Hash Function Balance and its Impact on Birthday Attacks

Mihir Bellare* Tadayoshi Kohno†

November 2002

Abstract

The standard estimate of $r^{1/2}$ trials to find a collision in a birthday attack on a hash function h with range size r is actually pessimistic, being correct only when h is *regular*, meaning all points in the range have the same number of pre-images under h. This paper provides a quantitative assessment of the extent to which the "amount of irregularity" in h improves the success-rate of the birthday attack. We show that the expected number of trials to find a collision is determined by a quantity we introduce called the *balance* of h, and can be significantly less than $r^{1/2}$ for functions of low balance. This leads us to examine popular design principles, such as the MD (Merkle-Damgård) transform, from the point of view of balance preservation, and to mount experiments to determine the balance of popular hash functions.

Keywords: Hash functions, birthday attacks, collision-resistance.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-mail: `tkohno@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/tkohno`

Contents

1 Introduction

Background on birthday attacks. Let $h\colon D \to R$ be a hash function. In a birthday attack, we pick points $x_1, \ldots, x_q$ from D and compute $y_i = h(x_i)$ for $i = 1, \ldots, q$. The attack is successful if there exists a collision, i.e. a pair i, j such that $x_i \neq x_j$ but $y_i = y_j$. We call q the number of *trials*.

There are several variants of this attack which differ in the way the points $x_1, \ldots, x_q$ are chosen (cf. [3, 7, 8]). The one we consider is that these points are chosen independently at random from D.[1] Stinson [7, Section 7.3] says that, due to the birthday phenomenon which gives the attack its name, a collision is expected within $r^{1/2}$ trials, where r denotes the size of the range of h. In particular, collisions in a hash function with output length m bits can be found in about $2^{m/2}$ trials.

However Stinson's analysis [7, Section 7.3], as well as all others that we have seen, assume the hash function is *regular*, meaning all points in the range have the same number of pre-images under h. These analyses note that if this is not the case then $r^{1/2}$ remains an upper bound on the expected number of trials, but there seems to be no quantitative information on how much faster the attack might find collisions as a function of the "amount of irregularity" in the hash function.

This paper. In this paper we provide precise quantitative information about the success-rate of the birthday attack as a function of the "amount of irregularity" in the hash function. To do this, we introduce a measure that we call the *balance* of a hash function. We show how the balance of h, a number between 0 and 1, characterizes the behavior of the birthday attack on h, by showing that the probability of finding a collision in q trials, as well as the number of trials at which the expected number of collision is one, are simple functions of the balance of the hash function being attacked. Then we pursue some other questions that arise as a consequence of these results.

The balance measure. View the range R of hash function $h\colon D \to R$ as consisting of $r \geq 2$ points $R_1, \ldots, R_r$. For $i = 1, \ldots, r$ we let $h^{-1}(R_i)$ be the pre-image of R_i under h, meaning the set of all $x \in D$ such that $h(x) = R_i$, and let $d_i = |h^{-1}(R_i)|$ be the size of the pre-image of R_i under h. We let $d = |D|$ be the size of the domain. We define the balance of h as

$$\mu(h) \;=\; \log_r \left[\frac{d^2}{d_1^2 + \cdots + d_r^2} \right] \,,$$

where $\log_r(\cdot)$ denotes the logarithm in base r. Proposition 3.2 shows that for any hash function h, the balance of h is a real number in the range from 0 to 1. Furthermore, the maximum balance of 1 is achieved when h is regular (meaning $d_i = d/r$ for all i) and the minimum balance of 0 is achieved when h is a constant function (meaning $d_i = d$ for some i and $d_j = 0$ for all $j \neq i$). Thus regular functions are well-balanced and constant functions are poorly balanced, but there are lots of possibilities in between these extremes.

Results. We are interested in the probability C of finding a collision in q trials of the birthday attack, and also in the *threshold* Q, defined as the number of trials required for the expected number of collisions to be one. (Alternatively, the expected number of trials to find a collision.)

[1] One might ask how to mount the attack (meaning how to pick random domain points) when the domain is a very large set as in the case of a hash function like SHA-1 whose domain is the set of all strings of length at most 2^{64}. We would simply let h be the restriction of SHA-1 to inputs of some reasonable length, like 161 bits or 320 bits. A collision for h is a collision for SHA-1, so it suffices to attack the restricted function.

Theorems 4.1 and Theorem 4.3, respectively, say that, up to constant factors,[2]

$$C = \binom{q}{2} \cdot \frac{1}{r^{\mu(h)}} \qquad \text{and} \qquad Q = r^{\mu(h)/2} . \tag{1}$$

These results indicate that the performance of the birthday attack can be characterized, quite simply and accurately, via the balance of the hash function h being attacked.

Note that when $\mu(h) = 1$ (meaning, h is regular) then Equation (1) says $Q = r^{1/2}$, which agrees with the above-discussed standard estimate for this case. At the other extreme, when $\mu(h) = 0$, meaning h is a constant function, the attack finds collisions in $O(1)$ trials so $Q = 1$. The value of the general results of Equation (1) is that they show the full spectrum in between the extremes of regular and constant functions. As the balance of the hash function drops, the threshold Q of the attack decreases, meaning collisions are found faster. For example the birthday attack on a hash function of balance $\mu(h) = 1/2$ will find a collision is about $Q = r^{1/4}$ trials, which is significantly less than $r^{1/2}$. Thus, we now have a way to quantitatively assess how irregularity in h impacts the success-rate of the birthday attack.

Good bounds rather than approximate equalities. It may not be hard to confirm, via some "back-of-the-envelope calculations," that C, Q have approximately the values claimed in Equation (1). However, we are claiming much more than "approximate equality." Theorem 4.1 provides both upper and lower bounds on C that are tight in the sense of being within a constant factor (specifically, a factor of four) of each other. (And Lemma 4.5 does even better, providing bounds within a factor two of each other, but the expressions are a little more complex.) Similarly, Theorem 4.3 provides upper and lower bounds on Q that are within a constant factor of each other. Deriving tight bounds, and in particular a good lower bound on C, required significantly more analytical work than merely producing a rough estimate of "approximate equality."

We claim good bounds are important. The estimates of how long the birthday attack takes to succeed, and the ensuing choices of output-lengths of hash functions, have been based so far on approximate equality calculations of the threshold that are usually upper bounds but not lower bounds on the exact value. Yet, the relevant parameter is actually a lower bound on the threshold since otherwise the attack might be doing better than we estimate. Thus one needs to make the analytical effort to derive good lower bounds.

Does the attacker need to know the balance? The attacker does not need to know the balance of the hash function in order to mount the attack. (The attack itself remains the birthday attack outlined above, where one simply picks random domain points and looks for a collision amongst them.) However, the balance might be revealed rather dramatically if the attack happens to terminate faster than expected.

Output lengths. Suppose we wish to design a hash function h for which the birthday attack threshold is 2^{80} trials. A consequence of our results above is that we must have $r^{\mu(h)/2} = 2^{80}$, meaning must choose the output-length of the hash function to be $160/\mu(h)$ bits. Thus to minimize output-length we must maximize balance, meaning we would usually want to design hash functions that are almost regular (balance close to one).

The general principle that hash functions should be as close to regular as possible is, we believe, well-known as a heuristic. Our results, however, provide a way of quantifying the loss in security as a function of deviations from regularity, and re-enforce this principle for practice.

Random hash functions. Designers of hash functions often have as target to make the hash

[2] This assumes $d \geq 2r$ and, in the case of C, that $8 \leq q \leq Q$. As Remark 4.2 argues, however, these conditions are not restrictive in practical settings.

function have "random" behavior. Proposition 4.4 together with Equation (1) enable us to estimate the impact of this design principle on birthday attacks. As an example, they imply that if h is a random hash function with $d = 2r$ then the expected probability of a collision in q trials is about 3/2 times what it would be for a regular function, while the expected threshold is about two-thirds what it would be for a regular function. (In particular, random functions are *worse* than regular functions from the point of view of protection against birthday attacks.) If one wants the best possible protection against both birthday and cryptanalytic attacks, one should design a function that is not entirely random but random subject to being regular.

Again, the qualitative fact that random functions are worse than regular functions with regard to protecting against the birthday attack is probably known to designers, but, as above, our results provide a quantitative reflection of this based on which we can measure how far randomness takes us from the ideal birthday behavior as a function of the domain and range sizes of the hash function.

Does the MD transform preserve balance? Given these results we would like to be building hash functions that have high balance. We look at some elements of current design to see how well they reflect this requirement.

Hash functions like MD5 [6], SHA-1 [5] and RIPEMD-160 [2] are designed by applying the Merkle-Damgård (MD) [4, 1] transform to an underlying compression function. Designers could certainly try to ensure that the compression function is regular or has high balance, but this turns out not to be enough to ensure high balance of the hash function because Proposition 5.1 shows that the MD transform does not preserve regularity or maintain balance. (We give an example of a compression function that has balance one, yet the hash function resulting from the MD transform applied to this compression function has balance zero.) Proposition 5.2 is more positive, showing that regularity not only of the compression function but also of certain associated functions does suffice to guarantee regularity of the hash function. But Proposition 5.3 notes that if the compression and associated functions have even minor deviations from regularity, meaning balance that is high but not equal to one, then the MD transform can amplify the imbalance and result in a hash function with very low balance. Given that a random compression function has balance close to but not equal to one, and we expect practical compression functions to be similar, our final conclusion is that we cannot recommend, as a general design principle, attempting to ensure high balance of a hash function by only establishing some properties of the compression function and hoping the MD transform does the rest.

