Measuring Hash Trustworthiness via Collision Utility Metrics: Logical Cryptanalysis of MD4

Alexander M. Scheel, Eric W. D. Rozier
Department of Computer Science
Iowa State University of Science and Technology
Ames, Iowa, USA 50011
Email: alexander.m.scheel@gmail.com, erozier@iastate.edu

***Abstract*—The discovery of fast collision attacks in cryptographic hash functions has traditionally resulted in the immediate deprecation of that hash function. In this paper we propose five scalable and practical metrics for evaluating the utility of collision classes based on boolean constraints and show that the published attacks by X. Wang, Y. Sasaki, P. Kasselman, H. Dobbertin, and M. Schläffer in MD4 have high utility. We expand on existing attacks by developing a series of techniques based on logical cryptanalysis to find over 35,000 collisions in MD4 based on existing collisions, through the novel definition of a collision neighborhood. We demonstrate new techniques for inductively building full collisions from reduced round variants of MD4. We propose these techniques as a mechanism for measuring hash trustworthiness and discuss potential applications to real-world systems.**

I. Introduction

Cryptographic hash functions form the core of many protocols which trust that cryptographic hash functions preserve certain basic properties. Cryptograpic hash functions see widespread use for file integrity checks to verify long term storage of data, as cache invalidation techniques, and as a building block in network protocols such as Kerberos and TLS. The classes of functions suitable for these use cases necessarily have strong guarantees about properties of its members. An ideal cryptographic hash has five main features: determinism - the same message always results in the same hash; performability - computing a hash for a message should not be algorithmically difficult; non-reversability - it should be infeasible to generate a message from its hash except by trying all possible messages; sensitivity to initial conditions - small changes to a message should change the hash value to a degree that the new hash value appears uncorrelated with the old hash value; unique digests - it should be infeasible to find two different messages with the same hash value.

The most important aspect of these can be formally defined under three properties which hash function implementations must have to be considered cryptographically secure in practice:

- **Preimage Resistance**: It should be computationally hard to find the inverse of a hash function.
- **Second Preimage Resistance**: Given an input block, it should be computationally hard to find a second block which hashes to the same value as the first block.

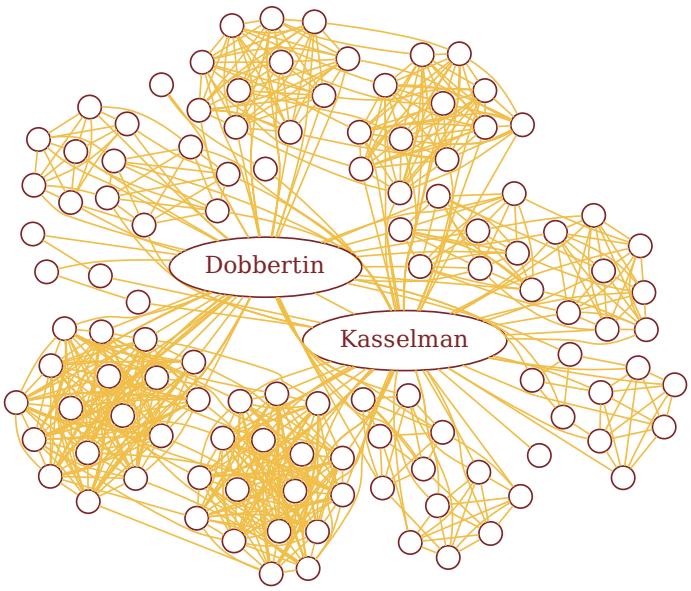


Fig. 1: A visualization of the heuristic unit-distance neighborhood of valid collisions from the Dobbertin and Kasselman collisions.

- **Collision Resistance**: It should be computationally hard to find two blocks which hash to the same value.

Note that while a second preimage is necessarily a collision, it is also a stronger result with higher use for an attacker.

From an attacker's perspective, finding a preimage or second preimage has been difficult: the current published bound for a preimage attack in MD5 is a cost of $2^{123.4}$ by Y. Sasaki and K. Aoki [1]. In MD4, the best second preimage attack by H. Yu et al. gives a bound of $\frac{1}{2^{56}}$ likelihood, but uses destructive message modification techniques to achieve collisions for all other messages [2]. Further, while J. Kelsey and B. Schneier proposed a method for finding second preimages in less than 2^n, we note that this requires rather long messages [3] which limits their utility to attackers. Collision attacks, however, are well within reach of adversaries, and have progressed past theoretical attacks into feasibility. The work of H. Dobbertin [4], X. Wang [5], M. Schläffer [6] and others demonstrate the ease with which collisions can be found.

From a cryptographic perspective, the existence of a feasible collision attack breaks the requisite properties, thus deprecating the function's use for trustworthy applications. In many practical instances, however, communities continue to use deprecated hash functions in a widespread fashion. One such example of widespread use post-deprecation is the continued use by Git of SHA-1, despite M. Stevens et al. publishing a full SHA-1 collision in February of 2017 [7]. This attack had a bound of $2^{63.1}$ — faster than the theoretical 2^{80} but still requiring significant compute resources to find.

We seek to address the problem of evaluating the use cases for a collision, to determine whether or not a collision affects a particular system. The central thesis of this research is that logical cryptanalysis allows us to formalize the structure inherent to the underlying hash function such that new collisions can be found within reasonable time on commodity hardware. We assert that when such structure exists, it is **imperitive to discontinue use of such hash functions as they are fundamentally untrustworthy for critical applications.** An example of such structure showing the unit distance neighborhood (defined in Section III) of the known collisions from Dobbertin and Kasselman is presented in Figure 1. This paper contributes several techniques to logical cryptanalysis and uses these techniques to derive metrics of the trustworthiness for hash functions. We claim the following results are novel in the field:

- We define the neighborhood of a class of collisions and show that it is frequently non-empty.
- We discuss techniques for finding useful collisions within a class using logical cryptanalysis.
- We propose additional techniques for building new classes of collisions and show that classes of collisions in MD4 can be found inductively.
- We propose five universal metrics for evaluating the utility of a class of collisions.

