Keccak[$r = 20$, $c = 80$, $n_r = 2$] におけるビット頻度および原像計算時間に関する実験

戚 莘鈺[1,a)]　藤岡 淳[1]　青木 和麻呂[2]

概要：Keccak は, スポンジ構造を採用し, 吸収フェーズ (absorbing phase) と搾出フェーズ (squeezing phase) に分かれ, 内部状態は, 入出力に直接影響する部分のレート (r), 入出力に直接は影響しない部分のキャパシティ (c) からなり, 計 $r + c$ ビットである. そして, ブロック置換関数により, 内部状態をかき混ぜることが n_r 回 (ラウンド数) 行われ, 出力長に応じて更にブロック置換が繰り返される. 提案者は, Keccak[$r = 40$, $c = 160$, $n_r = 1$] のように表現している. 本研究では, まだ有効な攻撃方法がない Keccak のパラメータと内部状態の組である $c/(r + c)$ が大きい原像攻撃耐性を評価することを目的とする. そのために, Keccak[$r = 20$, $c = 80$, $n_r = 2$] に対して, Hash 値におけるビット頻度やハッシュ値の各ビットにおけるビット頻度などを調べる. さらに, 最適化ソルバを利用した解析プログラムを作成し, ビット頻度が原像計算にかかる時間にどのように影響しているかを考察する.

キーワード：Keccak，原像攻撃耐性，キャパシティ比，ビット頻度，最適化ソルバ

Experiments on bit frequency and pre-image computation time in Keccak[$r = 20$, $c = 80$, $n_r = 2$]

Xinyu Qi[1,a)]　Atsushi Fujioka[1]　Kazumaro Aoki[2]

Abstract: Keccak adopts a sponge construction and is divided into an absorbing phase and a squeezing phase. Keccak has an internal state with a total length of $r + c$ bits, which consists of a rate (r) part that directly affects input/output and a capacity (c) part that does directly not affect input/output. Then, by the block permutation function, the internal state is mixed in n_r times (rounds), and further block permutation is applied according to the output length. The proposers express it as the style, Keccak[$r = 40$, $c = 160$, $n_r = 1$]. In this paper, we aim at pre-image attack resistance with large $c/(r + c)$ in Keccak's parameter, which has no effective attack method yet. For that purpose, we check bit frequency in the Hash value and bit frequency in each bit of the Hash value for Keccak[$r = 20$, $c = 80$, $n_r = 2$]. In addition, we make an analysis program using optimization solver and consider how the bit frequency of the pre-image affects the time required for pre-image derivation.

Keywords: Keccak, Pre-image resistance, Capacity Ratio, Bit Frequency, Optimization Slver

1. はじめに

一方向性 Hash 関数は, 任意の長さの入力を（通常は）固定長の出力に変換する Hash 関数である. Hash 値から元のメッセージを推測できない, 異なるメッセージに対し異なる Hash 値が対応する, といった特徴を持ち, 改竄の検出や電子署名などに用いられている. Keccak のパラメータの一

[1] 神奈川大学 大学院 工学研究科 工学専攻
Course of Engeneering, Graduate School of Engeneering, KANAGAWA University
[2] 文教大学 情報学部 情報システム学科
Department of Information Systems, Faculty of Information and Communications, BUNKYO University
[a)] r202070124jp@jindai.jp

部は 2015 年にアメリカ国立標準技術研究所 (NIST) により SHA-3 として選出された.

1.1 Keccak の構成

Keccak はスポンジ構造 (sponge construction) を採用し, 吸収フェーズ (absorbing phase) と搾出フェーズ (squeezing phase) に分かれ, 内部状態は, 入出力に直接影響する部分のレート (r), 入出力に影響しない部分のキャパシティ (c) からなり, 合計 $r+c$ ビットの長さの内部状態を持つ [1]. メッセージのブロックは内部状態の初期値との xor を取って, ブロック置換が行われる (図 1).

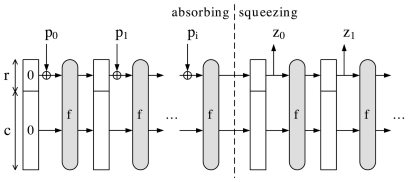


図 1 Keccak のスポンジ構造
出典: Wikipedia (`https://en.wikipedia.org/wiki/SHA-3`)

ブロック置換では, $r+c$ ビットの内部状態をかき混ぜることが n_r 回 (ラウンド数) 行われる. 基本的なブロック置換関数は 5 つある. 擬似コードは以下の通り.

