# A Collection of Thoughts on the Keccak/SHA-3 Construction

**Alexander Scheel**

Iowa State University

**TODO**

MATH 490 | Advisor: Dr. Clifford Bergman | SHA-3 | hash_framework

## 1. A Series of Introductions

This paper presents the work of the author towards the completion of the requirements for the MATH 490 Independent Study course under Dr. Clifford Bergman at Iowa State University of Science and Technology. This work was an extension of the author's Honors Project under Dr. Eric Bergman and utilized the resulting hash_framework project. Additional artifacts related to this project can be seen in the keccak-attacks repository.

Within this section are a series of introductions which provide necessary background on the topics of cryptography, hash functions, the development of Keccak/SHA-3, and its structure. While certain sections can be skipped if the reader has the prerequisite knowledge, hopefully all readers find the material engaging and useful. Following these introductions, this paper presents the analysis of the Keccak/SHA-3 hash function before concluding with a final evaluation and further work.

**Introduction on the Topics of Cryptography.** While there are many applications of pure mathematics, few are as demanding and shrouded in secrecy as cryptography. Cryptography exists because of the fundamental need of civilizations, governments, and individuals to keep secrets secure from devoted adversaries. While modern cryptography combines the disciplines of mathematics, computer science, and computer engineering, prior to the turn of the 20th century, cryptography lacked much of its modern rigor.

Prior to the late 19th century, cryptography was split into three major branches: ciphers, codes, and stenography.

**Introduction on the Topics of Hash Functions.** Within cryptography's collection of algorithms, few are are as useful as hash functions have proven to be to cryptographers and non-cryptographers alike. A hash function maps arbitrary length binary strings to binary strings of a fixed length.

**Introduction on the Development of Keccak/SHA-3.** FIPS 202 [1].

**Introduction to Terminology.** This section contains a collection terminology useful for discussing SHA-3 and Boolean Satisfiability.

E-mail: alexander.m.scheelgmail.com

We define $\Sigma = \{0, 1\}$ to be the alphabet.

We define the set $W = \{1, 2, 4, 8, 16, 32, 64\}$ to be the powers of two typically used in constructing the Keccak widths; $w = 64$ is standardized for use in SHA-3, though any power of two can be used.

For a given $w \in W$, we define $S_w = \Sigma^{25w}$, to be the set of all binary strings of length $25w$; these represent the possible states (which are binary strings that map onto an indexible array $A$ described later) and each round permutation maps $S_w \mapsto S_w$.

A binary string $s \in S_w$ can be indexed directly (in a zero-indexed manner, i.e., the starting bit of $s$ is denoted $s[0]$), or via a 3 dimensional structure of size 5 by 5 by $w$. This is typically done in conjunction with an uppercase letter, $A = s$, and then $A[x, y, z] = s[w(5y + x) + z]$. This is in accordance with FIPS 202 [1].

**Introduction on the Structure of SHA-3.** SHA-3 consists of two parts: a core permutation function, KECCAK-$f$, and a domain extender, the KECCAK sponge function. As discussed by the Keccak authors, additional security is given by choosing the permutation functions to be bijective, though they need not be (the domain need only be $S_w$).

## 2. Mathematical Properties of the Five Round Functions

In the following section, we detail various mathematical properties of the five permutation functions which make up the core round function of Keccak. In most cases, we seek to provide mathematical proofs of these properties. In all cases, we rely on external code and Boolean Satisfiability for computerized proofs of these theorems, in ways independent of the ways proved here. In a few cases, we provide references into the existing literature.

**Properties of $\theta$.** In this section, we show that $\theta$ is bijective, that XOR distributes through $\theta$, give a method for finding the inverse of $\theta$, and give the order of $\theta$.

Let $w$ be a fixed power of 2. Since $S_w$ is of finite size, it suffices to show that, $\forall x, y \in S_w$, $x \neq y \Rightarrow \theta(x) \neq \theta(y)$. Assume the hypothesis: then $x \oplus y \neq 0^{25w}$.