We propose that collision classes of high utility are closer to a second preimage attack in the amount of flexibility they provide to an attacker, whereas classes of low utility may not affect all systems which use a particular hash function.

The remainder of this paper is organized as follows. In Section II we discuss previous results and show how our research expands upon prior work. In Section III, we discuss the terminology and notations we use throughout the remainder of the paper. In Section IV, we give the intuition behind measuring the utility of a collision and motivate why it is useful. In Section V, we present new techniques in logical cryptanalysis to analyze collisions in hash functions. In Section VI, we discuss the utility of existing collisions under the metrics described in IV. In Section VII, we discuss new differential paths and the effectiveness of various techniques presented in Section V. Lastly, in Section VIII, we discuss the broader impacts and conclusions of this research, and conclude with Section IX discussing future work and extensions in Section X.

II. Related Work

This paper draws heavily upon the public cryptanalysis of MD4, the wide availability of high quality SAT solvers, and previous work in logical cryptanalysis. In particular, we look at ways that SAT solving can aid cryptanalysis instead of using cryptanalysis to benchmark SAT solvers. A critical assumption of our use of SAT solvers as a means to discover structures in hash functions which lead to untrustworthy behavior is the insight that while SAT is, in general, a hard problem for many specific and practical instances current SAT-solving technology is quite efficient [8]. We then apply them to aid the understanding of attacks against real world systems, and propose general metrics that target different types of systems. Towards this, we look at prior work in two main areas: differential and logical cryptanalysis.

A. On Differential Cryptanalysis

Differential cryptanalysis has been the major technique behind the discovery of collisions in cryptographic hash functions. Its importance can be seen everywhere from X. Wang's attacks on MD4 [9] to more recent cryptanalysis of general purpose hash functions such as Murmur3 by J. Aumasson, D. Bernstein, and M. Boßlet [10]. However, while differential cryptanalysis plays an important role, techniques for automated analysis of hash functions are still an emerging area. Further, techniques are being developed to make hash functions impervious to differential analysis.

Our work advances the current state of the art by removing the need for finding message modification techniques by specifying the differential path as part of the SAT formula and letting the solver find a pair of satisfying messages which follow the differential path and produce a collision. Further constraints can then be placed upon this model, such as a chosen prefix or desired start state. We believe performing these attacks by hand to be difficult, and note that developed message modification techniques may affect chosen prefixes. Thus, we seek to develop techniques to replace differential cryptanalysis with logical cryptanalysis.

B. On Logical Cryptanalysis

Logical cryptanalysis and its encoding as SAT likely started under the work of F. Massacci in 1999 with his paper titled "Using walk-SAT and rel-sat for cryptographic key search". [11]. However, much of the early work by F. Massacci was focused on symmetric and asymmetric ciphers [11]–[13]. It wasn't until the work of D. Jovanović and P. Janičić that exploring collisions in the context of SAT was introduced [14], and until the work of I. Mironov and L. Zhang that this was studied for an existing collision class [15].

Much of the work in this area is focused on the use of these problems to benchmark the performance of SAT solvers, and not to evaluate techniques for using SAT solvers to aid reasoning and understanding. E. Homsirikamol et al. show that SAT solvers can be used to brute force finding collisions in SHA-3, but with limited utility and limited to a small number of rounds [16]. On the other hand, advances by V. Nossum in

his master's thesis shows that additional work on encoding is also important: a new 32-bit modular addition circuit produced shorter run times [17].

We make use of the *bc2cnf* utility by T. Junttila as the basis for our models [18]. This allows us to encode hash functions as generic structures in the circuit description language and ease the algorithmic creation of models which build on top of them. We then use CryptoMiniSat5 by M. Soos to run the models and check for a satisfying witness, often an example collision [19], [20]. We deviate from evaluating the performance of SAT solvers or encodings of models by instead focusing on evaluating the properties of the hash function. We have developed new techniques for using a SAT solver to inductively build full collisions in MD4 from collisions in reduced-round MD4 and to build additional examples of collision classes from a single instance of a collision in MD4.

C. Trustworthy Computing

Hash functions see wide usage in modern trustworthy computing applications, where they are used to establish the authenticity and provenance of data, the credibility of image data [21], sensor identity and trust [22], meta-data validity [23], hardware authorization [24], and other measures of trust in system and mission critical systems. The use of deprecated hashes for commit verification in git [25], and for patch identity and trust verification [26], or segment verification in peer-to-peer decentralized networks [27] is quite common. The use of deprecated hashes poses a serious problem, and can compromise an otherwise trustworthy system or platform. A key component missing from the existing literature is a good metric which can be used to assess the trustworthiness of a hash, and its suitability for a given use case.

Such a metric needs to be one which allows for a variable which accounts for difficulty, so that trustworthiness can be assessed in a meaningful way in an environment with changing hardware capabilities.

III. Terminology & Notation

In order to define our define our metrics using logical cryptanalysis we first define a set of formal terminology and notation whn discussing our work, and the notation of collisions within a hash. We make the general assumption that ordered tuples and strings are indexable via square brackets. We utilize subscripts when referring to elements of tuples with named members, as opposed to just indexable members. When referring to the MD4 hash function, we use MD4 as specified in RFC 1320 [28].

