Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?

Daniel R. Simon*

Abstract

We prove the existence of an oracle relative to which there exist several well-known cryptographic primitives, including one-way permutations, but *ex*cluding (for a suitably strong definition) collision-intractible hash functions. Thus any proof that such functions can be derived from these weaker primitives is necessarily non-relativizing; in particular, no provable construction of a collision-intractable hash function can exist based solely on a "black box" one-way permutation. This result can be viewed as a partial justification for the common practice of treating the collision-intractable hash function as a cryptographic primitive, rather than attempting to derive it from a weaker primitive (such as a one-way permutation).

Key words: Hash functions, oracle, cryptography, complexity theory

1 Introduction

Modern practical cryptography is built on a number of basic primitives, which include, in addition to asymmetric cryptosystems, both symmetric ciphers and cryptographic hash functions. The theory and practice of ciphers has been explored extensively; much experience has been accumulated regarding practical design principles for them, and the simple theoretical primitive of the one-way function has been shown to be a necessary and sufficient building block for constructing one. In contrast, cryptographic hash functions are relatively new and unexplored as both theoretical and practical objects. It is therefore natural to ask whether the body of knowledge available for the former can be applied to the latter.

Cryptographic hash functions are actually used in a number of ways (so many, in fact, that they are often modeled as random oracles, as in [BR93]). Their most important attributed property, however (analogous to "security" for ciphers), is *collision intractability*, roughly defined as the intractability of finding

*Microsoft Research, One Microsoft Way, Redmond WA 98052 USA, email dansimon@microsoft.com

a pair of inputs that produce the same output. Whether used in message authentication codes (MACs) ([Tsu92]), signature schemes ([DP80]) or compound asymmetric primitives ([BR94]), collision intractability is the most important property which they are relied upon to provide. Collision intractability can be defined in a number of ways; functions exhibiting the weakest variant can be constructed easily from any one-way permutation ([NY89]), and somewhat less easily from any one-way function (see [Rom90]). However, the uses of collision-intractable hash functions generally require them to meet a stronger definition. Such strongly collision-intractable hash functions are only known to be provably constructible from specific assumptions such as the hardness of particular number-theoretic problems, or from stronger general primitives such as claw-free permutation pairs ([Dam87], [Rus92]) or fixed functions whose behavior is indistinguishable from that of random functions ([AV96], [AHV98]). The constructions in common use ([Riv92], [NIST94]) are typically obtained by applying the Merkle-Damgård iteration "meta-method" ([Mer89], [Dam89]) to fixed-length, heuristically designed "compression functions" whose collision-intractability is itself not inferrable from any weaker assumptions.

In this paper, we offer an explanation for this state of affairs: we present an oracle relative to which there exists a one-way permutation, and hence (for instance) a universal one-way hash function (as defined in [NY89]), but relative to which there are no strongly collision-intractable hash functions (in the sense of [Dam87]). It follows that any provable construction of the latter from one of the former would necessarily not relativize. Moreover, as we will show, the existence of the oracle implies that no provable construction of a collision-intractable hash function can exist based on a "black box" one-way permutation, i.e., a one-way permutation treated as an oracle, with no assumption made about it beyond its "one-wayness". (Constructions based on stronger assumptions about the permutation, such as certain algebraic or statistical properties, are not thus ruled out.) This result can therefore be viewed as a partial justification for the common practice of treating the collision-intractable hash function as a cryptographic primitive, rather than as a compound derived from a weaker primitive (such as the one-way permutation or any of its equivalents).

