Building a Collision-Resistant Compression Function from Non-Compressing Primitives

Thomas Shrimpton[1] and Martijn Stam[2]

[1] University of Lugano and Portland State University thomas.shrimpton@unisi.ch
[2] EPFL martijn.stam@epfl.ch

Abstract. We consider how to build an efficient compression function from a small number of random, non-compressing primitives (fixed-key blockciphers were our original motivation). Our main goal is to achieve a level of collision resistance as close as possible to the optimal birthday bound. We present a $2n$-to-n bit compression function based on three independent n-to-n bit random functions, each called only once. We show that if the three random functions are treated as black boxes (i.e., modelled as random oracles), finding collisions requires $\Theta(2^{n/2}/n^c)$ queries for $c \approx 1$. We also give a heuristic, backed by experimental results, suggesting that the security loss is at most four bits for block sizes up to 256 bits.
We believe this is the best result to date on the matter of building a collision resistant compression function from non-compressing functions. It also relates to an open question from Black et al. (Eurocrypt'05), who showed that compression functions that invoke a single non-compressing random function cannot suffice.

Keywords. Hash Functions, Random Oracle Model, Compression Functions, Collision Resistance.

1 Introduction

Hash functions are a central cryptographic primitive, appearing in countless protocols and applications. Dedicated hash functions such as MD5 and SHA1 have dominated practice, as these are relatively fast and, until recently, they were believed to resist collision-finding attacks. But this belief has been shown to be unfounded: successful attacks have been published against most members of the MD/SHA-family. This has prompted a renewed interest in design methodologies for hash functions, with a particular emphasis on providing formal guarantees of collision resistance (along with other properties).

The design of hash functions usually proceeds in two stages. First one designs a compression function with fixed domain, typically bitstrings of some small length. One then applies a domain extension method, such as the Merkle-Damgård transform [9, 21], to the compression function in order to construct a hash function for messages of arbitrary length. The first part has our interest; in particular, the central problem considered by this paper is the following one:

> Given a (small) number of independent n-to-n bit random (one-way) functions, construct a $2n$-to-n bit compression function with provable collision resistance as close as possible to the optimal $2^{n/2}$ birthday bound.

We consider information-theoretic adversaries in order to make a strong statement about collision resistance. That is, we measure the adversary's complexity only by the number of queries it makes to its oracles for the non-compressing primitives. Of course, in practice, time complexity or even time-space complexity (the product of space and time) are arguably more relevant. But query complexity is the most conservative resource measure: it yields a lower bound on the other two measures (and hence the actual costs of mounting an attack). Moreover, finding *lower* bounds in the other two models, other than by query complexity, is notoriously hard.

An earlier version[30] of this work was presented at the Ecrypt Workshop on Hash Functions (2007).

The underlying n-to-n bit primitives are modelled as random oracles. In particular, we only make available an oracle for the forward function evaluation and not for its inverse. Contrast this with a random two-way permutation, such as a blockcipher with its key fixed, where oracles for both computing the function and its inverse are available. To emphasize the difference we will explicitly distinguish between random one-way function, one-way permutation or two-way permutation. Our random one-way functions are in fact the same as Maurer and Tessaro's public random functions [19]. We note that standard caveats apply when instantiation the random functions in practice due to our use of random oracles.

Ideally we would like to make do with just one random function, and to invoke it once for each n-bit block of message digested; such a compression function would be called rate-1. Unfortunately, Black et al. [4] have given a negative result that all but rules this out. In particular, if one uses the MD-transform over a $2n$-to-n-bit compression function that makes only one call to a random n-to-n-bit function, then there always exists an information-theoretic adversary that finds collisions efficiently (i.e., using just few calls to the non-compressing primitive).[3]

OUR CONTRIBUTION. We give a strong *positive* result in this direction: we present a compression function that calls random n-to-n bit functions f_1, f_2, f_3, and that has almost optimal collision resistance (here n is a parameter that can be chosen freely). The construction is as follows:

$$H^{f_1,f_2,f_3}(V,M) = f_3\left(f_1\left(M\right) \oplus f_2\left(V\right)\right) \oplus f_1\left(M\right) \ .$$

A picture of the compression function is given in Figure 1.

When we consider the construction for increasing n, we show that any adversary making $\Theta(2^{n/2}/n^c)$ total oracle queries, for $c > 1$, has a vanishing probability of finding a collision in H. On the other hand, for $c < 1$ we provide compelling arguments that an adversary exists that will find a collision with high probability. Thus it is fair to say that finding collisions takes around $\Theta(2^{n/2}/n)$ queries.

We remark that our compression function can easily be transformed into a hash function with arbitrary domain while preserving the collision resistance (e.g., using the Merkle-Damgård transform). Moreover, although our proven bound on collision resistance falls somewhat short of the optimal $\Theta(2^{n/2})$, we believe it is still sufficiently close to be useful in practice. Indeed, for n of cryptographic relevance we give estimates of the exact collision resistance: it turns out that, for n up to 256 bits, the loss is at most 4 bits of collision resistance.

As a note of warning, we do not claim any "beyond-birthday" properties one might hope for from a hash function, such as resistance against multi-collisions and optimal preimage resistance. Indeed, preimages can typically be found in $O(2^{2n/3})$ queries, rather than the desired $\Omega(2^n)$.

RELATED WORK. Bellare and Micciancio [1] introduce incremental hash functions, which in principle could be built upon a non-compressing primitive. Crucial differences with our work are that they build an entire hash function, not just a compression function, and that the collision resistance of their schemes is not based on query complexity (usually just n queries suffice for a collision), but on the presumed computational hardness of combining the query answers into an actual collision.

Bernstein [3] bases his Rumba20 compression function on these ideas. He xor's together the output of four pseudorandom generators, components of the Salsa20 streamcipher. By modelling the underlying primitives as (independent) random one-way functions, he shows an *upper* bound (and estimate) on the full complexity for collision finding of $O(2^{n/3})$, well below the birthday bound (and our *lower* bound).

[3] Black et al. [4] phrase their results in terms of two-way random permutations, but their result holds for the random one-way functions we consider as well.

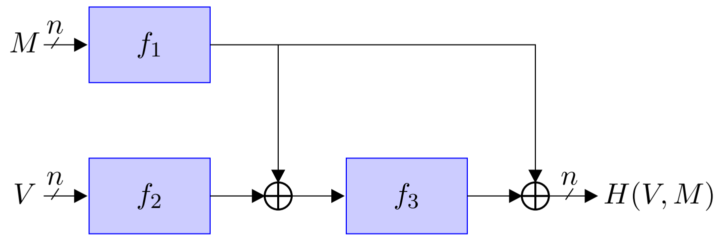

Fig. 1: The triple-function compression function. The functions f_1, f_2, f_3 are random n-to-n bit functions.

Note that the query complexity for finding collisions is $\Theta(2^{n/8})$ (cf. Wagner [32]). The expanding nature of Bernstein's primitives somewhat complicates theoretical efficiency comparisons, but the rate of his scheme is arguably 1/2, so more efficient than ours.

If one is willing to digest only *one bit* at a time (instead of n), then existing techniques already provide a way to construct an optimally collision-resistant hash function out of two random functions $f_0, f_1\colon \{0,1\}^n \to \{0,1\}^n$. Define a compression function $g\colon \{0,1\}^{n+1} \to \{0,1\}^n$ by $g(b \,||\, V) = f_b(V)$ that takes a single bit b of message to select which function will process the chaining value V. One can show that g is a completely random function iff both f_0 and f_1 were, and by using g in the block-chaining hash function from Damgård [9] one obtains a hash function that digests one bit per iteration.[4]

Maurer and Tessaro [19] consider the problem of constructing a function $C : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}$ given a n-to-n bit random one-way function f. They cast security in the indifferentiability framework and point out that the domain extension is actually far more challenging than a range extension. For all $\epsilon > 0$ they give a construction secure against adversaries making $\Theta(2^{n(1-\epsilon)})$ queries. Setting $m(n) = 2n$ and $l(n) = n$ gives rise to a $2n$-to-n bit compression function, setting $\epsilon = 1/2$ ensures optimal collision resistance. Thus from a security perspective this method beats ours. However, for these parameters, one seems to need 99 f-calls[5], meaning it has rate $1/99$. Our construction is considerably more efficient, but at the price of focussing primarily on collision resistance.

