Evaluation of hill climbing, simulated annealing, tabu search and random selection: search algorithms on cryptographic hash functions BLAKE, Grøstl and Keccak

Soham Sadhu

May 29, 2014

Abstract

- In October 2012, Keccak was chosen as the winner of SHA-3 competition amongst 64 candidates, including the finalists BLAKE and Grøstl.
- I have attempted to find near collisions in reduced versions of BLAKE, Grøstl and Keccak; using hill climbing, random selection, simulated annealing and tabu search.

Table of contents I

Table of contents II

Table of contents III

Hash function

A *hash family* is a four-tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$, satisfying the following conditions.[1]

- $\mathcal{X}$ is a set of possible messages
- $\mathcal{Y}$ is a finite set of hash function output
- $\mathcal{K}$, the *keyspace*, is a finite set of possible keys
- For each $K \in \mathcal{K}$, there is a hash function $h_k \in \mathcal{H}$. Each $h_k : \mathcal{X} \to \mathcal{Y}$

[1] Douglas R. Stinson. Cryptography Theory and Practice, chapter 4. Cryptographic Hash Functions. Chapman & Hall/CRC, Boca Raton, FL 33487-2742, USA, third edition, 2006.

Property of Hash function[2]

1. **Preimage resistance**
   Given: A hash function $h : \mathcal{X} \to \mathcal{Y}$ and an element $y \in \mathcal{Y}$.
   Find: $x \in \mathcal{X}$ such that $h(x) = y$.
2. **Second preimage**
   Given: A hash function $h : \mathcal{X} \to \mathcal{Y}$ and an element $x \in \mathcal{X}$.
   Find: $x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x) = h(x')$.
3. **Collision resistance**
   Given: A hash function $h : \mathcal{X} \to \mathcal{Y}$
   Find: $x, x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

[2] Douglas R. Stinson. Cryptography Theory and Practice, chapter 4. Cryptographic Hash Functions. Chapman & Hall/CRC, Boca Raton, FL 33487-2742, USA, third edition, 2006.

Security model

- **Random Oracle** model, proposed by Bellare and Rogaway. Algorithm is secure, modulo the way it creates the random outputs.[3]
- **Birthday paradox:** In a sample size of M, minimum N number of attempts to find, two elements with same value is given by equation $N \approx 1.17\sqrt{M}$.

[3] Gerrit Bleumer. Random oracle model. In HenkC.A. van Tilborg and Sushil Jajodia, editors, Encyclopedia of Cryptography and Security, pages 10271028. Springer US, 2011.

Application of hash functions

1. **Digital forensics:** take a hash value of evidence, to later prove that it has not been tampered. [4]
2. **Password stored:** is salted and hashed, before inserting to database.
3. **File integrity:** take hash value of files between time intervals, to make sure; they have not been tampered.
4. **Pseudo random:** generator, based on a seed value.

---

[4] Richard P. Salgado. Fourth Amendment Search And The Power Of The Hash, volume 119 of 6, pages 38 46. Harvard Law Review Forum, 2006.

SHA-0

- SHA-0 proposed by NSA in 1993, later standardised by NIST.
- In 1995 Florent Chabaud and Antoine Joux, found collisions in SHA-0 with complexity of 2^{61}.
- In 2004, Eli Biham and Chen found near collisions for SHA-0, about 142 out of 160 bits to be equal.
- Full collisions were also found, when the number of rounds for the algorithm were reduced from 80 to 62.

SHA-1

- In 1995, SHA-0 replaced by SHA-1, designed by NSA[5]. SHA-1 had block size of 512 bits, size of 160 bits; and additional circular shift operation, to rectify weakness from SHA-0.
- In 2005, team from Shandong University found collisions on full version of SHA-1 requiring 2^{69} operations[6].

[5] James Joshi. Network Security: Know It All: Know It All. Newnes Know It All. Elsevier Science, 2008.

[6] Bruce Schneier. Sha-1 broken. http://www.schneier.com/blog/archives/2005/02/sha1_broken.html, February 2005.

SHA-2

- SHA-2 was designed by NSA, and released in 2001 by NIST. Family of functions of SHA-224, SHA-256, SHA-384, SHA-512.
- Computational operations for finding collisions in SHA-256 for 23-step was found to be around $2^{11.5}$, and for 24 step was $2^{28.5}$ respectively.
- Computational operations for finding collisions in SHA-512 for 23-step was found to be around $2^{16.5}$, and for 24 step was $2^{32.5}$ respectively[7].

[7] Somitra Kumar Sanadhya and Palash Sarkar. New collision attacks against up to 24- step sha-2. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, INDOCRYPT, volume 5365 of Lecture Notes in Computer Science, pages 91103. Springer, 2008.

