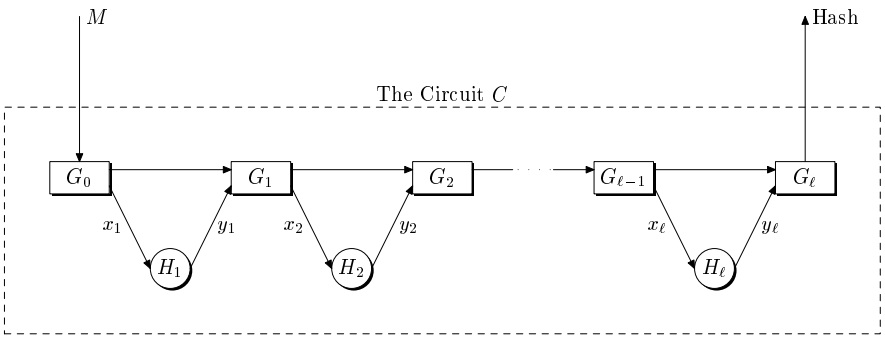

Fig. 1. Generic hash construction – the circuit C

Clearly the circuit C must use all hash functions $H_1, \ldots, H_\ell$. Otherwise, there is no hope of using a collision on C to get a collision on all H_i. Since we assumed that each H_i is used only once we know that C makes exactly ℓ oracle calls, one for each of $H_1, \ldots, H_\ell$. An inductive argument on the structure of C shows that, up to re-ordering the oracles, there exist circuits $G_0, G_1, \ldots, G_\ell$ so that C can be written as shown in Figure 1. Note that the wires connecting G_i and G_{i+1} can be quite thick, and in particular preserve all of the input and the state so far.

For an input message $M \in \{0,1\}^m$ let $\bar{y} = (y_1, \ldots, y_\ell)$ be the outputs of $H_1, \ldots, H_\ell$ while evaluating $C(M)$. Clearly the output of C is fully determined by M and $\bar{y}$. For $i = 1, \ldots, \ell$ we define the function:

$$X_i(M, y_1, \ldots, y_{i-1}) \quad : \quad \{0,1\}^{m+v_1+\ldots+v_{i-1}} \quad \to \quad \{0,1\}^*$$

to be the input x_i given to H_i when evaluating $C(M)$ and assuming that the output of the hash functions $H_1, \ldots, H_{i-1}$ is $y_1, \ldots, y_{i-1}$ respectively.

Theorem 3 can now be restated more directly. The theorem states that there exist functions $H_1, \ldots, H_\ell$ and inputs $M \neq M'$ such that $C(M) = C(M')$ and the following condition holds. Let $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ be the input/output pairs for $H_1, \ldots, H_\ell$ while evaluating $C(M)$. Similarly let $(x'_1, y'_1), \ldots, (x'_\ell, y'_\ell)$ be the input/output pairs while evaluating $C(M')$. Then there exists a $j \in \{1, \ldots, \ell\}$ for which

$$(y_j = y'_j) \rightarrow (x_j = x'_j)$$

In other words, no collision occurred on H_j.

Proof (of Theorem 3). The plan is to consider the set S of *all* 2^{m+w} tuples of the form $(M, y_1, \ldots, y_\ell)$. We often write $\bar{y} = (y_1, \ldots, y_\ell)$. Observe that a tuple $(M, \bar{y})$ in S completely defines the output of $C(M)$: simply define the output of hash function i to be y_i for all $i = 1, \ldots, \ell$. We view a tuple $(M, \bar{y})$ as an abstract object that is not derived from a specific instantiation of the hash functions.

We first partition S into equivalence classes based on the output of C. That is, two tuples $(M, \bar{y})$ and $(M', \bar{y}')$ are in the same equivalence class if C evaluates to the same output on both. Note that this partition of S is dependent only on C. It is independent of the choice of $H_1, \ldots, H_\ell$. Since C outputs elements in $\{0,1\}^n$ there are at most 2^n equivalence classes. It follows that there exists an equivalence class $E \subseteq S$ of size at least 2^{m+w-n}. Since $w > n$ this equivalence class E must be of size strictly greater than 2^m. We will use this E extensively in the remainder of the proof. First, we argue in the following simple lemma that E must contain tuples with certain properties.

