Motivating a Constraint-Based Approach to Colliding Hash Functions

Alexander Scheel
alexander.m.scheel@gmail.com
Iowa State University

Eric Rozier
erozier@iastate.edu
Iowa State University

Abstract

TODO

1 Introduction

1.1 Brief History of SAT in Cryptography

Much of the original research for developing the use of SAT with cryptography was done by Fabio Massacci currently at the University of Trento [4]. The focus of this work was largely two fold: cryptoanalysis against the DES and faking signatures in the RSA with $e = 3$. More information can be found here [5] and here [1], but note that this work predates all of the collisions discussed in this article.

- CryptoSAT
- cryptominisat
- SHA-1

1.2 Motivating Logical Cryptanalysis on Hash Functions

We wish to provide the motivations driving our work. To begin, while collisions in cryptographic hash functions have a wide reach beyond the academic community, most of the discovered collisions are driven by academics. As such, most papers begin by introducing a single hash function, the author's preferred notation, the methods used in whole or part used to discover the collision, the constraints on the differential and the differential path, and usually end with an example of a found collision. However, while community's goal is to improve techniques by better differentials as measured by higher likelihoods of occurrence, little work is done to determine the utility of the collision. We seek to push the community in a new direction, towards measuring, and later improving, the utility of collisions.

Thus we seek to answer the following questions: What does the input space of a collision look like? Are blocks only of high entropy, likely reflecting complex dependencies on bits of the input? Or, can fixed strings be required at positions of an input? Can these blocks be constrained to only inputs of ASCII or other character sets? Is the collision fixed to a single input differential, or can many different input differentials be used? Are there constraints on possible starting states?

By answering these questions, we begin to develop a sense of utility of collisions. We can see whether they work well at the beginning or end of longer messages; we can see how much flexibility we have to fit existing data (towards second-preimage attacks matching existing collisions); and, we can see how much structure we can enforce for members of a collision. However, what we cannot answer are the relationships between these facets of utility, and whether there are more facets we are overlooking.

Ultimately, we seek to develop a framework for pushing collisions from an academic exercise to a practical exercise in a constrained environment. The use is two fold: to develop better hash functions with highly-fractured collision spaces, and to allow users of hash functions to understand the weaknesses better. Our belief is that, while there should be an abundance of possible collisions, hash functions can be designed so that possible collisions do not pose security risks in all use cases. Further, we seek to demonstrate the usefulness of discussing collisions in a common language of SAT. This will allow properties of collisions to be easily verified, and find collision examples targeting a specific use cases.

We leave it as an open challenge to find a set of collisions which lie outside of our proposed methodology but which can be demonstrated to have high utility.

1.3 Terminology of Utility

We seek to extend the literature about the utility of collisions and thus propose the following new terminology.

The **constraints** of a collision are the necessary and/or sufficient boolean formulas for generating a collision, typically as a function of the state blocks.

A **state transition constraint** is a constraint on the intermediate state variables, i.e., round variables. A constraint is **weak** if it is a any difference, and a constraint is **strong** if it is a signed difference. A equality constraint can either be weak or strong. This is roughly equivalent to a differential path.

Input block constraints, **input state constraints**, and **output state constraints** are all analogously defined as state transition constraints.

A **property** of a collision is any constraint placed upon the input block which exists independent from the constraints of a collision.

An attack is **constraint-based** if there exists a SAT encoding of the attack which can generate multiple collisions.

A collision is **extensible** if one example of a collision will lead to other examples under the same differential and weak state transition constraints. Note that if a collision is extensible, it is also constraint-based.

A collision is **differential-independent** if there are multiple differentials which satisfy the state transition constraints.

A collision is **malleable** if collisions can be found with different properties such as ASCII or JSON input blocks, or with fixed internal text.

A collision is **state-independent** if there exist alternative starting states which produce collisions. A state-independent collision is termed **strong** if one block can collide under two or more starting states.

Thus, a collision is said to have **high utility** if it is differential-independent, malleable, and state-independent.