Algorithm ブロック置換の擬似コード

$Round[b](A, RC)\{$
θ step
$C[x] = A[x,0]\ xor\ A[x,1]\ xor\ \cdots\ xor\ A[x,4], \quad for\ x\ in\ 0,\ldots,4$
$D[x] = C[x-1]\ xor\ rot(C[x+1],1), \quad for\ x\ in\ 0,\ldots,4$
$A[x,y] = A[x,y]\ xor\ D[x], \quad for\ (x,y)\ in\ ((0,0),\ldots,(4,4))$
ρ step and π step
$B[y, 2\times x + 3\times y] = rot(A[x,y], r[x,y]),$
$for(x,y)\ in\ ((0,0),\ldots,(4,4))$
χ step
$A[x,y] = B[x,y]\ xor\ (not\ B[x+1,y]\ and\ B[x+2,y]),$
$for(x,y)\ in\ ((0,0),\ldots,(4,4))$
ι step
$A[0,0] = A[0,0]\ xor\ RC$
$return A$
$\}$

擬似コードの rot 以外の操作に, x, y は mod 5 で行われている. $A[x,y]$ は各ステップ state 状態のビット列 ($r+c$/25 bits) を示し, $B[x,y]$, $C[x]$, $D[x]$ は中間変数である. $r[x,y]$ は Keccak の仕様で規定されている回転定数である, $rot(W,r)$ は位置 i のビットを位置 $i+r\left(\mathrm{mod}\ \dfrac{r+c}{25}\right)$ に移動する回転演算である. RC はラウンドによるラウンド定数である.

1.2 攻撃コンテスト

Keccak team による攻撃コンテスト (The Keccak Crunchy Crypto Collision and Pre-image Contest) では, SHA-3 仕様通りの Keccak ではなく, ラウンド数 n_r を削減し, 内部状態が異なる (様々な c や r に対する) 各種の Keccak のバリアント (variant) を対象として, 衝突耐性に対する衝突 (collision) 攻撃と特定の Hash 値を持つメッセージを探索する原像 (pre-image) 攻撃を公募している. 攻撃コンテストで, 攻撃対象とする Keccak のバリアントはラウンド数は 1 から 12 まで, キャパシティ (c) として 160bits を指定し, 内部状態の長さ ($r+c$) を 200, 400, 800, 1600 としている.

2. 関連研究

2013 年, Morawiecki は SAT solver を利用し, SAT ベース攻撃により, コンテスト中の Keccak に対する原像攻撃に成功した [2].

同年, Morawiecki は差分解読法を利用し, コンテスト中の 3 ラウンド Keccak に対する原像攻撃に成功した [3].

Guo は 2016 年に, 線形構造法 (linear structure) を利用し, Keccak に対する原像攻撃の時間計算量を削減した [4]. 2017 年, Ting Li は Guo の攻撃手法に基づき, 交差線形構造法 (cross-linear structures) を発表し, 原像攻撃の時間計算量を更に削減した [5].

表 1 関連研究

提案者	バリアント	攻撃手法	発表年
Morawiecki	Keccak[$r=240$, $c=160$, $n_r=1/n_r=2$] Keccak[$r=1440$, $c=160$, $n_r=1/n_r=2$] Keccak[$r=640$, $c=160$, $n_r=1/n_r=2$] Keccak[$r=640$, $c=160$, $n_r=1/n_r=2$] Keccak[$r=1440$, $c=160$, $n_r=1/n_r=2$]	SAT ベース攻撃	2013
Morawiecki	Keccak[$r=576$, $c=1024$, $n_r=3$] Keccak[$r=1024$, $c=576$, $n_r=3$]	差分解読法	2013
Guo	Keccak[$r=576$, $c=1024$, $n_r=2$] Keccak[$r=1344$, $c=256$, $n_r=3$] Keccak[$r=640$, $c=160$, $n_r=3$] Keccak[$r=1440$, $c=160$, $n_r=3/n_r=4$]	線形構造法	2016
Li	Keccak[$r=240$, $c=160$, $n_r=3$]	交差線形構造法	2017

一方, Keccak[$r=40$, $c=160$, $n_r=2$] のような, 2 ラウンド以上で 内部状態における c の割合 ($c/(r+c)$. 以降, 本稿では, キャパシティ比と呼ぶ) が大きい Keccak バリアントへの攻撃の発表はされていない.