**Lemma 2.1.**

$$\theta(a) = 0^{25w} \iff a = 0^{25w}$$

*Proof.* This follows from the definition of $\theta$: note that $\theta(0^{25w}) = 0^{25w}$. If $a \neq 0^{25w}$, then there exists an index set $I_1$ such that $\forall i \in I_1$, $a[i] = 1$, and $\forall j \notin I_1$, $a[j] = 0$. Then $\theta(a) \neq 0^{25w}$, which follows from the construction of $\theta$. (Note that each output bit of $\theta$ is composed of the XOR of 11 values in a fixed pattern).

□

**Lemma 2.2.** $\forall a, b \in S_w$,

$$\theta(a \oplus b) = \theta(a) \oplus \theta(b)$$

*Proof.* This follows from the definition of $\theta$: note that $\theta$ is composed entirely of XORs and that XOR is commutative and associative.

□

Combining Lemma 2.1 and Lemma 2.2, we have that:

$$\begin{aligned} x \oplus y = 0^{25w} &\iff \theta(x \oplus y) = \theta(0^{25w}) \\ &\iff \theta(x \oplus y) = 0^{25w} \\ &\iff \theta(x) \oplus \theta(y) = 0^{25w} \end{aligned}$$

and hence $\theta$ is bijective.

To construct the inverse of $\theta$, note that $A'[x,y,z] = A[x,y,z] \oplus D[x,z]$; hence, $A[x,y,z] = A'[x,y,z] \oplus D'[x,z]$ for some $D' = D$. Since $D[x,z]$ is composed of several $C[x,z]$, where $C[x,z] = \oplus_{y=0}^{4} A[x,y,z]$, we can similarly define $C'[x,z]$ to be $C'[x,z] = \oplus_{y=0}^{4} A'[x,y,z]$. Then, we expect the inverse of $\theta$ to be of a similar form. This reduces to a linear algebra problem over boolean variables. We know that $D'[x,z] = D[x,z]$ in order to recover $A[x,y,z]$. Hence we can represent $D[x,z]$ as a series of bits of length $5 \times w$, where bit $i = z' + 5 \times x'$ is 1 if and only if $C[x',z']$ is used in the construction of $D[x,z]$. Further, we can view each of the $C'[x,z]$ as being the conjunction of three $C[x',z']$ in $A$, and thus can represent these as bit strings where bit $j = z' + 5 \times x'$ is 1 if and only if $C[x',z']$ is used to construct $C'[x,z]$. (That is, since $C'[x,z] = \oplus_{y=0}^{4} A'[x,y,z]$, $C'[x,z] = \oplus_{y=0}^{4}(A[x,y,z] \oplus D[x,z])$ and hence $C'[x,z] = \oplus_{y=0}^{4}(A[x,y,z] \oplus C[x',z'] \oplus C[x'',z''])$, and thus $C'[x,z] = C[x,z] \oplus C[x',z'] \oplus C[x'',z'']$, for some $x, z, x', z', x'', z''$ based on the definition of $\theta$. Thus, giving each $C'[x,z]$ a constant $c_{x,z}$ for whether it is used in constructing $D'[x,z]$, we can form a system of linear equations and solve for the constants $c_{x,z}$ in each expression. Since there are $5 \times w$ variables and $5 \times w$ equations in each equation for $D'[x,z]$, this can be solved easily, yielding the inverse of $\theta$.

Lastly, we have computed the order of the permutation $\theta$ for all $w$. We reproduce them here without proof; they were found by randomized search, and verified with SAT for $w = 1, 2, 4$ and 8. In general, the order is given by the expression $3 \times w$.

| w | Order |
|---|---|
| 1 | 3 |
| 2 | 6 |
| 4 | 12 |
| 8 | 24 |
| 16 | 48 |
| 32 | 96 |
| 64 | 192 |

***Of*** $\rho$***.*** Since $\rho$ is a simple permutation of the location of bits, it holds that $\rho(a \oplus b) = \rho(a) \oplus \rho(b)$.

It is obvious that the order of the $\rho$ permutation is $w$: this follows from the construction of $\rho$.

***Of*** $\pi$***.*** Since $\pi$ is a simple permutation of the location of bits, it holds trivially that $\pi(a \oplus b) = \pi(a) \oplus \pi(b)$.

It is obvious that the order of the $\pi$ permutation is 24: this follows from the construction of $\pi$.

***Of*** $\chi$***.*** It is non-trivial that the order of the $\chi$ function is a constant 4. However, this can be verified through the use of SAT to prove that $\chi^4$ is the identity function. Hence $\chi^{-1} = \chi^3$. However, due to the construction of $\chi$, XOR does not distribute over it.

