Collision Attack on the Hamsi-256 Compression Function

Mario Lamberger Florian Mendel Vincent Rijmen

presented by Kerem Varici

Outline

1 Description of Hamsi-256

2 Existing Attacks

3 (New) Improved Attack

4 Future Work

Outline

1 Description of Hamsi-256

2 Existing Attacks

3 (New) Improved Attack

4 Future Work

The Hamsi-256 Hash Function

- Hamsi-256 is one of the 14 round 2 candidates of the SHA-3 competition designed by Küçük
- It is an iterated hash function based on the Merkle-Damgård design principle
- It processes message blocks of 32 bits and produces a 256-bit hash value

The Compression Function

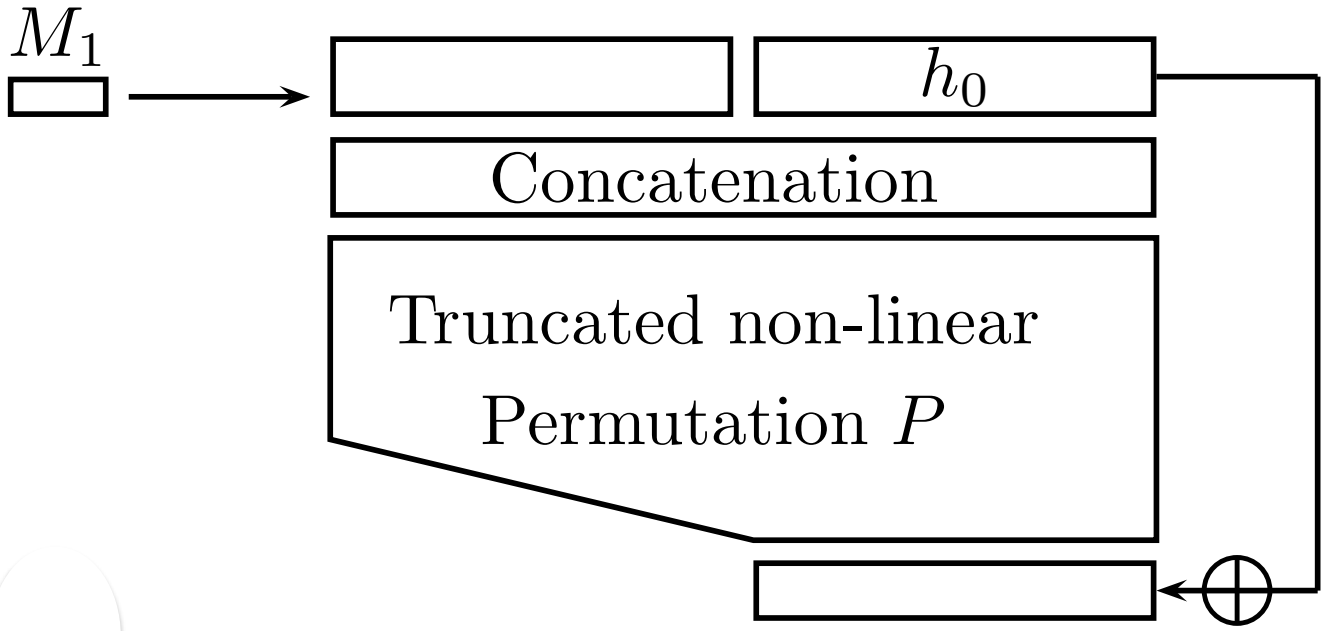


Figure : Compression function [6].

The Concatenation

The expanded message words m_i and the chaining words c_i with $0 \leq i < 8$ are arranged in a 4×4 state.

m_0	m_1	c_0	c_1
c_2	c_3	m_2	m_3
m_4	m_5	c_4	c_5
c_6	c_7	m_6	m_7

Figure : Concatenation [6].

The Non-Linear Permutation P

The non-linear permutation P consists of 3 rounds. In each round the state is updated by 3 steps:

- Addition of constants and counter
- Non-linear substitution layer
- Linear diffusion layer

The Substitution Layer

Parallel application of 128 identical 4×4-bit Sboxes.

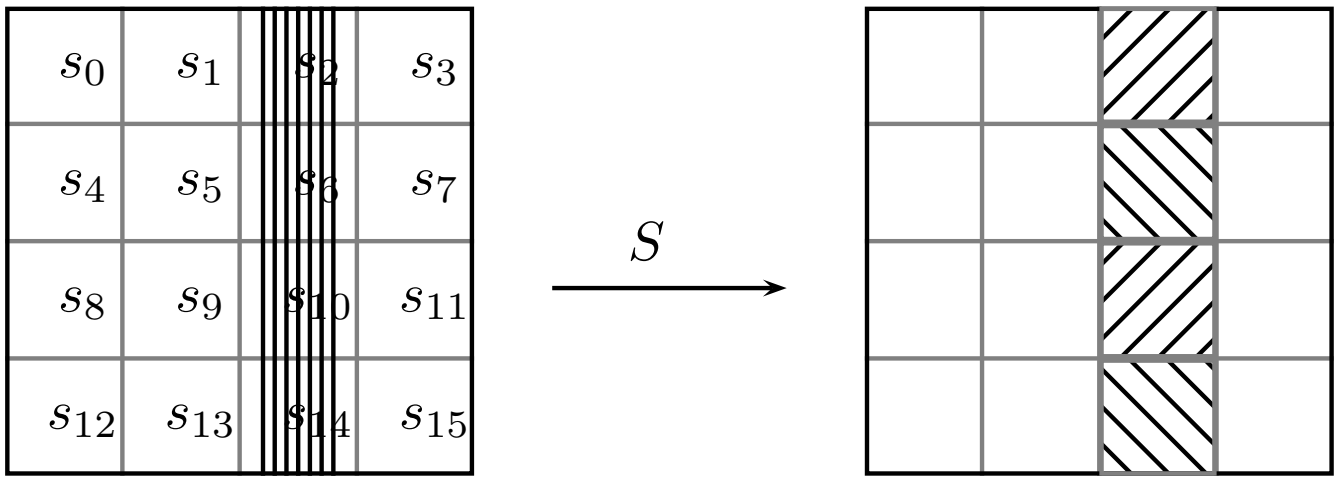


Figure : Substitution layer [6].