2 Definitions

The weakest definition of collision intractability in common use is the one given for *universal one-way hash function families* in [NY89]: under this definition, no polynomial-time algorithm can find a collision with a random input, given a hash function selected randomly from a family. (In [NY89] it is shown that a family of hash functions consisting of a one-way permutation composed with a random 2-universal hash function meets this definition.) Alternatively, one could require the intractability of finding a collision with a random input given a fixed function, rather than one selected randomly from a family; we will call such a function a *universal one-way hash function.* A stronger definition appears in [Dam87] for *collision-intractable hash function families*: no polynomial-time

algorithm can find a pair of inputs which form a collision, given a hash function selected randomly from a family. Finally, the strongest definition is the asymptotic version of the one assumed for real-world cryptographic hash functions: a single function for which it is intractable to find a single collision pair. We will call such a function a *collision-intractable hash function.*

Definition 1 *A* hash function *is a uniform family $F = \{C_n\}$ of circuits of size polynomial in n, taking input of size n and producing output of size $m < n$. A* hash function family *is a hash function in which the input is divided into two pieces, x and k, such that n is polynomial in $|x|$ and $m < |x|$.*

Definition 2 *An* n-specific collision *for a hash function F and an n-bit input x (resp., for a hash function family F and an n-bit input (k, x)) is a value y such that $C_n(x) = C_n(y)$ (resp., a pair (k, y) such that $C_n(k, x) = C_n(k, y)$). An* n-existential collision *for a hash function F (resp., for a hash function family F and a partial input k) is a pair (x, y) such that $C_n(x) = C_n(y)$ (resp., such that $C_n(k, x) = C_n(k, y)$).*

Definition 3 *([NY89]) A hash function (resp., hash function family) F is a* universal one-way hash function *(resp.,* universal one-way hash function family*) if every probabilistic polynomial-time algorithm finds an n-specific collision for F and an input x (resp., (k, x), for any x with k) chosen uniformly from $\{0,1\}^n$ with probability $n^{-\omega(1)}$ (over the algorithm's probabilistic choices and the choice of input).*

Note that a universal one-way hash function family is itself automatically a universal one-way hash function. Also, a construction similar to the one in [NY89] can be used to construct the former from the latter. Hence, in the case of universal hash functions (as opposed to collision-intractable hash functions), the existence of a family and the existence of a single function are equivalent, in that either can be constructed from the other. The same is not known to hold for collision-intractable hash functions and function families.

Definition 4 *([Dam87]) A hash function (resp., hash function family) F is a* collision-intractable hash function *(resp., a* collision-intractable hash function family*) if every polynomial-time algorithm finds an n-existential collision for F (resp., for F and a partial input k chosen uniformly from $\{0,1\}^{|k|}$) with probability $n^{-\omega(1)}$ (over the algorithm's probabilistic choices and the choice of k).*

3 The Main Result

The intuition underlying the theorem and proof is quite simple: since our goal is to find an oracle relative to which no collision-intractable function exists, we define an oracle which, given any function description, returns a collision (two inputs with the same output). And since we also wish to show that one-way

permutations exist relative to this oracle, we also provide a random permutation oracle f. It remains only to show that the collision-finding oracle does not help to invert f.

It turns out that the wrong choice of collision-finding oracle actually makes f invertible. For example, an oracle which simply returns a random collision (i.e., a uniformly chosen entry in the list of colliding pairs) would allow f to be inverted. (The space of collisions can be manipulated by use of a cleverly chosen query circuit, so that a constant fraction of all collisions involve an exponentially small fraction of the outputs.) On the other hand, if a collision is chosen by selecting a uniformly chosen input followed by a uniformly chosen collision with it (including perhaps itself), then the individual colliding inputs are themselves uniformly distributed. Receiving such a pair of uniformly (though not independently) distributed inputs gives essentially no information about the inverse x of any particular permutation output y.

In order to prove that this information really is of negligible value, we must measure the information obtained from each such query about x in particular, as opposed to any other miscellaneous information we might obtain about f in the process. To do so, we imagine that f has been composed with a transposition δ that transposes $y = f(x)$ with a randomly chosen image value. We are then given the resulting permutation $\pi = f \circ \delta$, together with a sequence R of query results, and ask what distribution on such transpositions δ can be inferred from π and R. For example, if we know x exactly, then we know exactly which δ was chosen to produce π. Conversely, if we know nothing about x, then any δ is as likely to be correct as any other. We will show that even after polynomially many queries to the collision oracle, the inferred likelihood of the correct value of δ, given π and R, remains exponentially small except with exponentially small probability. Hence the probability that x can be guessed correctly almost always remains negligible.