We stress that none of this implies any weaknesses in specific existing hash functions such as those mentioned above. But it does indicate a weakness in the MD transform based design principle from the point of view of ensuring high balance, and means that if we want to ensure or verify high balance of a hash function we might be forced to analyze it directly rather than being able to concentrate on the possibly simpler task of analyzing the compression function. We turn next to some preliminary experimental work in this vein with SHA-1.

Experimenting with SHA-1. The hash function SHA-1 was designed with the goal that the birthday attack threshold is about 2^{80} trials. As per the above, this goal would only be met if the balance of the hash function was close to one. More precisely, for any $n < 2^{64}$ let $\mathrm{SHA}_n\colon \{0,1\}^n \to \{0,1\}^{160}$ denote the restriction of SHA-1 to inputs of length n. We would like to know whether SHA_n has balance close to one for practical values of n, since otherwise a birthday attack on SHA_n will find a collision for SHA-1 in less than 2^{80} trials. The balance of SHA_n is however hard to compute, and even to estimate experimentally, when n is large. Section 6 however reports on some experiments that compute $\mu(\mathrm{SHA}_n)$ for small values of n and find it to be extremely close to one. Towards estimating the balance of SHA_n for larger values of n, Section 6 reports on some experiments on $\mathrm{SHA}_{n;t_1\ldots t_2}$ for small values of n and $t_2 - t_1$, where $\mathrm{SHA}_{n;t_1\ldots t_2}\colon \{0,1\}^n \to \{0,1\}^{t_2-t_1+1}$ is the function which returns the t_1-th through t_2-th output bits of SHA_n. Broadly speaking, the experiments indicate that these functions have high balance. This can be taken as some indication that SHA_n also has high balance, meaning SHA-1 is well-designed from the balance point of view.

IS ANYTHING BROKEN? We stress that none of the results in this paper uncover any weaknesses, or demonstrate improved performance of attacks on any *specific, existing* hash functions such as those mentioned above. We view our results rather as providing analytical tools and refining design principles towards the goal of better understanding the security of existing hash functions or building new ones.

2 Notation and Terminology

If n is a non-negative integer then we let $[n] = \{1, \ldots, n\}$. If S is a set then $|S|$ denotes its size. We denote by $h\colon D \to R$ a hash function mapping domain D to range R, and throughout the paper we assume that R has size at least two. We usually denote $|D|$ by d and $|R|$ by r. A *collision* for h is a pair x_1, x_2 of points in D such that $x_1 \neq x_2$ but $h(x_1) = h(x_2)$. For any $y \in R$ we let

$$h^{-1}(y) \;=\; \{\, x \in D \,:\, h(x) = y \,\}\ .$$

We say that h is *regular* if $|h^{-1}(y)| = d/r$ for every $y \in R$, where $d = |D|$ and $r = |R|$.

3 The Balance Measure and its Properties

We introduce a measure that we call the *balance*, and establish some of its basic properties.

Definition 3.1 Let $h\colon D \to R$ be a hash function whose domain D and range $R = \{R_1, \ldots, R_r\}$ have sizes $d, r \geq 2$, respectively. For $i \in [r]$ let $d_i = |h^{-1}(R_i)|$ denote the size of the pre-image of R_i under h. The *balance* of h, denoted $\mu(h)$, is defined as

$$\mu(h) \;=\; \log_r \left[\frac{d^2}{d_1^2 + \cdots + d_r^2} \right]\ , \qquad (2)$$

where $\log_r(\cdot)$ denotes the logarithm in base r. ▮

It is easy to see that a regular function has balance 1 and a constant function has balance 0. The following says that these are the two extremes. In general, the balance is a real number that could fall somewhere in the range between 0 and 1.

Proposition 3.2 *Let h be a hash function. Then*

$$0 \;\leq\; \mu(h) \;\leq\; 1\ . \qquad (3)$$

Furthermore, $\mu(h) = 0$ iff h is a constant function, and $\mu(h) = 1$ iff h is a regular function. ▮

Proof of Proposition 3.2: Let $h\colon D \to R$ where $R = \{R_1, \ldots, R_r\}$. Let $d_i = |h^{-1}(R_i)|$ for $i \in [r]$. Let $d = |D|$ and let

$$S = \{\, (x_1, \ldots, x_r) \in \mathbb{R}^r \,:\, x_1 + \cdots + x_r = d \,\}\ .$$

Define the function $f\colon S \to \mathbb{R}$ by $f(x_1, \ldots, x_r) = x_1^2 + \cdots + x_r^2$ for any $x_1, \ldots, x_r \in S$ and let

$$\begin{aligned} \mathsf{Min}_S(f) &= \min\{\, f(x_1, \ldots, x_r) \,:\, (x_1, \ldots, x_r) \in S \,\} \\ \mathsf{Max}_S(f) &= \max\{\, f(x_1, \ldots, x_r) \,:\, (x_1, \ldots, x_r) \in S \,\}\ . \end{aligned}$$

We note that

$$\mathsf{Min}_S(f) \;\leq\; \frac{d^2}{r^{\mu(h)}} \;\leq\; \mathsf{Max}_S(f)\;.$$

The extremums of f over S are well studied, and it is known that f achieves its minimum on S when $d_i = d/r$ for all $i \in [r]$, which implies $\mathsf{Min}_S(f) = r(d/r)^2 = d^2/r$ and corresponds to h being regular, with all points in the range having pre-image size d/r. On the other hand f achieves its maximum when $x_i = d$ for some $i \in [r]$ and $x_j = 0$ for all $j \in [r] - \{i\}$, which implies $\mathsf{Max}_S(f) = d^2$ and corresponds to h being a constant function that maps all d points in the domain to some single point in the range. We thus get

$$\frac{d^2}{r} \;\leq\; \frac{d^2}{r^{\mu(h)}} \;\leq\; d^2\;.$$

Dividing by d^2 and re-arranging terms we get

$$1 \;\leq\; r^{\mu(h)} \;\leq\; r\;.$$

Taking logarithms to base r yields the Proposition. ▮

The following will be useful later.

Lemma 3.3 Let h: $D \to R$ be a hash function. Let $d = |D|$ and $r = |R|$ and assume $d \geq r \geq 2$. Then

$$r^{-\mu(h)} - \frac{1}{d} \;\geq\; \left(1 - \frac{r}{d}\right) \cdot r^{-\mu(h)}\;, \tag{4}$$

where $\mu(h)$ is the balance of h as per Definition 3.1. ▮

Proof of Lemma 3.3: Note that

$$r^{-\mu(h)} - \frac{1}{d} \;=\; \left(1 - \frac{r^{\mu(h)}}{d}\right) \cdot r^{-\mu(h)}\;.$$

Proposition 3.2 says that $\mu(h) \leq 1$, and this implies that $r^{\mu(h)} \leq r$. This in turn implies

$$1 - \frac{r^{\mu(h)}}{d} \;\geq\; 1 - \frac{r}{d}\;.$$

This concludes the proof. ▮

4 Balance-based Analysis of the Birthday attack

The attack is presented in Figure 1. (Note that it picks the points $x_1, \ldots, x_q$ independently at random, rather than picking them at random subject to be being distinct as in some variants of the attack [7]. The difference in performance is negligible as long as the domain is larger than the range.)

We are interested in two quantities: the probability C of finding a collision in a given number q of trials, and the *threshold* Q, defined as the number of trials at which the expected number of collisions is 1. Both will be estimated in terms of the balance of the hash function being attacked. Note that although Q is a simpler metric it is less informative than C since the latter shows how the success-rate of the attack grows with the number of trials. We begin with Theorem 4.1 below, which gives both upper and lower bounds on C that are within small constant factors of each other. The proof of Theorem 4.1 is in Section 4.1 below.

For $i = 1, \ldots, q$ do   // q is the number of trials

  Pick x_i at random from the domain of h

  $y_i \leftarrow h(x_i)$ // Hash x_i to get y_i

  If there exists $j < i$ such that $y_i = y_j$ but $x_i \neq x_j$ then return x_i, x_j EndIf   // collision found

  EndFor

  Return $\perp$ // No collision found

Figure 1: Birthday attack on a hash function $h\colon D \to R$. The attack is successful in finding a collision if it does not return $\perp$. We call q the number of *trials*.

Theorem 4.1 *Let $h\colon D \to R$ be a hash function. Let $d = |D|$ and $r = |R|$ and assume $d \geq 2r \geq 4$. Let C denote the probability of finding a collision for h in q trials of the birthday attack of Figure 1. Let $\mu(h)$ be the balance of h as per Definition 3.1. Then*

$$\frac{1}{4} \cdot \binom{q}{2} \cdot \frac{1}{r^{\mu(h)}} \;\leq\; C \;\leq\; \binom{q}{2} \cdot \frac{1}{r^{\mu(h)}}\,, \tag{5}$$

under the assumption that $8 \leq q \leq r^{\mu(h)/2}$. ▮

As we mentioned before, we believe it is important to have close upper and lower bounds rather than approximate equalities when it comes to computing the success rate of attacks, since we are making very specific choices of parameters, such as hash function output lengths, based on these estimates, and if our estimates of the success rates are not specific too we might choose parameters incorrectly.