***Of*** $\iota$***.*** Note that since $\iota$ is an XOR with a fixed value, it is obvious that $\iota$ is a bijection: for any $w$, for any $i$, and for all $x \in S_w$, $\iota(\iota(x,i),i) = x$, since $x \oplus \iota_i \oplus \iota_i = x$. Hence, $\iota$ is its own inverse and hence $\iota$ is bijective since the inverse is well defined for all $x \in S_w$.

Lastly, note that it is trivial that the order of the $\iota$ permutation is 2 by construction (due to the XOR).

**Evaluation of the Orders of Composition of Permutations.** In this section, we discuss how the above five permutation functions compose, and the orders of the resulting compositions. We limit our discussion to $w = 1$ for the interests of exhaustive search: $2^{25}$ is possible on commodity hardware, $2^{50}$ would require additional resources. In general, we find that these permutations interact non-trivially, resulting in cycles of mixed size. We reproduce these results in the table below:

| Function | Order | Fixed Points | Number of Cycles | List of Cycles |
|---|---|---|---|---|
| $\theta$ | 3 | 2097152 | 2 | 1, 3 |
| $\rho$ | 1 | 33554432 | 1 | 1 |
| $\pi$ | 24 | 4 | 8 | 1, 2, 3, 4, 6, 8, 12, 24 |
| $\chi$ | 4 | 32 | 3 | 1, 2, 4 |
| $\chi \circ \pi$ | 17360392635484575518934418947500880 $\approx 2^{113.741}$ | 3 | 28 | Not Reproduced |
| $\pi \circ \rho \circ \theta$ | 24 | 4 | 8 | 1, 2, 3, 4, 6, 8, 12, 24 |
| $\chi \circ \pi \circ \rho \circ \theta$ | 418144575651966378899040573720 $\approx 2^{98.399}$ | 3 | 27 | Not Reproduced |
| $r_1$ | 320185339723133697023127516600 $\approx 2^{98.014}$ | 0 | 14 | Not Reproduced |
| $r_2$ | $\approx 2^{130.726}$ | 0 | 12 | Not Reproduced |
| $r_3$ | $\approx 2^{164.242}$ | 1 | 16 | Not Reproduced |
| $r_4$ | $\approx 2^{211.609}$ | 0 | 16 | Not Reproduced |
| $r_5$ | $\approx 2^{131.878}$ | 2 | 16 | Not Reproduced |
| $r_6$ | $\approx 2^{190.743}$ | 3 | 18 | Not Reproduced |
| $r_7$ | $\approx 2^{218.017}$ | 0 | 18 | Not Reproduced |
| $r_8$ | $\approx 2^{190.137}$ | 0 | 18 | Not Reproduced |
| $r_9$ | $\approx 2^{188.483}$ | 0 | 14 | Not Reproduced |
| $r_{10}$ | $\approx 2^{154.432}$ | 0 | 19 | Not Reproduced |
| $r_{11}$ | $\approx 2^{223.948}$ | 1 | 18 | Not Reproduced |
| $r_{12}$ | $\approx 2^{108.579}$ | 0 | 12 | Not Reproduced |
| $r_{13}$ | $\approx 2^{209.785}$ | 1 | 18 | Not Reproduced |
| $r_{14}$ | $\approx 2^{170.621}$ | 0 | 14 | Not Reproduced |
| $r_{15}$ | $\approx 2^{196.507}$ | 2 | 22 | Not Reproduced |
| $r_{16}$ | $\approx 2^{172.643}$ | 1 | 16 | Not Reproduced |
| $r_{17}$ | $\approx 2^{221.160}$ | 1 | 22 | Not Reproduced |
| $r_{18}$ | $\approx 2^{203.510}$ | 0 | 20 | Not Reproduced |
| $r_{19}$ | $\approx 2^{250.748}$ | 3 | 20 | Not Reproduced |
| $r_{20}$ | $\approx 2^{183.937}$ | 1 | 19 | Not Reproduced |
| $r_{21}$ | $\approx 2^{158.852}$ | 2 | 15 | Not Reproduced |
| $r_{22}$ | $\approx 2^{111.874}$ | 1 | 12 | Not Reproduced |
| $r_{23}$ | $\approx 2^{230.807}$ | 0 | 24 | Not Reproduced |
| $r_{24}$ | $\approx 2^{140.355}$ | 1 | 14 | Not Reproduced |

Note that, for $w = 1$, $\rho$ is the identity function and is thus omitted from the tables above. Note that $r_n$ denotes the first $n$ rounds of SHA-3.