3. 目標

攻撃コンテストにおいて, キャパシティ比が大きい 2 ラウンドの Keccak バリアントに対しては, まだ有効な攻撃方法は発表されていない.

そこで, Keccak[$r=40$, $c=160$, $n_r=2$] に対する原像攻撃耐性の検証を最終目的とし, 同じキャパシティ比である 2 ラウンドの小さいバリアント Keccak[$r=20$, $c=80$, $n_r=2$] における実験を行い, 特徴の抽出よりキャパシティ比が大きい 2 ラウンドの Keccak に対する Hash 値および

原像のビット頻度が攻撃にかかる時間にどのように影響しているかを考察する.

4. 方針

予備実験で Hash 値から原像を求める計算プログラムを実装し, Keccak の以下の各バリアントに対する Hash 値から原像を求める計算実験 10 回から 1000 回程度行った.

Keccak[$r = 40$, $c = 160$, $n_r = 1$] 入力 block $= 1$
Keccak[$r = 40$, $c = 160$, $n_r = 1$] 入力 block $= 2$
Keccak[$r = 40$, $c = 160$, $n_r = 1$] 入力 block $= 3$

実験で Hash 値により原像攻撃の計算時間は 1,000 倍以上変化することが確かめられた. Keccak[$r = 40$, $c = 160$, $n_r = 2$] に対し, 時間計算量が大きいので, Hash 値から原像を工夫なしに求めるのは難しい.

そこで, まずは小さいバリアント Keccak[$r = 20$, $c = 80$, $n_r = 2$] に対し, 原像攻撃時間が少ない Hash 値の特徴を抽出する.

計算時間が少ない Hash 値および原像の特徴を捉えることを目標とし, 実験を二つに分けて行う.

入力メッセージ block の増加により, Hash 値から原像を求める計算時間が増えるので, 入力は 1 block の時, 計算時間が少ない.

まず実験 1 は入力が 1 block の時, 全ての入力 (2^{20} 個) における Hash 値を対象として実験する.

そして, 実験 2 は入力を 1 block とすることに加え, 更に計算時間の少ない Hash 値および原像に対象として実験する.

5. 実験準備

今回の実験は 1 台の PC (8-thread 並列) で行った. 実験に用いた PC の仕様は以下の通り (表 2).

表 2 hardware/software

PC	OS	Windows 10
	CPU	Intel Core i7-6700 CPU @ 3.40GHz
	Physical Core	4
	Logical Processor	8
	RAM	3957MB
	Thread Count	8
Programming language		JAVA
Solver		Gurobi Optimizer 9.5.0

Gurobi Optimizer [6] は最適化ソルバであり, 線形計画法, 二次計画法, 二次制約付き計画法, 混合整数線形プログラミング, 混合整数二次計画法, および混合整数二次制約付き計画法を実現している.

Keccak の各ステップを二進変数 (Hash 値および原像のビット表現) の式として表し, Gurobi Optimizer を利用して, それらの連立方程式を解き, Hash 値から, 原像を求める.

6. 実験 1

6.1 実験 1.0

6.1.1 対象データ

Keccak[$r = 20$, $c = 80$, $n_r = 2$, block $= 1$] 1 block 全ての入力 (2^{20} 個) における Hash 値 (40bits) を対象として実験する.

6.1.2 目的

0 と 1 の分布により, Hash 値に偏りがあるかどうかを明らかにしたい.

6.1.3 期待する結果

理想的な Hash 関数であれば 0, 1 の出現頻度が正規分布に近づく. Keccak の 0 と 1 の出現頻度は正規分布でなければ, Hash 値に偏りがあることが証明できる. それにより, 1block の入力に対して Hash 値の特徴が探せる.

6.1.4 実験内容

まずは 2^{20} 個 Hash 値に対し, 各 Hash 値の 1 の数を数える. 各 Hash 値の 1 の数の分布は以下の図になる. (図 2)

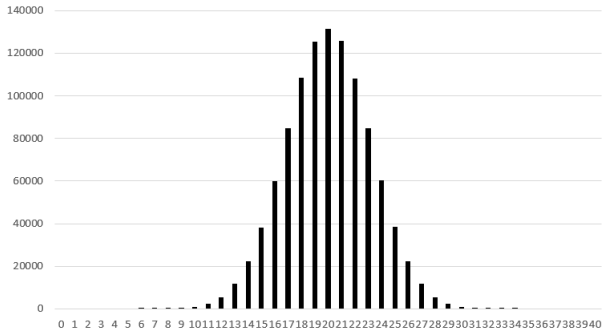


図 2 1block 全探索-Hash 値の 1 の数の分布

χ 二乗検定により, 各 Hash 値の 1 の数が正規分布かどうかを検証する.