The Diffusion Layer

Applications of the linear transformation $L : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$.

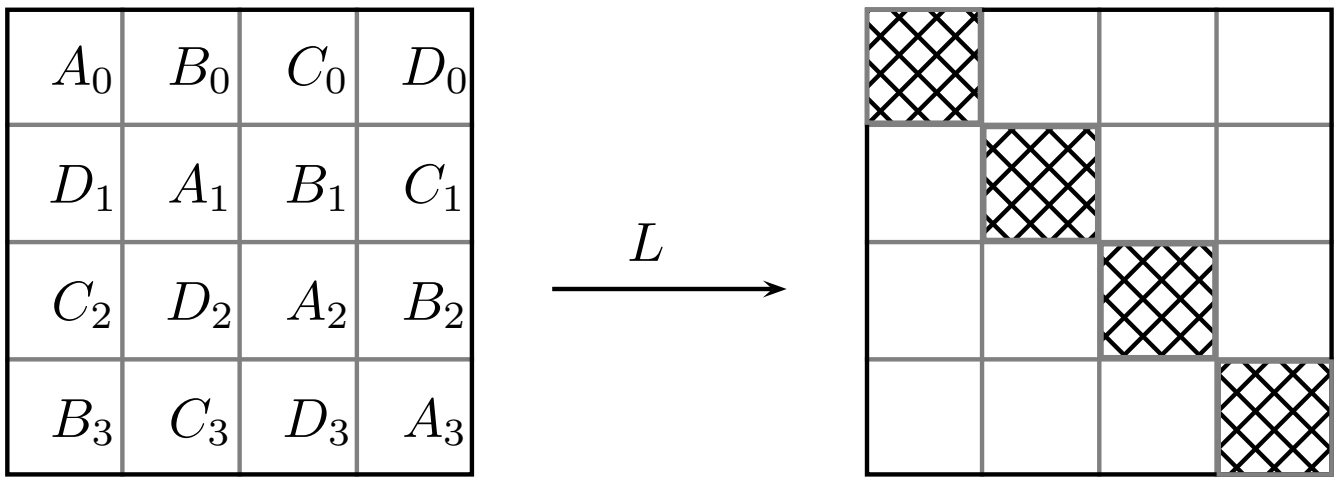


Figure : Diffusion layer [6].

The Truncation

Truncation is applied after the last round of the nonlinear permutation.

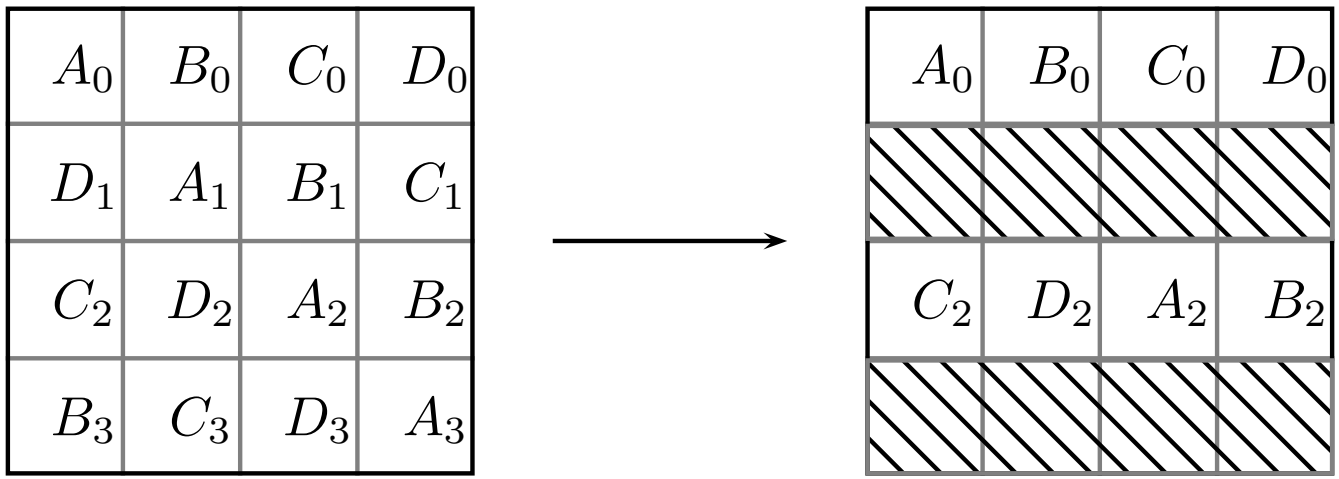


Figure : Truncation [6].

Outline

1 Description of Hamsi-256

2 Existing Attacks

3 (New) Improved Attack

4 Future Work

Attacks on the Hash Function

type of analysis	hash size	parameters	complexity	reference
2nd-preimage	256	(3,6)	2^{247}	Dinur, Shamir
2nd-preimage	256	(3,6)	$2^{251.3}$	Fuhr
pseudo 2nd-preimage	256	(3,6)	$2^{254.25}$	Çalık, Turan

Table : Attacks on the hash function.

Attacks on the Compression Function

type of analysis	hash size	parameters	complexity	reference
distinguisher	256	6	2^{10}	Boura, Canteaut
distinguisher	224, 256	6	2^{28}	Aumasson et al.
distinguisher	224, 256	6	2^{27}	Aumasson, Meier
distinguisher	384, 512	12	2^{729}	Aumasson, Meier
distinguisher	224, 256	3	-	Çalık, Turan
near-collision	256	2	-	Turan, Uyan
near-collision	224, 256	3	2^{26}	Aumasson et al.
near-collision	224, 256	3	2^{21}	Nikolic
near-collision	224, 256	3	2^{5}	Wang et al.
near-collision	224, 256	4	2^{32}	Wang et al.
near-collision	224, 256	5	2^{125}	Wang et al.
distinguisher	224, 256	6	$2^{124.3}$	Aumasson et al.

