Study of near collisions in reduced versions of BLAKE, Grøstl and Keccak

Soham Sadhu

Capstone Project Committee
Prof. Stanisław Raziszowski
Prof. Leon Reznik
Prof. Xumin Liu

August 22, 2014

Summary

- Reduced versions of BLAKE, Grøstl and Keccak, compared on basis of near collisions.
- Hill climbing, simulated annealing, tabu search and random selection used to find near collisions.

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.

Application of hash functions

1. Digital signature
2. Digital forensics [3]
3. Password storage
4. File integrity
5. Pseudo random string generator

[3] 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

1. SHA-0 proposed by NSA in 1993, later standardized by NIST.
2. SHA-1 designed by NSA in 1995, digest 160 bits, block size 512 bits [4].
3. SHA-2 released by NIST in 2001. Family of hash functions of size 224, 256, 384 and 512 bits.
4. Proposal for SHA-3 in FIPS PUB 202, based on Keccak, winner of SHA-3 competition.

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

FIPS PUB 202[5]

$$\text{SHA3-224}(M) = \text{KECCAK}[448](M \,||\, 01, 224)$$
$$\text{SHA3-256}(M) = \text{KECCAK}[512](M \,||\, 01, 256)$$
$$\text{SHA3-384}(M) = \text{KECCAK}[768](M \,||\, 01, 384)$$
$$\text{SHA3-512}(M) = \text{KECCAK}[1024](M \,||\, 01, 512)$$
$$\text{SHAKE128}(M, d) = \text{KECCAK}[256](M \,||\, 1111, d)$$
$$\text{SHAKE256}(M, d) = \text{KECCAK}[512](M \,||\, 1111, d)$$

[5] Information Technology Laboratory, National Institute of Standards and Technology. DRAFT FIPS PUB 202, SHA-3 Standard: Permutation Based Hash and Extendable-Output Functions. May 2014.

Sponge construction

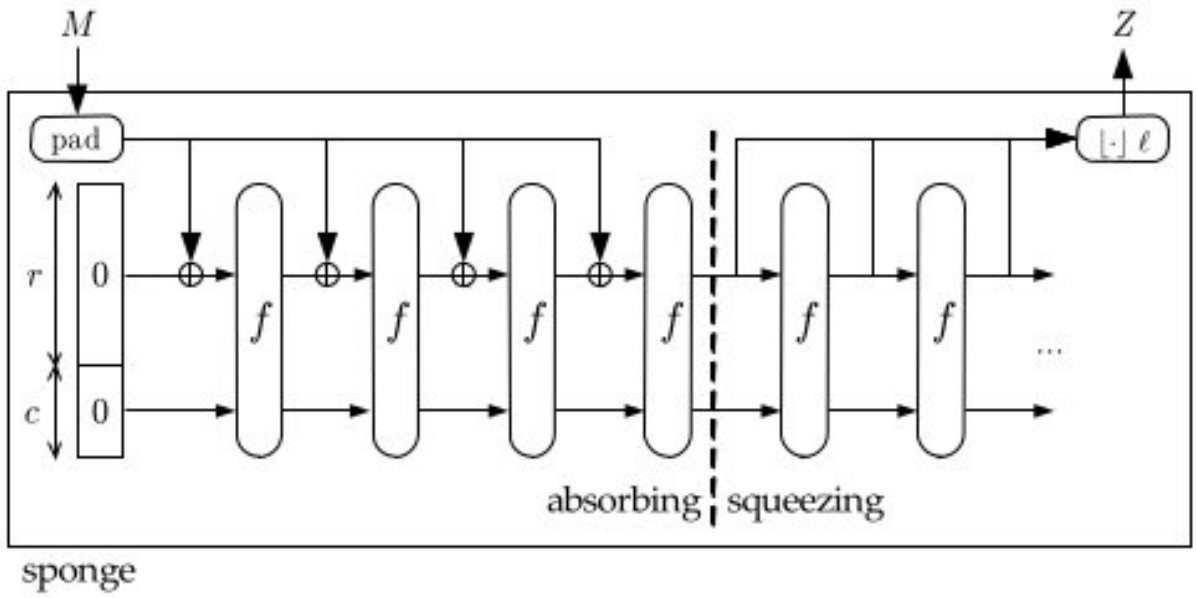
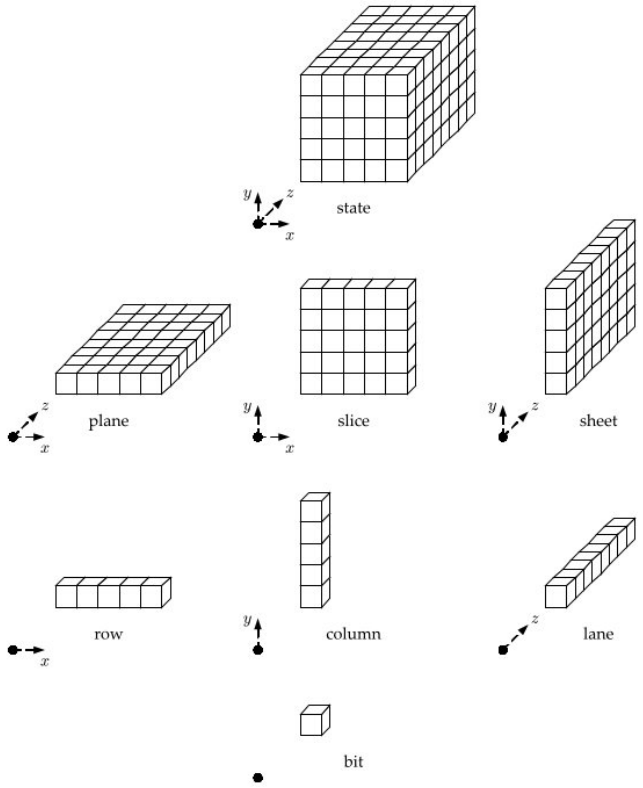