6.1.5 結果

χ^2 値は 26279.361, p 値は 10^{-6} 未満である. 1block の全ての入力 (2^{20} 個) に対して Hash 値のの分布 1 は正規分布に従っていないので, それによる Hash 値の偏りがあることを証明できる. 偏りの更に探索すれば, Hash 値の特徴を

6.2 実験 1.1

6.2.1 対象データ

Keccak[$r = 20$, $c = 80$, $n_r = 2$, block $= 1$] 1 block 全ての入力 (2^{20} 個) における Hash 値 (40bits) を対象として実験する.

6.2.2 目的

1 block 全ての入力は 20bits, `00000` から `fffff` までであり, 全部 2^{20} 個の各 bit は 2^{19} 個の 0 と 2^{19} 個の 1 に分けられる. しかし, Keccak を通し, かき混ぜることより, 0 と 1

は半分ずつにはならないことが期待される．更に Hash 値は 40bits になり，ある bit の偏りが検出できる可能性がある．Keccak を通し，0 と 1 の分布はどのように変わるのを検証する．

6.2.3 期待する結果

ある bit の偏りが検出できるれば，Hash 値の特徴が把握できる．

6.2.4 実験内容

2^{20} 個 Hash 値の各 bit に対し，1 の数を数える (表 3)．表の中に，黄色は 1 の数が 524288 $(= 2^{19})$ に等しく，赤色は 1 の数が 524288 未満，緑色は 1 の数が 524288 より大きいことを意味している．

表 3　1block 全探索-Hash 値 bit ごとの 1 の数

524288	524288	524288	524288	524288
524288	524288	524288	524288	524288
524288	523776	524288	524800	524800
524288	524288	524800	524800	523776
523418	524378	524566	524672	524728
524434	524540	523702	523976	524636
524260	524700	524108	523860	524390
523094	524408	524866	524104	524414

表は i $(i = 1, \ldots, 5)$ 列と j $(j = 1, \ldots, 8)$ 行で構成され，Hash 値 bit 列の第 $5 \times (j-1) + i$ bit の 1 の数を表している．

更にデータを 524288(2^{19}) との差値で表す.(表 4)

表 4　1block 全探索-Hash 値 bit ごとの 1 の数と 524288 の差

0	0	0	0	0
0	0	0	0	0
0	−512	0	512	512
0	0	512	512	−512
−870	90	278	384	440
146	252	-586	−312	348
−28	412	−180	−428	102
−1194	120	578	−184	126

表は i $(i = 1 \cdots 5)$ 列と j $(j = 1 \cdots 8)$ 行で構成され，Hash 値 bit 列の第 $5 \times (j-1) + i$ bit の 1 の数と 524288 の差を表している．

6.2.5 結果

表 3 中の 524288 $(= 2^{19})$ は 0 と 1 の数が等しいことを意味している．Keccak のブロック置換を通し，0 と 1 の頻度の増減は多少ある．特に Hash 値の後半に 0 と 1 の頻度を維持する bit がない．Hash 値の中に，0 と 1 の頻度が増減しない bit は 14 箇所ある．この 14 箇所は 0 と 1 の頻度の偏りがないが，それぞれの箇所は 1 の頻度が同じく 2^{19} なので，互いに相関するかもしれない．この 14 箇所の相関性を更に調査するべきである．

6.3 実験 1.2

6.3.1 対象データ

Keccak$[r = 20, c = 80, n_r = 2, \text{block} = 1]$ 1 block 全ての入力 (2^{20} 個) に対する Hash 値の偏りがない 14 箇所を対象として実験する．

6.3.2 目的

実験 1.1(6.2 節) の結果より，14 箇所の 1 の数は 524288$(= 2^{19})$ である．それら Hash 値の中，各 bit の間に互いに相関関係があるかどうかを解明したい．

6.3.3 期待する結果

二つの bit に互いに相関関係がある，Hash 値の特徴が把握できる．

6.3.4 実験内容

2^{20} 個 Hash 値の 14 箇所に対し，互いに xor 演算し，結果の 1 の数を数える．更にデータを 524288$(= 2^{19})$ との差値で表す (表 5)．