There has also been extensive research into the construction hash functions based on blockciphers. Davies-Meyer [20], Matyas-Meyer-Oseas [18], and Miyaguchi-Preneel [23, 28] are all well-known $2n$-to-n bit compression functions based on a single call to a blockcipher with n-bit key operating on n-bit blocks. These type of *rate-1* constructions were later systematically studied by Preneel et al. [28] and Black et al. [5], who identified twelve distinct constructions that provide optimal collision resistance when the blockcipher is modelled as an ideal cipher.[6]

Most of the work related to blockcipher-based compression functions allows per-round rekeying [5, 14, 25, 28]. As such, the primitive already compress in the sense that they take more input (key and plaintext data) than that they provide output. This significantly eases design and proof. Rekeying also has the drawback of entailing a significant computational cost.[7] Thus a lower rate fixed-key solution might actually be more efficient. Fixing the key would make use of the blockcipher in a more natural way, namely setting up

[4] Note that the same hash function emerges by using Damgård's construction based on a pair of claw free permutations[8], but using the functions f_0 and f_1 instead.

[5] If we charge a single call to a n-to-$12n$ $\tilde{f}$ for 12 calls to f using domain separation and subsequent concatenation.

[6] Black et al. [5] showed that an additional eight constructions do not yield collission resistant compression functions, yet still lead to collision resistant hash functions when properly MD-iterated.

[7] Gladman's implementation survey [13] shows that AES key scheduling would account for nearly 50% of the overall runtime.

a key once and then processing relatively large amounts of data with it (the way that blockciphers are used for encryption). A fixed-key blockcipher would be modelled as a random two-way permutation.

Black et al. [4] explicitly consider fixed-key blockciphers and show an impossiblity result for rate-1 schemes, but they do not consider compression functions that call out to more than one primitive per message block (and thus have a rate less than one).

Preneel et al. [27] propose a family of fixed-key constructions, but no formal security proof is given. The rate of their scheme is always strictly smaller then 1/2 and typically between 1/4 and 1/8.

The emphasis on blockcipher based hashing can be understood both historically and practically. Blockciphers have long been the central primitive in symmetric key cryptography, and there exists some measure of confidence in blockcipher designs. From a practical perspective, one might like to reuse optimized code or hardware implementations of blockciphers that are already implemented for encryption and message authentication within certain applications. That said, there seems to be no intrinsic theoretical reason to restrict designs to using blockciphers, hence our focus on simple random one-way functions (cf. [3, 19]).

The work on constructing double-length hash functions [14, 25] could be used as an alternative means to turn a number of non-compressing random functions into a compressing one, as explained in Appendix A. Although it is not the focus of this paper, one can conversely use our method to get a rate-1/3 $4n$-to-$2n$ bit double-length compression function with close to optimal collision resistance (in the output size) based on a set of $2n$-to-n bit random functions.

2 Preliminaries

GENERAL NOTATION. For a positive integer n, we write $\{0,1\}^n$ for the set of all bitstrings of length n. When X and Y are strings we write $X \,||\, Y$ to mean their concatenation and $X \oplus Y$ to mean their bitwise exclusive-or (xor). Unless specified otherwise, we will consider bitstrings as elements in the group $(\{0,1\}^n, \oplus)$.

For positive integers m and n, we let $\mathrm{Func}(m,n)$ denote the set of all functions mapping $\{0,1\}^m$ into $\{0,1\}^n$. We write $f \xleftarrow{\$} \mathrm{Func}(m,n)$ to denote random sampling from the set $\mathrm{Func}(m,n)$ and assignment to f. Unless otherwise specified, all finite sets are equipped with a uniform distribution.

DISTRIBUTIONS AND TENSORS. With $(\{0,1\}^n)^q$ we denote the set of q-element vectors, or *q-vectors*, in which each element is an n-bit string. When $\mathbf{a} \in (\{0,1\}^n)^q$ and $\mathbf{b} \in (\{0,1\}^n)^q$, we will write $\mathbf{a} = (a_1, \ldots, a_q)$ and $\mathbf{b} = (b_1, \ldots, b_q)$ when we wish to stress its components.

Fix a value q, and let $Q = q^2$. We define $\mathbf{a} \otimes \mathbf{b} \in (\{0,1\}^n)^Q$ as the tensor product under exclusive-or, where we identify $(\{0,1\}^n)^{q \times q}$ with $(\{0,1\}^n)^{q \cdot q} = (\{0,1\}^n)^Q$. More concretely $(\mathbf{a} \otimes \mathbf{b})_{i,j} = a_i \oplus b_j$ for i and j in $[1, \ldots, q]$. (Whenever possible we will try to use dummy i to refer to elements of $\mathbf{a}$ and dummy j to refer to those of $\mathbf{b}$.) If A and B are both distributions over $(\{0,1\}^n)^q$, this tensor operation induces a distribution over $(\{0,1\}^n)^Q$, which we will denote by the symbol $A \oplus B$. Unless otherwise specified, we will assume throughout that A and B are two distributions induced by sampling from $(\{0,1\}^n)^q$ *without* replacement. We will use U to denote the uniform distribution over $(\{0,1\}^n)^Q$ (where n and Q will often follow from the context). Thus U corresponds to sampling Q strings from $\{0,1\}^n$ uniformly and independently *with* replacement.

If in a random sample some value appears *exactly* k times, we say there is a *k-way collision* in that sample. Let $M_U(k)$ be the random variable describing the number of k-way collisions when the samples are drawn according to the distribution U. Similarly, let $M_{A \oplus B}(k)$ be the random variable describing the number of k-way collisions when the samples are drawn according to the distribution $A \oplus B$.

COMPRESSION FUNCTION SECURITY. When algorithms are provided with oracles, we write them as superscripts. A *collision-finding adversary* is an algorithm with access to one or more oracles, whose goal it is to find collisions in some specified hash function. All adversaries in this paper are computationally unbounded (information theoretic). Without loss of generality, we assume that adversaries do not repeat queries to oracles, and that they do not query an oracle outside of its specified domain.

A *compression function* is a mapping from $\{0,1\}^m \times \{0,1\}^\ell$ to $\{0,1\}^n$ for some $m, \ell, n > 0$ where $m + \ell \geq n$. For us, a compression function H must be given by a program that, given (V, M), computes $H^{\cdots}(V, M)$ via access to a finite number of specified oracles.

Definition 1 *Let $n > 0$ be an integer parameter, and fix an integer $k > 0$. Let $H\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a compression function taking k oracles. Let $\mathcal{A}$ be a collision-finding adversary for H that takes k oracles. The collision-finding advantage of $\mathcal{A}$ is defined to be*

$$\mathbf{Adv}^{\mathrm{coll}}_{H(n)}(\mathcal{A}) = \Pr\Big[f_1, \ldots, f_k \xleftarrow{\$} \mathrm{Func}(n,n), (V, M), (V', M') \leftarrow \mathcal{A}^{f_1(\cdot), \ldots, f_k(\cdot)} : \\ (V, M) \neq (V', M') \textit{ and } H^{f_1, \ldots f_k}(V, M) = H^{f_1, \ldots, f_k}(V', M')\Big]$$

We will require that the compression function H is in fact defined for arbitrary positive integers n, thus enabling asymptotic statements about collision resistance. In this case, we can meaningfully consider the non-uniform adversaries, regarding the functions f_1, f_2, and f_3 as length-preserving random oracles. The goal is then to limit the advantage against adversaries whose query-complexity falls within a certain class.

3 Some Background on Collision Probabilities

COLLISIONS IN UNIFORM SAMPLES. The susceptibility of hash functions to collisions has been heavily studied. The generic case can be termed in the language of occupancy urn models, a well-known tool from discrete probability theory. (Johnson and Kotz [15] and Feller [10] are the standard references, Girault et al. [12] and Preneel [26, Appendix B] are cryptographically oriented.) We give a brief overview of the main results that are relevant to our present cause.