Figure 1: Sponge construction $Z = Sponge[f, pad, r](M, l)$[6]

[6] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. http://sponge.noekeon.org/CSF-0.1.pdf, January 2011.

State [7]



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

Permutation round

$$\theta : a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1],$$

$$\rho : a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2],$$

$$0 \leq t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } GF(5)^{2\times 2},$$

$$\pi : a[x][y] \leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

$$\chi : a[x] \leftarrow a[x] + (a[x+1] + 1)\, a[x+2],$$

$$\iota : a \leftarrow a + RC[i_r].$$

$$RC[i_r][0][0][2^j - 1] = rc[j + 7i_r] \text{ for all } 0 \leq j \leq l,$$

HAIFA construction

BLAKE is built on HAIFA (HAsh Iterative FrAmework) structure [8]which is an improved version of Merkle-Damgård function.

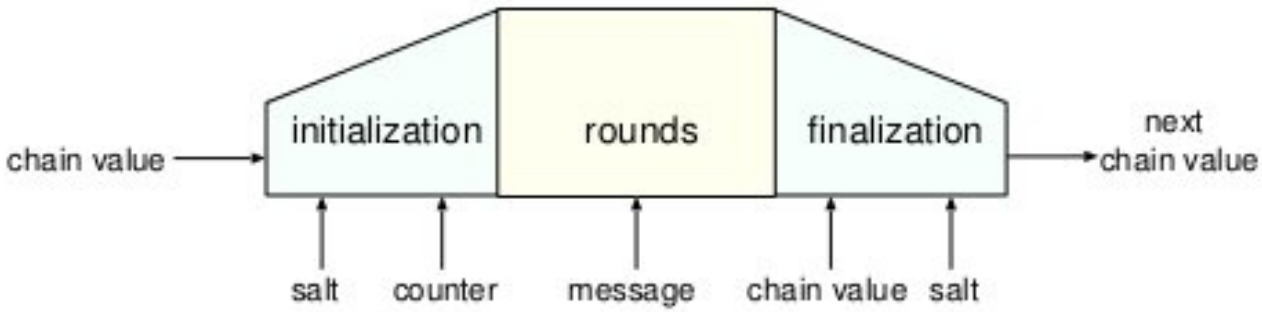


Figure 2: Local wide construction of BLAKE's compression function[9]

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

[9]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 $G_i(a, b, c, d)$

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

Permutation operations, ChaCha

$$a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)})$$
$$d \leftarrow (d \oplus a) \ggg [16, 32]$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg [12, 25]$$
$$a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)})$$
$$d \leftarrow (d \oplus a) \ggg [8, 16]$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg [7, 11]$$

$+$ addition in modulo $[2^{32}, 2^{64}]$
$\ggg k$ rotate to right by k bits
$\oplus$ bitwise XOR
r permutation round