SHA-3

- NIST announced competition for choosing SHA-3 on November, 2007. Entries accepted till October, 2008.
- 51 candidates from 64 submissions, were accepted for first round on December 9, 2008.
- Out of 5 finalists, on October 2, 2012; Keccak was announced winner amongst other four finalist, which included BLAKE and Grøstl.
- Keccak was chosen for large security margin, flexibility, and efficient hardware implementation.

Properties of BLAKE hash function

Algorithm	Word	Message	Block	Digest	Salt
BLAKE-224	32	$< 2^{64}$	512	224	128
BLAKE-256	32	$< 2^{64}$	512	256	128
BLAKE-384	64	$< 2^{128}$	1024	384	256
BLAKE-512	64	$< 2^{128}$	1024	512	256

Table: Specification of available input, output, block and salt sizes for various BLAKE hash functions, size in bits. [9]

[9] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Blake. http://www.131002.net/blake/blake.pdf, April 2012.

BLAKE construction

BLAKE is built on HAIFA (HAsh Iterative FrAmework) structure [10]which is an improved version of Merkle-Damgård function.

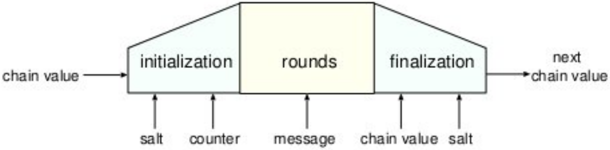


Figure: Local wide construction of BLAKE's compression function[11]

[10] Eli Biham and Orr Dunkelman. A framework for iterative hash functions - haifa. Cryptology ePrint Archive, Report 2007/278, 2007.

[11] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Blake. http://www.131002.net/blake/blake.pdf, April 2012.

Padding rule

- For variant producing digest size 224, 256 input message is padded with '1' followed by '0' bits, so that length is 447 modulo 512. Followed by bit '1', and 64 bit unsigned big endian representation of block length.
- For variant producing digest size 384, 512 input message is padded by bit '1', followed by '0' bits till length is 895 modulo 1024. Followed by bit '1', and 128 bit unsigned big endian representation of block length in bits.

Compression algorithm

Algorithm 1 BLAKE Compression procedure[12]

1: $h^0 \leftarrow IV$
2: **for** $i = 0, \dots, N-1$ **do**
3: $\quad h^{i+1} \leftarrow compress(h^i, m^i, s, l^i)$
4: **end for**
5: **return** h^N

[12] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Blake. http://www.131002.net/blake/blake.pdf, April 2012.

Initialization of the state

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

After initialization the matrix is operated for 14 or 16 rounds depending on version, on the following groups represented as $G_i(a, b, c, d)$

$$\begin{array}{ccc} G_0(v_0, v_8, v_{12}) & G_1(v_1, v_5, v_9, v_{13}) & G_2(v_2, v_6, v_{10}, v_{14}) \\ G_3(v_3, v_7, v_{11}, v_{15}) & G_4(v_0, v_5, v_{10}, v_{15}) & G_5(v_1, v_6, v_{11}, v_{12}) \\ G_6(v_2, v_7, v_8, v_{13}) & G_7(v_3, v_4, v_9, v_{14}) & \end{array}$$

BLAKE permutation operation for 224 and 256 variant

$$
\begin{aligned}
&a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\
&d \leftarrow (d \oplus a) \ggg 16 \\
&c \leftarrow c + d \\
&b \leftarrow (b \oplus c) \ggg 12 \\
&a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\
&d \leftarrow (d \oplus a) \ggg 8 \\
&c \leftarrow c + d \\
&b \leftarrow (b \oplus c) \ggg 7
\end{aligned}
$$

$+$ addition in modulo 2^{32}

$\ggg k$ rotate to right by k bits

$\oplus$ bitwise XOR

r permutation round

Permutation round selection, σ function[13]

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

[13] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Blake. http://www.131002.net/blake/blake.pdf, April 2012.

BLAKE permutation operation for 384 and 512 variant

$$
\begin{aligned}
& a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\
& d \leftarrow (d \oplus a) \ggg 32 \\
& c \leftarrow c + d \\
& b \leftarrow (b \oplus c) \ggg 25 \\
& a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\
& d \leftarrow (d \oplus a) \ggg 16 \\
& c \leftarrow c + d \\
& b \leftarrow (b \oplus c) \ggg 11
\end{aligned}
$$

$+$ addition in modulo 2^{64}

Finalization

$$
\begin{aligned}
h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\
h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\
h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\
h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\
h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\
h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\
h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\
h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}
\end{aligned}
$$

Padding

- The message is split into blocks of 512 bits for variants of digest size upto 256 bits; and into 1024 bit block for variant of digest size above 256 bits.
- Bit '1' is appended, then $w = -N - 65 \; mod \; l$, 0 bits are appended; followed by 64 bit representation of $(N + w + 65)/l$.
- Due to encoding of message in padded block, the maximum size of message for short variants is $2^{73} - 577$ bits, and that for variants above 256 is $2^{74} - 1089$bits.