One central concern for collision-finding attacks is to determine how many k-way collisions are expected for some fixed k, typically $k = 2$. If the hash function is modelled as a random oracle with range $\{0,1\}^n$, and Q domain points are hashed, then the number of k-way collisions is $M_U(k)$ with expected value $\mathbb{E}[M_U(k)] = N\binom{Q}{k}\left(\frac{1}{N}\right)^k\left(1 - \frac{1}{N}\right)^{Q-k}$, where $N = 2^n$. Thus, $\mathbb{E}[M_U(k)]$ follows a (scaled) binomial distribution with parameters $1/N$ and Q. If we let Q and N grow such that $Q/N < 1$, the binomial distribution can be approximated by a Poisson distribution with parameter $\lambda = \frac{Q}{N}$. This well known "urns and balls" result is captured in the following theorem, which also gives a more general result on the distribution of k-way collisions.

Theorem 2 *Let Q and N be positive integers, and let $\lambda = \frac{Q}{N}$. Then for $k \geq 2$ the random variable $M_U(k)$ follows asymptotically a Poisson distribution with parameter λ_k, where*

$$\Pr[M_U(k) = t] = e^{-\lambda_k}\frac{\lambda_k^t}{t!} \qquad \text{and} \qquad \lambda_k = Ne^{-\lambda}\frac{\lambda^k}{k!}$$

when Q, N tend to infinity such that $\lambda \to 0$.

The expected number of k-way collisions is therefore λ_k (since the parameter of a Poisson distribution equals its expected value). Furthermore, the λ_k themselves are also distributed according to a scaled Poisson distribution with parameter λ, as expected. The probability of finding any sort of collision is approximately $\frac{Q^2}{2N}$.

Sometimes one is more interested in the expected waiting time before the first k-way collision occurs. If the probability for finding a collision after q queries is at most ϵ, then the expected waiting time is at least $(1-\epsilon)q$. Thus upper bounding collision probabilities simultaneously lower bounds the expected waiting time, which is why we restrict our attention to the former.

In bounding the probability of finding a collision in our compression function, we will also need the distribution of k-way collisions when the samples from $\{0,1\}^n$ are distributed according to $A \oplus B$. It seems that occupancy is very similar to that of a fully uniform distribution, as formalized in Conjecture 3.

Conjecture 3 *For some positive integers q, n, let $Q = q^2$ and $N = 2^n$, and let $\lambda = Q/N$. Let q-vectors $\mathbf{a}$ and $\mathbf{b}$ have elements drawn according to A and B, respectively (i.e., uniformly from $\{0,1\}^n$ without replacement). Then for $k > 1$, $M_{A\oplus B}(k)$ follows asymptotically a Poisson distribution with parameter λ_k, where*

$$\Pr[M_{A\oplus B}(k) = t] = e^{-\lambda_k}\frac{\lambda_k^t}{t!} \qquad \text{and} \qquad \lambda_k = Ne^{-\lambda}\frac{\lambda^k}{k!} ,$$

when q, n tend to infinity such that $\lambda \to 0$.

Although we could not find a proof for the conjecture in full, proofs for the parts of it used in the sequel (to bound the collision finding probability either upwards or downwards) will be shown, so our results do *not* depend on the validity of the conjecture. In particular, we will upper bound $\Pr[M_{A\oplus B}(k) > 0]$ and—also serving as partial justification of the conjecture—we show in Appendix B that asymptotically $\mathbb{E}(M_{A\oplus B}(k)) = \lambda_k$.

4 A Construction Based on Three Length-Preserving Random Functions

Let $f_1, f_2, f_3 \in \mathrm{Func}(n,n)$ be three n-to-n bit functions, for some $n > 0$. We define the following $2n$-to-n bit compression function (again, see Figure 1):

$$H^{f_1,f_2,f_3}(V,M) = f_3\left(f_1\left(M\right) \oplus f_2\left(V\right)\right) \oplus f_1\left(M\right) .$$

We want to show that H is collision-resistant, so that it can be iterated to create a collision-resistant hash function [2, 9, 11, 21]. In the sequel, we will model f_1, f_2, and f_3 as three independent, uniform elements of $\mathrm{Func}(n,n)$. We will then bound the probability that a computationally unbounded adversary can find a collision as a function of the number of its oracle queries. We will also provide a matching lower bound by exhibiting an attack.

For a classical birthday attack, an attacker would need to evaluate the compression function H on roughly $2^{n/2}$ inputs to succeed. Clearly, one can obtain this many evaluations by querying each of the f_1, f_2, f_3 oracles on this many points. However, the structure of the compression function may make things easier for the adversary. In particular, asking q queries to each of the oracles can provide more than q evaluations of H, due to internal xor-collisions at the input to f_3. In the next section we will introduce the *yield* of a query set, and use it to measure the number of H evaluations an adversary can make given q queries to each of the oracles.

But before beginning any proofs, let us attempt to build some intuition about the compression function and the necessary requirements on f_1, f_2, f_3 when instantiated in practice.

PRACTICAL CONSIDERATIONS. Firstly, a collision in either f_1 or f_2 easily leads to a collision on the full compression function. A collision in f_3 does not appear to be useful, since to make a collision in H one needs to control the output of f_1 as well.

Secondly, if either f_1 or f_2 are invertible, an adversary can find collisions in H by making $O(2^{n/4})$ oracle queries. Say that f_2 is invertible. Then the adversary makes $2^{n/4}$ queries to both f_1 and f_3. With reasonable probability this will result in an internal xor-collision $f_1(M) \oplus f_3(Z) = f_1(M') \oplus f_3(Z')$. Inverting f_2 on $Z \oplus f_1(M)$, resp. $Z' \oplus f_1(M')$ will give a collision for H. Similarly if f_1 is invertible, call f_2 and f_3 each $2^{n/4}$ times to find an internal xor-collision $f_2(V) \oplus f_3(Z) \oplus Z = f_2(V') \oplus f_3(Z') \oplus Z'$. Now inverting f_1 on $Z \oplus f_2(V)$ and $Z' \oplus f_2(V')$ will complete the collision. (Clearly, if both f_1 and f_2 are invertible, only two calls to f_3 are needed to find a collision.) Invertibility of f_3 does not appear to be useful for attacking H.

Thus we will need (at least) for f_1 and f_2 to be collision-resistant and one-way. In particular, this rules out the straightforward blockcipher implementation $f_i(M) = E_{K_i}(M)$ for fixed (distinct) keys K_i, $i \in \{1, 2, 3\}$, as this violates the one-way requirement. One could consider instantiating the functions f_1, f_2 and f_3 with a simple blockcipher-based function, for example as $f_i(X) = E_{K_i}(X) \oplus X$, ($i \in \{1, 2, 3\}$, where K_i is a fixed and publicly known key). These are optimally inversion and collision-resistant in the ideal cipher model [5]. They are not, however, indifferentiable from a random oracle, even when the blockcipher E is modelled as an ideal cipher [7].[8] As such it would be interesting to see whether our construction can be proven secure (with similar collision resistance), in the ideal cipher model with the f_i replaced with a blockcipher-based instantiation. We note that there is no need to restrict oneself blockcipher-based implementations of functions f_1, f_2, f_3. Faster alternatives might be available by using for instance streamcipher-based components (cf. [3]).

VARIANTS. Several variants to this construction are possible. From the proof, it will be clear that our construction would work if the random functions were replaced by random one-way permutations as well. Equally clear is that V and M can be interchanged (or more generally any bijection can be applied to the inputs, and similarly any bijection can be applied to the output). More interesting is the case when two of the three functions are identical, in particular if $f_1 = f_2$. We have reasons to believe that this only results in a marginally decreased security (the proof carries through, apart from the bounding of a k-way collision).