Lemma 1. *Under the conditions of Theorem 3 the set E must contain tuples satisfying the following properties:*

1. *There exist distinct $M^{(0)}, \ldots, M^{(\ell)} \in \{0,1\}^m$ and some $\bar{y} \in \{0,1\}^w$ such that $(M^{(i)}, \bar{y})$ is in E for all $i = 0, \ldots, \ell$. In other words, E contains $\ell + 1$ tuples that all share the same $\bar{y}$.*
2. *There exists a tuple $(M', \bar{y}')$ in E such that $\bar{y}' \neq \bar{y}$.*

Proof. The first property is easy to see. Let us partition E into equivalence classes based on $\bar{y}$. Two tuples are in the same equivalence class if they have identical $\bar{y}$ components. If every class had only ℓ tuples in it then the size of E would be at most $2^w \ell$. But $2^w \ell \leq 2^m$ which contradicts the fact that $|E| > 2^m$. Hence, there must exist $\ell + 1$ tuples in E with the same $\bar{y}$.

Similarly, we prove the second property by a counting argument. Since $|E| > 2^m$ there must exist in E two tuples $T_1 = (M, \bar{y}_1)$ and $T_2 = (M, \bar{y}_2)$ where $\bar{y}_1 \neq \bar{y}_2$. Then either $\bar{y} \neq \bar{y}_1$ or $\bar{y} \neq \bar{y}_2$. Thus, either T_1 or T_2 is the required tuple. This completes the proof of the lemma. □

Note that M' may be contained in $\{M^{(0)}, \ldots, M^{(\ell)}\}$. This lemma relies on the fact that C outputs fewer than w bits. Part (2) of the lemma would not hold

otherwise. For example, part (2) does not hold for the concatenation construction.

Back to the proof of Theorem 3, we will use the tuples that are guaranteed to exist by the lemma to complete the proof of the theorem. We need a little more notation. For $i = 0, 1, \ldots, \ell$ and $\bar{y} = (y_1, \ldots, y_\ell)$ we let:

$$\begin{aligned} x_1^{(i)} &= X_1(M^{(i)}), \\ x_2^{(i)} &= X_2(M^{(i)},\ y_1), \\ &\vdots \\ x_\ell^{(i)} &= X_\ell(M^{(i)},\ y_1,\ \ldots,\ y_{\ell-1}) \end{aligned}$$

and write $\bar{x}^{(i)} = (x_1^{(i)}, \ldots, x_\ell^{(\ell)})$. These are the inputs to the hash functions corresponding to the tuple $(M^{(i)}, \bar{y}^{(i)})$. Similarly, we write $\bar{y}' = (y'_1, \ldots, y'_\ell)$, and let:

$$\begin{aligned} x'_1 &= X_1(M'), \\ x'_2 &= X_2(M',\ y'_1), \\ &\vdots \\ x'_\ell &= X_\ell(M',\ y'_1,\ \ldots,\ y'_{\ell-1}) \end{aligned}$$

Again, we write $\bar{x}' = (x'_1, \ldots, x'_\ell)$.

We now construct the functions $H_1, \ldots, H_\ell$ and the inputs M, M' needed to prove Theorem 3. For $i = 1, \ldots, \ell$ let H_i be some arbitrary function from $\{0,1\}^*$ to $\{0,1\}^{v_i}$. We show how to modify these functions in at most two points to obtain a collision on C without an internal collision on some H_j. The difficulty is in ensuring that our modifications to the H_i are legal. That is, we never map the same input to two different outputs, otherwise we will not be able to instantiate the H_i. We consider two cases.

Case 1: There exist $j \in \{1, \ldots, \ell\}$ and $a, b \in \{0, \ldots, \ell\}$ such that $x_j^{(a)} = x_j^{(b)}$ and $a \neq b$. In this case, for $i = 1, \ldots, \ell$ we define $\hat{H}_i$ as follows:

$$\hat{H}_i(x) = \begin{cases} y_i & \text{if } x = x_i^{(a)} \\ y_i & \text{if } x = x_i^{(b)} \\ H_i(x) & \text{otherwise} \end{cases}$$

Defining $\hat{H}_i(x)$ in this way is legal since we never map the same input to two different outputs.