Theorem 1 *There exists an oracle A relative to which there exist universal one-way hash functions and universal one-way hash function families, but no collision-intractable hash functions or collision-intractable hash function families.*

Proof The oracle A will "contain" a permutation f, and accept queries which consist of a circuit description; the circuit may contain special "f-gates" which denote a request to the oracle ("f-query") to compute f on the gate's input, as well as oracle gates ("A-gates") which denote submission of the gate's input as a normal query of A ("A-query"). Given such a circuit description, the oracle first verifies that the input length is greater than the output length. If so, it chooses a random input x to the circuit, selecting uniformly from the set of inputs of the correct length, and then selects a value x' chosen uniformly from the set of inputs (including x itself) for which the circuit produces the same output as x. The oracle returns the *collision* $(x, x', C(x))$, together with a *query list* consisting of the inputs and outputs of all f-queries made during the computation of the

circuit's outputs on inputs x and x'. Note that both x and x' are uniformly distributed over all possible inputs to C (although not independently).

A-queries, rather than being unit-cost, cost a number of computation steps equal to the cost of computing the circuit's output and verifying the collision–that is, twice the size of the input circuit (f-gates being treated as unit-cost gates). This cost prevents the oracle from being used to speed up ordinary computations, rather than to find collisions.

We will consider the permutation f, as well as the input values chosen by the oracle, to be chosen randomly. More precisely, we define for every n a family $\{A_n\}$ of oracles of this type "containing" a permutation $f : \{0,1\}^n \rightarrow \{0,1\}^n$, with each using a table of the necessary length to determine its choices for every possible query circuit of size up to $n^{\omega(1)}$, and prove that the theorem is true with probability $1 - n^{-\omega(1)}$over the choices of A_n taken uniformly from this family.

We model polynomial-time algorithms relative to this oracle by uniform polynomial-size circuit families that may contain both f-gates and A -gates. It is clear that relative to this oracle, there are no collision-intractable hash functions or function families, since n-existential collisions can be found (with constant probability) for any such function or family by constructing the circuit for the function and using it as the input for an A-gate. (Since the function is length-decreasing, at least half of its inputs must collide with some other input, and the oracle's response to the query will therefore provide a collision with constant probability.) We will show, moreover, that the permutation f, which can easily be computed in polynomial time by a simple circuit that includes a single A-gate, is, for any polynomial-time algorithm, invertible only with negligible probability (over the choices of A). Thus (by a simple counting argument) there exists an A for which f is a one-way permutation (from which a universal one-way hash function family can be constructed using the method of [NY89]).

The outline of this proof is as follows: Consider a polynomial-size oracle-querying circuit family C; it inverts f if it successfully outputs the pre-image of, say, 0^n under f with significant probability (over choices of A; since we are considering f as a random permutation, the choice of image to invert is arbitrary). It may make f-queries and A-queries in any order, including A-queries that themselves contain A-query gates (up to logarithmically many recursions, given the defined cost of queries). At the end of its computation it outputs a guess for $f^{-1}(0^n)$, and succeeds if its guess is correct with probability at least n^{-c}, for some constant c.

Now, the only source of information available to the circuit about $f^{-1}(0^n)$ comes from the results of A-queries. These results consist of collision/query list pairs. Both of these may give information about f that may be useful in guessing $f^{-1}(0^n)$. We measure the information obtained so far from these sources as follows: suppose that a permutation π is chosen by com-

posing f with a transposition δ that transposes 0^n with an image chosen uniformly at random (including possibly itself). Then for a sequence R of query results, the value $\Pr[\delta|\pi, R]$ is a measure of the information about $f^{-1}(0^n)$ obtained from R, given π (as distinct from all other information gleaned about f from R). In particular, an optimal circuit cannot guess $f^{-1}(0^n)$ with probability greater than $\Pr[\delta|\pi, R]$ if R is the complete sequence of query results it has obtained. Similarly, no individual f-query Q in the circuit can get a response of 0^n with probability greater than $\Pr[\delta|\pi, R_Q]$, where R_Q is the sequence of query results preceding Q in the computation of C, and hence available to C when determining its input to Q.