Table : Attacks on the compression function.

Attacks on the Compression Function

type of analysis	hash size	parameters	complexity	reference
distinguisher	256	6	2^{10}	Boura, Canteaut
distinguisher	224, 256	6	2^{28}	Aumasson et al.
distinguisher	224, 256	6	2^{27}	Aumasson, Meier
distinguisher	384, 512	12	2^{729}	Aumasson, Meier
distinguisher	224, 256	3	-	Çalık, Turan
near-collision	256	2	-	Turan, Uyan
near-collision	224, 256	3	2^{26}	Aumasson et al.
near-collision	224, 256	3	2^{21}	Nikolic
near-collision	224, 256	3	2^{5}	Wang et al.
near-collision	224, 256	4	2^{32}	Wang et al.
near-collision	224, 256	5	2^{125}	Wang et al.
distinguisher	224, 256	6	$2^{124.3}$	Aumasson et al.

Table : Attacks on the compression function.

Attack of Çalık and Turan

- Differential attack (distinguisher)
- Find a set of output bits that are not affected by a given input difference (with probability 1)
  - only one active sbox at the input
  - only differences in the chaining input

$\Rightarrow$ Distinguisher for the compression function

Example 1 – Input difference in one bit

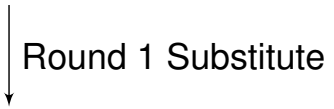
column: 14

00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Example 1 – Input difference in one bit

column: 14

00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000



00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000

Example 1 – Input difference in one bit

column: 14

00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Round 1 Substitute

00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000

Round 1 Diffuse

00000018	20000000	04000001	00800000
00040000	80000000	00000000	00200000
00060400	00008000	00000000	00004000
01000000	08000000	00000000	00000000

Example 1 – Input difference in one bit

column: 14

00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Round 1 Substitute

00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000

Round 1 Diffuse

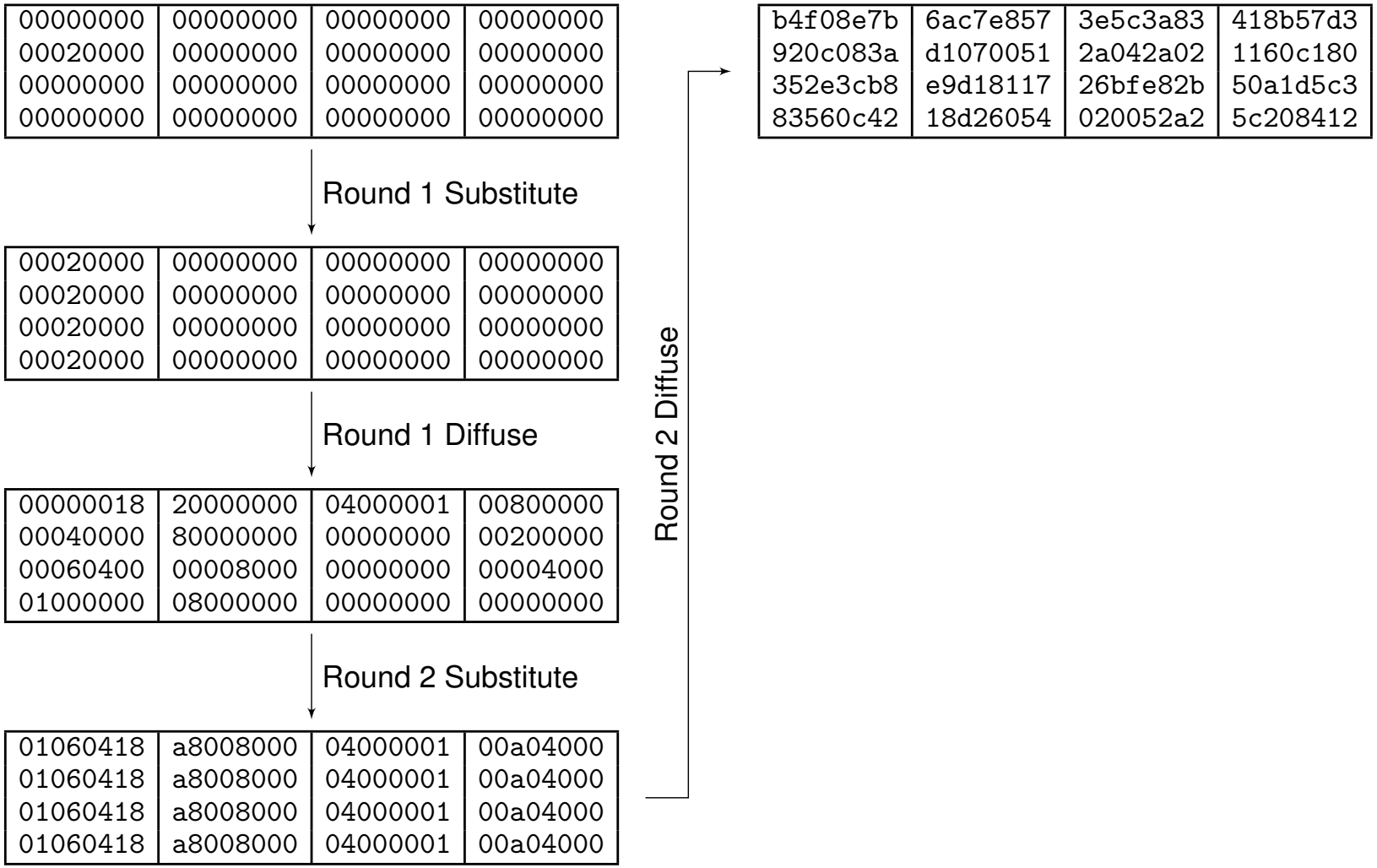
00000018	20000000	04000001	00800000
00040000	80000000	00000000	00200000
00060400	00008000	00000000	00004000
01000000	08000000	00000000	00000000