1.4 Common Notation for Constraint-Based Collisions

A SAT model, $S = (E, A)$, is an ordered tuple consisting of a set, E, of boolean equations of the form $var := expression$, and a set, A, of expressions which must be asserted to true. It should be obvious that this is equivalent to $3-CNF-SAT$. This notation is used as it is most similar to the "circuits" language [3]; further, unline $3-CNF-SAT$, this notation is composable in the sense that adding new constraints does not impact other existing constraints.

We now introduce a series of notations to ease construction of cryptographic SAT models.

We define

$$c_l^{l+k-j} := EQ_{i=j}^{k}(a_i, b_i)$$

for $j \geq 0, k \geq j, l \geq 0$ to be equivalent to:

$$\begin{aligned} \{c_l &:= a_j == b_j, \\ c_{l+1} &:= a_{j+1} == b_{j+1}, \\ &\ldots, \\ c_{l+k-j} &:= a_k == b_k\} \end{aligned}$$

In particular, c is a tuple of boolean equations with cardinality $|c| = k - j$, defining an equality relation between input sets a and b, at index starting at j and ending at k (inclusive).

Discuss encodings of hash functions? No, leave to section in

Use notation in other sections. Use Overview subsection to discuss the particular quirks of encoding of hash function into SAT.

2 MD4

2.1 Brief History and Overview

The MD4 hashing algorithm was developed by Ron Rivest and first released as RFC 1186 in 1990 [6]. This was later superseded by RFC 1320 in 1992 because the reference implementation was "more portable" [7]. As early as 2004, Wang et al, demonstrated the first full collision in MD4 and MD5 [10]. However, it wasn't until 2005 that details of the differential and constraints were published [11]. It is believed by some that these differentials and constraints were found "by hand" [2]. Others thus strove to automate these attacks with algorithmic approaches [9] [2], and [8].

In this section, we will analyze Wang's 2005 and Sasaki's 2007 differentials to MD4.

2.2 SAT Model of MD4

Base model.

2.3 Wang's MD4 Collisions

2.3.1 Overview and Motivation

While many have studied Wang's attack from the point of view of improving the probability of collision or validating constraints, we seek to extend Wang's work by showing properties about colliding blocks. In particular, we wish to show that Wang's attack is constraint-based, extensible, highly malleable, differential-independent, and

state-independent. Thus, we venture that Wang's attack is a strong attack against MD4, and that MD4 has no remaining collision resistance.

2.3.2 Extensible

We note that Wang's attack is extensible. With a loose fitting on the original example as referenced by 'md4-wang-extensible', we generated an additional 40,000 new collisions. We note that, while not fully explored, the loose fitting on the original hash produces valid collisions satisfying the closer reconstruction of the original constraints; that is, no further differential paths were discovered. By analyzing the new collisions for fixed properties, an alternative construction of Wang's constraints can be created. Further, by then negating individual constraints and testing for UNSAT, we can validate that they are indeed necessary. See table 1 for the set of differential path constraints and table 2 for the set of fixed-value constraints. Together these form an equivalent constraint set as what is seen in [11].

2.3.3 Constraint-Based

- Simple encoding
- Advanced constraints
- Time difference

testing

2.3.4 Differential-Independent

2.3.5 State-Independent

h1: f7f3fe7437e33bee3590392936aeab37e0b848329173bf0f08a252a0281e3284f4e4fd90626c4cbb3350ac3e36694d34703d31498806e5
h2: f7f3fe7437e33b6e3590399936aeab37e0b848329173bf0f08a252a0281e3284f4e4fd90626c4cbb3350ac3e36694d34703d30498806e5
h3: f7f3fe7437e33bee3590392936aeab37e0b848329173bf0f08a252a0281e3284f4e4fd90626c4cbb3350ac3e36694d34703d31498806e5
h4: f7f3fe7437e33b6e3590399936aeab37e0b848329173bf0f08a252a0281e3284f4e4fd90626c4cbb3350ac3e36694d34703d30498806e5
c Total time : 127.12

2.3.6 Malleable

- ASCII
- JSON
- substring

Testing

2.3.7 Additional Notes

2.4 Sasaki's MD4 Collisions

2.4.1 Overview and Motivation

We study Sasaki's collisions as a means of analyzing a more complete attack, and as a point of comparison to Wang's [8]. In particular, Sasaki boasts of substantial speed improvement over Wang's and an increased probability of collision. The latter should thus translate to improved results with respect to malleability.