Note that if some f-query in C has the input $\pi^{-1}(0^n)$, then δ is obvious given π, and $\Pr[\delta|\pi, R] = 1$ if R contains such a query result. (We will call such an event a "π-hit".) However, the probability that such a query is made at some point is *always* at most $2^{-n}|C|$, since R never reveals any information about $\pi^{-1}(0^n)$ (after all, for each f, $\pi^{-1}(0^n)$ is equally likely to be any value). Hence the effect of this event on our probability calculations will be negligible.

Similarly, if some f-query in C actually has the result 0^n, then δ is again obvious given π, and $\Pr[\delta|\pi, R] = 1$ if R contains such a query result. (We will call such an event a "δ-hit", or more succinctly, a "hit".) However, the probability that this event occurs for a given f-query preceded by result sequence R is at most $\Pr[\delta|\pi, R]$. We will show that for any C, $\Pr[\delta|\pi, R]$ always remains exponentially small (except with exponentially small probability, over choices of A's query results table). In particular, we will show that for each query Q, if $\Pr[\delta|\pi, R_Q]$ is exponentially small, then (except with exponentially small probability) $\Pr[\delta|\pi, R'_Q]$ is also exponentially small, where R'_Q is R_Q with the result of Q appended. This proof has two steps; first, we show that if for a given f-gate Q $\Pr[\delta|\pi, R_Q]$ is exponentially small, then not only will Q be a δ-hit with only exponentially small probability, but as long as it is not a δ-hit, $\Pr[\delta|\pi, R'_Q]$ will also be exponentially small. Second, we use this fact to show that if a given A-query circuit Q (for which $\Pr[\delta|\pi, R_Q]$ is exponentially small) produces a δ-hit with only exponentially small probability (over choices of its randomly, uniformly chosen input), then $\Pr[\delta|\pi, R'_Q]$ will also be exponentially small. From these two facts it follows that every query gate will produce an exponentially small $\Pr[\delta|\pi, R'_Q]$ from an exponentially small $[\delta|\pi, R_Q]$. Hence $[\delta|\pi, R_Q]$, which begins at 2^{-n} before any queries have been submitted, will never become non-negligible, and the circuit will therefore be able to produce a correct guess for $f^{-1}(0^n)$ with only negligible (i.e., exponentially small) probability.

An important fact we use is that the conditional probabilities on the possible values of δ are unlikely to be highly "misleading"; that is, $\Pr[\delta|\pi, R]$ is not likely to be too much less than $\Pr[\delta'|\pi, R]$ for some incorrect value of δ'. $\Pr[\delta|\pi, R]$ therefore can serve as a (very) rough approximation of

the overall entropy of the conditional distribution on δ given π and R.

Definition 5 *For oracle-querying uniform circuit family $C = \{C_n\}$ relative to an oracle A defined as above, with q_n queries in C_n, a* query ordering *is a polynomial-time-computable mapping from pairs (n, i) to particular f- or A-queries Q_i in C_n, such that if $a < b \leq q_n$, then (n, a) maps to a query Q_a, which can be computed without first receiving the entire response to the image Q_b of (n, b). For an oracle A and query ordering Ω, a* k-response prefix *$R_{\leq k}$ is the sequence of A's responses $R_1, ..., R_k$ to the k-*query prefix *$Q = Q_1, ..., Q_k$.*

Note that in recursive queries (say, an A-query whose circuit itself contains A-queries), one or the other of the inputs x and x' in a "higher-level" A-query may be required to compute the query contained within it (i.e., construct its query circuit); however, both x and x' need not be known. Moreover, the single inputs to the higher-level queries are nothing more than uniformly distributed variables. Hence we can consider the response to such an A-query as being distributed as if these higher-level query inputs were simply distributed uniformly.