Round 2 Substitute

01060418	a8008000	04000001	00a04000
01060418	a8008000	04000001	00a04000
01060418	a8008000	04000001	00a04000
01060418	a8008000	04000001	00a04000

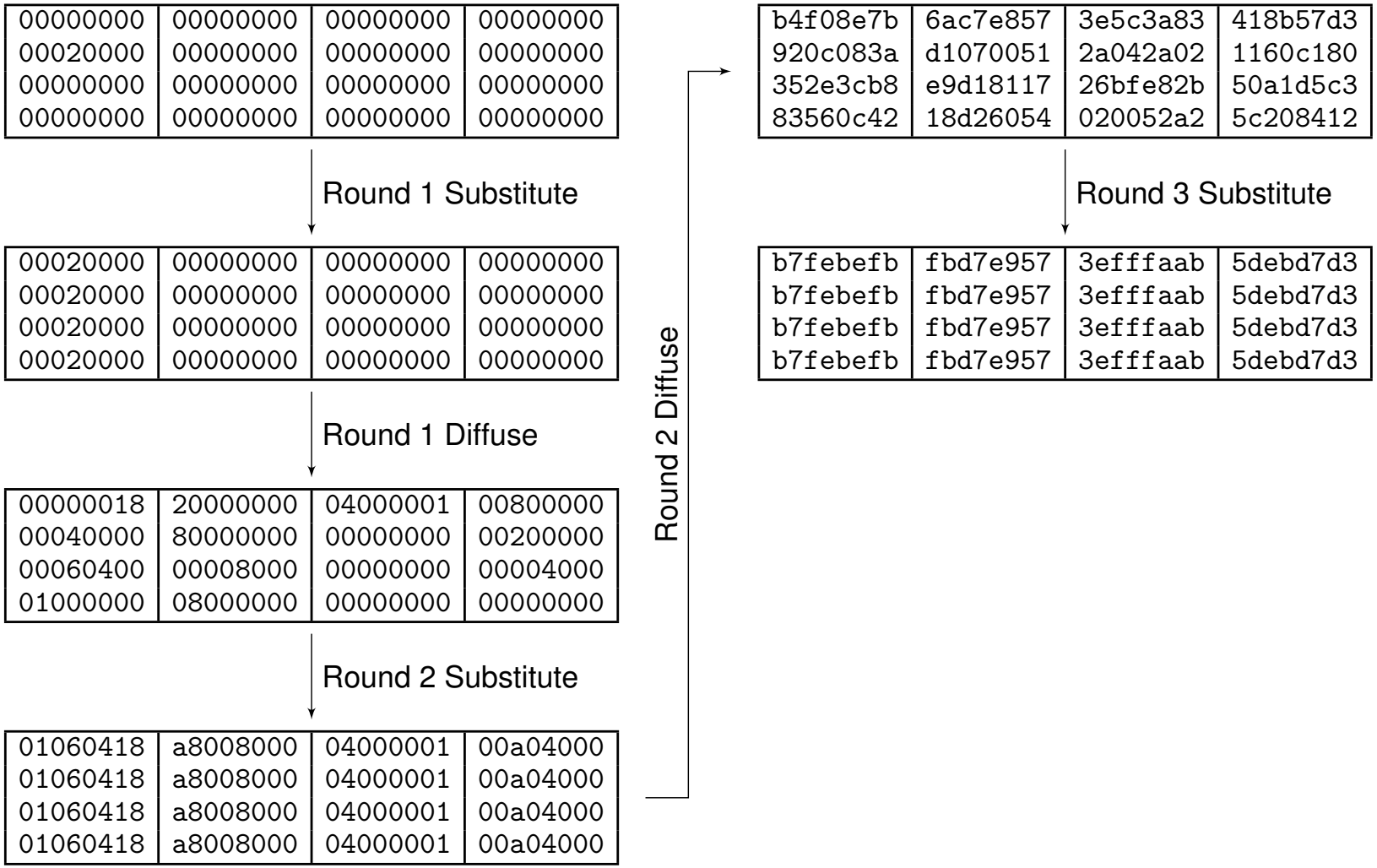
Example 1 – Input difference in one bit

column: 14



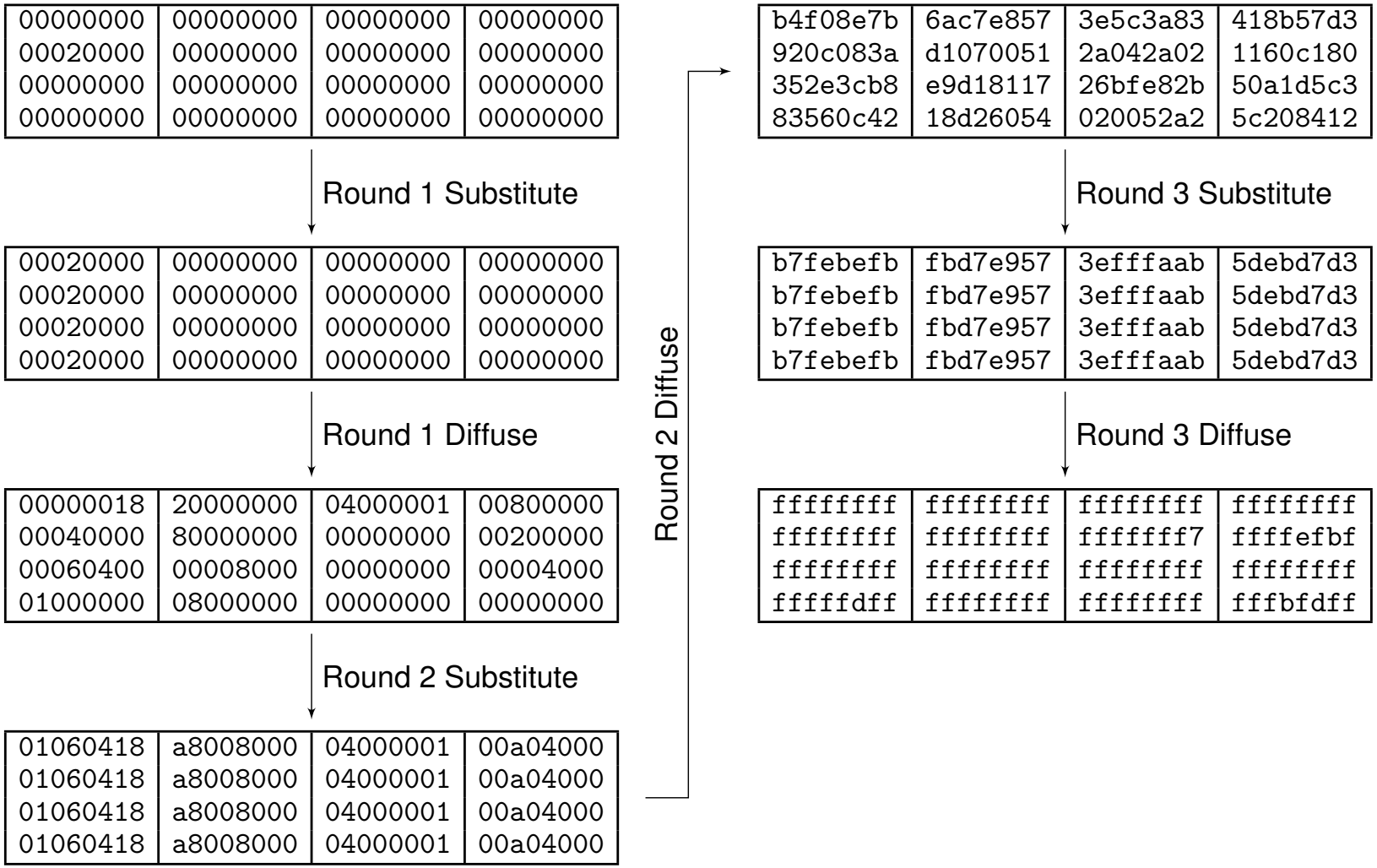
Example 1 – Input difference in one bit

column: 14