We use the following terminology when discussing the notion of collisions in a hash function.

A *cryptographic hash function* is a function:

$$h : S_i \times B \rightarrow S_o$$

which satisfies the usual properties of cryptographic hash functions (preimage, second preimage, collision, deterministic, etc.). We denote the input state space as S_i, the input block space as B, and the output state space as S_o. Usually the input state space and the output state space are the same, so we omit the subscripts, S. We assume that these are sets of binary strings of a fixed length. In the case of MD4, S is all binary strings of length 128, and B is all binary strings of length 512. We assume that the hash function can be reduced to an arbitrary number of rounds, r. We note the hash function reduced to r rounds as h_r.

An *input* to a hash function is an ordered pair, $i = (s, b)$, where $s \in S$ and $b \in B$. The *input space* is simply the domain of the hash function, which we denote as $\mathcal{I}$. If $i \in \mathcal{I}$ is an input, we reference the state as i_s and the input block as i_b.

We use the term *intermediate state variables* to refer to intermediate steps in the computation of a hash function under a specific input. Typically these are the outputs of one-way compression functions as part of the Merkel-Damgård construct. If i is an input to a hash function, we use $I(i)$ to denote the intermediate state variables. Formally, this is represented as an ordered tuple of binary strings of the size of the updated state. In the case of MD4, this is an ordered tuple of cardinality r, where each index is a binary string of length 32, for a maximum of 1536 intermediate state variables. We denote the space that $I(i)$ maps into as $\mathcal{V}$.

A *collision*, $L \subseteq \mathcal{I}$ is a subset of the input space of a hash function with $\left|L\right| \geq 2$, such that there exists an output $v \in S$ such that for all inputs $i \in L$, $h(i) = c$. We extend this to arbitrary round hash functions using the notation (r, L), to explicitly denote that L is a collision under h_r.

- We define a *multicollision* to be a collision which has multiple input states which hash to the same output under a single input block. That is, if L is a collision, then $\forall i \in L$, $i_b = c$ for some $c\ inB$ for L to be a multicollision.

For a simple collision, we say that the *differential path* is the difference between the two sets of intermediate state variables. This is formally represented by an ordered tuple of r elements. We use the simple xor difference throughout this paper.

A *collision class*, C, is an arbitrary differential path. If there exists at least one collision pair which has that differential path, we call that collision class non-empty. We denote the set of all non-empty collision classes as $\mathcal{C}$. When we wish to convert between a collision, $c \in L$ and a collision class, $C \in \mathcal{C}$, we use the notation $C = \mathcal{C}(c)$. When c is a tuple, (r, L), we say that the collision class, C must exist over same round collisions, and thus C becomes a two-tuple, (r, d), where d is the ordered tuple denoting differential path.

A *family of collision classes*, F, is the indices of a collision class, C, which have any non-zero difference. That is, $F = \{0 \leq i \leq R : C[i] \neq 0\}$. When C again is over a single round, r, we extend F likewise, to become (r, F). We denote the set of all families of collision classes as $\mathcal{F}$.

A. Known Collision Classes

We define the following set of known collision classes.

- C_{Wang}, to be the collision class introduced by X. Wang et al. in [9].
- C_{Sasaki}, to be the collision class introduced by Y. Sasaki et al. in [29].

- $C_{Dobbertin}$, to be the collision class introduced by H. Dobbertin in [4].
- $C_{Kasselman}$, to be the collision class introduced by P. Kasselman in [30].
- $C_{Schlaffer}$, to be the collision class introduced by M. Schläffer in [6].

B. Distance Heuristics

We introduce a distance function, δ, between collision classes by the number of differences in intermediate rounds deltas. That is, given two collision classes, $C_1, C_2 \in \mathcal{C}$:

$$\delta(C_1, C_2) = \left|\{i : C_1[i] \neq C_2[i]\}\right| \tag{1}$$

We extend this distance function to include an *absolute* distance, measuring the number of intermediate rounds with non-zero differences. We notate this as $\delta(C_1)$.

We can define a similar distance function, Δ, between families of collision classes by cardinality of the symmetric difference in the two collision families. That is, given two families of collision classes, $F_1, F_2 \in \mathcal{F}$:

$$\Delta(F_1, F_2) = \left|(F_1 \cup F_2) \setminus (F_1 \cap F_2)\right| \tag{2}$$

This is convenient for when the specifics of the differential path do not matter, merely that there exists at least one collision class of the specified form.

We introduce this distance function as a heuristic measure of the sensitivity of the hash function to small changes in the input.

C. Neighborhoods

We define the neighborhood of a collision class, $C \in \mathcal{C}$, to be the set of all other non-empty collision classes at a fixed distance, $d \in \mathbb{N}$, from C. That is:

$$N(C, d) = \{C_j \in \mathcal{C} : \delta(C, C_j) = d\} \tag{3}$$

$$N(C) = N(C, 1) \tag{4}$$

For instance, $C_{Wang} \in N(C_{Sasaki}, 21)$. The distance parameter, d, may optionally be omitted, in which case the unit distance is implied. Thus, $C_{Kasselman} \in N(C_{Dobbertin})$.

We can similarly define the neighborhood of a family of collision classes, $F \in \mathcal{F}$, to be the set of all other families of collision classes at a fixed distance, $d \in \mathbb{N}$, from F. That is:

$$\mathcal{N}(F, d) = \{F_j \in \mathcal{F} : \delta(F, F_j) = d\} \tag{5}$$

$$\mathcal{N}(F) = N(F, 1) \tag{6}$$

The distance parameter, d, may optionally be omitted, in which case the unit distance is implied.