Remark 4.2 Theorem 4.1 is only valid when $8 \leq q \leq r^{\mu(h)/2}$. In practice, this condition is hardly restrictive, since making $q \geq 8$ is not costly, and, on the other hand, the case of interest is q smaller than the expected number of trials to get a collision, which as per Theorem 4.3 is $r^{\mu(h)/2}$. For q higher than this, C is essentially 1. ▮

Next, we show that the threshold is $\Theta(r^{\mu(h)/2})$. Again, we provide explicit upper and lower bounds that are within a small constant factor of each other. The proof of Theorem 4.3 is in Appendix A.

Theorem 4.3 *Let $h\colon D \to R$ be a hash function. Let $d = |D|$ and $r = |R|$ and assume $d \geq 2r \geq 4$. Let Q denote the threshold, meaning the number of trials for which the expected number of collisions, in the birthday attack of Figure 1, is 1. Let $\mu(h)$ be the balance of h as per Definition 3.1. Then*

$$\sqrt{2} \cdot r^{\mu(h)/2} \;\leq\; Q \;\leq\; 1 + 2 \cdot r^{\mu(h)/2} \qquad ▮ \tag{6}$$

Designers of hash functions often have as target to make the hash function have "random" behavior. We now state a result which will enable us to gauge how well random functions fare against the birthday attack. (Consequences are discussed after the statement). Proposition 4.4 below says that if h is chosen at random then the expectation of $r^{-\mu(h)}$ is more than $1/r$ (what it would be for a regular function) by a factor equal to about $1+r/d$. The proof of Proposition 4.4 is in Appendix B.

Proposition 4.4 *Let D, R be sets of sizes d, r respectively, where $d \geq r \geq 2$. If we choose a function $h\colon D \to R$ at random then*

$$\mathbf{E}\left[r^{-\mu(h)}\right] \;=\; \frac{1}{r} \cdot \left(1 + \frac{r-1}{d}\right)\,. \qquad ▮$$

As an example, suppose $d = 2r$. Then the above implies that if h is chosen at random then

$$\mathbf{E}\left[r^{-\mu(h)}\right] \approx \frac{3}{2} \cdot \frac{1}{r} .$$

As per Theorem 4.1 and Theorem 4.3 this means that if h is chosen at random then the probability of finding a collision in q trials is expected to rise to about one-and-a-half times what it would be for a regular function, while the threshold is expected to fall to about two-thirds of what it would be for a regular function. Thus, from the point of view of protection against birthday attacks, it is better to have a function chosen at random subject to being regular than chosen entirely at random.

4.1 Proof of Theorem 4.1

We will establish a somewhat stronger result, namely:

Lemma 4.5 Let h: $D \to R$ be a hash function. Let $d = |D|$ and $r = |R|$ and assume $d > r \geq 2$. Let C denote the probability of finding a collision for h in $q \geq 2$ trials of the birthday attack of Figure 1. Let $\mu(h)$ be the balance of h as per Definition 3.1. Assume

$$q \;\leq\; \min\left(r^{\mu(h)/2} \,,\; (1/4) \cdot (1 - r/d) \cdot r^{\mu(h)} \right) . \tag{7}$$

Then

$$\frac{1}{2} \cdot \binom{q}{2} \cdot \left[\frac{1}{r^{\mu(h)}} - \frac{1}{d}\right] \;\leq\; C \;\leq\; \binom{q}{2} \cdot \left[\frac{1}{r^{\mu(h)}} - \frac{1}{d}\right] . \quad \blacksquare \tag{8}$$

We will first use Lemma 4.5 to prove Theorem 4.1 and then prove Lemma 4.5.

Proof of Theorem 4.1: To apply Lemma 4.5, we first verify that under the conditions of Theorem 4.1, Equation (7) is true. The assumption $d \geq 2r$, made in the statement of Theorem 4.1, implies

$$1 - \frac{r}{d} \;\geq\; \frac{1}{2} . \tag{9}$$

This implies

$$\frac{1}{4} \cdot \left(1 - \frac{r}{d}\right) \cdot r^{\mu(h)} \;=\; r^{\mu(h)/2} \cdot \frac{1}{4} \cdot \left(1 - \frac{r}{d}\right) \cdot r^{\mu(h)/2} \;\geq\; r^{\mu(h)/2} \cdot \frac{1}{8} \cdot r^{\mu(h)/2} .$$

The assumption $8 \leq q \leq r^{\mu(h)/2}$ made in the statement of Theorem 4.1 implies $8 \leq r^{\mu(h)/2}$ and thus $(1/8) \cdot r^{\mu(h)/2} \geq 1$. So from the above we get

$$\frac{1}{4} \cdot \left(1 - \frac{r}{d}\right) \cdot r^{\mu(h)} \geq r^{\mu(h)/2} .$$

The assumption $q \leq r^{\mu(h)/2}$ made in the statement of Theorem 4.1 thus suffices to yield Equation (7), and we can apply Lemma 4.5. The upper bound on C claimed in Theorem 4.1 follows directly from the upper bound on C in Lemma 4.5. On the other hand, applying first Lemma 3.3 and then Equation (9) we have

$$\frac{1}{2} \cdot \binom{q}{2} \cdot \left[\frac{1}{r^{\mu(h)}} - \frac{1}{d}\right] \;\geq\; \frac{1}{2} \cdot \binom{q}{2} \cdot \left(1 - \frac{r}{d}\right) \cdot \frac{1}{r^{\mu(h)}} \;\geq\; \frac{1}{2} \cdot \binom{q}{2} \cdot \frac{1}{2} \cdot \frac{1}{r^{\mu(h)}} .$$

Thus the lower bound on C in Lemma 4.5 implies the lower bound on C claimed in Theorem 4.1. $\blacksquare$

Proof of Lemma 4.5: We let $[q]_2$ denote the set of all two-element subsets of $[d]$. Recall that the attack picks $x_1, \ldots, x_q$ at random from the domain D of the hash function. We associated to any two-element set $I = \{i, j\} \in [q]_2$ the random variable X_I which takes value 1 if x_i, x_j form a collision (meaning $x_i \neq x_j$ and $h(x_i) = h(x_j)$), and 0 otherwise. We let

$$X = \sum_{I \in [q]_2} X_I .$$

The random variable X is the number of collisions. (We clarify that in this manner of counting the number of collisions, if n distinct points have the same hash value, they contribute $n(n-1)/2$ towards the value of X.) For any $I \in [q]_2$ we have

$$\mathbf{E}[X_I] = \Pr[X_I = 1] = \sum_{i=1}^{r} \frac{d_i(d_i - 1)}{d^2} = \sum_{i=1}^{r} \frac{d_i^2}{d^2} - \sum_{i=1}^{r} \frac{d_i}{d^2} = r^{-\mu(h)} - \frac{1}{d} . \tag{10}$$

By linearity of expectation we have

$$\mathbf{E}[X] = \sum_{I \in [q]_2} \mathbf{E}[X_I] = \binom{q}{2} \cdot \left[r^{-\mu(h)} - \frac{1}{d} \right] . \tag{11}$$

Let

$$p = r^{-\mu(h)} - \frac{1}{d} .$$

The upper bound of Lemma 4.5 is a simple application of Markov's inequality and Equation (11):

$$\Pr[C] = \Pr[X \geq 1] \leq \frac{\mathbf{E}[X]}{1} = \binom{q}{2} \cdot p .$$

We proceed to the lower bound. Let $[q]_{2,2}$ denote the set of all two-elements subsets of $[q]_2$. Via the inclusion-exclusion principle we have

$$\Pr[C] = \Pr\left[\bigvee_{I \in [q]_2} X_I = 1 \right] \geq \sum_{I \in [q]_2} \Pr[X_I = 1] - \sum_{\{I,J\} \in [q]_{2,2}} \Pr[X_I = 1 \wedge X_J = 1] . \tag{12}$$

Equation (11) tells us that the first sum above is

$$\sum_{I \in [q]_2} \Pr[X_I = 1] = \sum_{I \in [q]_2} \mathbf{E}[X_I] = \mathbf{E}[X] = \binom{q}{2} \cdot p . \tag{13}$$

We now claim that

$$\sum_{\{I,J\} \in [q]_{2,2}} \Pr[X_I = 1 \wedge X_J = 1] \leq \frac{1}{2} \binom{q}{2} \cdot p . \tag{14}$$

This completes the proof because from Equations (12), (13) and (14) we obtain Equation (8) as follows:

$$\begin{aligned} \Pr[C] &\geq \binom{q}{2} \cdot p - \sum_{\{I,J\} \in [q]_{2,2}} \Pr[X_I = 1 \wedge X_J = 1] \\ &\geq \binom{q}{2} \cdot p - \frac{1}{2} \binom{q}{2} \cdot p \\ &= \frac{1}{2} \binom{q}{2} \cdot p . \end{aligned}$$

It remains to prove Equation (14).