Finalization

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

Hashing the message

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

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

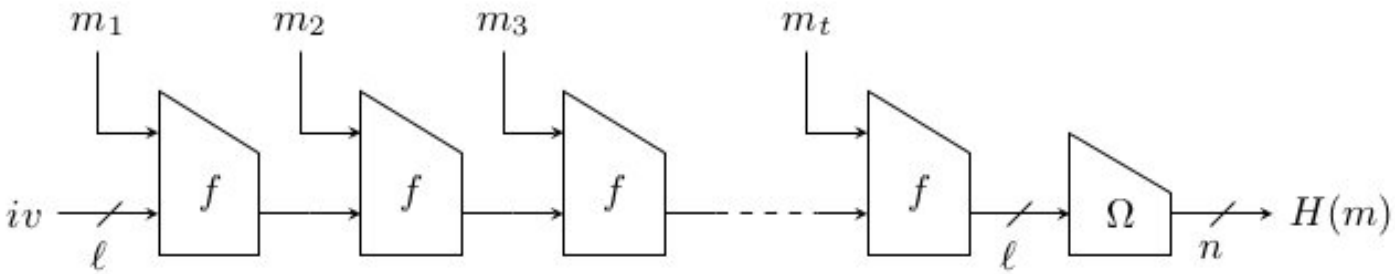


Figure 3: Grøstl hash function [10]

[10] Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram. Grstl - a sha-3 candidate version 2.0.1. http://www.groestl.info/Groestl.pdf

Permutation f function

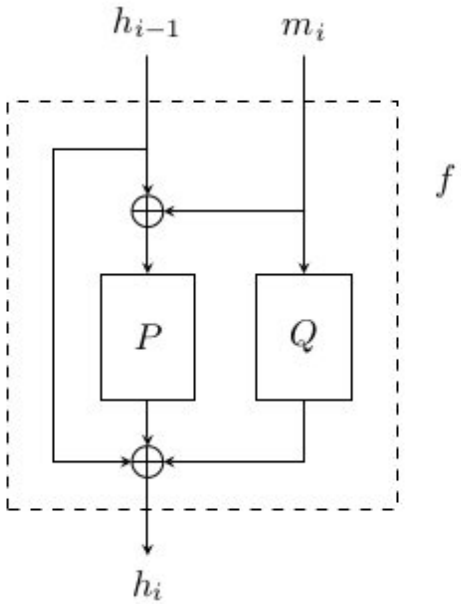


Figure 4: P and Q are $l - bit$ permutations [11]

[11] Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram. Grstl - a sha-3 candidate version 2.0.1. http://www.groestl.info/Groestl.pdf

Rounds

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

$A \leftarrow A \oplus C[i]$ 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

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

Substition and mixing

Substitution $a_{i,j} \leftarrow S(a_{i,j}), 0 \leq i < 8, 0 \leq j < v$; where S is S-box is from AES.
Mixing: $A \leftarrow B \times A$ where B is a finite field over $\mathbb{F}_{256}$ defined over $\mathbb{F}_2$ by polynomial $x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$.