While on first glance, large orders make it appear that the hash function is strong, SHA-3 also defines an XOF (or eXtensible Output Function) construct. This allows SHA-3 to act as a pseudo-random number generator: after hashing an input, the internal state of SHA-3 has reached some value $S$. Suppose we want to request $m$ bits of state. If $k$ exceeds our security margin $k/2$, we take $k/2$ bits, permute $S \to S'$ according to KECCAK-$f$, and repeat until all $m$ bits have been retrieved.

An ideal permutation function for an $XOF$ would ensure that there exists one random cycle through all possible states (and thus contain a cycle of $2^{25}$, in this case). However, the order of the $r_n$ rounds of SHA-3 far exceed $2^{25}$ and all of them have several cycles. This suggests that the actual security margin of the XOF construct is far less than the theoretical value. However, this analysis needs to be extended to at least $w = 4$ (performing $2^{100}$ iterations of the hash function core which is not feasible) to fully verify this.

## 3. Marginal and Differential Properties of the Five Round Functions

This section studies three approaches to extracting information about the five permutations functions using SAT. None of these approaches can directly lead to a collision in SHA-3, however, by combining them, possible differential trails can be approximated. Sadly for $w \geq 4$, several of these attacks take too long to run (even for individual permutation functions) and thus do not emit a valid attack for full Keccak.

**Marginal Properties.**

**Differential Properties.**

**Input Margin Impact on Differential Properties.**

**Output Margin Impact on Differential Properties.**

## 4. Exhaustive Collision Searches

By modeling the problem with SAT, we were able to recreate the work of prior authors ([2], [3]). However, like previous authors, our results do not scale to useful results. For $w = 1$, we have exhaustively searched the collision space; the results of this search are reproduced below:

| $w$ | $r$ | Number of collisions |
|---|---|---|
| 1 | 1 | 1024 |
| 1 | 2 | 502 |
| 1 | 3 | 543 |
| 1 | 4 | 525 |
| 1 | 5 | 488 |
| 1 | 6 | 518 |
| 1 | 7 | 532 |
| 1 | 8 | 498 |
| 1 | 9 | 506 |
| 1 | 10 | 522 |
| 1 | 11 | 506 |
| 1 | 12 | 503 |
| 1 | 13 | 495 |
| 1 | 14 | 522 |
| 1 | 15 | 485 |
| 1 | 16 | 540 |
| 1 | 17 | 467 |
| 1 | 18 | 490 |
| 2 | 1 | $> 2^{20}$ |
| 2 | 2 | $> 2^{14}$ |
| 2 | 3 | $> 74$ |

Note that the results for $w = 2$ are incomplete; the search was terminated after one week. Further, all of these used an effective margin of 512-bits. In our limited time, we were unable to find any useful patterns in this data.

## 5. Correlation Matrices

Another area of study was the resistance of Keccak to correlation attacks. That is, given some structured input (in this case, a valid block where the remaining bits are all zeros), does SHA-3 emit any useful two bit correlations across rounds? Here, we constructed a program to generate matricies of correlation for all two-variable boolean functions given some state. We then performed Monte-Carlo simulation; for $w = 8$ and $r = 24$, we found no useful correlations under sufficiently large starting states ($> 16$ million). Thus, Keccak is secure from cross-round first and second order correlations; all distinguisher attacks must thus rely on at least third-order correlations. For $w = 8$, checking all third-order correlations across 24 rounds becomes computationally infeasible and thus was not performed, since it is not likely to yeild useful results.

## 6. Fixed Point Attacks

In this section, we introduce two potential attacks against Keccak/SHA-3 using fixed points in the underlying round function. However, neither of these attacks have been proven possible with current analysis except for small values of $w$ and small numbers of rounds. The first is a possible attack using full fixed points in the core round function, while the latter describes a category of partial fixed points.