Let E be the set of all $\{I,J\} \in [q]_{2,2}$ such that $I \cap J = \emptyset$, and let N be the set of all $\{I,J\} \in [q]_{2,2}$ such that $I \cap J \neq \emptyset$. Then

$$\begin{aligned} & \sum_{\{I,J\}\in[q]_{2,2}} \Pr[\, X_I = 1 \wedge X_J = 1 \,] \\ = & \underbrace{\sum_{\{I,J\}\in E} \Pr[\, X_I = 1 \wedge X_J = 1 \,]}_{S_E} + \underbrace{\sum_{\{I,J\}\in N} \Pr[\, X_I = 1 \wedge X_J = 1 \,]}_{S_N} . \end{aligned} \tag{15}$$

We now claim that

$$S_E \;\leq\; \frac{1}{4}\binom{q}{2} \cdot p \tag{16}$$

$$S_N \;\leq\; \frac{1}{4}\binom{q}{2} \cdot p . \tag{17}$$

Equation (14) follows from Equations (15), (16) and (17), so it remains to prove the last two inequalities.

To upper bound S_E, we note that if $\{I,J\} \in E$ then the random variables X_I and X_J are independent. Using Equation (10) we get

$$S_E \;=\; \sum_{\{I,J\}\in E} \Pr[\, X_I = 1 \wedge X_J = 1 \,] \;=\; \sum_{\{I,J\}\in E} \Pr[\, X_I = 1 \,] \cdot \Pr[\, X_J = 1 \,] \;=\; |E| \cdot p^2 .$$

Computing the size of the set E and simplifying, we get

$$S_E \;=\; \frac{1}{2}\binom{q}{2}\binom{q-2}{2} \cdot p^2 \;=\; \binom{q}{2} \cdot p \cdot \left[\frac{1}{2}\binom{q-2}{2} \cdot p\right] \;=\; \binom{q}{2} \cdot p \cdot \frac{q^2 - 5q + 6}{4} \cdot p .$$

We now upper bound this as follows:

$$S_E \;<\; \binom{q}{2} \cdot p \cdot q^2 \cdot \frac{p}{4} \;\leq\; \binom{q}{2} \cdot p \cdot r^{\mu(h)} \cdot \frac{p}{4} \;\leq\; \frac{1}{4}\binom{q}{2} \cdot p .$$

Above the first inequality is true because Lemma 4.5 assumes $q \geq 2$. The second inequality is true because of the assumption made in Equation (7). The third inequality is true because $r^{\mu(h)} \cdot p < 1$. We have now obtained Equation (16).

The remaining task is to upper bound S_N. The difficulty here is that for $\{I,J\} \in N$ the random variables X_I and X_J are not independent. We let $d_i = |h^{-1}(R_i)|$ for $i \in [r]$ where $R = \{R_1, \ldots, R_r\}$ is the range of the hash function. If $\{I,J\} \in N$ then the two-elements sets I and J intersect in exactly one point. (They cannot be equal since I, J are assumed distinct.) Accordingly we have

$$S_N \;=\; \sum_{\{I,J\}\in N} \Pr[\, X_I = 1 \wedge X_J = 1 \,] \;=\; |N| \cdot \sum_{i=1}^{r} \frac{d_i(d_i-1)^2}{d^3} \;<\; \frac{|N|}{d^3} \cdot \sum_{i=1}^{r} d_i^3 . \tag{18}$$

We now compute the size of the set N:

$$
\begin{aligned}
|N| &= \frac{1}{2}\binom{q}{2}\binom{q}{2} - \frac{1}{2}\binom{q}{2} - \frac{1}{2}\binom{q}{2}\binom{q-2}{2} \\
&= \binom{q}{2} \cdot \left[\frac{1}{2}\binom{q}{2} - \frac{1}{2}\binom{q-2}{2} - \frac{1}{2}\right] \\
&= \binom{q}{2} \cdot \left[\frac{q(q-1)}{4} - \frac{(q-2)(q-3)}{4} - \frac{1}{2}\right] \\
&= \binom{q}{2} \cdot (q-2) \, .
\end{aligned}
$$

Putting this together with Equation (18) we have

$$
S_N \; < \; \binom{q}{2} \cdot q \cdot \left[\frac{1}{d^3} \cdot \textstyle\sum_{i=1}^{r} d_i^3\right] \, . \tag{19}
$$

To upper bound the sum of Equation (19), we view $d_1, \ldots, d_r$ as variables and consider the problem of maximizing $d_1^3 + \cdots + d_r^3$ subject to the constraints

$$
\sum_{i=1}^{r} d_i \; = \; d \qquad \text{and} \qquad \sum_{i=1}^{r} d_i^2 \; = \; d^2 \cdot r^{-\mu(h)} \, .
$$

The maximum occurs when $d_i = d \cdot r^{-\mu(h)}$ for $i = 1, \ldots, r^{\mu(h)}$ and $d_i = 0$ for $i = r^{\mu(h)}, \ldots, r$. The value of the maximum is

$$
r^{\mu(h)} \cdot d^3 r^{-3\mu(h)} \; = \; d^3 r^{-2\mu(h)} \, .
$$

Returning to Equation (19) with this information we get

$$
S_N \; < \; \binom{q}{2} \cdot q \cdot \left[\frac{1}{d^3} \cdot \textstyle\sum_{i=1}^{r} d_i^3\right] \; \le \; \binom{q}{2} \cdot q \cdot \frac{1}{d^3} \cdot d^3 r^{-2\mu(h)} \; = \; \binom{q}{2} \cdot q \cdot r^{-2\mu(h)} \, .
$$

We now use the assumption made in Equation (7), and finally use Lemma 3.3, to get

$$
S_N \; < \; \binom{q}{2} \cdot \frac{1}{4} \cdot \left(1 - \frac{r}{d}\right) \cdot r^{\mu(h)} \cdot r^{-2\mu(h)} \; \le \; \binom{q}{2} \cdot \frac{1}{4} \cdot \left(1 - \frac{r}{d}\right) r^{-\mu(h)} \; \le \; \binom{q}{2} \cdot \frac{1}{4} \cdot p \, .
$$

This proves Equation (17) and thus concludes the proof of Lemma 4.5. ▌

5 Does the MD transform preserve balance?

We consider the following popular paradigm for the construction of hash functions. First build a *compression function* $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$, where $b \ge 1$ is called the *block-length* and $c \ge 1$ is called the *chaining-length*. Then transform H into a hash function $\overline{H}\colon D_b \to \{0,1\}^c$, where

$$
D_b \; = \; \{\, M \in \{0,1\}^* \; : \; |M| = nb \text{ for some } 1 \le n < 2^b \,\} \, ,
$$

via the Merkle-Damgård (MD) [4, 1] transform depicted in Figure 2. (In this description and below, we let $\langle i \rangle_b$ denote the representation of integer i as a string of length *exactly* b bits for $i = 0, \ldots, 2^b - 1$.) In particular, modulo details, this is the paradigm used in the design of popular hash functions including MD5 [6], SHA-1 [5] and RIPEMD-160 [2].

For the considerations in this section, we will focus on the restriction of $\overline{H}$ to strings of some particular length. For any integer $1 \le n < 2^b$ (the number of blocks) we let $\overline{H}_n\colon D_{b,n} \to \{0,1\}^c$

Function $\overline{H}(M)$
  Break M into b-bit blocks $M_1 \| \cdots \| M_n$
  $M_{n+1} \leftarrow \langle n \rangle_b$; $C_0 \leftarrow 0^c$
  For $i = 1, \ldots, n+1$ do $C_i \leftarrow H(M_i \| C_{i-1})$ EndFor
  Return C_{n+1}

Figure 2: Hash function $\overline{H}\colon D_b \to \{0,1\}^c$ obtained via the MD transform applied to compression function $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$.

---

denote the restriction of $\overline{H}$ to the domain $D_{b,n}$, defined as the set of all strings in D_b that have length exactly nb bits.

Our results lead us to desire that $\overline{H}_n$ has high balance for all practical values of n. Designers could certainly try to ensure that the compression function is regular or has high balance, but to be assured that $\overline{H}_n$ has high balance it would need to be the case that the MD transform is "balance preserving." Unfortunately, the following shows that this is not true. It presents an example of a compression function H which has high balance (in fact is regular, with balance one) but $\overline{H}_n$ has low balance (in fact, balance zero) even for $n = 2$.

Proposition 5.1 *Let b, c be positive integers. There exists a compression function $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$ such that H is regular ($\mu(H) = 1$) but $\overline{H}_2$ is a constant function ($\mu(\overline{H}_2) = 0$).* ▮

Proof of Proposition 5.1: Let $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$ map $B\|C$ to C for all b-bit strings B and c-bit strings C. Clearly $\mu(H) = 1$ since each point in $\{0,1\}^c$ has exactly 2^b pre-images under H. Because the initial vector (IV) in the MD transform is the constant $C_0 = 0^c$, and by the definition of H, the function $\overline{H}_2$ maps all inputs to 0^c. ▮

This example might be viewed as contrived particularly because the compression function H above is not collision-resistant (although it is very resistant to birthday attacks), but in fact it still serves to illustrate an important point. The popularity of the MD paradigm arises from the fact that it *provably* preserves collision-resistance [4, 1]. However, the above shows that it does not provably preserve balance. Even though Proposition 5.1 does not say that the transform will *always* be poor at preserving balance, it says that we cannot count on the transform to preserve balance in general. This means that simply ensuring high balance of the compression function is not a suitable general design principle.