Shift byte

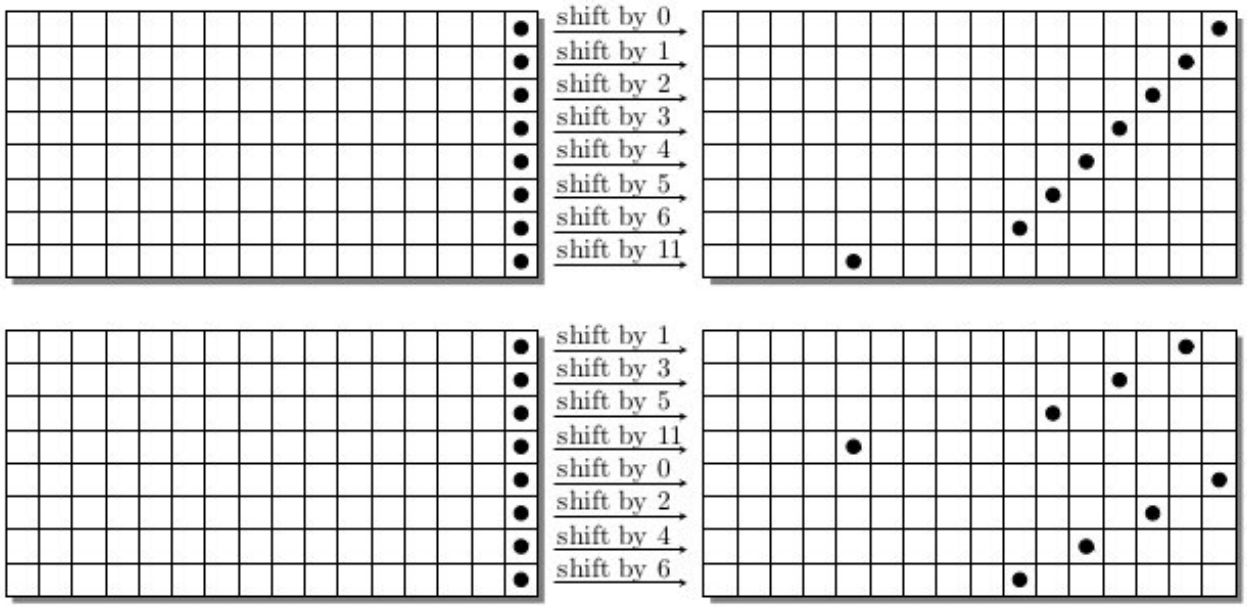


Figure 5: P_{1024}(top) and Q_{1024}(bottom)[12]

[12]Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram. Grstl - a sha-3 candidate version 2.0.1. http://www.groestl.info/Groestl.pdf

Collisions

1. Collision in hash function, is when we have two different messages hashing to the same value, using same initial value.
2. Full collision is when all the bits agree in the hash output for two different messages.
3. Semi-free start collision is where you change the initial value and obtain collisions for two different message.

Near collisions with hill climbing

1. A 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 $HW(h(M_1, CV) \oplus h(M_2, CV)) = n - \epsilon$.
2. 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.

Hill Climbing algorithm [13]

Algorithm 1 Hill Climbing algorithm (M_1, M_2, k)

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

[13] 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-3/Round2/Aug2010/documents/papers/TURAN_Paper_Erdener.pdf.

Simulated annealing, tabu search and random selection

1. Simulated annealing poor choices made initially with higher probability to escape local minima.
2. Tabu search stores a list of heuristics in tabu list. Pick the best solution from space, and compare with tabu list to avoid stuck in local minima.
3. Random selection just randomly selects a solution, good or bad till given number of trials.

Hypothesis

- Confirmed
  1. State size has no effect on efficiency of Keccak permutation rounds.
- Disproved
  1. Reduced state Keccak, has better resistance to near collisions than BLAKE and Grøstl, for search algorithms hill climbing, simulated annealing, tabu search and random selection.
  2. Simulated annealing and tabu search, are better at finding near collisions compared to hill climbing and random selection.

Input and output

1. 60 pairs made from seed string "The quick brown fox jumps over the lazy dog" by toggling a bit in seed, and pairing with seed.
2. 20 bits toggled at the start, middle and end of seed each.
3. Output folder is arranged Output/CV/search/size/SHA3/rounds/category/file.
4. Output file contains 8 parameters, 3 each for either success or failure. The rest 2 are total iterations and average iterations.

Chaining value and number of trials

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. Chaining values of 32 and 64 bits.
3. k for neighbourhood value is limited to less than equal to 2.

Demo

Demo

Iterations for BLAKE

Digest Size	Rounds			
	1	2	3	4
224	803	891	886	883
256	804	891	892	885
384	1137	898	899	902
512	1118	902	903	906

Table 1: Hill Climbing for BLAKE for CV bit length 32

Digest Size	Rounds		
	1	2	3
224	4190	3465	3483
256	4264	3456	3431
384	3885	3515	3528
512	3984	3535	3559

Table 2: Hill Climbing for BLAKE for CV of bit length 64

Iterations for Grøstl

Digest Size	Rounds			
	1	2	3	4
224	875	888	886	885
256	889	887	891	889
384	872	897	898	897
512	896	904	904	902

Table 3: Hill Climbing for Grøstl for CV of bit length 32

Digest Size	Rounds		
	1	2	3
224	3687	3468	3469
256	3714	3479	3473
384	3594	3522	3512
512	3581	3544	3559

Table 4: Hill Climbing for Grøstl for CV of bit length 64

Iterations for Keccak

Digest Size	Rounds			
	1	2	3	4
224	532	880	883	888
256	532	888	889	895
384	533	892	899	904
512	533	900	905	902

Table 5: Hill Climbing for Keccak for CV of bit length 32

Digest Size	Rounds		
	1	2	3
224	2118	3535	3457
256	2118	3557	3466
384	2134	3676	3521
512	2139	3764	3562

Table 6: Hill Climbing for Keccak for CV of bit length 64

Iterations for other search algorithms

1. Number of iterations for random simulation, simulated annealing for 32 bit chaining value is fixed at 1024 iterations.
2. For 64 bit chaining value, iterations are set to 11 times the digest size. That is 2464, 2816, 4224, 5632 iterations for digest sizes 224, 256, 384 and 512 bits.
3. For tabu search for Grøstl digest length 224, and for rounds 1 and 2, averaged in iterations about 335254.443.

