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

Soham Sadhu

May 25, 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

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.

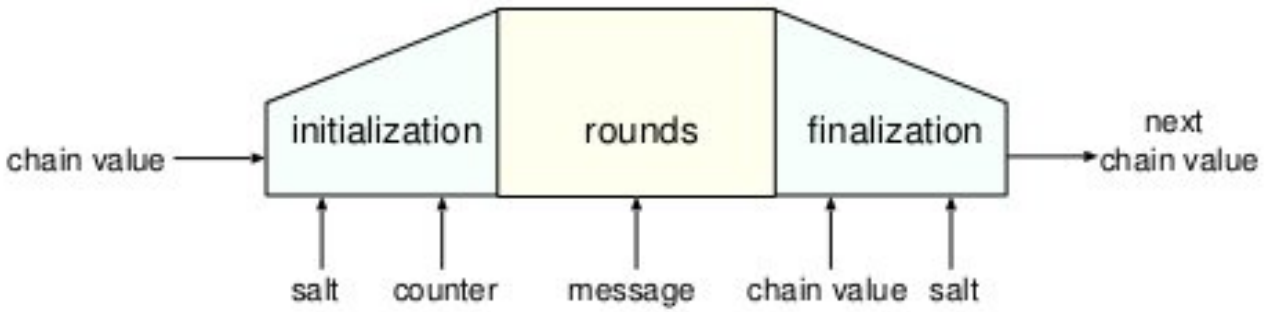


Figure: Local wide construction of BLAKE's compression function[11]

[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** $i = 0, \ldots, N-1$ **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 $G_i(a, b, c, d)$

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

BLAKE permutation operation for 224 and 256 variant

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

$+$ addition in modulo 2^{32}

$\ggg k$ rotate to right by k bits

$\oplus$ bitwise XOR

r permutation round

i is from G_i

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

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

$+$ 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 Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram.

Hashing the message

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

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

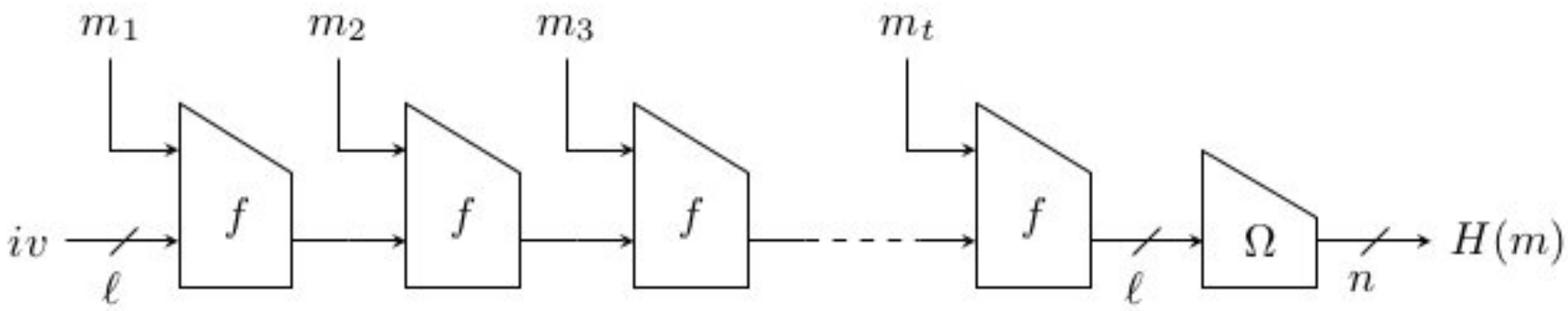


Figure: Grøstl hash function [15]

[15] Søren Steffen Thomsen, Martin Schläffer, Christian Rechberger, Florian Mendel, Krystian Matusiewicz, Lars R. Knudsen, and Praveen Gauravaram.