1 の数は 0 に近い，または 2^{20} に近いならば，二つの bit は互いに相関することが証明できる．表中の黄色は 1 の数と 524288$(= 2^{19})$ の差は 0，赤色は 1 の数と 524288 の差は 0 未満，緑色は 1 の数と 524288 の差は 0 より大きいことを意味している．

表 5　14bits の互いに xor 演算した 1 の数と 524288 の差

	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
X1	640	0	−128	0	−256	−896	−256	−128	896	896	−128	128	−256
X2		640	1024	256	0	256	128	−384	128	−640	0	128	128
X3			−640	128	256	0	−256	−128	384	−128	384	−128	128
X4				256	−128	256	0	640	−128	384	−128	768	384
X5					−256	128	−256	−256	128	0	−256	256	−896
X6						256	−12	0	256	−128	−384	128	−256
X7							256	0	0	256	256	0	−128
X8								−128	0	0	−640	−384	0
X9									896	−768	0	640	256
X10										896	0	0	−128
X11											128	0	1024
X12												0	−512
X13													0

Xi $(i = 1 \cdots 14)$ は 1 の数が 524288 となる Hash 値の箇所を表している．

6.3.5 結果

特に相関関係がある bit はない．しかし，その中の 7, 10, 13bit の xor 結果は 524288 $(= 2^{19})$ である．それはこの 3bits 互いに xor 結果は完全に偏りがないを示すことかもしれない．であれば 7, 10, 13bit は計算時間に影響がないかもしれない．

6.4 実験 1.3

6.4.1 対象データ

Keccak[$r = 20$, $c = 80$, $n_r = 2$, block $= 1$] 1 block 全ての入力 (2^{20} 個) における state($5 \times 5 \times 4$) を対象として実験する.

6.4.2 目的

実験 1.1(6.2 節) は 1 block 全ての入力 (2^{20} 個) に対して Hash 値を対象としたが, 実際に Keccak により state 全体の 1 と 0 の分布はどのよう変わるのかを調べたい.

6.4.3 期待する結果

state の中にある bit は偏りがあるかどうかを解明解明する, または 1 と 0 の分布の変化の法則を捕らえる.

6.4.4 実験内容

各ステップにおける state の各 bit に対し, 1 の数を数える.

Keccak のブロック関数にを施し, Hash 値の bit 長は $2r$(40bit) なので, 出力は二つの state に分けられる. データを 524288($= 2^{19}$) との差値で表す.

表中の黄色は 1 の数と 524288 ($= 2^{19}$) の差は 0, 赤色は 1 の数と 524288 の差は 0 未満, 緑色は 1 の数と 524288 の差は 0 より大きいことを意味している.

表 6 出力 1 state(Hash 前半)

0	0	0	0	0
0	0	0	0	0
0	−512	0	512	512
0	0	512	512	−512
512	0	512	512	−256
256	−256	−256	0	0
0	0	−256	256	−256
−256	256	256	256	256
256	−256	256	256	512
0	512	512	−512	512
−512	−512	256	256	256
256	0	−512	0	0
512	−512	−512	−512	−256
−256	256	−256	0	0
0	0	0	0	0
0	0	-1024	−1024	−1024
1024	−512	−512	−512	0
−512	−512	−512	256	−256
−256	−256	256	256	−256
256	256	256	−256	256

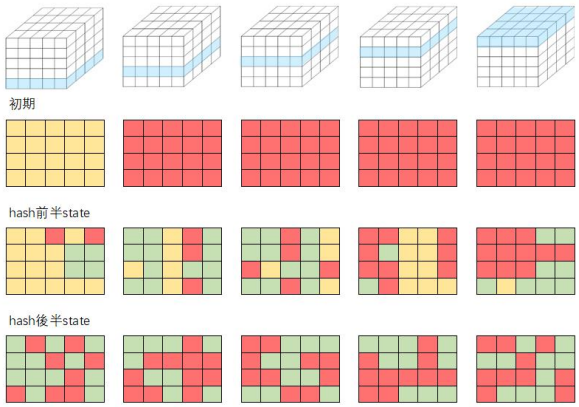
表 7 出力 2 state(Hash 後半

−870	90	278	384	440
146	252	−586	−312	348
−28	412	−180	−428	102
−1194	120	578	−184	126
−468	−980	282	1110	−210
58	−410	548	12	1064
−212	904	−106	−890	−48
−362	154	−574	−98	422
128	−36	−564	−68	1010
−848	16	−448	−758	166
496	48	−28	192	−10
550	460	−166	−362	640
−172	−30	−300	38	−556
−430	248	292	556	−38
284	628	200	−278	−598
−108	270	−166	718	288
−500	−446	866	−298	1094
−110	548	−176	132	−26
−98	340	684	592	480
−164	−410	−210	574	282

表は i ($i = 1, \ldots, 5$) 列と j ($j = 1, \ldots, 20$) 行で構成され, state の bit 列の第 $5 \times (j-1) + i$ bit の 1 の数と 524288 の差を表している.