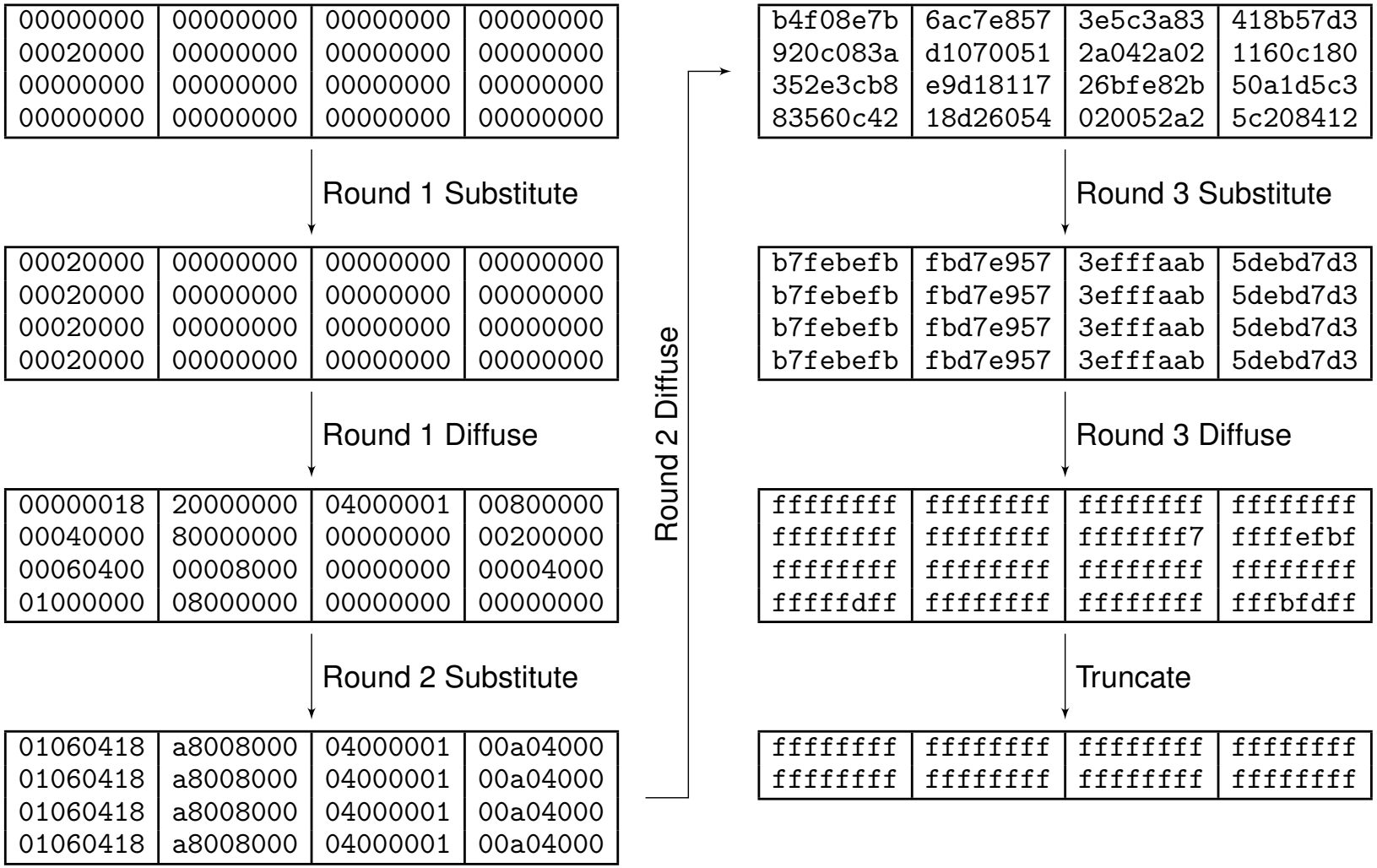
Example 1 – Input difference in one bit

column: 14



Example 1 – Input difference in one bit

column: 14



Example 1 – Input difference in one bit

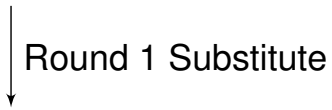
column: 14, $m_{14} = 1, m_{142} = 1$

00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Example 1 – Input difference in one bit