We have already mentioned that a collision in f_1 or f_2 yields a collision in the full compression function. In fact, it is even worse, since a single colliding pair M, M' for f_1 can be used for any chaining value V. That is, if $f_1(M) = f_1(M')$, then for all V it holds that $H(V, M) = H(V, M')$. The precise ramifications of such an attack are unclear, although it for instance allows finding k-way collisions in a standard (strengthened) MD-iterate in query complexity $\Theta(2^{n/2})$, which is an improvement over Joux' [16] $\Theta(2^{n/2} \log k)$. We will not delve into this in great detail; actual attacks based on this property will also crucially rely on the iteration method used (and if multicollisions are an issue one should not rely on the standard MD-transform). We do note that one could change f_2's input to $V \oplus M$ (without losing any of the ordinary collision resistance). This way a single collision in f_1 still leads to many collisions in H, but in contrast with the previous version they will be of the form $H(V, M) = H(V', M')$ with $V \neq V'$. This might hinder chaining the collisions when iterated.

Efforts to design a provably collision-resistant $2n$-to-n bit compression function of rate-1/2 (i.e., using only two calls to length-preserving random functions) and simple non-cryptographic operations (like xor

[8] The recent results of Chang et al. [6] extend this to all of the 20 provably collision-resistant constructions from [5].

and addition modulo-2^n, cf. [28]) all failed. Specifically, none achieved collision resistance anywhere near the optimal $\Theta(2^{n/2})$; all fell to attacks requiring at most $O(2^{n/4})$ queries.[9] That said, our work does not rule out the possibility of such constructions. We leave open this question.

5 Proof of Collision-Resistance

In this section we will show two things. On the one hand, for any $c > 1$, any adversary making at most $O(2^n/n^c)$ queries has a vanishing advantage, that is, $\mathbf{Adv}^{\text{coll}}_{H(n)}(\mathcal{A}) = o(1)$. On the other hand, for any $c < 1$ we exhibit an adversary that asks $\Theta(2^n/n^c)$ queries and whose advantage can be bounded away from 0, that is $\mathbf{Adv}^{\text{coll}}_{H(n)}(\mathcal{A}) = \Omega(1)$. This provides a fairly complete asymptotic characterization of the newly proposed construction.

SETTING UP THE PROOF. We will distinguish between three ways for an adversary to find a collision in H. It can try to find a collision in f_1 or f_2, since either would lead to a collision in H, as already shown above. Failing that, it can try to find a collision in the final output. This leads to the following upper bound

$$\begin{aligned}\mathbf{Adv}^{\text{coll}}_{H(n)}(\mathcal{A}) \leq\, & \Pr[\mathcal{A} \text{ finds collision in } f_1] + \Pr[\mathcal{A} \text{ finds collision in } f_2] \\ & + \Pr[\mathcal{A} \text{ finds collision in } H | \text{no collisions in } f_1 \text{ or } f_2]\end{aligned}$$

and the corresponding lower bound

$$\max \begin{pmatrix} \Pr[\mathcal{A} \text{ finds collision in } f_1], \\ \Pr[\mathcal{A} \text{ finds collision in } f_2], \\ \Pr[\mathcal{A} \text{ finds collision in } H| \text{ no collisions in } f_1 \text{ or } f_2] \end{pmatrix} \leq \mathbf{Adv}^{\text{coll}}_{H(n)}(\mathcal{A}) \;.$$

The probabilities of finding a collision in f_1 or f_2 are ordinary collision-finding problems and hence well understood. For $q \leq \sqrt{N}$ these probabilities roughly sum up to $\frac{q^2}{N}$. In particular, for $q = O(2^{n/2}/n^c)$ with $c > 0$ the probability will tend to zero for random functions f_1 and f_2. Needless to say, if f_1 and f_2 are (random) permutations, no collisions exist and both probabilities are always zero.

In any case, we can concentrate on the probability of $\mathcal{A}$ finding a collision in H when f_1 and f_2 are collision free. First, some assumptions. We assume that the adversary makes exactly q queries to each of the three random oracles, f_1, f_2, f_3. We do so without any loss of generality, since given any adversary that makes q_i queries to f_i there is an adversary that makes $q = \max(q_1, q_2, q_3)$ queries to *each* of the oracles with identical success probability. Finally, still without loss of generality, we will assume that adversaries actually compute $H^{f_1,f_2,f_3}(V, M)$ and $H^{f_1,f_2,f_3}(V', M')$ before outputing their candidate collisions. (In particular, this means that all necessary queries to f_1, f_2 and f_3 are made before halting.)

REMOVING THE ADVERSARY. Normally we would imagine that the adversary makes queries to all three oracles in some adaptive, probabilistic manner. But here we cannot only argue away the adversary's adaptivity, but we can remove the adversary altogether. Recall that f_1, f_2, f_3 are independent random oracles, and let us focus for a moment on the adversary's queries to f_1, f_2.

Knowing that $\mathcal{A}$ makes q queries to each, we can imagine preparing the answers in advance. That is, before the adversary starts querying the oracles, we make two lists, each of q random elements, and when the adversary makes a query to one of the two oracles f_1 or f_2, we supply it with the next element of the respective list. Moreover, since the actual correspondence between query and response is irrelevant (we can

[9] Moreover, finding collisions in $f_1(M) \oplus f_2(V)$ requires $\Theta(2^{n/4})$ queries, cf. [32].

arbitrarily permute the unused elements of a list without altering the observed behaviour of the f_1 and f_2 oracles) we might as well have provided the two lists to the adversary *before* any queries to f_1 or f_2. (Of course, we charge the adversary for the $2q$ queries). In fact, since f_3 is independent of f_1 and f_2, we can give these lists to the adversary at the very beginning of the collision-finding game, in advance of any f_3 queries.

DEFINING THE YIELD. In the proof, we will find it useful to define a quantity called the *yield*. Formally, given a vector $\mathbf{c} = (c_1, \dots, c_Q) \in (\{0,1\}^n)^Q$, define

$$\mathrm{yield}(\mathbf{c}) = \max_{\substack{G \subseteq \{0,1\}^n \\ |G| = q}} \sum_{g \in G} \sum_{i=1}^{Q} [c_i = g]$$

where $[\text{true}] = 1$ and $[\text{false}] = 0$. Thus the yield counts the total number of occurences of the q most frequent elements in a vector. We also define the yield over the tensor of two q-vectors. Given vectors $\mathbf{a} = (a_1, \dots, a_q)$ and $\mathbf{b} = (b_1, \dots, b_q)$ in $(\{0,1\}^n)^q$, we will define the yield of tensor $\mathbf{a} \otimes \mathbf{b}$ to be

$$\mathrm{yield}(\mathbf{a} \otimes \mathbf{b}) = \max_{\substack{G \subseteq \{0,1\}^n \\ |G| = q}} \sum_{g \in G} \sum_{i=1}^{q} \sum_{j=1}^{q} [a_i \oplus b_j = g] \, .$$

Let us give some intution for this latter definition, in particular. Recall that we will give the response lists of f_1 and f_2, call these $\mathbf{a} = (a_1, \dots, a_q)$ and $\mathbf{b} = (b_1, \dots, b_q)$ (resp.), to the adversary prior to its making any f_3 queries. These q queries to f_3 can be made according to any strategy. One such strategy, already mentioned in the previous section, is to query the f_3 oracle on those values for which it knows the greatest total number of xor-preimages. In this case, the yield of $\mathbf{a} \otimes \mathbf{b}$ gives an upper bound on the number of compression function outputs that the adversary can evaluate by asking q queries to f_3.