Definition 6 *For oracle A, a* k-event *(δ, π, k, R) consists of a choice of a random transposition δ (one of whose transposed elements is 0^n) that turns the permutation f contained in A into the permutation π, and a k-response prefix R.*

Definition 7 *For a circuit family C with query ordering Ω, a* (δ-)hit *is an f-query response whose value is 0^n, or an A-query response containing an f-query input-output pair the output of which is 0^n. Within k-event (δ, π, k, R), a* π-hit *is a query response that would have been a δ-hit had δ been the identity transposition.*

First we prove that $\Pr[\delta'|\pi, R]$ for some incorrect δ' is not likely to lead the querying circuit "down the garden path" by being too much larger than $\Pr[\delta|\pi, R]$ for the correct value of δ.

Lemma 2 *For any fixed δ, π and query prefix length k, let S be the set of all k-response prefixes R for which there exists a δ' such that $\Pr[\delta|\pi, R] < m \Pr[\delta'|\pi, R]$. Then $\Pr[R_{\leq k} \in S|\delta, \pi] \leq m$.*

Proof If $\Pr[\delta|\pi, R] < m \Pr[\delta'|\pi, R]$, then $\Pr[\delta, \pi, R] < m \Pr[\delta', \pi, R]$; hence $\Pr[R_{\leq k} \in S|\delta, \pi] = \sum_{R \in S} \Pr[R|\delta, \pi] = \sum_{R \in S} \Pr[\delta, \pi, R] / \Pr[\delta, \pi] \leq m \sum_{R \in S} \Pr[\delta', \pi, R] / \Pr[\delta, \pi] \leq m$ (since $\Pr[\delta, \pi] = \Pr[\delta', \pi]$ for all δ, δ', π).

Definition 8 *The k-event (δ, π, k, R) is a* c-garden path *if there exists a δ' such that $\Pr[\delta|\pi, R] < m \Pr[\delta'|\pi, R]$, where $m = 2^{-cn}$ $(0 < c < 1)$.*

Note that for any particular k, a garden path occurs with probability bounded above by $2^{-n/c}$. Next we show that individual f-queries almost never reveal more than a negligible amount of extra information about the correct value of δ.

Lemma 3 *If* $\Pr[\delta|\pi, R_{\leq k}] = h \leq 1/2$ *for the* k*-response prefix* $R_{\leq k}$, $(\delta, \pi, k, R_{\leq k})$ *is not a* c*-garden path, and the* $(k+1)$*th query* Q_{k+1} *is an* f*-query, then if the response* R_{k+1} *to* Q_{k+1} *is neither a* δ*-hit nor a* π*-hit (an event which occurs with probability at least* $1-h-2^n$*), then* $\Pr[\delta|\pi, R_{\leq k+1}]$ *for the* $(k+1)$*-response prefix* $R_{\leq k+1}$ *is at most* $h(1+2h/m)$, *where* $m = 2^{-cn}$.

Proof Assume that Q_{k+1} is not a δ-hit or a π-hit; then

$$\begin{aligned}\Pr[\delta|\pi, R_{\leq k+1}] &= \Pr[\delta|\pi, R_{\leq k}]\left(\frac{\Pr[R_{k+1}|\delta, \pi, R_{\leq k}]}{\Pr[R_{k+1}|\pi, R_{\leq k}]}\right) \\ &= \frac{\Pr[\delta|\pi, R_{\leq k}]\Pr[R_{k+1}|\delta, \pi, R_{\leq k}]}{\sum_{\{\delta'\}}\Pr[\delta'|\pi, R_{\leq k}]\Pr[R_{k+1}|\delta', \pi, R_{\leq k}]}\end{aligned}$$

But $\Pr[R_{k+1}|\delta', \pi, R_{\leq k}] = 0$ when $\delta' = \delta$, and 1 otherwise (since, given π, R_{k+1} is fixed for a particular Q_{k+1} unless R_{k+1} is altered by δ). Hence the above expression simplifies to $\Pr[\delta|\pi, R_{\leq k}](1/(1 - \Pr[\delta'|\pi, R_{\leq k}])) \leq h(1/(1-h/m) \leq h(1+2h/m)$.