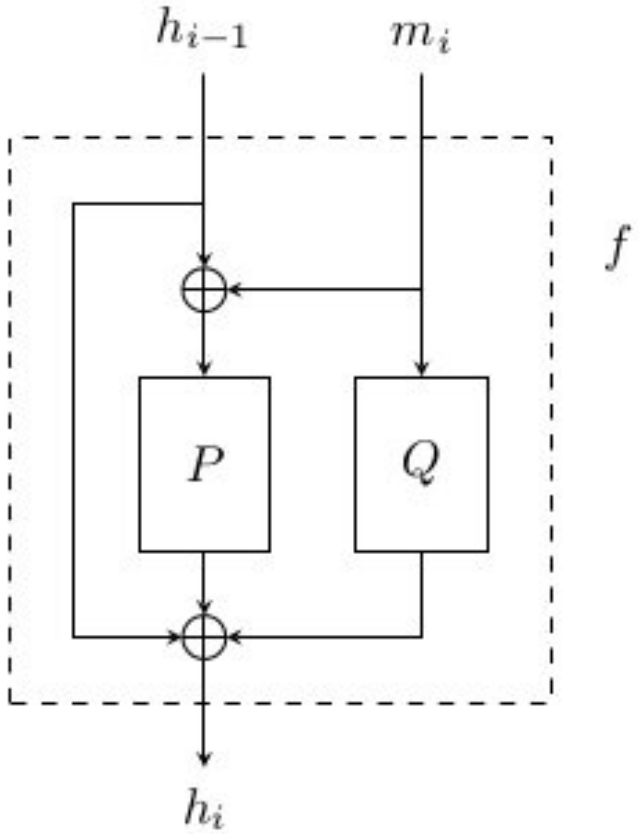
Input mapping

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

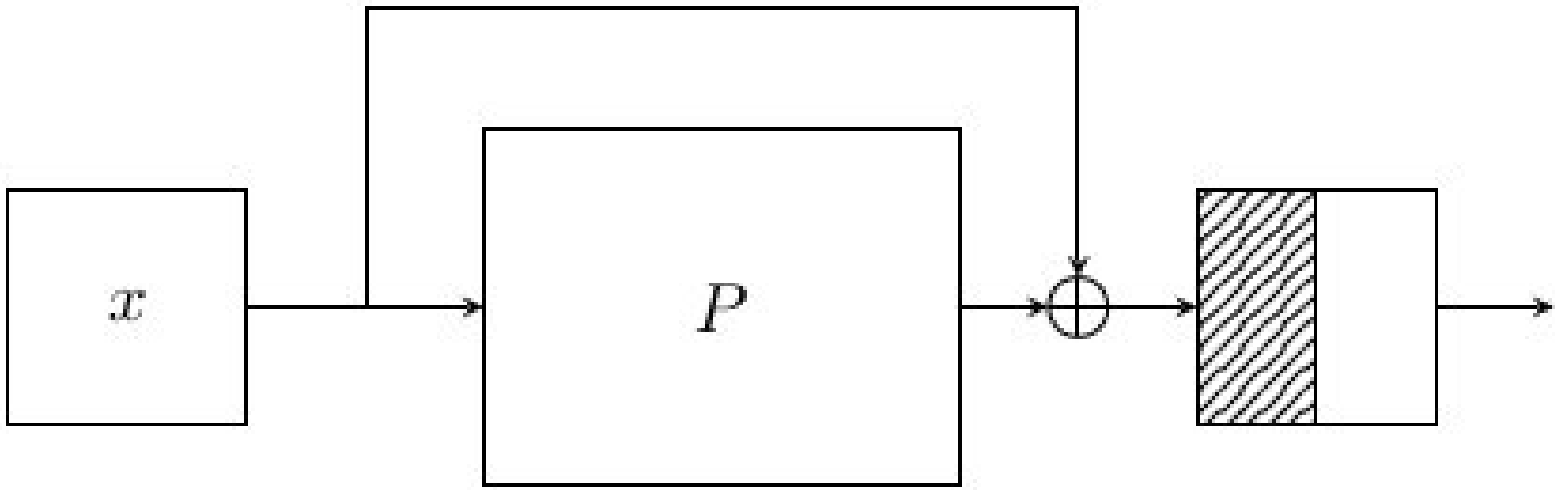
$$f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h.$$