CONNECTING THE PIECES. We will now show how the yield relates to the collision-finding probability of the adversary. Let d_r, for $r = 1, \dots, q$, denote the number of pairs (i, j) such that $a_i \oplus b_j$ equals the input of the r-th query to f_3. Suppose that after $r - 1$ queries to f_3, the adversary still has not found a collision. Then it will be able to output preimages for $\sum_{s=1}^{r-1} d_s$ hash values (or, equivalently, it will be able to output the hash value for that many preimages). With its rth call to f_3 it will be able to evaluate d_r new hash values, and the probability that one collides with one of the older values is therefore upper bounded[10] by $d_r \sum_{s=1}^{r-1} d_s / 2^n$. Summing over all queries to f_3 leads us to the following upper bound

$$\Pr[\mathcal{A} \text{ finds collision in } H | \text{ no collisions in } f_1 \text{ or } f_2] \leq \sum_{r=1}^{q} \sum_{s=1}^{r-1} d_r d_s / 2^n \, .$$

What can we say about this value? Firstly, the possible values of d_r are determined by $\mathbf{a}$ and $\mathbf{b}$ and the maximum $\sum_{s=1}^{r} d_s = \mathrm{yield}(\mathbf{a} \otimes \mathbf{b})$. Suppose we allow the adversary to partition $\mathrm{yield}(\mathbf{a} \otimes \mathbf{b})$ arbitrarily in q (real) parts d_r. The optimal way, in the sense of maximizing the sum above, is then to choose $d_r = \mathrm{yield}(\mathbf{a} \otimes \mathbf{b})/q$ for all $r = 1, \dots, q$ (optimality of this choice can be shown by induction). In that case we

[10] The upper bound is not always tight. Consequently, picking the elements corresponding to the maximal yield is sometimes not the optimal strategy for finding a collision; picking elements that are slightly less common might actually increase the chances of finding a collision, though the same upper bound will apply.

have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ finds collision in } H | \text{ no collisions in } f_1 \text{ or } f_2] &\leq \sum_{r=1}^{q} \sum_{s=1}^{r-1} d_r d_s / 2^n \\
&\leq \sum_{r=1}^{q} \sum_{s=1}^{r-1} (\text{yield}(\mathbf{a} \otimes \mathbf{b})/q)^2 / 2^n \\
&\approx \text{yield}(\mathbf{a} \otimes \mathbf{b})^2 / 2^{n+1} \; .
\end{aligned}$$

Our task is to put bounds on the expected value of $\text{yield}(\mathbf{a} \otimes \mathbf{b})$, or even better its square, where the elements in $\mathbf{a}$ and $\mathbf{b}$ are chosen independently, uniformly at random from $\{0, 1\}^n$ (without replacement).

UPPER BOUNDING THE YIELD. We now upper bound the yield, and hence the above collision-finding probability. We recall that the yield is the sum of the frequencies of the q most frequent elements in $\mathbf{a} \otimes \mathbf{b}$. As such, the trivial upper bound on the yield is the cardinality of $\mathbf{a} \otimes \mathbf{b}$, that is $Q = q^2$. Moreover, if all collisions in $\mathbf{a} \otimes \mathbf{b}$ are less than k-way, then the yield is (strictly) smaller than kq.

Let p be an upper bound on the probability that at least one collision that is at least k-way occurs in $\mathbf{a} \otimes \mathbf{b}$. Then conditioning on this event and employing the above observations yields that

$$\Pr[\mathcal{A} \text{ finds collision in } H | \text{ no collisions in } f_1 \text{ or } f_2] \leq (kq)^2 / 2^n + p \; .$$

Thus, if we show that for a particular choice of k and q we have that both terms in the sum are vanishing, we are done. It turns out that $k = n^d$ and $q = 2^{n/2}/n^c$ suffice for suitably chosen constants c, d; let's work it out.

Substitution in the first term, gives $n^{2(d-c)}$ which tends to zero iff $c > d$.

For the second term, we need to upper bound p, the probability of a k-way collision in $\mathbf{a} \otimes \mathbf{b}$ when $\mathbf{a}$ and $\mathbf{b}$ are drawn independent of each other and both uniformly at random from $\{0, 1\}^n$ without replacement. Now suppose that we have a k-way collision, then we can look at the positions in the matrix $\mathbf{a} \otimes \mathbf{b}$ that contribute to this collision. Suppose both (i, j) and (i', j') are involved, so $a_i \oplus b_j = a_{i'} \oplus b_{j'}$. We claim that if $i = i'$ (or $j = j'$), then also $(i, j) = (i', j')$. Clearly, if $i = i'$ then also $a_i = a_{i'}$ and, because both pair of indices xor to the same value, $b_j = b_{j'}$. But $\mathbf{b}$ does not contain any collisions, thus $j = j'$.

This allows us to bound the probability on a k-way collision. Firstly, we need to choose the k indices (i, j) that lead to the collision. For the indices i and j there are $\binom{q}{k}$ possibilities each, moreover given k indices i and k indices j, there are $k!$ ways of hooking them up. Given the locations of the collision, we have 2^n different values the collision can take. Then the values of a_i are still unrestricted, but the values of b_j are now determined. Without restrictions, there would have been $N!/(N - q)!$ ways to choose $\mathbf{b}$, but this now reduces to $(N - k)!/(N - q)!$. This leads us to:

$$p \leq \binom{q}{k}^2 k! 2^n \frac{(2^n - k)!}{(2^n)!} = \frac{(q!)^2 2^n (2^n - k)!}{((q - k)!)^2 k! (2^n)!} \; .$$

Now we need that this quantity tends to zero. Taking logarithms, using Stirling's approximation formula,[11] and filling in $q = 2^{n/2}/n^c$ and $k = n^d$ leads to

$$\begin{aligned}\ln p &\le (2q+1)\ln q - 2q + n\ln 2 + (2^n - k + \frac{1}{2})\ln(2^n - k) - (2^n - k)\\&\quad - (2(q-k)+1)\ln(q-k) + 2(q-k) - (2^n + \frac{1}{2})\ln 2^n + 2^n - k\ln k + k + o(1)\\&\le 2k\ln qk - k\ln k + n\ln 2 - kn\ln 2 + o(1)\\&\le 2k(\frac{n}{2}\ln 2 - c\ln n) - k\ln k + n\ln 2 - kn\ln 2 + o(1)\\&\le -(2c+d)n^d\ln n + n\ln 2 + o(1)\ .\end{aligned}$$

Hence for $d \ge 1$ this logarithm tends to minus infinity, and thus the probability of a k-way collision occuring, and with it the advantage of the adversary in finding a collision in H, tends to zero.

LOWER BOUNDING THE YIELD. For given n and q, let $\text{yield}_n^{A\oplus B}(q)$ be the expected value of $\text{yield}(\mathbf{a} \otimes \mathbf{b})$, for appropriately defined $\mathbf{a}$ and $\mathbf{b}$. Using the results from Appendix B we can lower bound the yield, specifically showing that for any $c < 1$, $q = \Omega(2^{n/2}/n^c)$ is sufficient for $\text{yield}_n^{A\oplus B} q$ to be $\Omega(2^{n/2})$. Thus, one can reasonably expect that this number of queries suffices to find collisions in H.

Let $c < 1$ be given. Pick $c < d < 1$ and set $q = 2^{n/2}/n^c$ and $k = n^d$. If the expected number of distinct k-way collisions is larger than q, we know that the yield is at least $kq = 2^{n/2}n^{d-c}$, which is $\Omega(2^{n/2})$.

By Conjecture 3, the expected number of k-way collisions is given by λ_k. We will show that λ_k/q tends to infinity for increasing n. Now, since

$$\lambda_k/q = \frac{2^{n/2}(\frac{1}{n^{2c}})^{n^d} n^c}{(n^d)!}$$

we can take logarithms, use Stirling's formula to approximate the factorial, and ignore smaller order terms, thus arriving at

$$\ln(\lambda_k/q) \approx \frac{\ln 2}{2} n - (2c+d)n^d \ln n\ .$$

Since $d < 1$ the term $\frac{\ln 2}{2}n$ will eventually dominate, showing that the expected number of k-way collisions asymptotically exceeds q as required.

A NOTE ON PREIMAGE RESISTANCE. Although our goal is to demonstrate a construction that yields a compression function with good collision-resistance, other useful properties should also be mentioned. Ideally, finding a preimage takes expected time 2^n for an n-bit primitive. To get an idea of the preimage resistance of the current proposal, we can look at the value of q for which the yield is around 2^n. If $q > 2^{n/2}$, a lower bound (and reasonable estimate) for the yield is $q^3/2^n$. Since $q^3/2^n \approx 2^n$ for $q \approx 2^{\frac{2}{3}n}$ it follows that our construction is not as preimage resistant as one might wish for.