Recall that both tuples $(M^{(a)}, \bar{y})$ and $(M^{(b)}, \bar{y})$ are in E. Moreover, we just forced each $\hat{H}_i$ to evaluate to y_i when C is given $M^{(a)}$ or $M^{(b)}$ as input. Therefore,

$$C^{\hat{H}_1, \ldots, \hat{H}_\ell}(M^{(a)}) = C^{\hat{H}_1, \ldots, \hat{H}_\ell}(M^{(b)})$$

Since $a \neq b$, we know by the lemma that $M^{(a)} \neq M^{(b)}$. But $x_j^{(a)} = x_j^{(b)}$ and hence there is no collision on $\hat{H}_j$. Thus, $M^{(a)}$ and $M^{(b)}$ cause a collision on C but not a collision on $\hat{H}_j$, as required.

Case 2: Suppose Case 1 did not happen. In this case we know that the following holds:

- Since $\bar{y} \neq \bar{y}'$, there exists an $r \in \{1, \ldots, \ell\}$ where: $y_r \neq y'_r$.
- Since Case 1 did not happen, then for all $j = 1, \ldots, \ell$ the set $\left\{x_j^{(i)}\right\}_{i=0}^{\ell}$ contains $\ell + 1$ distinct elements.

Observe that the second bullet implies that for any vector $\bar{x} = (x_1, \ldots, x_\ell)$ there exists some $k \in \{0, 1, \ldots, \ell\}$ such that $x_j \neq x_j^{(k)}$ for all $j = 1, \ldots, \ell$. The reason is that there are $\ell + 1$ possible values for k, but there are only ℓ coordinates in $\bar{x}$. Therefore, there must exist some $k \in \{0, 1, \ldots, \ell\}$ such that $\bar{x}$ differs from $\bar{x}^{(k)}$ in all the coordinates.

In particular, there exists $k \in \{0, 1, \ldots, \ell\}$ such that $\bar{x}'$ differs from $\bar{x}^{(k)}$ in all the coordinates. Then for $i = 1, \ldots, \ell$ we define $\hat{H}_i$ as follows:

$$\hat{H}_i(x) = \begin{cases} y'_i & \text{if } x = x'_i \\ y_i & \text{if } x = x_i^{(k)} \\ H_i(x) & \text{otherwise} \end{cases}$$

Since the vectors $\bar{x}'$ and $\bar{x}^{(k)}$ differ in all the coordinates, defining $\hat{H}_i(x)$ in this way is legal. We never map the same input to two different outputs.

Both tuples $(M^{(k)}, \bar{y})$ and $(M', \bar{y}')$ are in E and therefore

$$C^{\hat{H}_1, \ldots, \hat{H}_\ell}(M^{(k)}) = C^{\hat{H}_1, \ldots, \hat{H}_\ell}(M')$$

Furthermore, since $x_1^{(k)} \neq x'_1$ we know that $X_1(M^{(k)}) \neq X_1(M')$ and hence $M^{(k)} \neq M'$. In addition, since $y_r \neq y'_r$ there is no collision on $\hat{H}_r$. Thus, once again, M' and $M^{(k)}$ cause a collision on C but not a collision on $\hat{H}_r$, as required.

We see that in both cases we were able to produce the required functions $\hat{H}_1, \ldots, \hat{H}_\ell$ and pair of inputs M, M'. This completes the proof of Theorem 3. □

4 Discussion and Future Work

We discuss a few directions for future work in this area.

Stronger impossibility results. Our current impossibility results apply to any construction C that uses each of the underlying hash functions at most once. Is there a similar impossibility result for constructions that use each hash function multiple times? To extend our results all that is needed is a stronger existence argument, namely a version of Theorem 3 that applies to more general circuits. Ideally, Theorem 3 can be strengthened to handle adaptive constructions, namely constructions that evaluate the underlying hash functions a variable number times. For example, the number of times that C evaluates the function H_1 may depend on the length of M. Nevertheless, even an existence argument for circuits C that evaluate the hash functions a *constant* number of times would be progress. Until a complete impossibility argument is obtained, we cannot rule out the existence of space-efficient combiners.

Non-blackbox results. Our impossibility results build *generic* hash functions $H_1, \ldots, H_\ell$ that cause a construction C and its proof of security P to fail. In practice, however, hash functions typically follow the Merkle-Damgård paradigm. Thus, a natural question is whether one can give an impossibility proof where the counter-example hash functions $H_1, \ldots, H_\ell$ are Merkle-Damgård functions. Alternatively, is there a space efficient combiner that is proven secure only when instantiated with Merkle-Damgård functions? (even though the combiner is necessarily insecure when instantiated with general hash functions).