column: 14, $m_{14} = 1, m_{142} = 1$

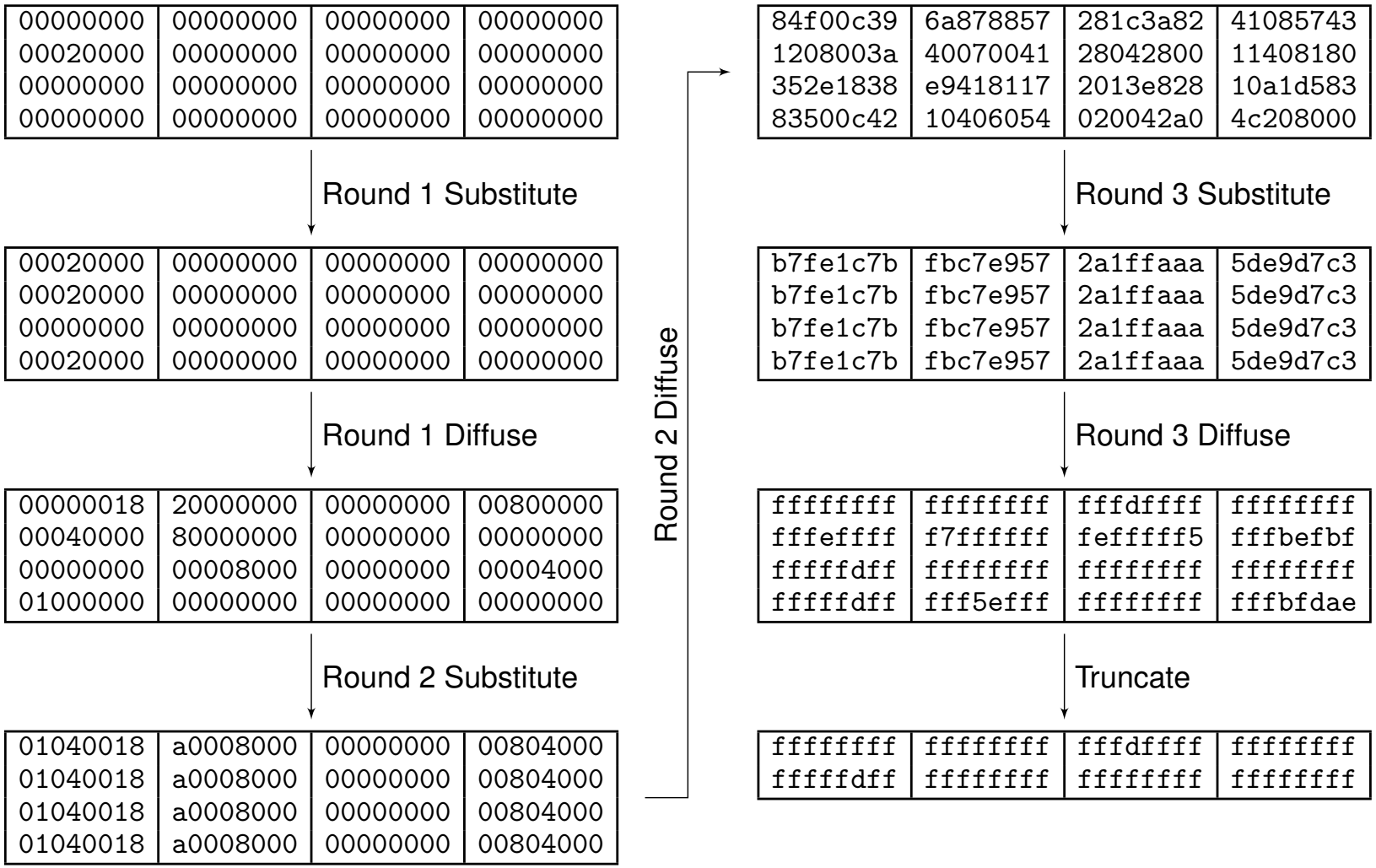
00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000



00020000	00000000	00000000	00000000
00020000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00020000	00000000	00000000	00000000

Example 1 – Input difference in one bit

column: 14, $m_{14} = 1, m_{142} = 1$



Example 1

Column	CV bit	Unaffected output bits	Condition
14	78	78,150	$m_{14} = 1, m_{142} = 1$
15	79	79,151	$m_{15} = 1, m_{143} = 1$
46	110	110, 182	$m_{46} = 1, m_{174} = 1$
47	111	111, 183	$m_{47} = 1, m_{175} = 1$
78	130	14, 214	$m_{78} = 0, m_{206} = 0$
79	131	15, 215	$m_{79} = 0, m_{207} = 0$
110	184	46, 246	$m_{110} = 0, m_{138} = 0$
111	185	47, 247	$m_{111} = 0, m_{139} = 0$

Figure : List of unaffected outputs for one input difference [3].