Neighborhoods can be classified into three types: *expansion*, *internal*, and *mixed*. Let d be fixed. An *expansion* neighborhood of a collision class, $C \in \mathcal{C}$, is the neighborhood restricted only to those collision classes which only differ in rounds external to the collision family of C. An *internal* neighborhood of C is the neighborhood restricted only to those collision classes which differ in rounds internal to the collision family of C. A *mixed* neighborhood of C is the neighborhood restricted only to collisions which differ in a round internal and a round external to the collision family of C. That is:

$$N_{exp}(C, d) = \{C_j \in N(C, d) : \forall i \in F(C), C_j[i] = C[i]\}$$

$$N_{int}(C, d) = \{C_j \in N(C, d) : \forall i \notin F(C), C_j[i] = C[i]\}$$

$$N_{mix}(C, d) = \{C_j \in N(C, d) : \exists i \in F(C), C_j[i] \neq C[i] \text{ and } \exists k \notin F(C), C_j[k] \neq C[k]\}$$

When the neighborhood defined by $\delta = 1$ is frequently non-empty we assert that sufficient structure exists in the underlying hash function such that logical cryptanalysis via 3-CNF-SAT becomes feasible. This condition is thus sufficient to render the underlying hash function **untrustworthy**. Conversely it is a necessary condition (but possibly insufficient) that a trustworthy hash function has an empty neighborhood defined by $\delta = 1$.

Furthermore this condition extends to $\delta = d$ for some small value of d as determined by computational resources such that d becomes a trustworthiness metric under constraints of current 3-CNF-SAT solving capabilities within reasonable time bounds.

IV. Flexibility and the Utility of a Collision Class

We seek to build new intuition about collision classes in hash functions and particularly collisions in MD4. By developing techniques using logical cryptanalysis, we complement and extend the results from differential cryptanalysis. In particular, we use logical cryptanalysis to discuss the impacts of collisions in real-world systems, and to give justification for general techniques, and the results of these techniques.

Real world systems are most vulnerable to two attacks: preimage and second preimage. For systems relying on password validation via comparing two hashes, preimage attacks would allow inverting the hash, increasing the risk associated with losing a database of hashed passwords. On the other hand, systems relying on message validation, such as signature checks or file integrity checks would be easily broken with fast second preimage techniques. Both of these attacks are theoretically intractable, thus protecting real-world systems. In practice, however, research efforts have been successfully devoted to the production of collision attacks, producing efficient results. These attacks have real consequences for trustworthy systems, and evaluating their impact and risk is an important and underserved area of the existing literature. We seek to evaluate trustworthiness by building a measure of the relative difference between a collision attack and second preimage attacks. The literature describes a number of successful high profile attacks against popular, and industrially relevant hash functions. For MD5, these include cloning a CA certificate [31] and creating malicious executables [32]. For SHA-1, a pair of PDFs with the same hash was successfully produced [7]. Many of these techniques, however, exploit not the hash but the flexibility of the file format relying instead on embedding arbitrary, collidable data and later checking for the presence of one of the matching blocks. This is especially

obvious in the MD5 executable and SHA-1 PDF attacks. The work by M. Stevens to perform the CA Collision, on the other hand, was a chosen prefix attack extending previous work from 2007 [33].

Evaluating flexibility becomes important as it serves as a contrapositive to trustworthiness. A highly flexible attack implies a larger set of families of collision classes, and decreasing the trust that should be placed in a hash function. To evaluate how much flexibility exists in a collision, we propose the following metrics for evaluating the utility of a collision class:

1) The number of unique differentials a collision class has.
2) The number of unit-step neighbors a collision class has.
3) The maximum count of zeros in a the binary representation of a colliding block (and likewise with ones).
4) Whether there exists a block which collides under multiple initial values.
5) Whether or not zero, one, or both of the blocks in a collision may be of ASCII values under any input block difference.

Note that the first three are quantitative measures providing some measure of flexibility of a collision class, whereas the latter two are merely looking for a single witness having such a property. Depending on the scenario, specific properties of a collision may be of more interest than others.

The first metric evaluates the flexibility of the differential path. A differential path with more flexibility will have more input-block differentials which produce blocks with the given differential path. More differentials implies a greater flexibility in choice of colliding block, and possibly allows for multiple collisions for a given colliding block. Furthermore, a collision with more differentials is more likely to satisfy the last metric, having a pair of blocks — both ASCII — which produce a collision.

The second metric evaluates the density of the neighborhood of a collision class. If a collision resides in a dense neighborhood, it provides more possible collision classes to search for a second preimage, chosen prefix, or other structure desired in a collision. If, however, a collision class has no neighbors, then it cannot be used to find other possible classes for other input blocks.

The maximum quantity of zeros (or ones) in the binary representation of a block serve as a measure of the extremes to which a collision can be pushed. This can additionally be extended to any suitable bit pattern in any base to provide a more relevant metric as desired by the system under study.

The fourth metric evaluates the utility of a collision when the internal state of the hash function is unknown. If a collision occurs under multiple initial values, this could be used to attack some systems where the input provided by a user is appended to unknown data and then hashed. If a block collides under a suitably large number of initial values, the attack becomes highly likely to occur successfully. Measuring the exact number of initial values a block collides under, however, is beyond the scope of this work.