For example, if $h = \Pr[\delta|\pi, R_{\leq k}]$ is 2^{-dn} $(0 < d < 1)$, and Q_{k+1} is an f-query, then Q_{k+1} is a hit with probability at most 2^{-dn}, and if it is not a hit, then $\Pr[\delta|\pi, R_{\leq k+1}]$ is at most $2^{-dn}(1 + 2^{1+cn-dn})$. Clearly, then, even after a polynomially large sequence of consecutive f-queries $\Pr[\delta|\pi, R]$ remains exponentially small as long as $d > c$ (except possibly in the case of a hit or a garden path, both of which occur with exponentially small probability).

Now consider the case where Q_{k+1} is an A-query. The query circuit may itself contain both A-queries and f-queries, whose results are hits with probabilities that depend on previously gathered query results (that is, on $R_{\leq k}$) and on the uniformly distributed inputs to Q_{k+1} and to the "higher level" query circuits of which Q_{k+1} is a part. The effect of the query's result on $\Pr[\delta|\pi, R]$ depends in this case not only on whether the result is a hit, but also on the probability that the particular collision (call it $(x, x', Q_{k+1}(x))$) would have been chosen given a different δ. Since x is chosen uniformly over possible inputs, its probability of being chosen is independent of δ. The probability that x' is chosen is simply the reciprocal of the size of the set $S_{x,\delta}$ of inputs that collide with x. But as long as x does not result in a hit, $|S_{x,\delta}|$ correlates significantly with δ only to the extent that elements of $S_{x,\delta}$, if chosen, would result in hits, since if y is in $S_{x,\delta}$ without resulting in a hit, then it would be out of the set only for at

most polynomially many possible δ values (those that would change the output of one of the polynomially many f-queries in the A-query with y as input).

We thus define a set S_x for each x whose elements would be in S_x for all but polynomially many values of δ. We will show that except with exponentially small probability (assuming exponentially small $\Pr[\delta|\pi, R_{\leq k}]$), $|S_{x,\delta}|$ does not vary much from $|S_x|$ as δ varies, and hence little information is gathered about δ, and $\Pr[\delta|\pi, R_{\leq k+1}]$ is thus increased by only an exponentially small amount.

Lemma 4 *Let $\Pr[\delta|\pi, R_{\leq k}] = h \leq 1/2$ for the k-response prefix $R_{\leq k}$, and the $(k+1)$th query Q_{k+1} be an A-query whose result includes collision $(x, x', Q_{k+1}(x))$. Let n^e be an upper bound on the total size of the circuit attempting to compute $f^{-1}(0^n)$; let $r \leq 1/2$. Then*

1. *the probability p that x or x' is a δ-hit or a π-hit is at most $2n^e(h + 2^{-n})$;*
2. *the probability that $|S_{x,\delta'}| > (1 + r_1)|S_x|$ for more than a fraction r_1 of all values of δ' is at most $n^e 2^{-n}/r_1^2$;*
3. *the probability that $|S_{x,\delta}| < (1 - r_2)|S_x|$ is at most p/r_2.*

Moreover, if none of the above events occur, then (except in the case of a garden path),

$$\Pr[\delta|\pi, R_{\leq k+1}] \leq \frac{h(1 + r_1)}{(1 - r_2)(1 - r_1 h 2^n/m)}.$$