Example 2 – Input differences in two bits

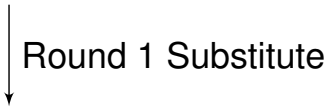
column: 7, no message conditions

00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000

Example 2 – Input differences in two bits

column: 7, no message conditions

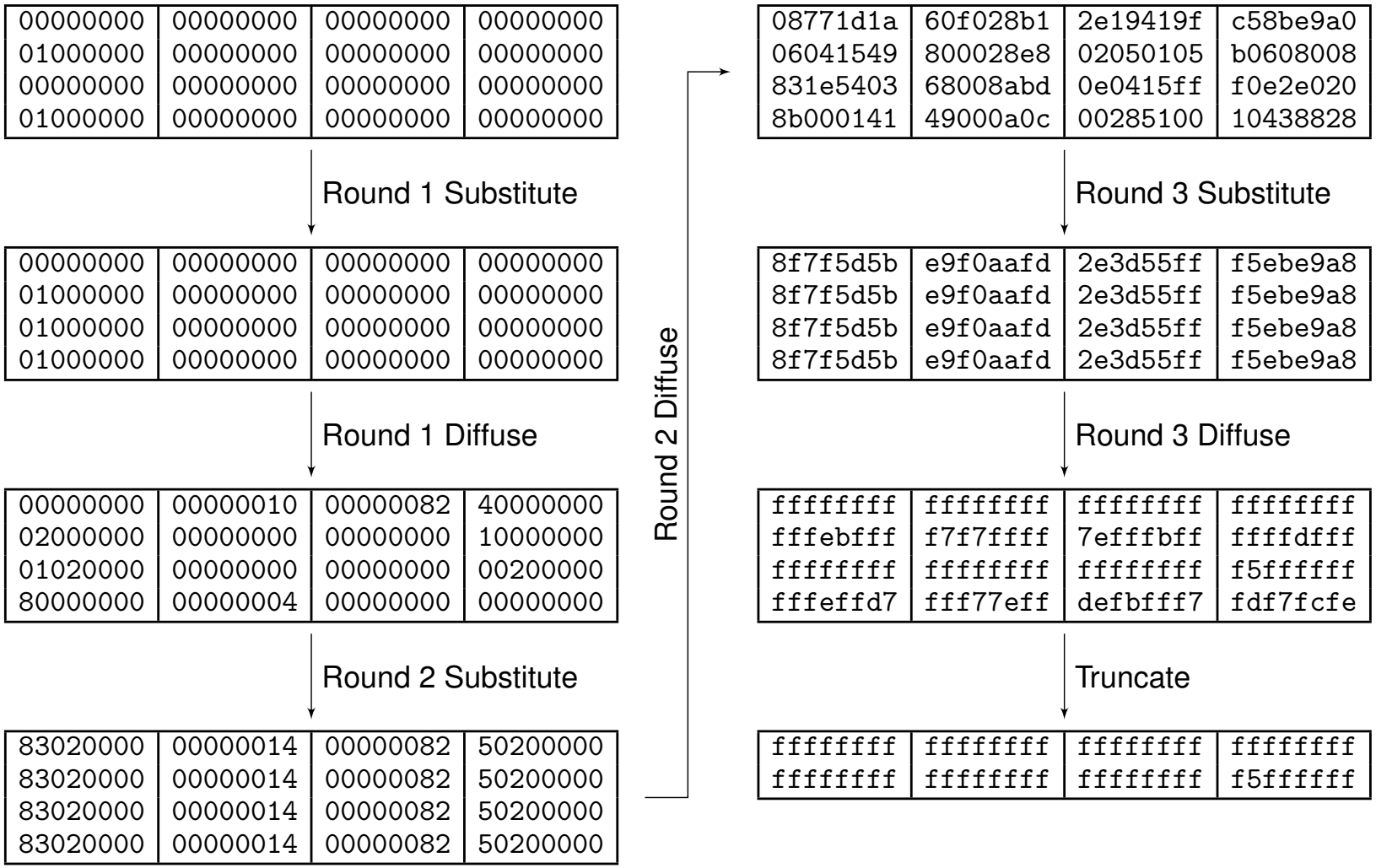
00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000



00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000

Example 2 – Input differences in two bits

column: 7, no message conditions



Example 2

Column	CV bits	Unaffected bits	Column	CV bits	Unaffected bits
0	64,192	165, 201	32	96,224	197, 233
1	65,193	166	33	97,225	198
2	66,194	167	34	98,226	199
3	67,195	168	35	99,227	200
4	68,196	169	36	100,228	201
6	70,198	227	38	102,230	131
7	71,199	228, 230	39	103,231	132, 134
8	72,200	229	40	104,232	133
9	73,201	131, 230	41	105,233	134, 163
10	74,202	132, 231	42	106,234	135, 164
11	75,203	133, 232	43	107,235	136, 165
12	76,204	134, 233	44	108,236	137, 166
13	77,205	135	45	109,237	167
14	78,206	136	46	110,238	168
15	79,207	137	47	111,239	169
26	90,218	195	58	122,250	227
27	91,219	196	59	123,251	228
28	92,220	197	60	124,252	229
29	93,221	198	61	125,253	230
30	94,222	163, 199	62	126,254	195, 231
31	95,223	164, 200	63	127,255	196, 232

Figure : List of unaffected output bits for two input difference [3].

Outline

1 Description of Hamsi-256

2 Existing Attacks

3 (New) Improved Attack

4 Future Work

The Attack - Basic Idea

- Assume we can find a independent input differences not affecting the same b output bits (with $a \approx b$)
- Then we can control b bits of the output of the compression function with an average complexity of 2

The Attack - Basic Idea

- Assume we can find a independent input differences not affecting the same b output bits (with $a \approx b$)
- Then we can control b bits of the output of the compression function with an average complexity of 2

$\Rightarrow$ Collision/Preimage attack on the compression function

Suitable Input Differences

Input differences with Hamming weight 2

- Fix additional conditions on the message and chaining bits
- Then the unaffected output bits can be increased
  - up to 62 unaffected output bits
  - on average 30 unaffected output bits

Suitable Input Differences

Input differences with Hamming weight 2

- Fix additional conditions on the message and chaining bits
- Then the unaffected output bits can be increased
  - up to 62 unaffected output bits
  - on average 30 unaffected output bits