TABLE I: Distance between Existing Collision Classes

	X. W.	Y. S.	P. K.	H. D.	M. S.	Absolute
X. Wang's	0	21	26	27	12	18
Y. Sasaki's		0	25	26	2	16
P. Kasselman's			0	1	25	11
H. Dobbertin's				0	26	12
M. Schläffer's					0	17

The final metric is similar to the third in that it looks for specific bit patterns in a collision. ASCII is one example of a widely used constraint system. Further examples, such as JSON, XML, etc., may likewise be supplemented based on the specifics of the system, and the domain of its application and implementation.

If a collision class satisfies many of these properties, then it is more flexible and thus more likely to be used to target deployed systems. If, however, a collision class does not satisfy these properties, its impact is likely severely limited in scope, and may not provide useful information to find other collision classes which have higher utility.

V. Techniques for Logical Cryptanalysis

The following techniques have been extensively tested on MD4 and partially tested on MD5 and are believed to apply fully to MD5. They likely apply to any hash function with similar structure, but as of yet, are untested.

A. Distance Heuristics

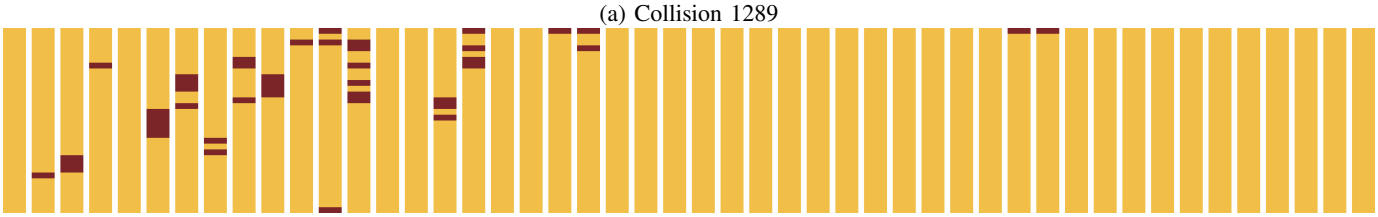
We find justification for the distance heuristic, δ, defined in Section III-B in the existing literature on MD4: H. Dobbertin's [4] and P. Kasselman's [30] collision classes have distance $\delta(C_{Dobbertin}, C_{Kasselman}) = 1$ under this metric. Since others have shown that MD4 is untrustworthy, we can limit our search space to small distances, exploiting the structure of MD4. This reduction in search space decreases the time to solve a single 3-CNF-SAT instance at the expense of restricting ourselves to highly connected, highly similar differential paths. Further, we can achieve parallel solving capabilities by noting that this problem is embarrassingly parallel: each candidate collision class can be independently checked on separate machines.

The highly connected, highly similar differential paths are highlighted in Figure 3: when the neighbors of H. Dobbertin's and P. Kasselman's collision classes from Figure 1 are viewed independently, the large cliques become apparent. Depending on the search strategies utilized, cliques may be expanded before new neighbors, or a significant portion of a vertices neighbors might belong to existing cliques, resulting in slow progress towards any external goal.

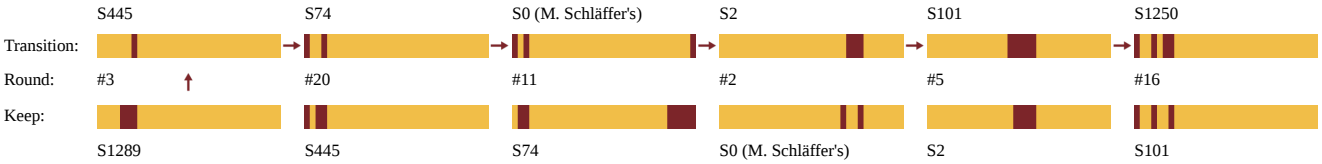
Refer to Table I for the distances between existing collisions in MD4.

We find justification for the distance heuristic Δ in the existing literature due to collisions by X. Wang [5] and M. Schläffer [6]. Their respective families are:

(a) Collision 1289



(b) Collision 1250



(c) Differential Path Transitions

$$F(C_{Wang}) = (1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, \mathbf{13}, 15, 16, 19, 20, 35, 36)$$
$$F(C_{Schlaffer}) = (1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 19, 20, 35, 36)$$

B. Family Similarity

We define a relation among families of collisions across rounds as follows. Let F_1 and F_2 be two different families of collisions. Then we say that F_1 and F_2 are *similar*, and notate it $F_1 \precsim F_2$, if $R(F_1) \leq R(F_2)$ and $F_1 \subseteq F_2$. Note that, when $R(F_1) = R(F_2)$, this is equivalent to saying that F_2 is in some expansion neighborhood of F_1. Further, when $R(F_1) < R(F_2)$ and $F_1 = F_2$, then we say that F_2 is the *trivial extension* of F_1.

We claim that the following statements are true for MD4:

1) For every $F \in \mathcal{F}$, there exists F' such that $F' \precsim F$ and $R(F') + 4 = R(F)$.
2) For every $F \in \mathcal{F}$, there exists F' such that $F \precsim F'$.

From an intuitive standpoint, what this says is that collisions have witnesses for their existence in reduced round versions of the function, and further, that it is sufficient to search reduced round versions to find candidates for extension into additional rounds. This likely derives from the iterative construction of MD4 and the particular choice of block schedule.

TODO – Empirical justification / graphical representation?

1) Collision Class Similarity: We claim that the aforementioned statements hold when considering individual collision classes instead of families of collision classes, but with less likelihood. That is, we can transplant an n round collision into an $n+4$ round and consider a neighborhood around it to find a new collision.