Proof The probability of a hit follows from the fact that $\Pr[\delta|\pi, R_{\leq k}]$ only increases as k increases; hence each f-query in the A-query results in a hit with probability at most h (given either input x or x', since both are uniformly distributed). Note that if neither x nor x' result in hits, then both are in the set S_x. Moreover, if x and x' are drawn from a set of size t, then there are at most $n^c t$ additions to all the sets $S_{x,\delta'}$ (since each x value changes sets for at most n^c different values of δ'). Hence for a collection of sets $\{S_{x,\delta'}\}$ to have their size increased by a factor of $(1 + r_1)$ for a fraction r_1 of all 2^n δ' values, the total number of x values in the collection of sets must be at most $n^c t/r_1^2 2^n$, meaning that the probability that x is in this collection is at most $n^c 2^{-n}/r_1^2$. Also, for any particular S_x and δ, $|S_{x,\delta}| < (1 - r_2)|S_x|$ only if a fraction r_2 of the elements of S_x are not in $S_{x,\delta}$–that is, if they result in hits. Hence the probability that a fraction at least r_2 of elements of S_x are hits is at most p/r_2 (otherwise a fraction greater than p of all possible x values would be hits).

Now assume that none of the aforementioned events occur. Define δ' as *good* if $|S_{x,\delta'}| \leq (1+r_1)|S_x|$. Recall that

$$\begin{aligned}
\Pr[\delta|\pi, R_{\leq k+1}] &= \frac{\Pr[\delta|\pi, R_{\leq k}]\Pr[R_{k+1}|\delta, \pi, R_{\leq k}]}{\sum_{\{\delta'\}}\Pr[\delta'|\pi, R_{\leq k}]\Pr[R_{k+1}|\delta', \pi, R_{\leq k}]} \\
&= \frac{h/|S_{x,\delta}|}{\sum_{\{\delta'\}}\Pr[\delta'|\pi, R_{\leq k}]/|S_{x,\delta'}|} \\
&\leq \frac{h}{(1-r_2)\sum_{\{\delta'\}}\Pr[\delta'|\pi, R_{\leq k}]|S_x|/|S_{x,\delta'}|} \\
&\leq \frac{(1+r_1)h}{(1-r_2)\sum_{\{\delta' \text{ good}\}}\Pr[\delta'|\pi, R_{\leq k}]}.
\end{aligned}$$

But since we are assuming that at most a fraction r_1 of δ' values are not good, and each has probability at most h/m (assuming no garden path), we have $\sum_{\{\delta' \text{ good}\}}\Pr[\delta'|\pi, R_{\leq k}] \geqslant 1 - rh2^n/m$, as required.

Now, setting, say, $r_1 = 2^{-un}$ and $r_2 = 2^{-vn}$, and choosing moreover c to be an arbitrarily small positive constant and u and v to be $1/2 - c$, we observe that d (from the example following lemma 3) begins at 1 and increases by $o(1)$ over polynomially many queries except with exponentially small probability. Hence $\Pr[\delta'|\pi, R]$ never becomes significantly larger than 2^{-n}, and the circuit therefore has only an exponentially small chance of guessing δ correctly.

4 The Oracle Separation and "Black Box" Constructions

The oracle A presented above can be used to show that no construction of a collision-intractable hash function can exist which assumes only a generic one-way permutation, treating it as a "black box" (i.e., an oracle) for the purposes of the construction. Consider, for instance, an oracle F which, for a given size input of which the first half of the input bits are ones, outputs the result of A on the latter half of the input, and otherwise, computes the one-way permutation f described above (which remains a one-way permutation even in the presence of A). A simple permutation-preserving trick (mapping inputs of the form $(11...1x, x, ..., x)$, for suitably many repetitions of x, to $(11...1x, A(x))$, and vice versa, for every x) can be used to turn F into a permutation oracle Π; Π preserves F's "one-wayness" (as long as most inputs still result in a simple computation of f) as well as F's feature of offering callers complete access to A (using polynomially larger-sized inputs). It follows that any proof of a construction of a collision-intractable hash function from a one-way permutation must implicitly assume that the permutation oracle is not Π (which can be used to find collisions in any hash function). Hence the proof cannot apply to an absolutely arbitrary one-way permutation.