Initial values and rounds [14]

Permutations	Digest size	Recommended value of r
P_{512} and Q_{512}	8 - 256	10
P_{1024} and Q_{1024}	264 - 512	14

Table: Recommended number of rounds

n	iv_n
224	00 . . . 00 00 e0
256	00 . . . 00 01 00
384	00 . . . 00 01 80
512	00 . . . 00 02 00

Table: Initial values for Grøstl-n function.

[14] Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian

Hashing the message

After padding, the message is broken to blocks, and processed sequentially. An initial $h_0 = \text{iv}$ is defined.

$$h_i \leftarrow f(h_{i-1}, m_i) \text{ for } i = 1, \ldots, t.$$

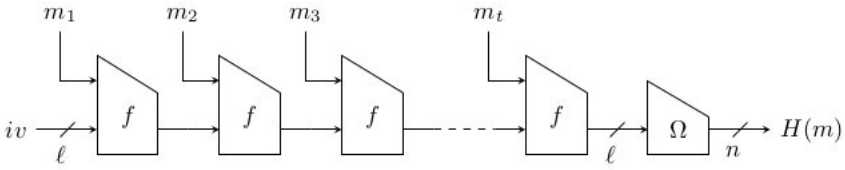


Figure: Grøstl hash function [15]

[15] Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian

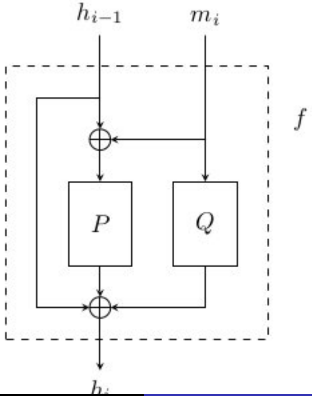
Input mapping

Mapping of the input bytes to the state in following order.

$$\begin{bmatrix} 00 & 08 & 10 & 18 & 20 & 28 & 30 & 38 \\ 01 & 09 & 11 & 19 & 21 & 29 & 31 & 39 \\ 02 & 0a & 12 & 1a & 22 & 2a & 32 & 3a \\ 03 & 0b & 13 & 1b & 23 & 2b & 33 & 3b \\ 04 & 0c & 14 & 1c & 24 & 2c & 34 & 3c \\ 05 & 0d & 15 & 1d & 25 & 2d & 35 & 3d \\ 06 & 0e & 16 & 1e & 26 & 2e & 36 & 3e \\ 07 & 0f & 17 & 1f & 27 & 2f & 37 & 3f \end{bmatrix}$$

Permutation f function

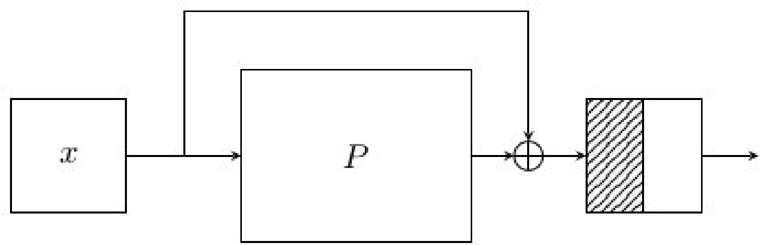
$$f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h.$$



Omega truncate function

The Ω function consists of a $trunc_n(x)$ that outputs only the trailing n bits of input x.

$$\Omega(x) = trunc_n(P(x) \oplus x).$$



Contents of P and Q function

The P and Q functions are represented by a round, with slight variation in variables they operate.

$$R = MixBytes \cdot ShiftBytes \cdot SubBytes \cdot AddRoundConstant$$

Add Round Constant

$A \leftarrow A \oplus C[i]$ where A is state matrix and C is constant matrix.