Is there any other design principle whereby some properties of the compression function suffice to ensure high balance of the hash function? Towards finding one we note that the behavior exhibited by the function $\overline{H}_2$ in the proof of Proposition 5.1 arose because the initial vector (IV) of the MD transform was $C_0 = 0^c$, and although H was regular, the restriction of H to inputs having the last c bits 0 was not regular, and in fact was constant. Accordingly we consider requiring regularity conditions not just on the compression function but on certain related functions as well. If $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$ then define $H_0\colon \{0,1\}^b \to \{0,1\}^c$ via $M \mapsto H(M\|0^c)$ for all $M \in \{0,1\}^b$, and for $n \geq 1$ define $H_n\colon \{0,1\}^c \to \{0,1\}^c$ via $M \mapsto H(\langle n \rangle_b \| M)$ for all $M \in \{0,1\}^c$. The following shows that if H, H_0, H_n are all regular, meaning have balance one, then $\overline{H}_n$ is also regular.

Proposition 5.2 *Let b, c, n be positive integers. Let $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$ and let H_0, H_n be as above. Assume H, H_0, and H_n are all regular. Then $\overline{H}_n$ is regular.* ▮

Proof of Proposition 5.2: The computation of $\overline{H}_n$ can be written as

Function $\overline{H}_n(M)$
Break M into b-bit blocks $M_1 \| \cdots \| M_n$; $C_1 \leftarrow H_0(M_1)$
For $i = 2, \ldots, n$ do $C_i \leftarrow H(M_i \| C_{i-1})$ EndFor
$C_{n+1} \leftarrow H_n(C_n)$; Return C_{n+1}

It is not hard to check that the assumed regularity of H_0, H and H_n imply the regularity of $\overline{H}_n$. ▮

Unfortunately Proposition 5.2 is not "robust." Although $\overline{H}_n$ has balance one if H, H_0, H_n have balance one, it turns out that if H, H_0, H_n have balance that is high but not quite one, we are *not* assured that $\overline{H}_n$ has high balance. Proposition 5.3 shows that even a slight deviation from the maximum balance of one in H, H_0, H_n can be amplified, and result in $\overline{H}_n$ having very low balance. The proof of the following is in Appendix C.

Proposition 5.3 *Let b, c be integers, $b \geq c \geq 2$, and let $n \geq c$. Then there exists a compression function H: $\{0,1\}^{b+c} \rightarrow \{0,1\}^c$ such that $\mu(H) \geq 1 - 1/c$, $\mu(H_0) = 1$, and $\mu(H_n) \geq 1 - 2/c$, but $\mu(\overline{H}_n) \leq 1/c$, where the functions H_0, H_n are defined as above.* ▮

A random compression function will have expected balance that is high but not quite 1 (cf. Proposition 4.4) and we expect that practical compression functions are in the same boat. Furthermore it seems harder to build compression functions that have balance exactly one than close to one. So the lack of robustness of Proposition 5.2, as exhibited by Proposition 5.3, means that Proposition 5.2 is of limited use.

The consequence of the results in this section is that we are unable to recommend any design principle that, to ensure high balance, focuses solely on establishing properties of the compression function. It seems one is forced to look directly at the hash function. We endeavor next to do this for SHA-1.

6 Experiments on SHA-1

Let SHA_n: $\{0,1\}^n \rightarrow \{0,1\}^{160}$ denote the restriction of SHA-1 to inputs of length $n < 2^{64}$. Because SHA-1's range is $\{0,1\}^{160}$, it is commonly believed that the expected number of trials necessary to find a collision for SHA_n is approximately 2^{80}. As Theorem 4.3 shows, however, this is only true if the balance of SHA_n is one or close to one for all practical values of n. If the balance is not close to one, then we expect to be able to find collisions using less work. It therefore seems desirable to calculate (or approximate) the balance of SHA_n for reasonable values of n (eg. $n = 320$). A direct computation of $\mu(\mathrm{SHA}_n)$ based on Definition 3.1 is however infeasible given the size of the range of SHA_n. Accordingly we focus on a more achievable goal. We look at properties of SHA_n that one can reasonably test and whose absence might indicate that SHA_n does not have high balance. Our experiments are not meant to be exhaustive, but rather representative of the types of feasible experiments one can perform with SHA-1.

Let $\mathrm{SHA}_{n;t_1 \ldots t_2}$: $\{0,1\}^n \rightarrow \{0,1\}^{t_2 - t_1 + 1}$ denote the function that returns the t_1-th through t_2-th output bits of SHA_n. We ask what exactly is the balance of $\mathrm{SHA}_{32;t_1 \ldots t_2}$ when $t_2 - t_1 + 1 \in \{8, 16, 24\}$. And we ask whether the functions $\mathrm{SHA}_{256;t_1 \ldots t_2}$ and $\mathrm{SHA}_{320;t_1 \ldots t_2}$ appear regular when $t_2 - t_1 + 1 \in \{8, 16, 24\}$. We only consider $t_1 \equiv 1 \bmod 8$. (Note that SHA_{256} is SHA-1 restricted to the domain $\{0,1\}^{256}$, not NIST's new SHA-256 hash algorithm.)

Balance of $\text{SHA}_{32;t_1\ldots t_2}$. We calculate the balance of $\text{SHA}_{32;t_1\ldots t_2}$ for all pairs t_1, t_2 such that $t_2 - t_1 + 1 \in \{8, 16, 24\}$ and t_1 begins on a byte boundary (ie. we look at all 1-, 2-, and 3-byte portions of the SHA-1 output). The calculated values of $\mu(\text{SHA}_{32;t_1\ldots t_2})$ appear in Appendix D.1. Characteristic values are $\mu(\text{SHA}_{32;1\ldots 8}) = 0.99999998893$, $\mu(\text{SHA}_{32;1\ldots 16}) = 0.999998623$, and $\mu(\text{SHA}_{32;1\ldots 24}) = 0.99976567$, indicating that, for the specified values of t_1, t_2, $\text{SHA}_{32;t_1\ldots t_2}$ is close to regular.

These results do not imply that the functions $\text{SHA}_{n;t_1\ldots t_2}$ or SHA_n, $n > 32$ and t_1, t_2 as before, are regular. But the fact that $\text{SHA}_{32;t_1\ldots t_2}$ is almost regular is encouraging — a small value for $\mu(\text{SHA}_{32;t_1\ldots t_2})$ for any of the specified t_1, t_2 pairs might indicate some unusual property of the SHA-1 hash function.

Experiments on SHA_{256} and SHA_{320}. Fix $m = 256$ or $m = 320$. Although we cannot calculate the balance of SHA_m (cf. the discussions above), we can compare the outputs of $\text{SHA}_{m;t_1\ldots t_2}$ on random inputs to what one would expect from a regular function. Assume that $t_2 - t_1 + 1 \in \{8, 16, 24\}$.

If the outputs of $\text{SHA}_{m;t_1\ldots t_2}$ on random bits are approximately the same as what one would expect from a regular function, it would support the view that SHA_m has high balance. However, a significant difference between the outputs of $\text{SHA}_{m;t_1\ldots t_2}$ on random inputs and what one would expect from a regular function might indicate some unusual behavior with SHA-1, and this unusual behavior would deserve further investigation.

We used χ^2 tests to compare the output of $\text{SHA}_{m;t_1\ldots t_2}$ on random inputs to the output of a regular function. For each test we picked 2^{32} distinct random strings from $\{0,1\}^m$, hashed those values using SHA_m, and counted the number of times we saw each $t_2 - t_1 + 1$-bit substring. (The random inputs were generated using 256-bit Rijndael in counter mode with a randomly chosen key.) If SHA_m is regular, then we expect each $t_2 - t_1 + 1$-bit output of $\text{SHA}_{m;t_1\ldots t_2}$ to occur $2^{32+t_1-t_2-1}$ times. The results of the experiments are summarized in Appendix D.2 and Appendix D.3. Recall that the P-value indicates the probability that $\text{SHA}_{m;t_1\ldots t_2}$ will have the observed χ^2 test statistic when the null hypothesis is true; ie. when $\text{SHA}_{m;t_1\ldots t_2}$ is indeed regular.

The results in Appendix D.2 and Appendix D.3 are consistent with SHA_{256} and SHA_{320} having high balance. However, we again point out that these tests were only designed to uncover gross anomalies and are not exhaustive.

References

[1] I. Damgård. A Design Principle for Hash Functions. *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[2] H. Dobbertin, A. Bosselaers, B. Preneel. RIPEMD-160, a strengthened version of RIPEMD. *Fast Software Encryption '96*, Lecture Notes in Computer Science Vol. 1039, D. Gollmann ed., Springer-Verlag, 1996.

[3] A. Menezes, P. van Oorschot and S. Vanstone. Handbook of applied cryptography. CRC Press, 1997.