2.4.2 Extensible

We note that Sasaki's MD4 Collisions is an extensible collision. With a loose fitting on the original example as referenced by 'md4-sasaki-extensible', we generated an additional 40,000 new collisions. Further, the constraints published on page 16, while sufficient, are not necessary; many of the negated forms have valid collisions as well [8].

2.4.3 Differential-Independent

We note that Sasaki's MD4 collisions are differential independent, finding examples of all 64 theoretical differentials.

2.4.4 State Independent

h1: 13d340f35ff2c0ace19bcd73f5c153aa816105701b5050a2c7e0ab260adf
h2: 13d340035ff2c0ace19bcdf3f5c153aa816105f01b5050a2c7e0ab260adf
h3: 13d340f35ff2c0ace19bcd73f5c153aa816105701b5050a2c7e0ab260adf
h4: 13d340035ff2c0ace19bcdf3f5c153aa816105f01b5050a2c7e0ab260adf
c Total time : 165.32

3 MD5

3.1 Brief History and Overview

3.2 Wang's MD5 Collisions

3.3 Stevens's MD5 Collisions

3.4 Xie's MD5 Collisions

3.5 Sasaki's MD5 Collisions

4 SHA1

4.1 Brief History and Overview

4.2 Wang's SHA1 Partial Collision

4.3 Stevens's SHA1 Full Collision

5 Collisions

5.1 Notation

We propose the convention of using both a hexadecimal and binary representation. Hex in a ”standard order”, i.e.,

```
echo ``<hex>'' | xxd -r -p | openssl
    ↪ md4
```

5.2 MD4

5.2.1 Wang's Collisions

6 Tables

6.1 Notation

Tabular notation notes go here. Meaning of +, -, T, F, and *. Note that missing numbers are replaced with ... when all the same.

6.2 MD4

6.2.1 Wang's Collision

hli	constraints
0	.....
1	.....+.....
2	.....+.-.....
3	.....+.....
4	.....
5	.....+.....
6	.....+-++.....
7	.....+--.....
8	.....+.....
9	.....-..+--+.....
10	..-.....
11	+.....
12	.....+..+.....
13	..+-.-.....
14	.....
15	.....+.....
16	-.-.+-.....
17	.....
18	.....
19	+.-.....
20	-.+-.....
21	.....
...	.....
47	.....

Table 1: Discovered differential path constraints in MD4 under Wang's Attack

hli	constraints
0	.....F.....
1	.....F.....
2	.....F..TT.....
3	.....F.....F..FT.....
4	.....F.....T..T.....
5	.....T.....F.....
6	.....FTFF....F.....
7	.....FFFF...FTT.....
8	.....TFFF.F.TTT.....
9	.....T..FTTF..F.TTT.....
10	..T...F..FFFF..T.....
11	F.F...T..FTTF.....
12	F.T...F..F.....
13	T.FT.TF..F.....
14	..FF.FT..T.....
15	..FT.TT.....F.....
16	T..T.FT.....
17	...T.TT.....
18	...T.TT.....
19	F.TT.....
20	T.FT.....
21	...T.....
22	...T.....
23	.....
...	.....
34	.....
35	T.....
36	T.....
37	.....
...	.....
47	.....