6 Poisson Heuristic to Approximate the Yield

In this section, we will derive an alternative characterization of $\text{yield}(\mathbf{a} \otimes \mathbf{b})$. We then combine it with the conjectured distribution of k-way collisions in $A\oplus B$ in order to recast the problem of finding the expected

[11] For a positive integer k: $k! \approx \sqrt{2\pi k}(\frac{k}{e})^k$, where $k!$ is the usual factorial.

value $\mathrm{yield}_n^{A\oplus B}(q)$ into that of determining a certain property of the tail of a Poisson distribution. The latter problem can be tackled much more easily numerically for larger values of n. Indeed, we provide experimental results to both validate our reformulation, as well as determining concrete estimates of the collision resistance of our proposal in practice.

THEORETICAL BACKGROUND. Let $\mathbf{c} \in (\{0,1\}^n)^Q$ be given (we can ignore for the moment from which distribution $\mathbf{c}$ has arisen). Recall that $\mathrm{yield}(\mathbf{c})$ is the sum of the frequencies of the most frequent elements in $\mathbf{c}$. Thus, to determine $\mathrm{yield}(\mathbf{c})$, it suffices to know how many k-way collisions there are, for all k. Recalling that $M_{\mathbf{c}}(k)$ denotes the number of k-way collisions in $\mathbf{c}$, then we have:

$$\mathrm{yield}(\mathbf{c}) = \max_{\substack{\mathbf{w}\in[0,\ldots,Q]^{Q+1} \\ w_k \le M_{\mathbf{c}}(k), \sum_{k=0}^{Q} w_k = q}} \sum_{k=0}^{Q} k w_k \ .$$

To approximate $\mathrm{yield}_n^{A\oplus B}(q)$, we can look what happens if we look at $\mathrm{yield}(\mathbf{c})$ for an (imaginary) sample $\mathbf{c}$ whose number of k-way collisions exactly equals the expectation of $M_{A\oplus B}(k)$ (for all k).[12] We refer Appendix B for the expectation of $M_{A\oplus B}(k)$. (If $\mathbf{c}$ were drawn uniformly at random, rather than from $A\oplus B$, we could use Theorem 2, leading to the same approximation.) Let $\mathrm{yield}_n^P(q)$ be the yield based on the Poisson heuristic corresponding to a sample size of $Q = q^2$ elements of $\{0,1\}^n$. Then we pose the following approximation, where $\lambda = Q/N$:

$$\mathrm{yield}_n^{A\oplus B}(q) \approx \mathrm{yield}_n^P(q) = \max_{\substack{\mathbf{w}\in[0,\ldots,Q]^{Q+1} \\ w_k \le \frac{N\lambda^k}{k! e^{\lambda}}, \sum_{k=0}^{Q} w_k = q}} \sum_{k=0}^{Q} k w_k \ .$$

Note that we make two types or error in this approximation. The distribution of $\mathbb{E}(M_{A\oplus B}(k))$ is not exactly a scaled Poisson distribution; moreover, the maximum of an expected vector is not the same as the expected maximum of a vector (it is a lower bound though).

In Appendix C we provide experimental results, supporting our claim on the behaviour of expected number of k-way collisions for the distribution $A\oplus B$ and suggesting that $\mathrm{yield}_n^P(q)$ is a very good estimator for the actual $\mathrm{yield}_n^{A\oplus B}(q)$. Since $\mathrm{yield}_n^P(q)$ can be easily computed, this gives us a handle to estimate how secure our construction is for particular values of n (of cryptographic relevance).

ESTIMATED COLLISION-RESISTANCE. We are now ready to estimate the actual level of collision-resistance our construction offers. For each value of n, we have determined the smallest q for which the resulting average yield exceeds $2^{n/2}$. (Note that this is not exactly equivalent to having probability half of finding collisions, but we are confident it gives a faithful indication.)

In Table 1a we have tabulated $\log_2 q$ for small values of n. Included are three versions. First we give the value of q that, when simulating the experiment of picking A and B and computing $\mathrm{yield}_{A\oplus B}$, gives an average yield exceeding $2^{n/2}$. Second up is the corresponding result for the experiment of picking U directly uniformly at random. Finally we also give the value of $\log_2 q$ that follows from the Poisson estimation. For the latter we also provide the value of c such that $q = 2^{n/2}/n^{-c}$, that is $c = \log_n \frac{2^{n/2}}{q}$.

In Table 1b we have only given the values corresponding to the Poisson estimation, for n of cryptographic relevance. Note the very small loss of actual security. Even for a 512-bit primitive the loss is less than five bits. The table also shows the very slow increase in c.

[12] In fact such an ideal $\mathbf{c}$ might not exist, since the expected value of $M_{A\oplus B}(k)$ could be fractional.

As a consequence of $\log_2 q$ being fairly close to $\frac{n}{2}$, one might want to take the probability of finding a collision in either f_1 or f_2 into account as well. Assuming f_1 and f_2 are random functions, the probability of finding a collision in either is about 2^{-6} for $n = 160$, decreasing even further to 2^{-7} for $n = 256$, where q is chosen according to Table 1. (Again, if f_1 and f_2 are random permutations, this issue is moot.)

n	q $(\log_2 q)$			c
	$A \oplus B$	U	Poisson	
8	9 (3.17)	12 (3.58)	9 (3.16993)	0.28
10	16 (4.00)	17 (4.09)	16 (4.00000)	0.30
12	30 (4.91)	31 (4.95)	30 (4.90689)	0.30
14	56 (5.81)	60 (5.91)	56 (5.80735)	0.31
16	105 (6.71)	114 (6.83)	105 (6.71425)	0.32
18	195 (7.61)	210 (7.71)	195 (7.60733)	0.33
20	360 (8.49)	363 (8.50)	360 (8.49185)	0.35
22	676 (9.40)	676 (9.40)	676 (9.40088)	0.36
24	1338 (10.39)	1343 (10.39)	1338 (10.3859)	0.35

(a) Small n

n	$\log_2 q$	c
32	14.2148	0.36
64	29.6518	0.39
96	45.2324	0.42
128	60.9975	0.43
160	76.8034	0.44
192	92.5954	0.45
224	108.415	0.46
256	124.3	0.46
384	187.907	0.48
512	251.601	0.49

(b) Large n

Table 1: The relative bit-security provided against collision resistance based on n-bit primitives.

7 Conclusion

In this paper we have proposed a rate-1/3 $2n$-to-n bit compression function based on three random n-to-n bit functions. If the three underlying functions are modelled as random oracles, finding collisions requires roughly $2^{n/2}/n$ queries. Preimage resistance is loosely estimated to be around $2^{2n/3}$. Since the attacks based on optimizing the yield are inherently time and space consuming, it is unclear whether in practice algorithms can be found with a time complexity matching these query complexities (meaning our scheme will be harder to break).

8 Acknowledgement

We would like to thank David Wagner and Thomas Ristenpart for clarifying discussions.

References

1. Bellare, M., and Micciancio, D. A new paradigm for collision-free hashing: incrementality at reduced cost. In *Advances in Cryptology – EUROCRYPT'97* (1997), vol. 1233 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 163–192.
2. Bellare, M., and Ristenpart, T. Multi-property-preserving hash domain extension and the emd transform. In *Advances in Cryptology – ASIACRYPT'06* (2006), vol. 4284 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 299–314.
3. Bernstein, D. The Rumba20 compression function. http://cr.yp.to/rumba20.html, 2007.
4. Black, J., Cochran, M., and Shrimpton, T. On the impossibility of highly efficient blockcipher-based hash functions. In *Advances in Cryptology – EUROCRYPT '05* (2005), vol. 3494 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 526–541.