[4] R. Merkle. One way hash functions and DES. *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[5] National Institute of Standards. FIPS 180-2, Secure hash standard. August 1, 2000.

[6] R. Rivest. The MD5 message-digest algorithm. IETF RFC 1321, April 1992.

[7] D. Stinson. Cryptography theory and practice, 1st Edition. CRC Press, 1995.

[8] G. Yuval. How to swindle Rabin. *Cryptologia* (3), 1979, 187–190.

A Proof of Theorem 4.3

We will establish a somewhat stronger result, namely:

Lemma A.1 Let h: $D \to R$ be a hash function. Let $d = |D|$ and $r = |R|$ and assume $d > r \geq 2$. Let Q denote the expected number of trials to find a collision for h in the birthday attack of Figure 1. Let $\mu(h)$ be the balance of h as per Definition 3.1. Then

$$Q = \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{8}{r^{-\mu(h)} - 1/d}} . \qquad \blacksquare \tag{20}$$

We will first use Lemma A.1 to prove Theorem 4.3 and then prove Lemma A.1.

Proof of Theorem 4.3: For the upper bound we have

$$\begin{aligned} Q &= \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{8}{r^{-\mu(h)} - 1/d}} && (21) \\ &\leq \frac{1}{2} + \frac{1}{2} \cdot \left(\sqrt{1} + \sqrt{\frac{8}{r^{-\mu(h)} - 1/d}} \right) && (22) \\ &= \frac{1}{2} + \frac{1}{2} + \frac{\sqrt{2}}{\sqrt{r^{-\mu(h)} - 1/d}} \\ &\leq 1 + \frac{\sqrt{2}}{\sqrt{(1 - r/d) r^{-\mu(h)}}} && (23) \\ &\leq 1 + \frac{\sqrt{2}}{\sqrt{r^{-\mu(h)}/2}} && (24) \\ &= 1 + 2 \cdot r^{-\mu(h)/2} . && (25) \end{aligned}$$

Equation (21) is by Lemma A.1. Equation (22) uses the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ which is valid for all non-negative reals a, b. Equation (23) is by Lemma 3.3. Equation (24) is true by the assumption, made in the statement of Theorem 4.3, that $d \geq 2r$. We now proceed to the lower bound. We have

$$Q = \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{8}{r^{-\mu(h)} - 1/d}} \geq \frac{1}{2} \cdot \sqrt{\frac{8}{r^{-\mu(h)} - 1/d}} \geq \frac{\sqrt{2}}{\sqrt{r^{-\mu(h)}}} = \sqrt{2} \cdot r^{-\mu(h)/2} .$$

This concludes the proof of Theorem 4.3. $\blacksquare$

Proof of Lemma A.1: Let the random variable X be as in the proof of Lemma 4.5. The threshold is the number of trials at which one expects there to be one collision, meaning is a value of q such that $\mathbf{E}[X] = 1$. Taking the value of $\mathbf{E}[X]$ from Equation (11), we proceed to solve the equation

$$\binom{q}{2} \cdot \left[r^{-\mu(h)} - \frac{1}{d} \right] \;=\; 1$$

for q. The above can be written as

$$q^2 - q - \frac{2}{r^{-\mu(h)} - 1/d} \;=\; 0 \,.$$

The positive root of this equation,

$$Q \;=\; \frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 + \frac{8}{r^{-\mu(h)} - 1/d}} \,,$$

is the value we seek. ▮

B Proof of Proposition 4.4

Proof of Proposition 4.4: As usual say $R = \{R_1, \ldots, R_r\}$, and also $D = \{D_1, \ldots, D_d\}$. For $i \in [r]$ let $d_i = |h^{-1}(R_i)|$. This is a random variable over the choice of h, and we begin by computing the expectation of d_i^2. For $j \in [d]$ let X_j be the random variable that takes value 1 if $h(D_j) = R_i$ and 0 otherwise. Then

$$\begin{aligned}
\mathbf{E}\left[d_i^2\right] &= \mathbf{E}\left[(X_1 + \cdots + X_d)^2\right] \\
&= \mathbf{E}\left[\textstyle\sum_{j=1}^{d} X_j^2 + \sum_{k \neq l} X_k X_l\right] \\
&= \sum_{j=1}^{d} \mathbf{E}\left[X_j^2\right] + \sum_{k \neq l} \mathbf{E}\left[X_k X_l\right] \qquad (26) \\
&= \sum_{j=1}^{d} \mathbf{E}\left[X_j\right] + \sum_{k \neq l} \mathbf{E}\left[X_k\right] \cdot \mathbf{E}\left[X_l\right] \qquad (27) \\
&= \sum_{j=1}^{d} \frac{1}{r} + \sum_{k \neq l} \frac{1}{r^2} \\
&= \frac{d}{r} + \frac{d^2 - d}{r^2} \\
&= \frac{d^2 + dr - d}{r^2} \,.
\end{aligned}$$

Equation (26) is by linearity of expectation. Since X_j is boolean valued we have $X_j^2 = X_j$, and on the other hand if $k \neq l$ then X_k, X_l are independent, which justifies Equation (27). Now from the above we have

$$\mathbf{E}\left[r^{-\mu(h)}\right] \;=\; \mathbf{E}\left[\frac{d_1^2 + \cdots + d_r^2}{d^2}\right] \;=\; \frac{1}{d^2} \cdot \sum_{i=1}^{r} \mathbf{E}\left[d_i^2\right] \;=\; \frac{1}{d^2} \cdot r \cdot \frac{d^2 + dr - d}{r^2} \;=\; \frac{1}{r} \cdot \left(1 + \frac{r-1}{d}\right) \,.$$

This completes the proof of Proposition 4.4. ▮

C Proof of Proposition 5.3

Proof of Proposition 5.3: Let $H\colon \{0,1\}^{b+c} \to \{0,1\}^c$ be defined as

$$H(B\|C) = \begin{cases} \langle B\rangle_c & C = 0^c \\ \langle C \ll 1\rangle_c \oplus (0^{c-1}1) & C \neq 0^c \end{cases}$$

where B is b-bits long, C is c-bits long, $\langle B\rangle_c$ is the right-most c bits of B, and $\langle C \ll 1\rangle_c$ is the left shift of C by one bit (ie. $\langle C \ll 1\rangle_c$ is a c-bit string, the left-most $c-1$ bits of which are the right-most $c-1$ bits of C, and the right-most bit of which is 0).

Clearly $\mu(H_0) = 1$. To see that $\mu(H) \geq 1 - 1/c$ we note that there are 2^{c-1} points $X \in \{0,1\}^c$ with a right-most bit of 0 and each of these points has 2^{b-c} preimages (corresponding to the set $\{0,1\}^{b-c}X0^c$). There is one point of the form $0^{c-1}1$ and it has $2^b + 2^{b-c}$ preimages (corresponding to the sets $\{0,1\}^b 10^{c-1}$ and $\{0,1\}^{b-c}0^{c-1}10^c$). There are $2^{c-1}-1$ additional points $Y \in \{0,1\}^c$ with a right-most bit of 1 and each of these points has $2^{b+1} + 2^{b-c}$ preimages (corresponding to the sets $\{0,1\}^{b+1}Y'$ and $\{0,1\}^{b-c}Y0^c$, where Y' is the left-most $c-1$ bits of Y). Let

$$\begin{aligned} S &= 2^{c-1}(2^{b-c})^2 + (2^b + 2^{b-c})^2 + (2^{c-1}-1)(2^{b+1}+2^{b-c})^2 \\ &\leq 2^{2b+c+1} . \end{aligned}$$

It follows that

$$\mu(H) \;=\; \log_{2^c}\left[\frac{2^{2b+2c}}{S}\right] \geq \log_{2^c}\left[\frac{2^{2b+2c}}{2^{2b+c+1}}\right] = \frac{c-1}{c} .$$

We lower bound $\mu(H_n)$ as follows. Let $R = \sum_{C\in\{0,1\}^c} d_C^2$, where d_C^2 is the number of preimages of $C \in \{0,1\}^c$ under H_n. We divide the analysis into three cases. In the first case we assume that the right-most bit of $\langle n\rangle_b$ is 0. This implies that there will be one point in $\{0,1\}^{c-1}0$ with one preimage and all the remaining points in $\{0,1\}^{c-1}0$ will have no preimage. Of the 2^{c-1} points in $\{0,1\}^{c-1}1$, all but point $0^{c-1}1$ will have two preimages, and $0^{c-1}1$ will have one preimage. Thus $R < 2^{c+1}$.