**Full Fixed Points.** One theoretical attack against SHA-3 is by using a fixed point. If there existed a state $x$ such that $h(x) = x$, for $h$ the round function, then for all additional blocks, $h(x|b) = h(x \oplus b)$. While fixed points can and do occur (see 24-rounds in Table 2), using a fixed-point attack is more subtle in practice: it is unlikely that a fixed point is a valid block (that is, $x$ contains a suffix that is $0^m$ for the current margin, $m$). Thus one must find a series of blocks $b_1$, $b_2$ such that $h(b_1) \oplus b_2 = x$; this in turn extends the attack from being $h(x|b) = h(x \oplus b)$ to:

$$\begin{aligned} x = h(h(b_1) \oplus b_2) &\Rightarrow \\ h(x|b) = h(b_1|b_2|b) &= h(b_1|h_2 \oplus b) \\ &= h(b_1|h_2|0^{25w}|b) \end{aligned}$$

However finding $b_1$, $b_2$ amounts to a preimage attack and is thus unlikely to occur.

For $w = 1, 2$, we have verified that no fixed points occur which are valid blocks for $1 \leq r \leq 24$, where $r$ is the number of rounds, at margins of 4 and 8 bits.

**Partial Fixed Points.** Another theoretical attack is using a partial fixed point. Suppose there exists a block $x$ such that $h(x) = y$ is also a block (that is, $y$ has a suffix of $0^m$). In this case, the following is a valid collision, for all blocks $b$:

$$h(x \oplus b) = h(y|b)$$

We have verified the existance of these partial fixed points for small $w$ and number of rounds. Furthermore, they are more readily found using SAT than finding a fixed point is.

For $w = 1$, we reproduce a table of quantities of partial fixed points per round below for the first 8 rounds:

| Rounds | Quantity | Input | Output |
|---|---|---|---|
| 1 | 512 | FFFFFFFFFFFFFFFFFFFFFFFFF | TFFFFFFFFFFFFFFFFFFFFFFFF |
| 2 | 567 | FTFTTTTFFTTTFFTFTFFFFFFFF | TTTFTFFTFTTTFTFFTFFFFFFFF |
| 3 | 527 | FFTFFFFTFTFFFFFFFFFFFFFFF | TTTFTTTTFTFTFTFTFFFFFFFFF |
| 4 | 545 | TTTFFTTTFFTTFTTFFFFFFFFFF | FFFTTTTFTTFFFTFTFFFFFFFFF |
| 5 | 488 | FFFFFFFFTFFFTFFFFFFFFFFFF | TFFTFFTFFFFFFTFFTFFFFFFFF |
| 6 | 533 | FFTFFFFTTFFFTFFFFFFFFFFFF | TTTFFFFTFFFFTTFFTFFFFFFFF |
| 7 | 510 | TFFFTTTFTTTTFFTFFFFFFFFFF | FTFFFFFFFTFTTTFTTFFFFFFFF |

Note that the above table is only for an effective margin of 256-bits (4 bits when $w = 1$). For an effective margin of 512-bits (8 bits), we found no such partial fixed points.

However, brute forcing a partial fixed point for larger values of $w$ and $r$ quickly grows impossible. A similar technique to the aforementioned differential matrices could extend these techniques to larger values of $w$ and $r$. Instead of building a model to check for a specific number of differences, instead build a model to check for a specific number of zeros in the inputs and outputs. This could provide a reduction in number of search paths for the SAT solver, and allow for more explicit parallelism.

## 7. Conclusions and Further Work

## 8. Bibliography

1. (2015) Fips pub 202, sha-3 standard: Permutation-based hash and extendable-output functions. U.S.Department of Commerce/National Institute of Standards and Technology. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.
2. Homsirikamol E, Morawiecki P, Rogawski M, Srebrny M (2012) *Security Margin Evaluation of SHA-3 Contest Finalists through SAT-Based Attacks*, eds. Cortesi A, Chaki N, Saeed K, Wierzchoń S. (Springer Berlin Heidelberg, Berlin, Heidelberg), pp. 56–67. https://eprint.iacr.org/2012/421.pdf.
3. Morawiecki P, Srebrny M (2010) A sat-based preimage analysis of reduced keccak hash functions (Cryptology ePrint Archive, Report 2010/285). https://eprint.iacr.org/2010/285.pdf.