2) Second Preimage Similarity: If the models are constrained for a second preimage, we note that attack again works, but requires searching wider and wider differential paths in small round sizes. In particular, despite strong evidence for this pattern holding in $n \leq 24$ rounds, outside of the collision family $\{0, 16\}$ in both 28 and 32 round MD4– which does not extend any further–we have been unable to find any other differential path in those rounds by exhaustive brute force search up to a differential path of length 8. Further searches have been done of 36, 40, 44, and 48 rounds up to a differential path of length 5 with a similar lack of results. Thus we believe the general technique to hold, but the size of the search space for a second preimage becomes prohibitively large, and further, that differential paths become sufficiently long to prohibit easy finding of second preimages.

3) Chosen Prefix Similarity: Unlike second preimage attacks, chosen prefix attacks have a higher likelihood of working, and give better results. However, we have found that most chosen prefixes work in the neighborhoods of existing collisions, and thus are not convinced that inductively building a chosen prefix collision is a useful technique by itself.

C. Unjustified Performance Observations

We share the general tips for working with models of MD4 from a SAT solver performance perspective, without justification for results. Note that these results may be limited in usefulness and may not be uniformly applicable.

- Unless searching for a second preimage or chosen prefix with sufficient number of bits (≥ 128), we find models perform better leaving the initial state unconstrained.
- Adding ASCII block constraints significantly and negatively impact the performance of finding a collision.

TABLE II: Number of Differentials for Existing Collision Classes

Attack	Size	Attack	Size
X. Wang's	64	Y. Sasaki's	4
H. Dobbertin's	32	P. Kasselman's	32
M. Schläffer's	64		

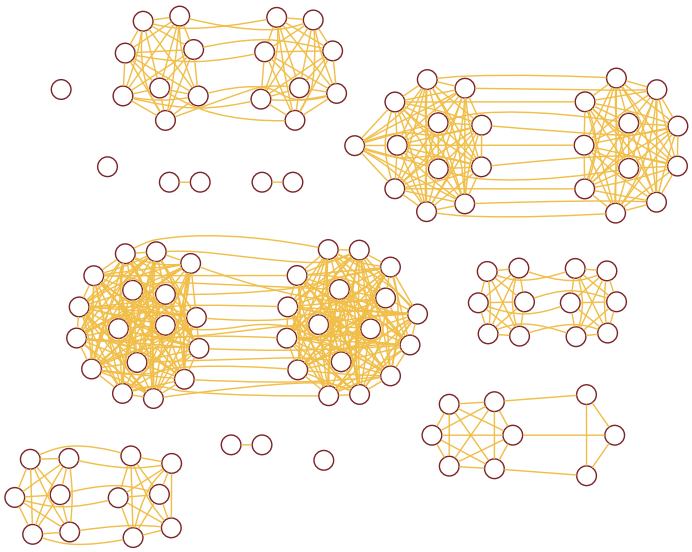


Fig. 3

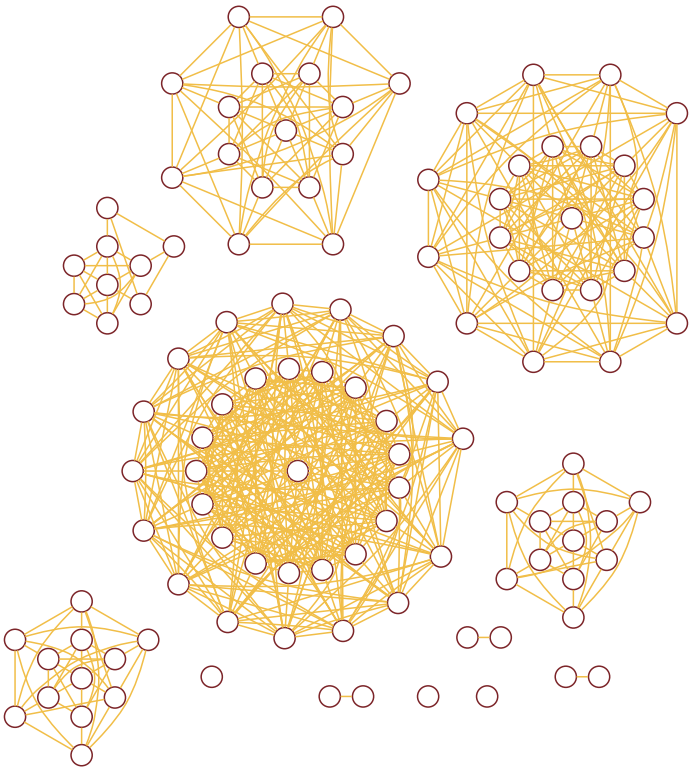


Fig. 4

- Keeping a small search radius on neighborhoods involving families of collision classes is important for good performance. However, once sufficient witnesses have been found across rounds, missing families within the round can be found by various filling-in techniques through applying the $\precsim$ relation.

VI. Empirical Results: The Utility of Existing Collisions

TODO – Introduction?

A. Unique Differentials

Table II states the number of unique differentials, including the specified one, which produces a differential path conforming to each of the collision classes. We note in particular that while Y. Sasaki's attack is generally an improvement upon X. Wang's, Y. Sasaki's attack has fewer input block differentials and thus less flexibility in this aspect. Further, M. Schläffer's attack – also building off the work of X. Wang – has the same number of differentials and is thus also more flexible than Y. Sasaki's.

B. Unit-Step Neighborhood

Refer to Table III for the sizes of neighborhoods of existing collisions. In particular, note that both Y. Sasaki's attack and M. Schläffer's attack – which both claimed to improve upon X. Wang's attack–have larger neighborhood sizes, and likewise with P. Kasselman's attack which improved upon H. Dobbertin's attack. Thus, the newer collisions occur in more dense subsets of the collision class space.