Keccak[$r = 20$, $c = 80$] の state は $5 \times 5 \times 4$ の直方体で表せられる. 直方体 $A[x][y][z]$ に $(5y + x) \times 4 + z$ の順番で排列している. 最下層は入出力に影響する r bit, 残りの 4 層は入出力に影響しない c bit である. state 直方体の各層の変化を図 3 に示す.

図 3 state 変化



6.4.5 結果

Keccak のブロック置換を通し, state 全体に 0 と 1 の頻度は増減は多少ある. 出力 1(Hash 前半 state) の偏りはまだ 2^n に限定され, 出力 2(Hash 後半 state) は不規則である. 全 1, 全 0 になる bit はないが, Keccak のブロック置換を通し, 0 と 1 の分布は非対称になる傾向があることがわかる.

7. 実験 2

7.1 実験 2.0

7.1.1 対象データ

2^{10} 個の Keccak[$r = 20$, $c = 80$, $n_r = 2$, block $= 1$] における Hash 値から原像を計算し, 計算時間とそれに対する Hash 値および原像を実験対象とする.

表 8 1 block 入力に対する計算時間

原像	Hash 値	計算時間 (秒)
00756	56 5F E7 55 54	6948.64
007D9	BD 2B FB 54 84	8132.68
007E5	45 A5 1F 4A 7C	334.12
009EE	2F 39 E1 E3 44	186.49
01317	5E 08 39 2B 70	119.49
01591	A7 35 38 39 5B	3821.69
01718	6F AA 89 59 4E	88.70
0178C	B2 1F BC 61 0C	259.03

.....

7.1.2 目的

Hash 値および原像の 1 と 0 の分布と計算時間の間の線形関係の有無を検証する.

7.1.3 期待する結果

Hash 値および原像の 1 と 0 の分布と計算時間は線形関係である.

7.1.4 実験内容

2^{10} 個の実験データ (表 8) により, Hash 値および原像の 1 の数と計算時間に対し線形回帰を計算する.

線形回帰の決定係数 R^2 は 1 に近いほど, 回帰式が実際のデータに当てはまっていることを表す.

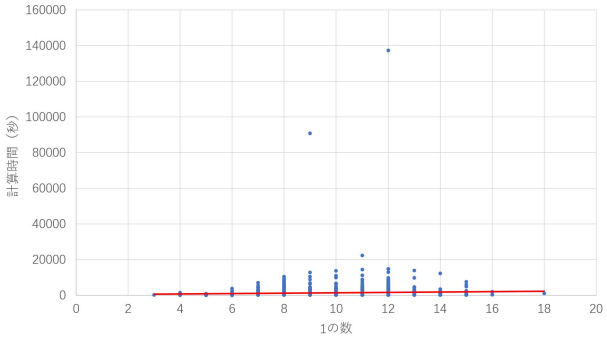


図 4 入力の 1 の数と計算時間の関係
回帰分析した式は $y = 108.21x + 210.17$(赤線)
決定係数 $R^2 = 0.00382$

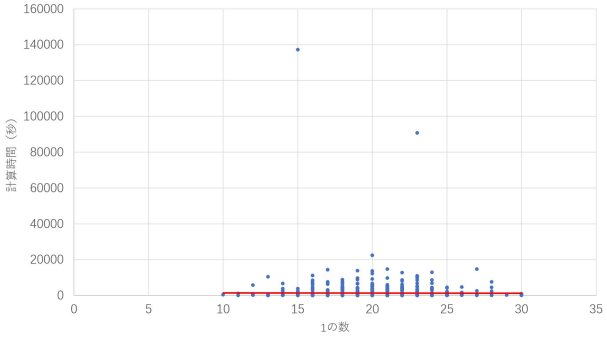


図 5 Hash 値の 1 の数と計算時間の関係
回帰分析した式は $y = -7.1249x + 1436.4$(赤線)
決定係数 $R^2 = 0.00006$

7.1.5 結果

Hash 値および原像 1 の数と計算時間は回帰分析をしたが, 決定係数 R^2 は 1 に近くないので, 線形関係がないことがわかる.