Let $\langle n\rangle_c$ be the right-most c bits of $\langle n\rangle_b$. In the second case we assume that the right-most bit of $\langle n\rangle_b$ is 1 and and that $\langle n\rangle_c \neq 0^{c-1}1$. All the points in $\{0,1\}^{c-1}0$ have no preimages, $2^{c-1}-2$ points in $\{0,1\}^{c-1}1$ have two preimages, the point $\langle n\rangle_c$ has three preimages, and the point $0^{c-1}1$ has one preimage. In this case $R < 2^{c+2}$. In the final case we assume that the right-most bit of $\langle n\rangle_b$ is 1 and that $\langle n\rangle_c = 0^{c-1}1$. All the points in $\{0,1\}^{c-1}0$ have no preimages and all the points in $\{0,1\}^{c-1}1$ have two preimages. In this case $R = 2^{c+1}$. These results imply that

$$\mu(H_n) \;=\; \log_{2^c}\left[\frac{2^{2c}}{R}\right] \geq \log_{2^c}\left[\frac{2^{2c}}{2^{c+2}}\right] = \frac{c-2}{c} .$$

Let us now consider the balance of $\overline{H}_n$. Let $M = M_1\|\cdots\|M_n \in \{0,1\}^{bn}$ be a string and let $|M_i| = b$. Then if $M_1 \notin \{0,1\}^{b-c}0^c$, we have that $\overline{H}_n(M) = 1^c$; i.e., $|\overline{H}_n^{-1}(1^c)| \geq 2^{bn} - 2^{bn-c}$. This allows us to upper bound $\mu(\overline{H}_n)$ as follows:

$$\mu(\overline{H}_n) \leq \log_{2^c}\left[\frac{(2^{bn})^2}{(2^{bn}-2^{bn-c})^2}\right] \leq \log_{2^c}\left[\frac{2^{2bn}}{2^{2bn}-2^{2bn-c+1}}\right]$$

Using the assumption that $c \geq 2$,

$$\mu(\overline{H}_n) \leq \log_{2^c}\left[\frac{2^{2bn}}{2^{2bn-1}}\right] = \frac{1}{c}$$

as desired. ▌

D Experimental Data

D.1 Balance of SHA-1 on 32-bit inputs

The following table shows the balance of $\mathrm{SHA}_{32;t_1\ldots t_2}$ when $t_2 - t_1 + 1 \in \{8, 16, 24\}$ and t_1 begins on a byte boundary.

$t_2 - t_1 + 1 = 8$	$t_2 - t_1 + 1 = 16$	$t_2 - t_1 + 1 = 24$
$\mu(\mathrm{SHA}_{32;1\ldots8}) = 0.99999998893$	$\mu(\mathrm{SHA}_{32;1\ldots16}) = 0.999998623$	$\mu(\mathrm{SHA}_{32;1\ldots24}) = 0.99976567$
$\mu(\mathrm{SHA}_{32;9\ldots16}) = 0.99999998941$	$\mu(\mathrm{SHA}_{32;9\ldots24}) = 0.999998604$	$\mu(\mathrm{SHA}_{32;9\ldots32}) = 0.99976548$
$\mu(\mathrm{SHA}_{32;17\ldots24}) = 0.99999998972$	$\mu(\mathrm{SHA}_{32;17\ldots32}) = 0.999998620$	$\mu(\mathrm{SHA}_{32;17\ldots40}) = 0.99976553$
$\mu(\mathrm{SHA}_{32;25\ldots32}) = 0.99999998884$	$\mu(\mathrm{SHA}_{32;25\ldots40}) = 0.999998627$	$\mu(\mathrm{SHA}_{32;25\ldots48}) = 0.99976561$
$\mu(\mathrm{SHA}_{32;33\ldots40}) = 0.99999999079$	$\mu(\mathrm{SHA}_{32;33\ldots48}) = 0.999998641$	$\mu(\mathrm{SHA}_{32;33\ldots56}) = 0.99976582$
$\mu(\mathrm{SHA}_{32;41\ldots48}) = 0.99999998909$	$\mu(\mathrm{SHA}_{32;41\ldots56}) = 0.999998620$	$\mu(\mathrm{SHA}_{32;41\ldots64}) = 0.99976559$
$\mu(\mathrm{SHA}_{32;49\ldots56}) = 0.99999998912$	$\mu(\mathrm{SHA}_{32;49\ldots64}) = 0.999998626$	$\mu(\mathrm{SHA}_{32;49\ldots72}) = 0.99976558$
$\mu(\mathrm{SHA}_{32;57\ldots64}) = 0.99999999083$	$\mu(\mathrm{SHA}_{32;57\ldots72}) = 0.999998625$	$\mu(\mathrm{SHA}_{32;57\ldots80}) = 0.99976581$
$\mu(\mathrm{SHA}_{32;65\ldots72}) = 0.99999998923$	$\mu(\mathrm{SHA}_{32;65\ldots80}) = 0.999998627$	$\mu(\mathrm{SHA}_{32;65\ldots88}) = 0.99976575$
$\mu(\mathrm{SHA}_{32;73\ldots80}) = 0.99999999083$	$\mu(\mathrm{SHA}_{32;73\ldots88}) = 0.999998637$	$\mu(\mathrm{SHA}_{32;73\ldots96}) = 0.99976577$
$\mu(\mathrm{SHA}_{32;81\ldots88}) = 0.99999998925$	$\mu(\mathrm{SHA}_{32;81\ldots96}) = 0.999998622$	$\mu(\mathrm{SHA}_{32;81\ldots104}) = 0.99976558$
$\mu(\mathrm{SHA}_{32;89\ldots96}) = 0.99999998987$	$\mu(\mathrm{SHA}_{32;89\ldots104}) = 0.999998617$	$\mu(\mathrm{SHA}_{32;89\ldots112}) = 0.99976554$
$\mu(\mathrm{SHA}_{32;97\ldots104}) = 0.99999998862$	$\mu(\mathrm{SHA}_{32;97\ldots112}) = 0.999998624$	$\mu(\mathrm{SHA}_{32;97\ldots120}) = 0.99976567$
$\mu(\mathrm{SHA}_{32;105\ldots112}) = 0.99999998826$	$\mu(\mathrm{SHA}_{32;105\ldots120}) = 0.999998626$	$\mu(\mathrm{SHA}_{32;105\ldots128}) = 0.99976562$
$\mu(\mathrm{SHA}_{32;113\ldots120}) = 0.99999998959$	$\mu(\mathrm{SHA}_{32;113\ldots128}) = 0.999998616$	$\mu(\mathrm{SHA}_{32;113\ldots136}) = 0.99976566$
$\mu(\mathrm{SHA}_{32;121\ldots128}) = 0.99999998999$	$\mu(\mathrm{SHA}_{32;121\ldots136}) = 0.999998634$	$\mu(\mathrm{SHA}_{32;121\ldots144}) = 0.99976556$
$\mu(\mathrm{SHA}_{32;129\ldots136}) = 0.99999999052$	$\mu(\mathrm{SHA}_{32;129\ldots144}) = 0.999998636$	$\mu(\mathrm{SHA}_{32;129\ldots152}) = 0.99976563$
$\mu(\mathrm{SHA}_{32;137\ldots144}) = 0.99999998916$	$\mu(\mathrm{SHA}_{32;137\ldots152}) = 0.999998615$	$\mu(\mathrm{SHA}_{32;137\ldots160}) = 0.99976554$
$\mu(\mathrm{SHA}_{32;145\ldots152}) = 0.99999998769$	$\mu(\mathrm{SHA}_{32;145\ldots160}) = 0.999998626$	
$\mu(\mathrm{SHA}_{32;153\ldots160}) = 0.99999998993$		

D.2 Random 256-bit inputs

The following table shows the P-values of the χ^2 test statistics for $\mathrm{SHA}_{256;t_1\ldots t_2}$ when $t_2 - t_1 + 1 \in \{8, 16, 24\}$ and t_1 begins on a byte boundary.

$t_2 - t_1 + 1 = 8$		$t_2 - t_1 + 1 = 16$		$t_2 - t_1 + 1 = 24$	
Function	P-value	Function	P-value	Function	P-value
$\mathrm{SHA}_{256;1\ldots8}$	0.100485	$\mathrm{SHA}_{256;1\ldots16}$	0.586626	$\mathrm{SHA}_{256;1\ldots24}$	0.692566
$\mathrm{SHA}_{256;9\ldots16}$	0.619285	$\mathrm{SHA}_{256;9\ldots24}$	0.140110	$\mathrm{SHA}_{256;9\ldots32}$	0.317284
$\mathrm{SHA}_{256;17\ldots24}$	0.935787	$\mathrm{SHA}_{256;17\ldots32}$	0.466384	$\mathrm{SHA}_{256;17\ldots40}$	0.021164
$\mathrm{SHA}_{256;25\ldots32}$	0.175975	$\mathrm{SHA}_{256;25\ldots40}$	0.439931	$\mathrm{SHA}_{256;25\ldots48}$	0.469674
$\mathrm{SHA}_{256;33\ldots40}$	0.748468	$\mathrm{SHA}_{256;33\ldots48}$	0.361088	$\mathrm{SHA}_{256;33\ldots56}$	0.040855
$\mathrm{SHA}_{256;41\ldots48}$	0.031226	$\mathrm{SHA}_{256;41\ldots56}$	0.184079	$\mathrm{SHA}_{256;41\ldots64}$	0.676174
$\mathrm{SHA}_{256;49\ldots56}$	0.583221	$\mathrm{SHA}_{256;49\ldots64}$	0.763258	$\mathrm{SHA}_{256;49\ldots72}$	0.670727
$\mathrm{SHA}_{256;57\ldots64}$	0.457368	$\mathrm{SHA}_{256;57\ldots72}$	0.376232	$\mathrm{SHA}_{256;57\ldots80}$	0.096921
$\mathrm{SHA}_{256;65\ldots72}$	0.329475	$\mathrm{SHA}_{256;65\ldots80}$	0.309056	$\mathrm{SHA}_{256;65\ldots88}$	0.744634
$\mathrm{SHA}_{256;73\ldots80}$	0.547074	$\mathrm{SHA}_{256;73\ldots88}$	0.686284	$\mathrm{SHA}_{256;73\ldots96}$	0.170097

