# Logical Cryptanalysis Benchmarks for Classical and Modern Hash Functions

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

## I. Introduction

This collection of benchmarks focuses on new techniques in logical cryptanalysis for analyzing the structure of various hash functions. We offer as a benchmark a new technique analyzing the security of a hash function of the Merkle-Dåmgard construction which are broadly applicable to all such constructions and several benchmarks analyzing the various security properties of the Keccak hash function. While the latter techniques have not yeilded useful cryptanalysis results, they pose as a source of variable scalable benchmarks due to several choices of parameters (the bitwidth, $w$ and the number of rounds).

## II. Classical Hash Functions

The MD4 hash function is composed of 48 iterated rounds, each updating one of four 32-bit state variables [4]. While authors such as I. Mironov have applied SAT solvers to finding collisions given an existing differential path [3] and D. Jovanović has applied applied SAT solvers to brute force finding collisions and preimages in hash functions [2], these techniques either require pre-existing collisions or are too hard for SAT solvers on a large number of rounds.

One technique that is efficient for even a modest number of rounds ($\leq 28$) is the notion of a differential family search. In most cases, the differential path between two blocks $b_1$ and $b_2$ is the simple XOR difference between the intermediate rounds. For a given path $p$, there are often many such blocks which create a collision; I. Mironov showed that SAT solvers can find such blocks in MD4 and MD5 [3] relatively quickly. We can extend this concept to differential paths as well: given a differential path $p$, its family is the ordered tuple of indices of the rounds with a non-zero difference. For example, the family of the differential path introduced by M. Schlaffer (in [6]) is:

$$(1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 19, 20, 35, 36)$$

Thus for a given family, $f$, there are multiple possible differential paths which have a structure described by $f$.

By using this concept of a differential family, we can divide the search space into multiple different SAT problems: for a given number of rounds, $r$, there are $2^{r-4}$ possible differential families (versus $2^{32w-32}$ possible differential paths). For $r \leq 28$ it is possible to exhaustively search large portions of the family space; for $32 \leq r \leq 36$ it becomes possible to find in select cases, and for $r > 36$, it remains impossible in nearly all cases (due to the large time limit). Our benchmarks are a sample of possible differential families for $r = 24$, mixing both SAT and UNSAT results. These benchmarks can be found in the `families` folder [5].

## III. Modern Hash Functions

The notion of a differential family is not as useful for Keccak due to the interaction between the sponge function and the internal permutation functions causing the collision space of families to be entirely different between successive rounds. From an algebraic perspective, and to study the security margin of the XOF consruct [1], it is important to show that the Keccak round functions are bijective and have high permutation order. Two of our benchmarks (`bijection` and `orders`) model these problems in SAT. In general, these instances are easy, but with a few notable outliers: for $w \geq 4$, proving the order of $\theta$ is $3w$ is difficult. Note that this should simplify to showing that $\forall x, x \neq x$ is UNSAT, and thus should be relatively trivial; however $w = 4$ produces runtimes in excess of one hour. Further, anything later involving $\theta$ (such as $\rho \circ \theta$, etc.) also becomes difficult for $w \geq 4$.

In `differences`, we study the differential properties of the permutation functions: for a given parameters $x, y \leq 25w$ and round function $f$, we seek to find witnesses $a$ and $b$ such that:

$$\#(1, a \oplus b) = x$$
$$\#(1, f(a) \oplus f(b)) = y$$

That is, find an input with difference $x$ which produces an output with difference $y$. There are a few interesting outliers in this model: for $w \geq 4$, most round functions cannot be individually analyzed this way. Further, while $\pi$ is a permutation of the order of the bits (and thus does not change the values of any bits), certain instances in $w = 2$ where $x \neq y$ produce runtimes in excess of an hour. This is surprising as the model is trivially SAT if and only if $x == y$ (and the model merely involves changing the orders of variables).

The benchmarks in `output-margins` consider the effects of the sponge function. Since all of the round functions are bijective, if two inputs differ, the interal state of Keccak must also differ. However, to produce a collision, only the first $y$ bits of the output must be the same. Thus we can create a model for a given number of differences $x, z \leq 25w$, security margin $y \leq 25w$, and round functions $f$:

$$\#(1, axorb) = x$$
$$\#(1, (f(a)xorf(b))[0 : y]) = z$$

If such a witness a and b exist for $z = 0$, then the input difference $x$ is possible of producing collisions at a security margin $m$. Our provided benchmarks sample the space for small values of $w$ and relatively few round functions; for $w \geq 16$ and for any set of functions including $\theta$, the runtimes become exceedingly long.

The benchmarks in `xof-state` attempt to recreate the internal state of Keccak given a series of outputs from the XOF (extensible output function at a given margin). In general, this is possible for either small values of $w$ or small numbers of rounds (for larger values of $w$). However, care must be selected in choosing the base seed, otherwise, there can possibly be multiple satisfying seeds. However, for a set of inputs with unique solution, these benchmarks can be extended to contain the output from several rounds of Keccak. After a threshhold dependent on the margin, these are redundant information and thus test the SAT solvers to work with larger models which overfit to the solution.

## IV. Thanks

A special thanks to Mate Soos and his CryptoMiniSat solver [7] for suggesting these problems be submitted as benchmarks to the conference.

This work was completed as part of an Undergraduate Honors Research project at Iowa State University under the guidance of Eric W. Davis and as part of MATH 490: Independent Study of SHA-3 under the guidance of Clifford Bergman.

## References

[1] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: The keccak reference - version 3.0 (2011), https://keccak.team/files/Keccak-reference-3.0.pdf

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

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

[4] Rivest, R., RSA Data Security, I.: The MD4 Message-Digest Algorithm. RFC 1320, IETF (April 1992), https://tools.ietf.org/html/rfc1320

[5] Scheel, A.: Sat competition 2018 submission. GitHub (2018), https://github.com/cipherboy/sat/tree/master/sat-competition-2018

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

[7] Soos, M.: Cryptominisat sat solver. GitHub (2017), https://github.com/msoos/cryptominisat