$$P_{1024} : C[i] = \begin{bmatrix} 00 \oplus i & 10 \oplus i & 20 \oplus i \ldots f0 \oplus i \\ 00 & 00 & 00 \ldots 00 \\ \vdots & \vdots & \vdots \ldots \vdots \end{bmatrix}$$

and

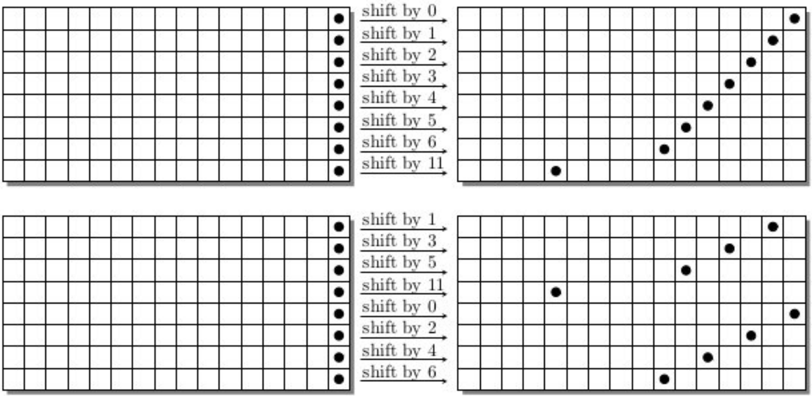
$$Q_{1024} : C[i] = \begin{bmatrix} ff & ff & ff \ldots ff \\ \vdots & \vdots & ff \ldots ff \\ ff \oplus i & ef \oplus i & df \oplus i \ldots 0f \oplus i \end{bmatrix}$$

Substitute byte

$$a_{i,j} \leftarrow S(a_{i,j}), 0 \leq i < 8, 0 \leq j < v.$$

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Shift byte


shift by 0
shift by 1
shift by 2
shift by 3
shift by 4
shift by 5
shift by 6
shift by 11
shift by 1
shift by 3
shift by 5
shift by 11
shift by 0
shift by 2
shift by 4
shift by 6

Mix byte

$A \leftarrow B \times A$

B is a finite field over $\mathbb{F}_{256}$, defined over $\mathbb{F}_2$ by polynomial $x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$.

$$B = \begin{bmatrix} 02 & 02 & 03 & 04 & 05 & 03 & 05 & 07 \\ 07 & 02 & 02 & 03 & 04 & 05 & 03 & 05 \\ 05 & 07 & 02 & 02 & 03 & 04 & 05 & 03 \\ 03 & 05 & 07 & 02 & 02 & 03 & 04 & 05 \\ 05 & 03 & 05 & 07 & 02 & 02 & 03 & 04 \\ 04 & 05 & 03 & 05 & 07 & 02 & 02 & 03 \\ 03 & 04 & 05 & 03 & 05 & 07 & 02 & 02 \\ 02 & 03 & 04 & 05 & 03 & 05 & 07 & 02 \end{bmatrix}$$

Keccak sponge construction

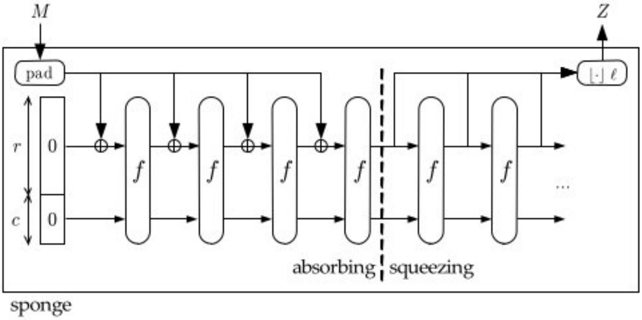
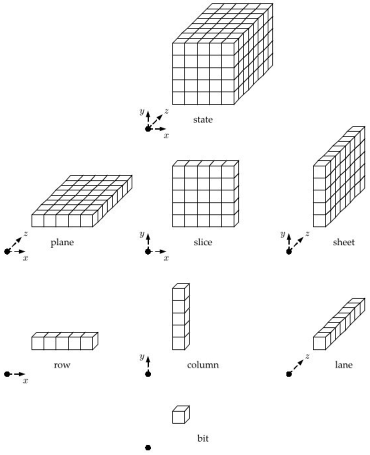


Figure: Sponge construction $Z = Sponge[f, pad, r](M, l)$[21]

[21] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. http://sponge.noekeon.org/CSF-0.1.pdf.

Keccak state



Padding and permutations

- The message is padded with 10^*1 to make it multiple of the block length.
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
- The permutation round R is repeated for $12 + 2l$ times, where l is the lane length. The default capacity size is 576.

Contents of Keccak permutation round

$$
\begin{aligned}
\theta : a[x][y][z] &\leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1\\
\rho : a[x][y][z] &\leftarrow a[x][y][z-(t+1)(t+2)/2],\\
&\quad t \textit{ satisfying } 0 \leq t < 24 \textit{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \textit{ in } G\\
&\quad \textit{or } t = -1 \textit{ if } x = y = 0,\\
\pi : a[x][y] &\leftarrow a[x'][y'], \textit{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},\\
\chi : a[x] &\leftarrow a[x] + (a[x+1]+1)\, a[x+2],\\
\iota : a &\leftarrow a + RC[i_r].
\end{aligned}
$$