7.2 実験 2.1

7.2.1 対象データ

Keccak$[r = 20,\ c = 80,\ n_r = 2,\ \text{block} = 1]$ における Hash 値から原像を求めたところ, 計算時間の範囲が 20 秒から 10000 秒以上となり, 非常に大きなばらつきがあった. 例えば, 計算時間が 10000 秒以上のデータは一個だけのに, 図 7.1.4 と図 7.1.4 のように横軸の範囲が 0 から 10000 以上となり, 1 の頻度と計算時間の関係が分析しにくい.

各 1 の頻度布最小計算時間を示すところおよび表 9 よる. 計算時間が 400 秒以上になると, 計算時間が 100 秒増えてもデータ頻度の増加が少ない. 故に, データが集中しているところ, 計算時間が 400 秒以下の Hash 値および原像を実験対象とする.

表 9 実験 2 のデータ分布

計算時間 (秒)	0–100	0–200	0–300	0–400	0–500
データ頻度 (%)	2.93%	14.94%	60.25%	71.97%	75.78%
計算時間 (秒)	0–600	0–700	0–800	0–900	900+
データ頻度 (%)	77.54%	78.42%	79.20%	80.27%	100.00%

7.2.2 目的

計算時間が 400 秒以下の時, Hash 値および原像の 1 と 0 の分布と計算時間の関係を検証する.

7.3 期待する結果

計算時間が 400 秒以下の時, Hash 値および原像の 1 と 0 の分布と計算時間に関係がある.

7.3.1 実験内容

737 個の実験データ (計算時間が 400 秒以下) により, Hash 値および原像 1 の数と計算時間の関係を解明する.

図 6 Hash 値の 1 の数と計算時間の関係

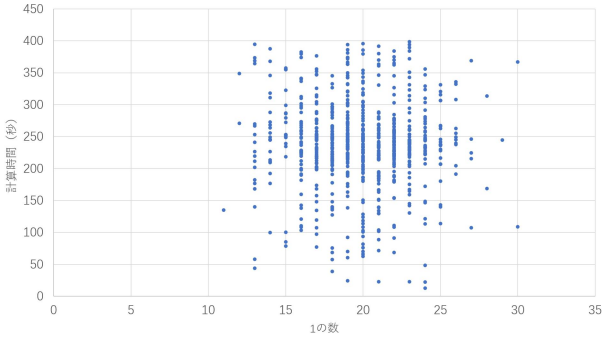
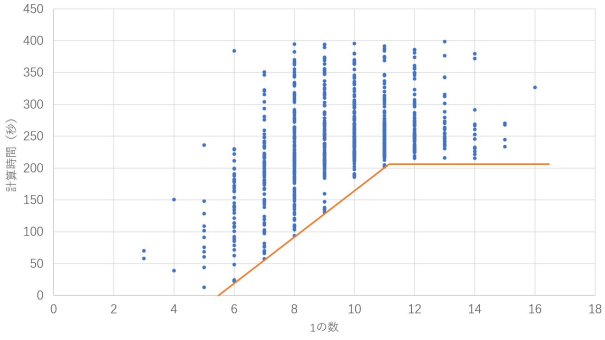


図 7 原像の 1 の数と計算時間の関係



7.3.2 結果

計算時間が 400 秒以下の際に, Hash 値の 1 の数と計算時間の関係 (図 6) は見えないが, 原像の 1 の数は、11 未満の場合最小計算時間と正比例し、原像の 1 数が 11 を超えても、最小計算時間はほとんど変わらない係 (図 7).

項の数が小さいところでは，計算時間は次数に依存し，ある程度項の数が大きくなると，計算時間は一定になることが予想できる．それより攻撃時間の少ない原像を判別できることを期待している．

8. まとめ

本研究は Gurobi Optimizer を利用し, 攻撃コンテスト中の問題の解答を求める Keccak[$r = 40$, $c = 160$, $n_r = 2$] に関する原像攻撃耐性の検証を目的とする．大きさを半分にした Keccak[$r = 20$, $c = 80$, $n_r = 2$] から, ビット頻度および原像計算時間に関する実験を行っている．

原像計算時間の少ない Hash 値および原像の特徴の抽出により, キャパシティ比が大きい Keccak に対して攻撃が成功することを期待している．

今後の課題は, 実験と解析を行い, Keccak[$r = 20$, $c = 80$, $n_r = 2$] に対する有効な原像攻撃を完成させることである．更に Keccak[$r = 40$, $c = 160$, $n_r = 2$] に拡張し, 原像攻撃耐性を検証する．

参考文献

[1] Bertoni, G. and Daemen, J. and Peeters, M. and Van Assche, G. *Keccak*, Annual International Conference on The Theory and Applications of Cryptographic Techniques , PP.313–314(2013).