Table 2: Discovered fixed-value constraints in MD4 under Wang's Attack

6.2.2 Sasaki's Collision

h1i	constraints
0	-.....+-
0	+.....-+
1	.....-.+.....
1	.....+.-.....
2	.....-.....
2	.....+.....
3	.....-++.....
3	.....+--.....
4	.....-+-.....-++++++...
4	.....-++.....-++++++...
4	.....+--.....+------...
4	.....+-+.....+------...
5	-.....-++.....
5	-.....+--.....
5	+.....-++.....
5	+.....+--.....
6	-.....-++.....-++++++++.
6	-.....-++.....+--------.
6	-.....+--.....-++++++++.
6	-.....+--.....+--------.
6	+.....-++.....-++++++++.
6	+.....-++.....+--------.
6	+.....+--.....-++++++++.
6	+.....+--.....+--------.
7	.....
8	.....-.-+.....
8	.....-.+-.....
8	.....+.-+.....
8	.....+.+-.....
9	---.+--.....-.....
9	-++.-++.....+.....
9	-++.+--.....+.....
9	+--.-++.....-.....
9	+--.+--.....-.....
9	+++.-++.....+.....
10	.....-.....
10	.....+.....
11	.....-.....
11	.....+.....
12	..-+..-.....
12	..-+..+.....
12	..+-..-.....
12	..+-..+.....
13	.....
14	.....
15	-.....
15	+.....
16	...-.....
16	...+.....
17	.....
18	.....
19	.....
20	-.....
20	+.....
21	.....
...	.....
31	.....
32	-.....
32	+.....
33	.....
...	.....
47	.....

Table 3: Discovered differential paths' constraints in MD4 under Sasaki's Attack. Includes duplicate paths.

h1i	constraints
0	.....T.T.....
1	T.....T
2	T.....F.F.....F
3	.....F.T.T.....
4	.....FFF.....F.....
5	.....T.....FFFFFFF...
6	.....FTF.....
7	T.....F.F.TFT.....F..FFFFFFTFF.
8	F.....F..TTTTFTTTT.
9	.....TT.....
10	FFF..TF.TT.....
11	FTT..FT.....
12	.....F.....
13	..FF..F.....F.....
14	..TT..T.....
15	...T.....
16	.....
17	...T.....
18	...T.....
19	.....
...	.....
47	.....

Table 4: Discovered fixed-value constraints in MD4 under Sasaki's Attack

References

[1] FIORINI, C., MARTINELLI, E., AND MASSACCI, F. How to fake an rsa signature by encoding modular root finding as a sat problem. *Discrete Applied Mathematics 130*, 2 (2003), 101 – 127. http://www.ing.unitn.it/~massacci/papers/fior-mart-mass-03-DAM.pdf.

[2] FOUQUE, P.-A., LEURENT, G., AND NGUYEN, P. Automatic search of differential path in md4. Cryptology ePrint Archive, Report 2007/206, 2007. http://eprint.iacr.org/2007/206.

[3] JUNTTILA, T. Tools for constrained boolean circuits. https://users.ics.aalto.fi/tjunttil/circuits/.

[4] MASSACCI, F. Logical cryptanalysis. http://disi.unitn.it/~massacci/CryptoSAT/. Accessed 04 Sep 2017.

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

[6] RIVEST, R. The MD4 Message-Digest Algorithm. RFC 1186, IETF, April 1990. https://tools.ietf.org/html/rfc1186.

[7] RIVEST, R., AND RSA DATA SECURITY, I. The MD4 Message-Digest Algorithm. RFC 1320, IETF, April 1992. https://tools.ietf.org/html/rfc1320.

[8] SASAKI, Y., WANG, L., OHTA, K., AND KUNIHIRO, N. *New Message Difference for MD4*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 329–348. https://link.springer.com/content/pdf/10.1007/978-3-540-74619-5_21.pdf.

[9] SCHLÄFFER, M., AND OSWALD, E. *Searching for Differential Paths in MD4*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 242–261. https://link.springer.com/content/pdf/10.1007%2F11799313_16.pdf.

[10] WANG, X., FENG, D., LAI, X., AND YU, H. Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/2004/199.

[11] WANG, X., LAI, X., FENG, D., CHEN, H., AND YU, X. *Cryptanalysis of the Hash Functions MD4 and RIPEMD*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 1–18. http://www.infosec.sdu.edu.cn/uploadfile/papers/Cryptanalysis%20of%20the%20Hash%20Functions%20MD4%20and%20RIPEMD.pdf.