$t_2 - t_1 + 1 = 8$ Function	P-value	$t_2 - t_1 + 1 = 16$ Function	P-value	$t_2 - t_1 + 1 = 24$ Function	P-value
$SHA_{256;81...88}$	0.085886	$SHA_{256;81...96}$	0.310768	$SHA_{256;81...104}$	0.313116
$SHA_{256;89...96}$	0.841003	$SHA_{256;89...104}$	0.898719	$SHA_{256;89...112}$	0.919306
$SHA_{256;97...104}$	0.508412	$SHA_{256;97...112}$	0.749241	$SHA_{256;97...120}$	0.806863
$SHA_{256;105...112}$	0.091271	$SHA_{32;105...120}$	0.159980	$SHA_{32;105...128}$	0.241736
$SHA_{256;113...120}$	0.972487	$SHA_{256;113...128}$	0.384936	$SHA_{256;113...136}$	0.113037
$SHA_{256;121...128}$	0.068753	$SHA_{256;121...136}$	0.914603	$SHA_{256;121...144}$	0.443723
$SHA_{256;129...136}$	0.580299	$SHA_{256;129...144}$	0.975337	$SHA_{256;129...152}$	0.136327
$SHA_{256;137...144}$	0.353030	$SHA_{256;137...152}$	0.317605	$SHA_{256;137...160}$	0.492472
$SHA_{256;145...152}$	0.675190	$SHA_{256;145...160}$	0.581100		
$SHA_{256;153...160}$	0.235566				

D.3 Random 320-bit inputs

The following table shows the P-values of the χ^2 test statistics for $SHA_{320;t_1...t_2}$ when $t_2 - t_1 + 1 \in \{8, 16, 24\}$ and t_1 begins on a byte boundary.

$t_2 - t_1 + 1 = 8$ Function	P-value	$t_2 - t_1 + 1 = 16$ Function	P-value	$t_2 - t_1 + 1 = 24$ Function	P-value
$SHA_{320;1...8}$	0.427497	$SHA_{320;1...16}$	0.815452	$SHA_{320;1...24}$	0.850009
$SHA_{320;9...16}$	0.929965	$SHA_{320;9...24}$	0.042673	$SHA_{320;9...32}$	0.669224
$SHA_{320;17...24}$	0.083470	$SHA_{320;17...32}$	0.423254	$SHA_{320;17...40}$	0.612206
$SHA_{320;25...32}$	0.011549	$SHA_{320;25...40}$	0.296112	$SHA_{320;25...48}$	0.307289
$SHA_{320;33...40}$	0.999038	$SHA_{320;33...48}$	0.866707	$SHA_{320;33...56}$	0.252888
$SHA_{320;41...48}$	0.776933	$SHA_{320;41...56}$	0.301196	$SHA_{320;41...64}$	0.817281
$SHA_{320;49...56}$	0.936904	$SHA_{320;49...64}$	0.778989	$SHA_{320;49...72}$	0.501343
$SHA_{320;57...64}$	0.329116	$SHA_{320;57...72}$	0.919067	$SHA_{320;57...80}$	0.732148
$SHA_{320;65...72}$	0.201503	$SHA_{320;65...80}$	0.157921	$SHA_{320;65...88}$	0.007878
$SHA_{320;73...80}$	0.782695	$SHA_{320;73...88}$	0.556592	$SHA_{320;73...96}$	0.855473
$SHA_{320;81...88}$	0.391338	$SHA_{320;81...96}$	0.963846	$SHA_{320;81...104}$	0.216442
$SHA_{320;89...96}$	0.916778	$SHA_{320;89...104}$	0.207163	$SHA_{320;89...112}$	0.904849
$SHA_{320;97...104}$	0.484870	$SHA_{320;97...112}$	0.128456	$SHA_{320;97...120}$	0.688773
$SHA_{320;105...112}$	0.912156	$SHA_{320;105...120}$	0.450265	$SHA_{320;105...128}$	0.216091
$SHA_{320;113...120}$	0.426536	$SHA_{320;113...128}$	0.786771	$SHA_{320;113...136}$	0.901526
$SHA_{320;121...128}$	0.885900	$SHA_{320;121...136}$	0.601405	$SHA_{320;121...144}$	0.616341
$SHA_{320;129...136}$	0.680423	$SHA_{320;129...144}$	0.153638	$SHA_{320;129...152}$	0.804534
$SHA_{320;137...144}$	0.351930	$SHA_{320;137...152}$	0.293502	$SHA_{320;137...160}$	0.213030
$SHA_{320;145...152}$	0.342827	$SHA_{320;145...160}$	0.131438		
$SHA_{320;153...160}$	0.111307				

Because the P-values for some of the above experiments (eg. $SHA_{320;65...88}$) were moderately small, we repeated the above experiment. The results of the second experiment are presented below.

$t_2 - t_1 + 1 = 8$ Function	P-value	$t_2 - t_1 + 1 = 16$ Function	P-value	$t_2 - t_1 + 1 = 24$ Function	P-value
$SHA_{320;1...8}$	0.883591	$SHA_{320;1...16}$	0.831152	$SHA_{320;1...24}$	0.306404

$t_2 - t_1 + 1 = 8$		$t_2 - t_1 + 1 = 16$		$t_2 - t_1 + 1 = 24$	
Function	P-value	Function	P-value	Function	P-value
$\mathrm{SHA}_{320;9...16}$	0.607924	$\mathrm{SHA}_{320;9...24}$	0.180552	$\mathrm{SHA}_{320;9...32}$	0.117292
$\mathrm{SHA}_{320;17...24}$	0.007593	$\mathrm{SHA}_{320;17...32}$	0.531028	$\mathrm{SHA}_{320;17...40}$	0.823950
$\mathrm{SHA}_{320;25...32}$	0.037554	$\mathrm{SHA}_{320;25...40}$	0.738748	$\mathrm{SHA}_{320;25...48}$	0.753472
$\mathrm{SHA}_{320;33...40}$	0.725501	$\mathrm{SHA}_{320;33...48}$	0.756219	$\mathrm{SHA}_{320;33...56}$	0.175559
$\mathrm{SHA}_{320;41...48}$	0.769757	$\mathrm{SHA}_{320;41...56}$	0.820646	$\mathrm{SHA}_{320;41...64}$	0.238651
$\mathrm{SHA}_{320;49...56}$	0.833762	$\mathrm{SHA}_{320;49...64}$	0.116576	$\mathrm{SHA}_{320;49...72}$	0.446928
$\mathrm{SHA}_{320;57...64}$	0.090176	$\mathrm{SHA}_{320;57...72}$	0.027964	$\mathrm{SHA}_{320;57...80}$	0.836329
$\mathrm{SHA}_{320;65...72}$	0.379653	$\mathrm{SHA}_{320;65...80}$	0.192988	$\mathrm{SHA}_{320;65...88}$	0.294248
$\mathrm{SHA}_{320;73...80}$	0.774100	$\mathrm{SHA}_{320;73...88}$	0.121466	$\mathrm{SHA}_{320;73...96}$	0.290684
$\mathrm{SHA}_{320;81...88}$	0.266744	$\mathrm{SHA}_{320;81...96}$	0.078667	$\mathrm{SHA}_{320;81...104}$	0.726818
$\mathrm{SHA}_{320;89...96}$	0.648729	$\mathrm{SHA}_{320;89...104}$	0.233293	$\mathrm{SHA}_{320;89...112}$	0.991511
$\mathrm{SHA}_{320;97...104}$	0.887169	$\mathrm{SHA}_{320;97...112}$	0.258782	$\mathrm{SHA}_{320;97...120}$	0.826019
$\mathrm{SHA}_{320;105...112}$	0.456699	$\mathrm{SHA}_{320;105...120}$	0.578998	$\mathrm{SHA}_{320;105...128}$	0.489699
$\mathrm{SHA}_{320;113...120}$	0.012146	$\mathrm{SHA}_{320;113...128}$	0.733158	$\mathrm{SHA}_{320;113...136}$	0.561124
$\mathrm{SHA}_{320;121...128}$	0.494374	$\mathrm{SHA}_{320;121...136}$	0.893367	$\mathrm{SHA}_{320;121...144}$	0.831479
$\mathrm{SHA}_{320;129...136}$	0.558424	$\mathrm{SHA}_{320;129...144}$	0.361668	$\mathrm{SHA}_{320;129...152}$	0.249553
$\mathrm{SHA}_{320;137...144}$	0.543976	$\mathrm{SHA}_{320;137...152}$	0.616025	$\mathrm{SHA}_{320;137...160}$	0.862948
$\mathrm{SHA}_{320;145...152}$	0.315248	$\mathrm{SHA}_{320;145...160}$	0.182013		
$\mathrm{SHA}_{320;153...160}$	0.538103				