TABLE III: Neighborhood Sizes for Existing Collision Classes

Attack	Size	Attack	Size
X. Wang's	54	Y. Sasaki's	157
H. Dobbertin's	55	P. Kasselman's	60
M. Schläffer's	100		

C. Zeroes

See table IV for the maximum count of zeroes in a block for existing collisions. Note that X. Wang's attack is the leader, with 509 maximum zero bits in the collision pair. Further, while Y. Sasaki improved the bound in other areas, this was not one of them. Of note too, H. Dobbertin's and P. Kasselman's collisions are essentially identical in this metric.

D. Multicollisions

We note that while these collisions are state independent and include multicollisions, none of these attacks are always multicollisions. Many of the blocks we've found only collide under the initial state value. Because of this, these collisions are only useful if you know the current state where a colliding pair is to be injected. Thus, this is one area in which collisions could potentially improve.

TABLE IV: Number of Zeroes in a Colliding Block

Attack	Count	Attack	Count
X. Wang's	509	Y. Sasaki's	494
H. Dobbertin's	504	P. Kasselman's	504
M. Schläffer's	506		

TABLE V: ASCII Blocks in Existing Collision Classes

Attack	Single Block	Both Blocks
X. Wang's	true	false
Y. Sasaki's	true	false
P. Kasselman's	true	**true**
H. Dobbertin's	true	**true**
M. Schläffer's	true	false

E. ASCII Blocks

See Table V for results of finding ASCII block pairs. Note that this table validates P. Kasselman's and H. Dobbertin's results of finding two ASCII block messages. However, these results do not hold for later collisions by X. Wang, Y. Sasaki, or M. Schläffer. Thus, while the latter attacks have been viewed as being of better quality, under this particular metric, P. Kasselman's and H. Dobbertin's collision classes are better.

F. Recapitulation

Overall, we feel that while the given collision classes have value and utility, they do not constitute full second preimage attacks. Further, simple neighborhood searches do not lead to collisions in every case. Therefore, additional techniques are necessary for expanding the collision neighborhood.

VII. New Collisions

Through the exploration of the neighborhood technique, we have found over 35,000 new differential paths which originated from the five original collisions in MD4. This shows that the neighborhood definition is sufficient to produce a large quantity of new collision classes. Further, we give evidence for propagating neighborhoods across reduced round versions of MD4.

A. Extensions of Prior Work

We noted in table III that the unit-step neighborhood of the existing attacks were non-empty. By repeatedly expanding neighborhoods of known collision classes, we can construct paths in the collision space. We made the following observations:

1) The space of collision classes is not smooth under the unit distance relation. That is, starting at $C_{Schlaffer}$ and evaluating successive neighborhoods, moving towards C_{Wang} whenever the distance decreases does not lead to a simple path of length $\delta(C_{Schlaffer}, C_{Wang}) = 17$. Our lower bound on the actual distance is 29, but after 14 rounds of expansion, we only reduced the distance to 15.
2) Even under successive unit neighborhoods, the family of input block differentials remains roughly the same among successive neighborhoods. This can be partially counteracted by moving to reduced-round spaces, evaluating the neighborhood, and expanding back to a collision in full-round MD4. That is, there is a detectable signature of sharing significant input block differential structure with the original collision. With 35,918 unique classes of collisions which had 176 unique families of collisions, there were only three unique families of input block differentials: those of the starting collision classes.

This last point shows that, while there may be a number of unique differentials for a given collision class, and potentially many collision classes within some neighborhood of the original collision class, by analyzing the structure of the input block differential, systems can potentially prohibit successful collision attacks by preventing structures of input block differences.

TODO - figures and graphics on dataset, cross round, etc.

VIII. Discussion

TODO - Recap impacts on real-world systems like MD5, SHA-1, etc.

IX. Conclusion

TODO: COMPLETE

X. Future Work

Moving forward, we seek to solidify the utility metrics by improving upon second preimage and chosen preimage bounds to validate assumptions about the utility of collisions. Further, we seek to apply these results to other hash functions in the same family, such as MD5, SHA-1, and SHA-2. Lastly, it remains to be seen how this work impacts dissimilar constructions such as SHA-3 and key derivation functions such as bcrypt and scrypt, and whether logical cryptanalysis techniques can be applied to these categories of hash functions as well.

This research and all associated code are open access and open source. The code is hosted on GitHub at https://github.com/XXXXXXXXXXXXXXXXX. For additional details or to request data associated with the project, please contact the authors.

Acknowledgments

The authors would like to thank the Department of Electrical and Computer Engineering and XXXXX XXXXX for providing hardware.

References

[1] Y. Sasaki and K. Aoki, *Finding Preimages in Full MD5 Faster Than Exhaustive Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 134–152, https://www.iacr.org/archive/eurocrypt2009/54790136/54790136.pdf.

[2] P.-A. Fouque, G. Leurent, and P. Nguyen, "Automatic search of differential path in md4," Cryptology ePrint Archive, Report 2007/206, 2007, http://eprint.iacr.org/2007/206.

[3] J. Kelsey and B. Schneier, *Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 474–490, https://www.schneier.com/academic/paperfiles/paper-preimages.pdf. [Online]. Available: https://doi.org/10.1007/11426639_28

[4] H. Dobbertin, "Cryptanalysis of md4," *Journal of Cryptology*, vol. 11, no. 4, pp. 253–271, Sep 1998, https://link.springer.com/content/pdf/10.1007/s001459900047.pdf. [Online]. Available: https://doi.org/10.1007/s001459900047

[5] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions md4, md5, haval-128 and ripemd," Cryptology ePrint Archive, Report 2004/199, 2004, http://eprint.iacr.org/2004/199.