θ step

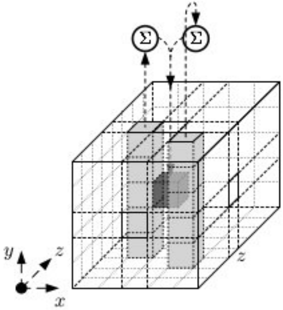


Figure: θ applied to a single row[23]

[23]Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January

ρ step

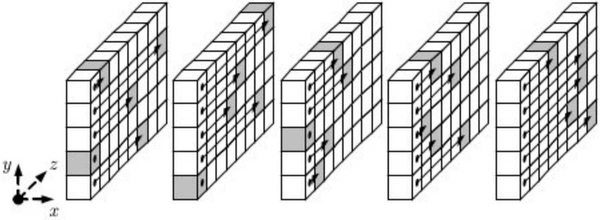


Figure: ρ transformation applied to lanes[24]

[24] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January 2011.

π step

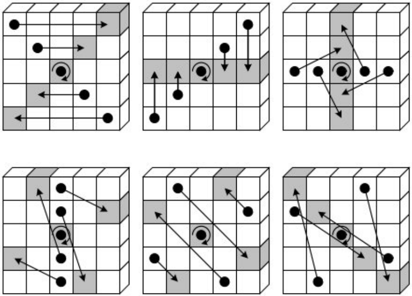


Figure: π applied to a single slice[25]

[25] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January 2011.

χ step

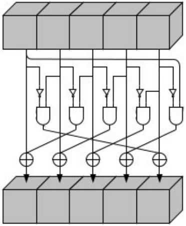


Figure: χ applied to a single row.[26]

[26] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January 2011.

ι step

- The round constants are given by
$$RC[i_r][0][0][2^j - 1] = rc[j + 7i_r] \text{ for all } 0 \leq j \leq l,$$
- $rc[t] = (x^t \; mod \; x^8 + x^6 + x^5 + x^4 + 1)$ mod x in GF(2)$[x]$

Rotational cryptanalysis in ARX

- Rotational cryptanalysis that studies propogation of rotational pair throug a primitive, is used on reduced version of Threefish, a core of Skein[27].
- Pair of two 1600-bit states $(A, A^{\leftarrow})$ are called rotational pair when each lane in state $A^{\leftarrow}$ is created by bitwise rotation of operation of corresponding lane in state A [28].

[27] Dmitry Khovratovich and Ivica Nikoli. Rotational cryptanalysis of arx. In Seokhie Hong and Tetsu Iwata, editors, Fast Software Encryption, volume 6147 of Lecture Notes in Computer Science, pages 333346. Springer Berlin Heidelberg, 2010.

[28] Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced keccak. Cryptology ePrint Archive, Report 2012/546, 2012. http://eprint.iacr.org/2012/546.pdf.

Definitions

1. Set S_n is a set of 2^{1600} pairs of states which are created by an operation of KECCAK-f[1600] applied to all possible rotational pairs.
2. Probability $p^n_{(x,y,z)}$ is the probability for pair of states $(A, A^{\leftarrow})$ randomly selected from the set S_n we have $A_{(x,y,z)} = A^{\leftarrow}_{(x,y,z+n)}$.
3. Given probability distribution $\mathcal{D}_n$ that assigns probability $\frac{1}{n!}$ for each $p \in \mathcal{P}_n$. A permutation is random, if chosen from $\mathcal{D}_n$.

Probability distribution and distinguisher

- Random permutation $p^{n}_{(x,y,z)}$ should follow binomial distribution $\mathcal{B}(t,s)$ that is equal to 0.5. Experimental values are supposed to fall within range of $0.5t \pm 2\sigma$ with 95% confidence interval.
- A 4 round rotational for Keccak was built, and 10,000 samples selected from it. $p^{54}_{(4,4,14)} = 0.5625$ had highest deviation, and the mean was 5682. Mean should not have exceeded from range of $\mathcal{B}(10000, 0.5)$ which is 5000$\pm$2.5.
- After four rounds however, $p^{n}_{(x,y,z)} = 0.5$, and hence the distinguisher cannot be directly extended.

Extend probabilities beyond 4 round Keccak

1. Find probability that relation between two pairs of states $(A_{(x,y,z)}, A^{\leftarrow}_{(x,y,z+n)})$ and $(A_{(x,y',z)}, A^{\leftarrow}_{(x,y'',z+n)})$ are observed, that should follow distribution $\mathcal{B}(10000, 0.5)$.
2. Bit wise operations like NOT, or rotation in Keccak do not affect the probabilities, but AND and XOR do.
3. AND operation $P_{out} = \frac{1}{2}(p_a + p_b - p_a p_b)$ [29]
4. XOR operation $P_{out} = p_a + p_b - 2p_a p_b$

[29] Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced keccak. Cryptology ePrint Archive, Report 2012/546, 2012. http://eprint.iacr.org/2012/546.pdf.