Iterations for Keccak state reduced 200 bits

Digest Size	Rounds			
	3	4	5	6
224	888	886	887	886
256	890	887	892	886
384	901	899	897	898
512	904	903	899	903

Table 7: Average iterations over all input cases for Hill Climbing for Keccak state reduced to 200 bits for chaining value of bit length 32

Iterations for Keccak state reduced 400 bits

Digest Size	Rounds			
	3	4	5	6
224	886	888	887	889
256	892	886	891	893
384	893	899	898	899
512	906	902	905	904

Table 8: Average iterations over all input cases for Hill Climbing for Keccak state reduced to 400 bits for chaining value of bit length 32

Iterations for Keccak state reduced 800 bits

Digest Size	Rounds			
	3	4	5	6
224	883	886	888	882
256	891	890	891	887
384	896	901	898	899
512	901	905	901	905

Table 9: Average iterations over all input cases for Hill Climbing for Keccak state reduced to 800 bits for chaining value of bit length 32

Iterations for 75% bits matching collision

Rounds	1				2			
Digest size	224	256	384	512	224	256	384	512
BLAKE	803	814	1137	1121	887	889	898	908
Grøstl	874	885	877	897	887	890	902	905
Keccak	532	532	533	533	882	885	897	902
Keccak-200	994	977	959	957	899	902	903	897
Keccak-400	836	899	972	955	886	903	903	906
Keccak-800	568	574	574	908	894	920	947	918

Table 10: Average iterations over all input cases for Hill Climbing for variations of Keccak and other hashing algorithms. Chaining value is bit length 32, and the near collision is 75% bit match.

Near collisions BLAKE

Digest Size	Rounds			
	1	2	3	4
224	41/306	13/5	34/8	25/5
256	39/256	4/3	10/4	3/2
384	58/384	0/0	0/0	0/0
512	58/384	0/0	0/0	0/0

Table 11: Near collisions BLAKE with 32 bit CV

Digest Size	Rounds		
	1	2	3
224	56/384	38/13	43/10
256	55/384	9/3	16/4
384	60/284	0/0	0/0
512	59/384	0/0	0/0

Table 12: Near collisions BLAKE with 64 bit CV

Near collisions Grøstl

Digest Size	Rounds			
	1	2	3	4
224	10/6	16/6	29/7	33/9
256	1/2	6/3	10/3	7/3
384	12/128	0/0	0/0	0/0
512	22/145	0/0	0/0	0/0

Table 13: Near collisions Grøstl with 32 bit CV

Digest Size	Rounds		
	1	2	3
224	43/23	40/12	49/11
256	25/10	14/5	15/3
384	14/256	0/0	0/0
512	28/158	0/0	0/0

Table 14: Near collisions Grøstl with 64 bit CV

Near collisions Keccak with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	60/384	60/384	40/11	33/8
256	60/384	60/384	11/4	8/3
384	60/384	60/384	1/1	0/0
512	60/384	60/384	0/0	0/0

Table 15: Near collisions Keccak with 32 bit CV

Digest Size	Rounds		
	1	2	3
224	60/384	60/384	55/21
256	60/384	60/384	20/7
384	60/384	60/384	1/1
512	60/384	60/384	0/0

Table 16: Near collisions Keccak with 64 bit CV

Collision for 75% bit match

Rounds	1				2			
Digest size	224	256	384	512	224	256	384	512
BLAKE	26/512	23/512	22/400	21/266	0/0	0/0	0/0	0/0
Grøstl	0/0	0/0	7/256	8/256	0/0	0/0	0/0	0/0
Keccak	60/768	60/768	60/768	60/768	60/768	60/768	60/768	60/768
Keccak-200	20/256	20/256	1/1	0/0	1/1	0/0	0/0	0/0
Keccak-400	20/256	20/256	20/256	4/2	9/123	7/99	0/0	0/0
Keccak-800	60/768	60/768	60/768	20/256	60/733	60/740	60/538	0/0

Table 17: Collisions for 75% bit matching, for 32 bit chaining value. Application of hill climbing search. Collision instances for start, middle and end are grouped together.