[6] M. Schläffer and E. Oswald, *Searching for Differential Paths in MD4*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 242–261, https://link.springer.com/content/pdf/10.1007%2F11799313_16.pdf. [Online]. Available: https://doi.org/10.1007/11799313_16

[7] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1," Cryptology ePrint Archive, Report 2017/190, 2017, http://eprint.iacr.org/2017/190.

[8] M. Y. Vardi, "on p, np, and computational complexity," *Communications of the ACM*, vol. 53, no. 11, pp. 5–5, 2010.

[9] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–18, http://www.infosec.sdu.edu.cn/uploadfile/papers/Cryptanalysis%20of%20the%20Hash%20Functions%20MD4%20and%20RIPEMD.pdf. [Online]. Available: https://doi.org/10.1007/11426639_1

[10] D. J. B. Jean-Philippe Aumasson and M. Boßlet, "Hash-flooding dos reloaded: attacks and defenses," Presentation at 29th Chaos Communications Congress, 2004, https://131002.net/siphash/siphashdos_29c3_slides.pdf.

[11] F. Massacci, "Using walk-sat and rel-sat for cryptographic key search," in *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, ser. IJCAI'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 290–295. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624218.1624261

[12] F. MASSACCI, L. MARRARO, and L. MARRARO, "Logical cryptanalysis as a sat problem: Encoding and analysis of the u.s. data encryption standard," 2000, http://www.ing.unitn.it/~massacci/papers/mass-marr-00-JAR.pdf.

[13] C. Fiorini, E. Martinelli, and F. Massacci, "How to fake an rsa signature by encoding modular root finding as a sat problem," *Discrete Applied Mathematics*, vol. 130, no. 2, pp. 101 – 127, 2003, http://www.ing.unitn.it/~massacci/papers/fior-mart-mass-03-DAM.pdf. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X02004006

[14] D. Jovanović and P. Janičić, *Logical Analysis of Hash Functions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 200–215. [Online]. Available: http://csl.sri.com/users/dejan/papers/jovanovic-hashsat-2005.pdf

[15] I. Mironov and L. Zhang, *Applications of SAT Solvers to Cryptanalysis of Hash Functions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 102–115, https://eprint.iacr.org/2006/254.pdf. [Online]. Available: https://doi.org/10.1007/11814948_13

[16] E. Homsirikamol, P. Morawiecki, M. Rogawski, and M. Srebrny, *Security Margin Evaluation of SHA-3 Contest Finalists through SAT-Based Attacks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 56–67. [Online]. Available: https://eprint.iacr.org/2012/421.pdf

[17] V. Nossum, "Sat-based preimage attacks on sha-1," Master's thesis, University of Oslo, 2012. [Online]. Available: https://www.duo.uio.no/handle/10852/34912

[18] T. Junttila, "Tools for constrained boolean circuits," https://users.ics.aalto.fi/tjunttil/circuits/.

[19] M. Soos, K. Nohl, and C. Castelluccia, "Extending sat solvers to cryptographic problems," in *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, ser. SAT '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 244–257, https://www.cs.virginia.edu/~kn5f/pdf/Extending_SAT_2009.pdf.

[20] M. Soos, "Cryptominisat sat solver," GitHub, 2017. [Online]. Available: https://github.com/msoos/cryptominisat

[21] G. L. Friedman, "The trustworthy digital camera: Restoring credibility to the photographic image," *IEEE Transactions on consumer electronics*, vol. 39, no. 4, pp. 905–910, 1993.

[22] K. H. Wong, Y. Zheng, J. Cao, and S. Wang, "A dynamic user authentication scheme for wireless sensor networks," in *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, vol. 1. IEEE, 2006, pp. 8–pp.

[23] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," *Sequences II: Methods in Communication, Security and Computer Science*, pp. 329–334, 1993.

[24] E. G. Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider, "Logical attestation: an authorization architecture for trustworthy computing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 249–264.

[25] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.

[26] U. Kühn, K. Kursawe, S. Lucks, A.-R. Sadeghi, and C. Stüble, "Secure data management in trusted computing," in *CHES*. Springer, 2005, pp. 324–338.

[27] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[28] R. Rivest and I. RSA Data Security, "The MD4 Message-Digest Algorithm," Internet Requests for Comments, IETF, RFC 1320, April 1992, https://tools.ietf.org/html/rfc1320.

[29] Y. Sasaki, L. Wang, K. Ohta, and N. Kunihiro, *New Message Difference for MD4*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 329–348, https://link.springer.com/content/pdf/10.1007/978-3-540-74619-5_21.pdf. [Online]. Available: https://doi.org/10.1007/978-3-540-74619-5_21

[30] P. R. Kasselman, "A fast attack on the md4 hash function," in *Communications and Signal Processing, 1997. COMSIG '97., Proceedings of the 1997 South African Symposium on*, Sep 1997, pp. 147–150, http://www.mathcs.emory.edu/~whalen/Hash/Hash_Articles/IEEE/A%20fast%20attack%20on%20the%20MD4%20hash%20function.pdf.

[31] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik, and B. de Weger, *Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 55–69. [Online]. Available: http://deweger.xs4all.nl/papers/[41]StSoApLeMoOsdW-RogueCA-Crypto[2009].pdf

[32] P. Selinger, "Md5 collision demo," http://www.mathstat.dal.ca/~selinger/md5collision.

[33] M. Stevens, A. Lenstra, and B. de Weger, *Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–22. [Online]. Available: https://doi.org/10.1007/978-3-540-72540-4_1