Extending probability beyond 4 rounds

- $p^{63}_{(2,1,37)}$ and $p^{63}_{(2,2,37)}$ have the highest deviation from 0.5 at end of fourth round.
- Probability that they are in the same relation is given by $p^{n}_{(x,y,z)}\dot{p}^{n}_{(x,y'',z)} + (1 - p^{n}_{(x,y,z)})(1 - p^{n}_{(x,y,z)})$ which is around 0.499024.
- To observe bias generate 403,000,000 samples obtained by selecting $\in$ error at 0.05 in Chernoff inequality

$$m \geq \frac{1}{(P_c - 0.5)^2} \ln \frac{1}{\sqrt{\in}}$$

Extending probability beyond 4 rounds

Algorithm 2 Find pair probabililty[30]

1: Generate 403,000,000 rotational pairs.
2: **for all** 403,000,000 rotational pairs **do**
3: Run Keccak for 5 rounds on states A and $A^{\leftarrow}$.
4: **if** $(A_{(1,2,43)} \oplus A^{\leftarrow}_{(1,2,44)} \oplus A_{(2,0,16)} \oplus A^{\leftarrow}_{(2,0,17)} = 0)$ **then**
5: mean := mean + 1
6: **end if**
7: **end for**
8: **return** mean

[30] Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced keccak. Cryptology ePrint Archive, Report 2012/546, 2012. http://eprint.iacr.org/2012/546.pdf.

Distinguisher Keccak and fin preimage

1. Mean for $\mathcal{B}(403000000, 0.5)$ with the standard deviation has range of 201,500,000$\pm$2.10037, but experimentally from above procedure it comes to around 201,450,503.
2. For primage, unknown message with cyclical property like 4 0's followed by 4 1's, alternatively of 512 bits is chosen. There are 256 possible message for rotational counterpart.
3. A rotational counterpart of the preimage is searched, that reduces complexity of random search.

Preimage for 3 rounds of Keccak [31] I

```
Guess first 8 lanes of A^←
Run Keccak-f[1600] for 3 rounds on state A^←
for for n:= 0 to n <64 do
    candidate := true
    for 10 sets of cordinates (x, y, z) being on list created on precomputation do
        if (p^n_(x,y,z) = 1) and (A_(x,y,z) = A^←_(x,y,z)) then
            candidate := false
        end if
        if (p^n_(x,y,z) = 0) and (A_(x,y,z) ≠ A^←_(x,y,z)) then
            candidate := false
        end if
```

Preimage for 3 rounds of Keccak [32] II

```
    end for
    if candidate = true then
        Rotate the guessed state by n bits
        Verify input to Keccak, that runs for 3 rounds.
    end if
end for
```

[32] Paweł Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced keccak. Cryptology ePrint Archive, Report 2012/546, 2012. http://eprint.iacr.org/2012/546.pdf.

How the search is better than random search

1. For guessing 512 bits, from 64 rotational pairs, the probability for guessing rotational counterpart is $A^{\leftarrow}$ is $2^{-512} \cdot 64 = 2^{-506}$.
2. There are 2^{256} messages of the cyclic pattern, and 10 sets of (x, y, z) cordinates for each of the rotational number.
3. The probability of the candidate having $p^n_{(x,y,z)}$ similar to that on list is 2^{-10}. So $2^{512}/2^{10} = 2^{502}$ number of checks are required at most.
4. You can extend this to 4 rounds of Keccak, but will have to drop the ι, round constant addition round, since it makes $p^n_{(x,y,z)} \neq 0, 1$.

Table of effort to find near collisions with random search

ϵ/n	Complexity ($\approx$)
128/256, 256/512, 512/1024	2^4
151/256, 287/512, 553/1024	2^{10}
166/256, 308/512, 585/1024	2^{20}
176/256, 323/512, 606/1024	2^{30}
184/256, 335/512, 623/1024	2^{40}
191/256, 345/512, 638/1024	2^{50}
197/256, 354/512, 651/1024	2^{60}

Table: Approximate complexity to find a ϵ/n-bit near collision by generic random search [34]

[34] Meltem Sönmez Turan and Erdener Uyan. Practical near-collisions for reduced round blake, fugue, hamsi and jh. Second SHA-3 conference, August 2010. http://csrc.nist.gov/groups/ST/hash/sha-

Near collisions found with hill climbing

1. Near collisions in which more than 75% of the bits were same for two different messages, were found for reduced rounds of BLAKE-32, Hamsi-256 and JH.
2. ϵ/n bit near collision for hash function for two messages M_1 and M_2, where $M_1 \neq M_2$ is defined as $HW(h(M_1, CV) \oplus h(M_2, CV)) = n - \epsilon$.
3. Hill Climbing starts with a random candidate, and then choosing a random successor that has a better fit to the solution. Ideally $HW(h(M, CV) \oplus h(M, CV + \delta)) = n/2$ where δ is n-bit vector with small Hamming weight.