5. Black, J., Rogaway, P., and Shrimpton, T. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Advances in Cryptology – CRYPTO '02* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag.
6. Chang, D., Lee, S., Nandi, M., and Yung, M. Indifferentiable security analysis of popular hash functions with prefix-free padding. In *Advances in Cryptology – ASIACRYPT'06* (2006), vol. 4284 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 283–289.
7. Coron, J.-S., Dodis, Y., Malinaud, C., and Puniya, P. Merkle-damgard revisited: How to construct a hash function. In *Advances in Cryptology – CRYPTO '05* (2005), vol. 3621 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 430–448.
8. Damgård, I. Collision free hash functions and public key signature schemes. In *Advances in Cryptology – EUROCRYPT '87* (1988), D. Chaum and W. L. Price, Eds., vol. 304 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 203–216.
9. Damgård, I. A design principle for hash functions. In *Advances in Cryptology – CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer-Verlag.
10. Feller, W. *An Introduction to Probability Theory and its Applications*, vol. 1. John Wiley and Sons, Inc., 1968.
11. Gauravaram, P., Millan, W., Dawson, E., and Viswanathan, K. Constructing secure hash functions by enhancing merkle-damgrd construction. In *Information Security and Privacy* (2006), vol. 4058 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 407–420.
12. Girault, M., Cohen, R., and Campana, M. A generalized birthday attack. In *Advances in Cryptology – EUROCRYPT '88* (1988), C. G. Guenther, Ed., vol. 330 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 129–156.
13. Gladman, B. Implementation experience with aes candidate algorithms. In *Second AES Conference* (1999).
14. Hirose, S. Provably secure double-block-length hash functions in a black-box model. In *Information Security and Cryptology – ICISC '04* (2005), Lecture Notes in Computer Science, Springer-Verlag, pp. 330–342.
15. Johnson, N. L., and Kotz, S. *Urn Models and Their Applications*. John Wiley and Sons, Inc., 1977.
16. Joux, A. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in Cryptology – CRYPTO '04* (2004), M. K. Franklin, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 306–316.
17. Knudsen, L., and Muller, F. Some attacks against a double length hash proposal. In *Advances in Cryptology – ASIACRYPT'06* (2006), vol. 4284 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 462–473.
18. Matyas, S., Meyer, C., and Oseas, J. Generating strong one-way functions with cryptographic algorithms. *IBM Technical Disclosure Bulletin 27*, 10a (1985), 5658–5659.
19. Maurer, U., and Tessaro, S. Domain extension of public random functions: Beyond the birthday barrier. In *Advances in Cryptology – CRYPTO '07* (2007), Lecture Notes in Computer Science, Springer-Verlag, pp. 187–204.
20. Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, 1996.
21. Merkle, R. One way hash functions and DES. In *Advances in Cryptology – CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer-Verlag.
22. Mironov, I., and Narayanan, A. Domain extension for random oracles: Beyond the birthday-paradox bound. Tech. rep., ECRYPT Hash Workshop 2007, Proceedings, 2007.
23. Miyaguchi, S., Iwata, M., and Ohta, K. New 128-bit hash function. In *Proceedings 4th International Joint Workshop on Computer Communications* (1989), pp. 279–288.
24. Nandi, M., Lee, W., Sakurai, K., and Lee, S. Security analysis of a 2/3-rate double length compression function in black-box model. In *Fast Software Encryption – FSE'05* (2005), vol. 3557 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 243–254.
25. Peyrin, T., Gilbert, H., Muller, F., and Robshaw, M. Combining compression functions and block cipher-based hash functions. In *Advances in Cryptology – ASIACRYPT'06* (2006), vol. 4284 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 315–331.
26. Preneel, B. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
27. Preneel, B., Govaerts, R., and Vandewalle, J. On the power of memory in the design of collision resistant hash functions. In *Advances in Cryptology – Auscrypt '92* (1992), Lecture Notes in Computer Science, Springer-Verlag.
28. Preneel, B., Govaerts, R., and Vandewalle, J. Hash functions based on block ciphers: A synthetic approach. In *Advances in Cryptology – CRYPTO '93* (1994), Lecture Notes in Computer Science, Springer-Verlag, pp. 368–378.
29. Seurin, Y., and Peyrin, T. Security analysis of constructions combining FIL random oracles. In *Fast Software Encryption (FSE'07)* (2007), vol. 4593 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 119–136.
30. Shrimpton, T., and Stam, M. Efficient collision-resistant hashing from fixed-length random oracles. ECRYPT Hash Workshop 2007, May 24–25, Barcelona, 2007.
31. Steinberger, J. The collision intractability of MDC-2 in the ideal-cipher model. In *Advances in Cryptology – EUROCRYPT'07* (2007), vol. 4515 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 34–51.
32. Wagner, D. A generalized birthday problem. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., Lecture Notes in Computer Science, Springer-Verlag, pp. 288–303.

A The Dual Relationship with Double-Length Constructions (or Range Extenders)

In the main body we have considered the problem of extending the domain of a cryptographic primitive. One could consider *range extension* as the dual problem to domain extension. A relevant example instance of this problem is:

> Given a $2n$-to-n bit compression function create a $4n$-to-$2n$ bit compression function with collision resistance close to the optimal 2^n.

Perhaps a more common approach is to take as the starting primitive a blockcipher operating on n-bit blocks and having keysize (some multiple of) n; see, for example, Peyrin et al. [25]. (This has the practical advantage that one can use blockciphers of 128-bit blocks, which would not be advisable if a square root attack running in time 2^{64} were possible.)

When hash functions are expected to behave as random oracles, Maurer and Tessaro [19] make the point that domain extension is much harder than range extension. There is an easy counting argument underlying this claim (extending the domain increases the number of possible functions far more than a comparable extension of the range). However, in the context of collision-resistant hashing some other issues come to the fore. The difficulty in range extension is that the new construction should satisfy considerably stronger collision-resistance bound than the underlying primitive. Said another way, why bother with range extension if the longer hash value does not provide comensurately better security against generic (in casu birthday) attacks? As a result, satisfactory bounds on the collision resistance of double-length hash functions have been elusive. For example, Steinberger [31] showed that the collision-resistance of the standardized MDC-2 construction is at least $\Omega(2^{3n/5})$ in the ideal cipher model, when iterated; still considerably shy of the desired $\Theta(2^n)$.[13]

Somewhat paradoxically, range extenders can also be used to solve the problem of building a compressing function out of non-compressing primitives, and vice versa. As such, they can truly be regarded as dual problems. The resulting constructions are not necessarily very elegant, however, as our two transformations below will show. Nonetheless, our construction (see Figure 2) can be used to produce a rate-1/3 *double-length* compression function[14] with collision resistance around $2^n/2n$. This compares favourably with Nandi et al.'s construction of rate 1/3 and collision-resistance at most $\Theta(2^{2n/3})$ [17, 24] or Peyrin et al.'s construction of rate 1/5 and collision-resistance $\Theta(2^{2n/3})$ [25, 29].

Let $H^{\cdots} : \{0,1\}^{2n} \to \{0,1\}^n$ be some construction to build an optimally collision resistant compression function out of non-compressing primitives $f_1, \ldots, f_k : \{0,1\}^n \to \{0,1\}^n$, for arbitrary n. For example, our triple function construction is such an H for $k = 3$. Now, let $g_1, \ldots, g_{2k} : \{0,1\}^{2m} \to \{0,1\}^m$ be some given (single length) compression functions; we desire to use these $2k$ functions to build a double length compressing function $G^{\cdots} : \{0,1\}^{4m} \to \{0,1\}^{2m}$. We do so as follows. Set $n = 2m$ and define $f_i = g_i || g_{2k+1-i}$ for $i \in \{1, \ldots, k\}$. If we model the g_i as $2m$-to$m$ bit, independent random oracles then the f_i are $2m$-to-$2m$ bit (ie., non-compressing) random oracles. Now we apply H to these f_i to obtain a $4m$-to-$2m$ bit compression function, ultimately based on the g_i, whose collision resistance is 2^m, as desired.

Conversely (and for even k and n), let $G^{\cdots} : \{0,1\}^{4m} \to \{0,1\}^{2m}$ be an optimally collision resistant double length combiner for primitives $g_1, \ldots, g_k : \{0,1\}^{2m} \to \{0,1\}^m$. Let $f_1, \ldots, f_{k/2} : \{0,1\}^n \to \{0,1\}^n$ be given. Again, let $n = 2m$ and define the g_i such that $f_i = g_i || g_{k+1-i}$ for $i \in \{1, \ldots, k/2\}$. Run the range extender G with components g_i, yielding a hash function $G : \{0,1\}^{2n} \to \{0,1\}^n$ with collision

[13] Mironov and Narayanan [22] recently claimed that if the MMO constructions in MDC-2 are replaced with random $2n$-to-n bit compression functions, the double length construction does have collision resistance $\Theta(2^n)$.