[2] Morawiecki, P. and Srebrny, M. *A SAT-based preimage analysis of reduced Keccak Hash functions*, International Workshop on Fast Software Encryption , PP.392–397 , Vol.113, No.10-11 (2013).

[3] Morawiecki, P. and Pieprzyk, J. and Srebrny, M. and Straus, M. *Preimage attacks on the round-reduced Keccak with the aid of differential cryptanalysis*, Cryptology ePrint Archive (2013).

[4] Guo, J. and Liu, M. and Song, L. *Linear structures: applications to cryptanalysis of round-reduced Keccak*, International Conference on the Theory and Application of Cryptology and Information Security , PP.249–274 (2016).

[5] Li, T. and Sun, Y and Liao, M. and Wang, D. *Preimage attacks on the round-reduced Keccak with cross-linear structures*, IACR Transactions on Symmetric Cryptology , PP.39–57 (2017).

[6] Bixby, B. *The gurobi optimizer*, Transp. Re-search Part B , pp.159–178, Vol.41, No.2 (2007)

付　　録

A.1 実験 1.0 の χ 二乗検定

理想的な Hash 関数であれば 0, 1 の出現頻度が正規分布に近づく．今回サンプルの数は 2^{20} あるので, 理想的な理論的確率は正規分布にかなり近い．

$$P = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{A.1}$$

μ はデータの平均値とする．

表 **A·1** χ 二乗検定

組	頻度	理論的確率 (%)	理論的頻度	差^2/理論値
0	0	0.0000000125	0.0001311	0.000131
1	0	0.0000000799	0.0008382	0.000838
2	0	0.0000005272	0.0055285	0.005529
3	0	0.0000031486	0.0330159	0.033016
4	0	0.0000170251	0.1785212	0.178521
5	0	0.0000833517	0.8740059	0.874006
6	7	0.0003694888	3.8743712	2.521585
7	11	0.0014830476	15.5508812	1.331791
8	74	0.0053898922	56.5171159	5.408118
9	280	0.0177370086	185.9860158	47.523085
10	842	0.0528518075	554.1913695	149.467878
11	2219	0.1426008375	1495.2781582	350.284863
12	5282	0.3483937819	3653.1735821	726.238554
13	11522	0.7707398896	8081.7935050	1464.405236
14	22151	1.5439638724	16189.6346147	2195.100637
15	38113	2.8006620567	29367.0701677	2604.661895
16	59692	4.6002417273	48237.0306941	2720.240444
17	84711	6.8422467428	71746.1572054	2342.803507
18	108414	9.2154460789	96630.9558761	1436.807983
19	125080	11.2391797181	117851.3411211	443.384935
20	131454	12.4123555751	130152.9815950	13.005072
21	125463	12.4129640882	130159.3623175	169.452421
22	108019	11.2408327990	117868.6749305	823.086340
23	84665	9.2177052457	96654.6449572	1487.270335
24	59978	6.8445952051	71770.7826181	1937.692705
25	38390	4.6022719144	48258.3187496	2017.967419
26	22085	2.8021728101	29382.9115649	1812.601624
27	11424	1.5449482214	16199.9562616	1408.013568
28	5383	0.7713069081	8087.7391251	904.531368
29	2227	0.3486842832	3656.2197094	558.683323
30	743	0.1427337410	1496.6717519	379.522837
31	260	0.0529062544	554.7622858	156.616279
32	59	0.0177570227	186.1958786	86.891244
33	25	0.0053965034	56.5864397	17.631489
34	3	0.0014850124	15.5714835	10.149463
35	0	0.0003700146	3.8798848	3.879885
36	0	0.0000834785	0.8753355	0.875336
37	0	0.0000170527	0.1788104	0.178810
38	0	0.0000031540	0.0330726	0.033073
39	0	0.0000005282	0.0055386	0.005539
40	0	0.0000000801	0.0008398	0.000840
			χ^2	26279.36152
μ	20.0005		自由度	40
σ	3.16119		p 値	10^{-6} 未満