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. With widespread use of hash functions as 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, this class of functions necessarily has strong guarantees about properties of its members. There are three properties hash functions must have to be considered cryptographically secure:

- **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.
- **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]. 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 [2]. However, collision attacks are well within reach of adversaries, and well past theoretical attacks. The work of H. Dobbertin [3], X. Wang [4], M. Schläffer [5] 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. In some instances however, there continues to be widespread use of deprecated hash functions. One widespread example is the continued use by Git of SHA-1, despite M. Stevens et al. publishing a full SHA-1 collision in February of 2017 [6]. However, 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. This paper contributes several techniques to logical cryptanalysis and uses these techniques to derive metrics of the trustworthiness of 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 of collisions.
- We propose additional techniques for building new classes of collisions and show that 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.

TODO - Finish Organization The remainder of this paper is organized as fol lows. 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 between 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.

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. 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 [7] to more recent cryptanalysis of general purpose hash functions such as Murmur3 by J. Aumasson, D. Bernstein, and M. Boßlet [8]. 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 removes 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". [9]. However, much of the early work by F. Massacci was focused on symmetric and asymmetric ciphers [9]–[11]. It wasn't until the work of D. Jovanović and P. Janičić that exploring collisions in the context of SAT was introduced [12], and until the work of I. Mironov and L. Zhang that this was studied for an existing collision class [13].

However, much of the work in this area is focused on benchmarking SAT solvers, and not on evaluating 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 [14]. 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 [15].

We make use of the *bc2cnf* utility by T. Junttila as the basis for our models [16]. 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 [17]. 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 addition examples of collision classes from a single instance of a collision in MD4.

C. On Trustworthy Computing

TODO - Assigned @erozier

III. Terminology & Notation

A. Conventions

- We assume ordered tuples and strings are indexable via square brackets. Occasionally we use subscripts for referring to elements of tuples with named members.

TODO

B. Collisions in Hash Functions

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

- A *cryptographic hash function* is a function:
  $$h : S_i \times B \to S_o$$
  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.
- 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 48, where each index is a binary string of length 32, for a total of 1536 intermediate state variables. We denote the space that $I(i)$ maps into as $\mathcal{V}$. We define the number of rounds as R.
- We use the notation $R(I)$
- 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 define a *simple collision* to be any collision with cardinality exactly two.
  - We define a *strong collision* to be a collision which has multiple blocks which hash to the same output

under the same input state. That is, if L is a collision, then $\forall i \in L$, $i_s = c$ for some $c \in S$ for L to be a strong collision.
  - 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. **TODO – HELP define formal notation?**
  - Signed - Wang's, we don't use.
  - Unsigned - more common, actually used by us.
- 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 $C = \mathcal{C}(c)$.
- A *family of collision classes*, F, is the indices of a collision class, C, which have any non-zero difference. That is, $F = \{0 \leq i \leq R : C[i] \neq 0\}$.

C. 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 [7].
- C_{Sasaki}, to be the collision class introduced by Y. Sasaki et al. in [18].
- $C_{Dobbertin}$, to be the collision class introduced by H. Dobbertin in [3].
- $C_{Kasselman}$, to be the collision class introduced by P. Kasselman in [19].
- $C_{Schlaffer}$, to be the collision class introduced by M. Schläffer in [5].

D. Distance Metrics

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 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 exist at least one collision class of the specified form.

E. Neighborhoods

We define the neighborhood of a collision class, $C \in \mathcal{C}$, to be the set of all other collisions 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. An *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:

$$\begin{aligned} N_{exp}(C, d) &= \{C_j \in N(C, d) : \forall i \in F(C), C_j[i] = C[i]\} \\ N_{int}(C, d) &= \{C_j \in N(C, d) : \forall i \notin F(C), C_j[i] = C[i]\} \\ N_{mix}(C, d) &= \{C_j \in N(C, d) : \exists i \in F(C), C_j[i] \neq C[i] \\ &\qquad \text{and } \exists k \notin F(C), C_j[k] \neq C[k]\} \end{aligned}$$

IV. Intuition on the Utility of a Collision

TODO - audit introductory paragraphs

We seek to build new intuition about collisions in hash functions, specifically, collisions in MD4. By developing techniques from logical cryptanalysis, we seek to complement and extend the results from differential cryptanalysis. In particular, we wish to use logical cryptanalysis to discuss the impacts of collisions in real-world systems. In this paper, we introduce and give justification for general techniques, and present some results of these techniques.