How hill climbing works

1. Hill climbing algorithm will be to minimize the function $f_{M_1,M_2}(x) = HW(h(M_1,x) \oplus h(M_2,x))$.
2. CV is chosen as any random chaining value. Then the set of k-bit neighbours for the CV are created $S^k_{CV} = \{x \in \{0,1\}^n \mid HW(CV \oplus x) \leq k\}$.
3. Hill climbing is used to obtain k-optimum condition from k-bit neighbours $f_{M_1,M_2}(CV) = \min_{x \in S^k_{CV}} f_{M_1,M_2}(x)$.

Hill Climbing algorithm

Algorithm 3 Hill Climbing algorithm (M_1, M_2, k)

Randomly select CV
$f_{best} = f_{M_1,M_2}(CV)$
while (CV is not k-opt) **do**
  CV $=$ x such that $x \in S^k_{CV}$ with $f(x) < f(best)$
  $f_{best} = f_{M_1,M_2}(CV)$
end while
return (CV, f_{best})

Simulated annealing I

function $\textsc{Simulated-annealing}(M_1, M_2, CV, \text{schedule})$
    current $\leftarrow$ CV
    for t $=$ 1 to ∞ **do**
        T $\leftarrow$ schedule(t)
        if T $=$ 0 **then**
            return current
        end if
        next $\leftarrow$ a randomly selected successor from set $S^k_{current}$
        $\Delta E \leftarrow f_{M_1,M_2}(current) - f_{M_1,M_2}(next)$
        if ΔE $>$ 0 **then**
            current $\leftarrow$ next
        else

Simulated annealing II

current ← next, with probability $e^{\Delta E/T}$
end if
end for
end function

Tabu search I

function TABU-SEARCH($TabuList_{size}, M_1, M_2, CV$)
    $S_{best} \leftarrow CV$
    $TabuList \leftarrow$ null
    while S_{best} not k-opt **do**
        CandidateList $\leftarrow$ null
        $S_{neighbourhood} \leftarrow S^k_{S_{best}}$
        for $S_{candidate} \in S_{best_{neighbourhood}}$ **do**
            if ($\neg$ContainsAnyFeatures($S_{candidate}, TabuList$)) **then**
                CandidateList $\leftarrow S_{candidate}$
            end if
        end for

Tabu search II

```
        S_candidate ← LocateBestCandidate( CandidateList )
        if Cost( S_candidate ) ≤ Cost( S_best ) then
            S_best ← S_candidate
            TabuList ← featureDifferences(S_candidate, S_best)
            while TabuList > TabuList_size do
                DeleteFeature( TabuList )
            end while
        end if
    end while
    return S_best
end function
```

Random selection I

function $\textsc{Random-Selection}(M_1, M_2, CV,$ number_of_trials)
  current $\leftarrow$ CV
  trial $\leftarrow 0$
  while trial $<$ number_of_trials **do**
    next $\leftarrow$ randomly selected candidate from $S^k_{current}$
    if $f_{M_1,M_2}(next) - f_{M_1,M_2}(current)$ **then**
      current $\leftarrow$ next
    end if
  end while
  return current
end function

Hypothesis

- Reduced state Keccak, has better resistance to near collisions than BLAKE and Grøstl. For the attack algorithms hill climbing, simulated annealing, tabu search and random selection.
- Simulated annealing and tabu search, are better at finding near collisions compared to hill climbing and random selection.

Input

- The seed message is "The quick brown fox jumps over the lazy dog". 20 pairs are made from the string, for each of the three categories start, middle and end.
- The pair contains the original seed message and the seed message toggled for one bit. For each category 20 bits are toggled. The bits are toggled at the start, middle and end of the seed string.
- For input string 01100010 00011000, then in input file 1.txt; pair will be made with this seed string and **1**1100010 00011000, in file 2.txt the other string will be 0**0**100010 00011000.

Input

1. For the middle section, the bits are toggled to either side of the middle bit, in the string. For example for seed string 01100010 00011000, the file 1.txt will have seed and string 0110001**1** 00011000, and file 2.txt will have modified string as 01100010 **1**0011000.
2. For the ending section the bit toggling starts from the least significant bits, so for seed string 01100010 00011000, file 1.txt has updated string 01100010 0001100**1**, and file 2.txt have updated string 01100010 000110**1**0.