[14] The rate is still 1/3, and not 1/6, since two blocks of n-bits (of message) are processed simultaneously.

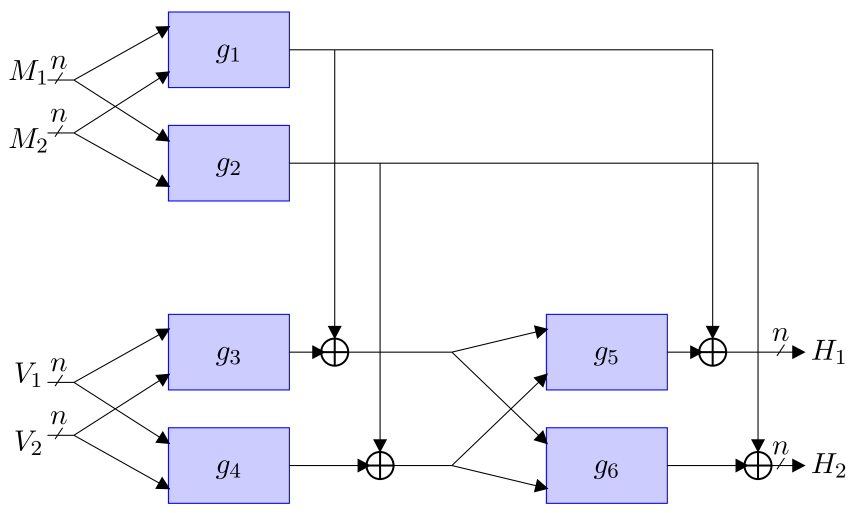

Fig. 2: The rate-1/3 double-length hash function. The functions $g_1, g_2, g_3, g_4, g_5, g_6$ are random $2n$-to-n bit functions.

resistance $2^{n/2}$. (Note that in this construction one should expect to throw half of f's output away all the time.)

B Partial Justification of Conjecture 3

Conjecture 3 states that the number of k-way collisions in the distribution $M_{A\oplus B}(k)$ asymptotically behaves as a Poisson distribution with parameter $\lambda_k = Ne^{-\lambda}\frac{\lambda^k}{k!}$. Here A and B are (independent) uniform distributions of q elements each over $\{0,1\}^n$ (without replacement), $Q = q^2, N = 2^n, \lambda = Q/N$ and the asymptotic statement assumes q and n tending to infinity such that $\lambda \to 0$. What we will show here is that asymptotically $\mathbb{E}(M_{A\oplus B}(k)) = \lambda_k$, thus the first moment of $M_{A\oplus B}(k)$ is in accordance with the stated Poisson distribution.

Proof: Define $F_{x,k}$ as the binary random variable taking on 1 iff the value $x \in \{0,1\}^n$ occurs exactly k times (when sampling from $A\oplus B$). This variable is related to the variable G_x denoting how often x occurs, indeed $\mathbb{E}(F_{x,k}) = \Pr(G_x = k)$. Let us try to determine $\Pr(G_x = k)$. This probability is the same for all x, so we can concentrate on $\Pr(G_0 = k)$. Since we also have that $M_{A\oplus B}(k) = \sum_x F_{x,k}$ and linearity of expectancy (even over dependent variables), we get

$$\mathbb{E}(M_{A\oplus B}(k)) = \sum_x \mathbb{E}(F_{x,k}) = N \Pr[G_0 = k] \ .$$

A collision occurs iff $a_i \oplus b_j = 0^n$, where a_i is an element in the vector $\mathbf{a}$ and b_j in the vector $\mathbf{b}$; equivalently, $a_i = b_j$. This problem has been studied, for example in the context of meet-in-the-middle attacks [12], and it turns out that the probability of a k-way collision follows a hypergeometric distribution, thus $\Pr(G_0 = k) = \binom{q}{k}\binom{N-q}{q-k}/\binom{N}{q}$. Asymptotically this means that $\mathbb{E}(M_{A\oplus B}(k))$ behaves as a scaled Poisson distribution with parameter Q/N, as claimed. *Q.E.D.*

C Experimental Justification of the Poisson Approximation

In this Appendix we provide experimental results to support our claim on the behaviour of expected number of k-way collisions for the distribution $A \oplus B$. It also suggests that $\text{yield}_n^P(q)$ is a very good estimator for the actual $\text{yield}_n^{A \oplus B}(q)$.

In order to get accurate estimates, we performed the experiment of picking from A and B a large number of times per pair (n, q). (Anywhere from 10^3 to 10^6 runs per pair.) Given that each run takes time 2^n with similar space requirements, we only have the data for n up to 22.

In Table 2 the data is given for query complexity $q = 2^{n/2}$. Since $N = 2^n$ and $Q = q^2$ this means that $\lambda = 1$. The first column gives the values of n, from which the other parameters (q, Q, and N) can be deduced. We then give three columns with the average yield (or rather the logarithms thereof). The first is the result of experiments based on the distribution $A \oplus B$, the second of experiments based on a uniform distribution U and the final is the Poisson estimation. The remaining five columns give the average numbers $M_{A \oplus B}(k)$, normalized by N. These are the numbers we conjecture are distributed according to a Poisson distribution with parameter Q/N. So, in the final row we give the Poisson distribution with $\lambda = 1$ for reference.

Table 3 gives the analogous results, but for $q = 2^{n/2-1}$, corresponding to $\lambda = \frac{1}{4}$.

n	$\log_2$ yield			Average nr. of k-way collisions / N				
	$A \oplus B$	U	Poisson	$M_{A \oplus B}(1)$	$M_{A \oplus B}(2)$	$M_{A \oplus B}(3)$	$M_{A \oplus B}(4)$	$M_{A \oplus B}(5)$
4	3.21	3.20	3.15	0.366	0.206	0.054	0.012	0.0016
6	4.47	4.49	4.45	0.367	0.189	0.060	0.014	0.0027
8	5.74	5.75	5.74	0.368	0.185	0.061	0.015	0.0030
10	6.90	6.90	6.90	0.368	0.184	0.061	0.015	0.0030
12	8.09	8.10	8.10	0.368	0.184	0.061	0.015	0.0031
14	9.19	9.19	9.19	0.368	0.184	0.061	0.015	0.0031
16	10.35096	10.35301	10.35384	0.3678	0.1839	0.0613	0.0153	0.00307
18	11.42019	11.42016	11.42020	0.3679	0.1839	0.0613	0.0153	0.00307
20	12.51232	12.51221	12.51233	0.3679	0.1839	0.0613	0.0153	0.00306
22	13.63089	13.63084	13.63085	0.3679	0.1839	0.0613	0.0153	0.00307
Poisson, $\lambda = 1$				0.3679	0.1839	0.0613	0.0153	0.00307

Table 2: Comparison of experimental yield and its Poisson estimate for $q = 2^{\frac{n}{2}}$ samples.

n	$\log_2$ yield			Average nr. of k-way collisions / N				
	$A \oplus B$	U	Poisson	$M_{A\oplus B}(1)$	$M_{A\oplus B}(2)$	$M_{A\oplus B}(3)$	$M_{A\oplus B}(4)$	$M_{A\oplus B}(5)$
6	2.52	2.53	2.54	0.787	0.0968	0.00543	0.000636	0.0000313
8	3.85	3.90	3.93	0.781	0.0975	0.00736	0.000517	0.0000302
10	5.10	5.10	5.10	0.780	0.0972	0.00791	0.000501	0.0000283
12	6.20	6.20	6.20	0.779	0.0974	0.00805	0.000505	0.0000261
14	7.37	7.37	7.37	0.779	0.0974	0.00810	0.000506	0.0000259
16	8.613	8.617	8.619	0.779	0.0974	0.00810	0.000507	0.0000254
18	9.653	9.652	9.652	0.779	0.0973	0.00811	0.000507	0.0000254
Poisson, $\lambda = \frac{1}{4}$				0.779	0.0974	0.00811	0.000507	0.0000254

Table 3: Comparison of experimental yield and its Poisson estimate for $q = 2^{\frac{n}{2}-1}$ samples.