We note that a stronger impossibility proof, as mentioned in the first paragraph, will also provide an impossibility proof for efficiently combining Merkle-Damgård functions. Simply view the ℓ Merkle-Damgård chains as part of the circuit C and then apply the impossibility argument to the ℓ compression functions. A complete impossibility proof will provide ℓ compression functions for which the proof of security P fails. Then the derived hash functions $H_1, \ldots, H_\ell$ cause the proof P to fail and are Merkle-Damgård functions as required.

Impossibility results for other hashing concepts. It seems natural that similar ideas can be applied to other hash function concepts such as second pre-image resistance. Can one prove that concatenation is the best secure combiner for second pre-image resistant functions?

5 Conclusion

We studied the problem of combining multiple hash functions into a single function that is collision resistant whenever at least one of the original functions is. The hope was that, given a number of plausibly secure hash constructions (*e.g.*, SHA-512 and Whirlpool), one might be able to hedge one's bet and build a new function that is at least as secure as the strongest of them. The combination should be space efficient in that the final output should be smaller than the concatenation of all hashes.

We showed that no such efficient black-box combination is possible assuming each hash function is evaluated once. We leave for future work the question of

generalizing our proof to the case where the same hash can be evaluated more than once — or building a working construction that exploits this condition.

Acknowledgments

We thank Anupam Datta and the anonymous reviewers for their helpful comments.

References

1. Charles Asmuth and G. R. Blakely. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers and Mathematics with Applications*, 7:447–450, 1981.
2. Paulo S. L. M. Barreto and Vincent Rijmen. The Whirlpool hashing function. In *First open NESSIE Workshop*, Leuven, Belgium, November 2000.
3. John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly efficient blockcipher-based hash functions. In *Proceedings of Eurocrypt '05*, volume 3494 of *LNCS*. Springer-Verlag, 2005.
4. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Proceedings of Crypto '02*, LNCS. Springer-Verlag, 2002.
5. Jean-Sebastian Coron, Yevgeniy Dodis, Cecile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: how to construct a hash function. In *Proceedings of Crypto '05*, volume 3621 of *LNCS*. Springer-Verlag, 2005.
6. Ivan B. Damgård. A design principle for hash functions. In *Proceedings of Crypto '89*, LNCS. Springer-Verlag, 1989.
7. Yevgeniy Dodis and Jonathan Katz. Chosen ciphertext security of multiple encryption. In *Proceedings of TCC '05*, pages 188–209, 2005.
8. Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Proceedings of Eurocrypt '05*, pages 96–113, 2005.
9. Amir Herzberg. Tolerant combiners: Resilient cryptographic design. Cryptology ePrint Archive, Report 2002/135, 2002. `http://eprint.iacr.org/`.
10. Jonathan Hoch and Adi Shamir. Breaking the ICE - finding multicollisions in iterated concatenated and expanded (ICE) hash functions. In *Proceedings of FSE '06*, 2006.
11. Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In *Proceedings of TCC '05*, pages 264–282, 2005.
12. Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Proceedings of Crypto '04*, volume 3152 of *LNCS*, pages 306–316. Springer-Verlag, 2005.
13. John Kelsey. Can we get a reduction proof for truncated hashes? Crypto '05 rump session, 2005.
14. Stefan Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004. `http://eprint.iacr.org/`.
15. Ralph C. Merkle. One way hash functions and DES. In *Proceedings of Crypto '89*, LNCS. Springer-Verlag, 1989.

16. Mridul Nandi and Doug R. Stinson. Multicollision attacks on some generalized sequential hash functions. Cryptology ePrint Archive, Report 2006/055, 2006.
17. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of STOC'89*, pages 33–43, 1989.
18. NIST. Secure hash standard. Federal Information Processing Standards (FIPS) Publication 180-2, 2002.
19. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *Proceedings of Crypto '93*, LNCS. Springer-Verlag, 1993.
20. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In *Proceedings of Eurocrypt '05*, volume 3494 of *LNCS*, pages 1–18. Springer-Verlag, 2005.
21. Xiaoyun Wang, Andrew Yao, and Frances Yao. New collision search for SHA-1. Rump Session Crypto'04, 2004.
22. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *Proceedings of Crypto '05*, volume 3621 of *LNCS*. Springer-Verlag, 2005.
23. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Proceedings of Eurocrypt '05*, volume 3494 of *LNCS*, pages 19–35. Springer-Verlag, 2005.