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

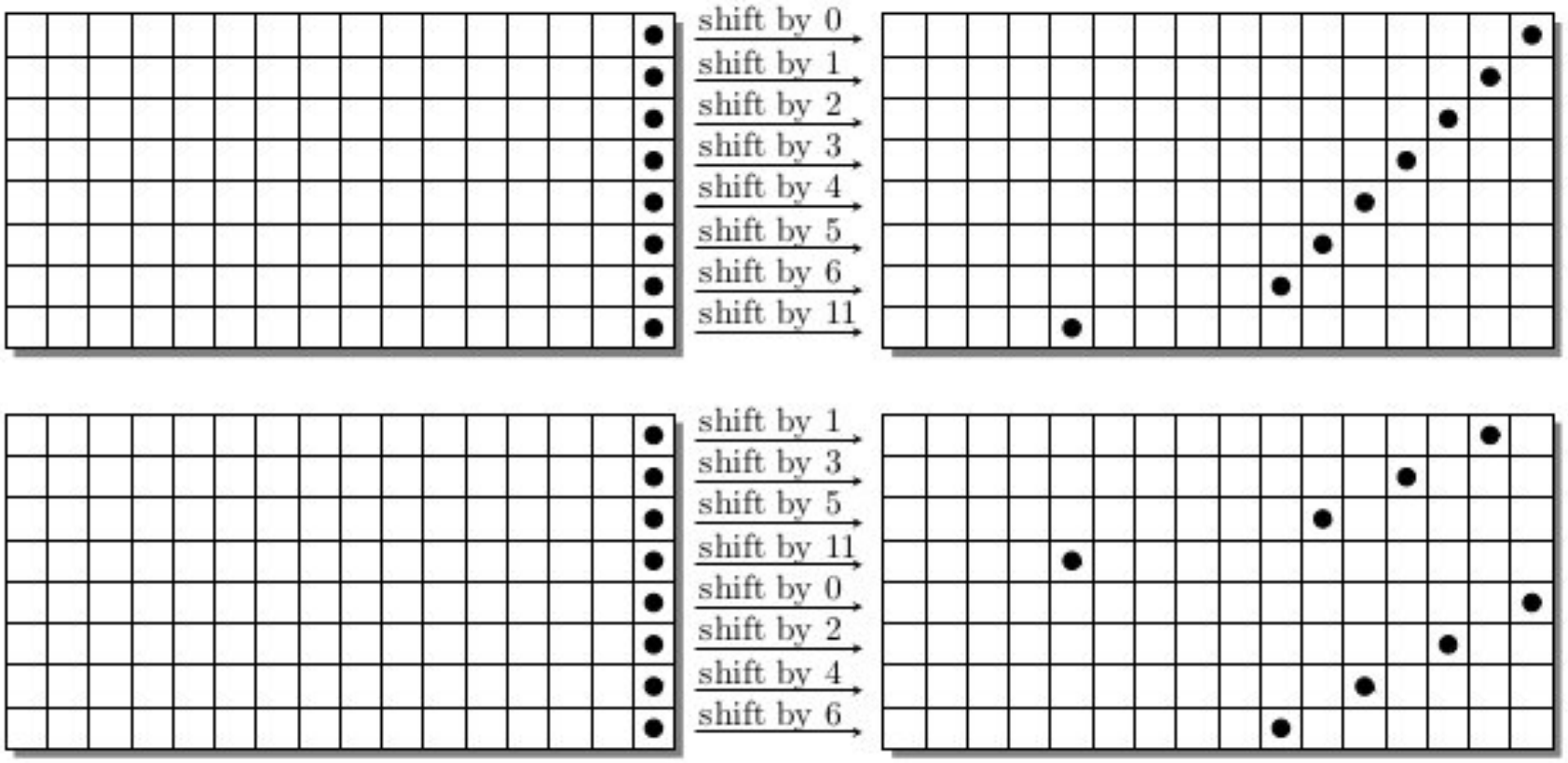


Figure: ShiftBytes transformation of permutation P_{1024}(top) and Q_{1024}(bottom)[20]

Mix byte

$$A \leftarrow B \times A$$

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