Output structure

1. The output folder is arranged to have the results in the following way starting from the upper most folder Output/chaining_value/collision_search/digest_size/SHA3_finalist/numbe
2. The results for the input message pair of the respective file are inserted to corresponding output file. Output for input file start/1.txt will be found in experiment parameter defined path and then start/1.txt

Ouput file

1. The file contains 8 parameters, 3 each for either success or failure. Success is defined as finding near collision that is more than 65% of the bits similar. The parameters are number of trials in success/failure, total iterations, and average iterations.
2. The rest 2 are total iterations and average iterations. These iteration numbers are mostly useful for hill climbing, since iterations for other algorithms are fixed.

Number of trials and iterations

1. The number of trials was initially kept at 128, but then increased to 256 to get more accurate numbers. In each trial of experiment, the chaining value is randomly selected.
2. The iterations are the number of times that that algorithm has to loop through for its' operation. This is chosen, over time required since differences in algorithm implementation may skew the time taken for finding effectiveness of the search algorithm.

Chaining value and neighbourhood

1. The chaining value length can be varied from 32, 64, 128, 256, 512 and 1024. We chose to do experiment with 32 bit chaining value, and then 64 bit chaining value.
2. Other higher chaining value lengths were not tried, given the expensive time computation, and relatively less success, in finding collisions.
3. k for k bit neighbourhood value is limited to less than equal to 2. Numbers above that are not optimal.

Demo

Demo

Observations on implementation

-

Hill climbing

Simulated annealing

Tabu search

Random selection

Validity of hypothesis

1. Both hypothesis are disproved. Keccak performs poorly for reduced rounds of 1, and 2. And equivalent performance to that of Grøstl and BLAKE, for rounds 3 and 4.
2. Simulated annealing does not perform as good as hill climbing for finding collisions in reduced rounds of the examined algorithm.
3. Random selection performance is almost comparable to simulated annealing.

Collision resistance in reduced rounds

1. For round 1 in all SHA-3 examined algorithm, collisions were found in almost all instances and trials.
2. For round 2, this behaviour is continued in Keccak for all collision search algorithm except tabu search. Thus Keccak is weaker for 2 rounds compared to BLAKE and Grøstl.
3. For rounds 3 and 4, Keccak seems to have equivalent resistance to that of BLAKE and Grøstl.

Feasibility of collision search algorithm

1. Hill climbing, with greedy ascent seems to be the most feasible search algorithm for finding near collisions, while tabu search is least feasible.
2. Hill climbing consistently finds more collisions than any other algorithm, with less iterations.
3. Random selection is comparable with simulated annealing, in finding near collisions in 3 selected hashing algorithms.

Effect of digest size

1. Collision resistance seems to be proportional to the digest size.
2. This could be due to exponential increase in the search space, it becomes harder to find better peaks for the search algorithm to move on.
3. A reason could be, that higher digest sizes have more internal state space, to execute the functions, thus making them more collision resistant.

Effect of number of rounds

1. The collision resistance increases as the number of rounds is increased.
2. No collisions are found in 256 trials, for digest sizes 384 and above; having processed more than 4 rounds, for any of the hashing algorithms.
3. It should be noted that for each algorithm permutation recommended permutation rounds differ. Grøstl has least number of permutation round, while Keccak has most of the permutation round.

Chaining value length

1. The effort to find collision, rises asymptotically for chaining value lengths.
2. Smaller chaining value lengths, like 32 bits are more effective in finding collision, than 64 bit chaining value.
3. Smaller chaining values, have smaller neighbourhoods to choose from thus increasing the feasibility.

Bit differences in particular position

1. On whole collision resistance is agnostic to bit difference in any specific position.
2. All the 3 algorithms inspected here with search algorithms, show little variation or bias, in finding collisions based on where the bit is updated.
3. The diffusion property seems to hold uniformly for all the 3 hashing algorithms.

Future work

1. The experiment can be repeated with Skein and JH, other SHA-3 finalist alongwith Keccak.
2. The state size can be reduced and experimented with.
3. Other parameters could manipulated could be, decrease digest size but increase rounds.
4. Instead of benchmark at 65% bits match for near collision, try for more exacting match for collisions.
5. Try to see a relationship between chaining values, if possible.

Acknowledgements

- Thanks to redditors p1mrx and deejaydarvin, and crypto stack exchange user Richie Frame for explanation of bit/byte ordering in Keccak.
- Crypto stack exchange user fgrieu for help in mix byte implementation in Grøstl.
- Larry Bugbee for fixing up his BLAKE Python implementation, which I used as reference for my Java implementation.
- Geoffrey De Smet for tabu lists document.
- Ted Hopp, for pointing out correct way of bit wise XOR for long in Java.
- Alexandre Yamajako for pointing specification to find inverse degree of Keccak function.

Questions

Questions?