Note that we are modeling the one-way permutation primitive here as a single oracle answering arbitrary-length queries. It is common for "black box" constructions based on abstract primitives to represent the primitive as a family of oracles with fixed input and output lengths, rather than as a single oracle; this is normally reasonable because such constructions are typically relativizing, meaning that the constructions are no less provable in the presence of longer-length oracles in the same family. A black-box construction with a non-relativizing proof that did not permit the presence of longer-length oracles could, in principle, exist (although it is difficult even to imagine one); however, it would say nothing of practical significance, since any feasible instantiation of the one-way permutation would necessarily be implementable for any length which is polynomial in the original one. Hence the conclusions drawn here based on the model of the one-way permutation as a single oracle still apply to all practically relevant constructions.

5 Conclusions and Open Problems

The result presented here suggests that it is unlikely that a convincing construction of a collision-intractable hash function can be built on nothing more than the assumption of a one-way permutation. One way to address this difficulty is to look for other primitives (such as the claw-free permutation pairs of [Dam87], or one-way permutations with some extra property) from which collision-intractable hash functions can be built, and find plausible assumptions under which such primitives would exist. Another approach is to examine the uses of collision-intractable hash functions, and find ways to use weaker primitives to accomplish the task. It may even turn out, in the long run, that collision-intractability is never in fact necessary.

6 Acknowledgements

Many thanks to Josh Benaloh, Wei Dai, Amit Khetan, Terence Spies, Ramarathnam Venkatesan, and the Eurocrypt '98 Program Committee for their helpful comments and useful discussions.

References

[AHV98] W. Aiello, S. Haber and R. Venkatesan, "New Constructions for Secure Hash Functions", Proc. Fifth Workshop on Fast Software Encryption (FSE5), 1998.

[AV96] W. Aiello and R. Venkatesan, "Foiling Birthday Attacks in Length-Doubling Transformations", Proc. EUROCRYPT '96, 1996.

[BR93] M. Bellare and P. Rogaway, "Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols", Proc. 1st Annual Conference on Computer and Communications Security, 1993.

[BR94] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption", Proc. Eurocrypt '94, 1994.

[Dam87] I. Damgård, "Collision-Free Hash Functions and Public-Key Signature Schemes", Proc. EUROCRYPT '87, 1987.

[Dam89] I. Damgård, "A Design Principle for Hash Functions", Proc. CRYPTO '89, 1989.

[DP80] D. Davies and W. Price, "The Application of Digital Signatures Based on Public-Key Cryptosystems", Proc. 5th International Computer Communications Conference, 1980.

[Mer89] R. Merkle, "One Way Hash Functions and DES", Proc. CRYPTO '89, 1989.

[NIST94] National Institute of Standards and Technology, NIST FIPS PUB 186, "Digital Signature Standard", U.S. Department of Commerce, 1994.

[NY89] M. Naor and M. Yung, "Universal Hash Functions and their Cryptographic Applications", Proc. 21st Annual Symposium on Theory of Computing, 1989.

[Riv92] R. Rivest, "The MD5 Message Digest Algorithm", RFC 1321, 1992.

[Rom90] J. Rompel, "One-Way Functions Are Necessary and Sufficient for Digital Signatures", Proc. 22nd Annual Symposium on Theory of Computing, 1990.

[Rus92] A. Russell, "Necessary and Sufficient Conditions for Collision-Free Hashing", Proc. CRYPTO '92, 1992.

[Tsu92] G. Tsudik, "Message Authentication with One-Way Hash Functions", ACM Computer Communications Review v 22, no. 5, pp. 29–38, 1992.

[ZMI90] Y. Zheng, T. Matsumoto and H. Imai, "Structural Properties of One-Way Hash Functions", Proc. CRYPTO '90, 1990.