To begin, real world systems are most vulnerable to two attacks: the preimage and second preimage. For systems relying on e.g., 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 intractable, thus protecting real-world systems. However, much research has been devoted to collision attacks, producing efficient results.

Thus, we seek to evaluate the relative difference between a collision attack and second preimage attacks, to give some measure of distance between them. Several high profile attacks against hash function usage have been made, including **TODO – cite**, with MD5, cloning a CA certificate and creating malicious executables, and with SHA-1, a pair of PDFs with the same hash. However, many of these techniques 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. In some cases, such as trying to produce a rogue CA certificate that matches an existing root, this is not possible and thus relies on searching a wide enough set of objects for a possible match to the collision.

Seeing the need 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 for having the 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 differentials which produce blocks with the given differential path. More differentials implies a greater flexibility in choice of colliding block, and possibly allowing 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 user provided input 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. However, measuring the exact number of initial values a block collides under 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 last 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.

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, it 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 believe to apply fully to MD5. They may or may not apply to any later hash function, such as SHA-1, SHA-2, or SHA-3, and have not been tested yet.

In the following sections, we use the collisions of X. Wang [7], Y. Sasaki [18], M. Schläffer [5], H. Dobbertin [3], and P. Kasselman [19] for examples.

A. Distance Metrics

We find justification for this metric in the existing literature on MD4: H. Dobbertin's [3] and P. Kasselman's [19] collision classes have distance 1 under this metric. We extend this distance function to include an *absolute* distance, measuring the number of intermediate rounds with non-zero differences. We denote this as $\delta(C_1)$.

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

B. Family Similarity

We define a relation among families of collisions across rounds. 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 $F_1 \lesssim F_2$, 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:

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		

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		

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

TODO - Describe justification? Later sections?

C. Class Similarity

We claim that the aforementioned statements hold when considering individual collision classes instead of families of collision classes, but with less likelihood.

TODO - Describe technique applied neighborhood extensions

Give some reasoning for why it works, example graphs.

D. Miscellaneous

TODO - Describe chosen prefix, second preimage, ASCII, and other tips and tricks for faster model runs.

VI. EMPIRICAL RESULTS

A. Unique Differentials

TODO Refer to Table II for the number of differential paths.

B. Unit-Step Neighborhood

TODO

Refer to Table III for the sizes of neighborhoods of existing collisions. In particular, note that Y. Sasaki's attack–which claimed to improve upon X. Wang's attack–has a larger neighborhood size, and likewise with P. Kasselman's attack which improved upon H. Dobbertin's attack.

C. Zeroes & Ones

TODO

Separate table for known attacks as data points. Examples as appendix.

D. Multicollisions

TODO

Include table detailing how many have multicollisions and how many don't. Hypothesis: all do since all existing datapoints do.

Separate table for known attacks as data points. Examples as appendix.

TABLE IV
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

TODO

Include table detailing how many have ASCII blocks and how many don't.

See Table IV for results. Note that the validation of P. Kasselman's and H. Dobbertin's results do not hold for the latter attacks 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.

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, and show a new collision in MD4 with new differential path.

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. 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 show 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

B. New Differentials

We present the following new differentials as part of our work:

TODO - figures and graphics on new differentials/differential paths

VIII. ASCII Collisions

A. Dobbertin

Default IV

Block 1:

3f474422 624f2b46 683b6c7e 48227d5c 4f2a4f61 206c573e 61622234 227a2633 53482622 706a6727 503e2b3e 7a636c76 7**b**752620 3f2e5728 236a6376 45654a75

Block 2:

3f474422 624f2b46 683b6c7e 48227d5c 4f2a4f61 206c573e 61622234 227a2633 53482622 706a6727 503e2b3e 7a636c76 7**c**752620 3f2e5728 236a6376 45654a75

B. Kasselman

Default IV

Block 1:

7e273957 597e7f4a 7c773d7c 4034376f 69643257 515f5269 6a4d665a 29416b3a 7f377a64 7e232455 6d436c56 26717321 3**6**65774f 6f2f4421 77787656 26236479

7e273957 597e7f4a 7c773d7c 4034376f 69643257 515f5269 6a4d665a 29416b3a 7f377a64 7e232455 6d436c56 26717321 3**7**65774f 6f2f4421 77787656 26236479

IX. Discussion of Impacts

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

X. Conclusion

The conclusion goes here.

XI. Future Work

The framework can be seen here **TODO**. Complete data set available upon request.

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] 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

[3] 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

[4] 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.

[5] 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

[6] 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.

[7] 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

[8] 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.

[9] 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

[10] 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.

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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

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

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

[18] 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

[19] 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.