Near collision numbers for Keccak internal state of 200 bits

Size	Rounds			
	3	4	5	6
224	33/8	30/6	30/8	31/9
256	12/4	5/3	8/3	11/4
384	0/0	0/0	0/0	0/0
512	0/0	0/0	0/0	0/0

Table 18: Collisions for Keccak state reduced to 200 bits, with hill climbing for 32 bit chaining value.

Near collision numbers for Keccak internal state of 400 bits

Size	Rounds			
	3	4	5	6
224	19/7	33/9	27/8	30/8
256	6/3	7/3	3/3	6/3
384	0/0	0/0	0/0	0/0
512	0/0	0/0	0/0	0/0

Table 19: Collisions for Keccak state reduced to 400 bits, with hill climbing for 32 bit chaining value.

Near collision numbers for Keccak internal state of 800 bits

Size	Rounds			
	3	4	5	6
224	38/9	27/9	32/9	32/7
256	7/4	9/5	5/2	8/4
384	2/2	0/0	0/0	0/0
512	0/0	0/0	0/0	0/0

Table 20: Collisions for Keccak state reduced to 800 bits, with hill climbing for 32 bit chaining value.

Near collisions BLAKE with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	39/256	0/0	1/1	1/1
256	38/256	0/0	0/0	1/1
384	39/353	0/0	0/0	0/0
512	50/342	0/0	0/0	0/0

Table 21: Near collisions BLAKE with 32 bit CV

Near collisions Grøstl with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	1/1	1/1	2/2	0/0
256	0/0	0/0	1/1	1/1
384	12/128	0/0	0/0	0/0
512	15/129	0/0	0/0	0/0

Table 22: Near collisions Grøstl with 32 bit CV

Digest Size	Rounds		
	1	2	3
224	0/0	0/0	0/0
256	0/0	0/0	0/0
384	13/256	0/0	0/0
512	19/129	0/0	0/0

Table 23: Near collisions Grøstl with 64 bit CV

Near collisions Keccak with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	60/384	60/384	0/0	0/0
256	60/384	60/384	0/0	1/1
384	60/384	60/384	0/0	0/0
512	60/384	60/384	0/0	0/0

Table 24: Near collisions Keccak with 32 bit chaining value

Near collisions BLAKE with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	41/291	10/5	24/6	28/6
256	39/256	3/2	4/2	7/3
384	57/384	0/0	0/0	0/0
512	58/384	0/0	0/0	0/0

Table 25: Near collisions BLAKE with 32 bit chaining value

Near collisions Grøstl with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	7/6	15/4	26/7	20/7
256	0/0	0/0	3/2	3/3
384	12/128	0/0	0/0	0/0
512	22/148	0/0	0/0	0/0

Table 26: Near collisions Grøstl with 32 bit chaining value

Near collisions Keccak with 32 bit chaining value

Digest Size	Rounds			
	1	2	3	4
224	60/384	60/384	30/11	24/5
256	60/384	60/384	13/4	8/3
384	60/384	60/384	0/0	0/0
512	60/384	60/384	0/0	0/0

Table 27: Near collisions Keccak with 32 bit chaining value

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.

State size reduction for Keccak

1. State size reduction does not negatively affect the security margin of Keccak.
2. This can be explained on the basis that lowering bit rate increases squeezing and absorbing phase, for a message to be hashed, thus subjecting it to more permutation rounds.
3. For low permutations rounds like 1, this increases margin of Keccak reduced rounds, but it gradually plateaus.

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. Other parameters could manipulated could be, decrease digest size but increase rounds.
3. Instead of benchmark at 65% bits match for near collision, try for more exacting match for collisions.
4. Try to see a relationship between chaining values, if possible.

Acknowledgements

- Thanks to Prof. Radziszowski for his patience and guidance.
- 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 updating his BLAKE Python implementation.
- Geoffrey De Smet for tabu lists document.
- Ted Hopp, for pointing out bit wise XOR for long in Java.
- Alexandre Yamajako for pointing specification to find inverse degree of Keccak function.

References

1. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak reference. http://keccak.noekeon.org/Keccak-reference-3.0.pdf, January 2011.
2. Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Blake. http://www.131002.net/blake/blake.pdf, April 2012.
3. Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram. Grstl - a sha-3 candidate version 2.0.1. http://www.groestl.info/Groestl.pdf, March 2011.
4. Meltem Sönmez Turan and Erdener Uyan. Practical near-collisions for reduced round blake, fugue, hamsi and jh.

Questions

Questions?