$\Rightarrow$ 198 candidates with at least 10 unaffected output bits

The Attack

Finding a set of suitable input differences

(0) Precomputation: Generate a large set S of input differences where each only affect a few output bits

(1) Find a subset $S' \subseteq S$ not affecting the same b output bits (with $a \approx b$ and $a = |S'|$)

(2) Use Gaussian elimination to find a solution for the message expansion

(3) Use it in an attack on the compression function

The Attack

Solution with $a = 6, b = 6$

column	1	2	38	39	110	111
difference	$\mathtt{a}_x$	$\mathtt{a}_x$	$\mathtt{a}_x$	$\mathtt{a}_x$	5_x	5_x
message bits	2_x	2_x	3_x	1_x	$2_x, 3_x$	$2_x, 3_x$
chaining bits	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$0_x, 3_x$	$1_x, 2_x$

- None of them affects the output bits: $131, 200, 201, 202, 237, 238$

The Attack

Solution with $a = 6, b = 6$

column	1	2	38	39	110	111
difference	a_x	a_x	a_x	a_x	5_x	5_x
message bits	2_x	2_x	3_x	1_x	$2_x, 3_x$	$2_x, 3_x$
chaining bits	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$0_x, 3_x$	$1_x, 2_x$

- None of them affects the output bits: $131, 200, 201, 202, 237, 238$

$\Rightarrow$ Collision attack with complexity of 2^{126}

Improving the Attack

- Find a subset S' for larger values of a and b

Improving the Attack

- Find a subset S' for larger values of a and b
- Therfore, we need to find a set S, where the number of unaffected output bits is larger than 30 on average

Improving the Attack

- Find a subset S' for larger values of a and b
- Therfore, we need to find a set S, where the number of unaffected output bits is larger than 30 on average
  - Consider candidates with differences in more than only one column $\Rightarrow$ huge search space

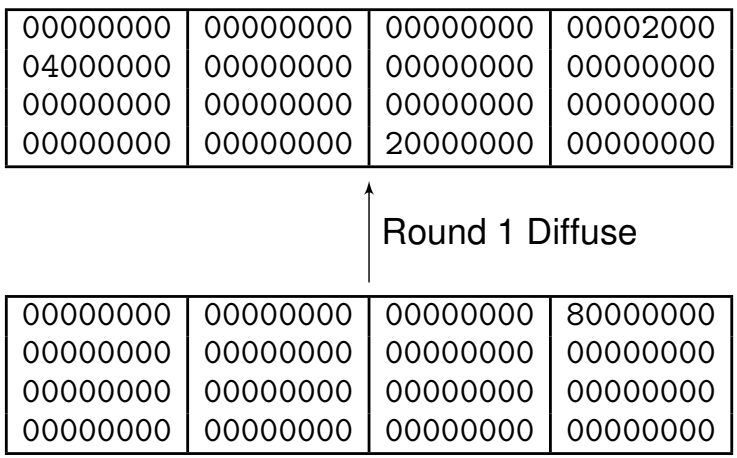
Improving the Attack

- Find a subset S' for larger values of a and b
- Therfore, we need to find a set S, where the number of unaffected output bits is larger than 30 on average
  - Consider candidates with differences in more than only one column $\Rightarrow$ huge search space
  - We are only interested in candidates with a sparse difference at the input of round 2 $\Rightarrow$ search space is reduced

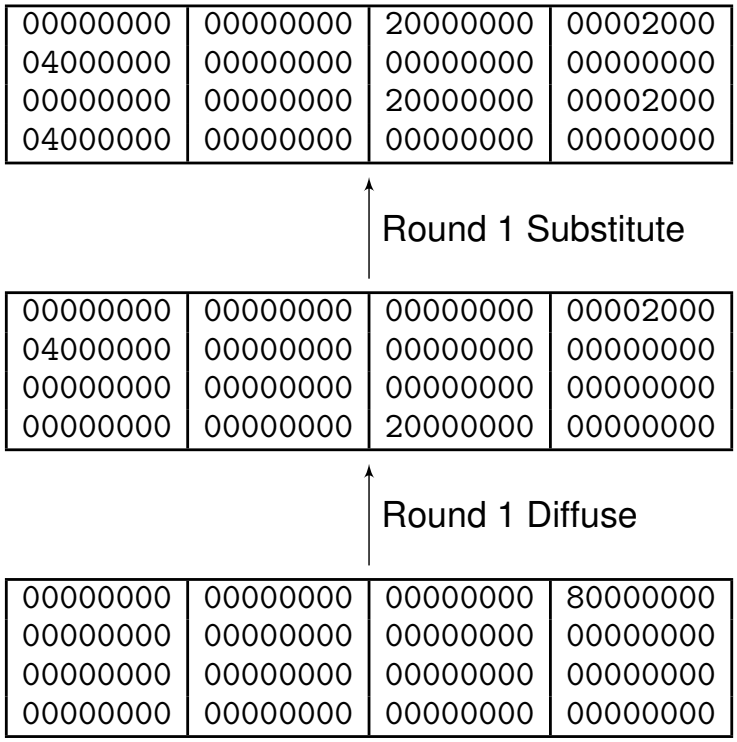
Improving the Attack

00000000	00000000	00000000	80000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Improving the Attack



Improving the Attack



Improving the Attack

3 active columns: 5, 66, 114

00000000	00000000	20000000	00002000
04000000	00000000	00000000	00000000
00000000	00000000	20000000	00002000
04000000	00000000	00000000	00000000

↑ Round 1 Substitute

00000000	00000000	00000000	00002000
04000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	20000000	00000000

↑ Round 1 Diffuse

00000000	00000000	00000000	80000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Improving the Attack

3 active columns: 5, 66, 114

00000000	00000000	20000000	00002000
04000000	00000000	00000000	00000000
00000000	00000000	20000000	00002000
04000000	00000000	00000000	00000000

↑ Round 1 Substitute

00000000	00000000	00000000	00002000
04000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	20000000	00000000

↑ Round 1 Diffuse

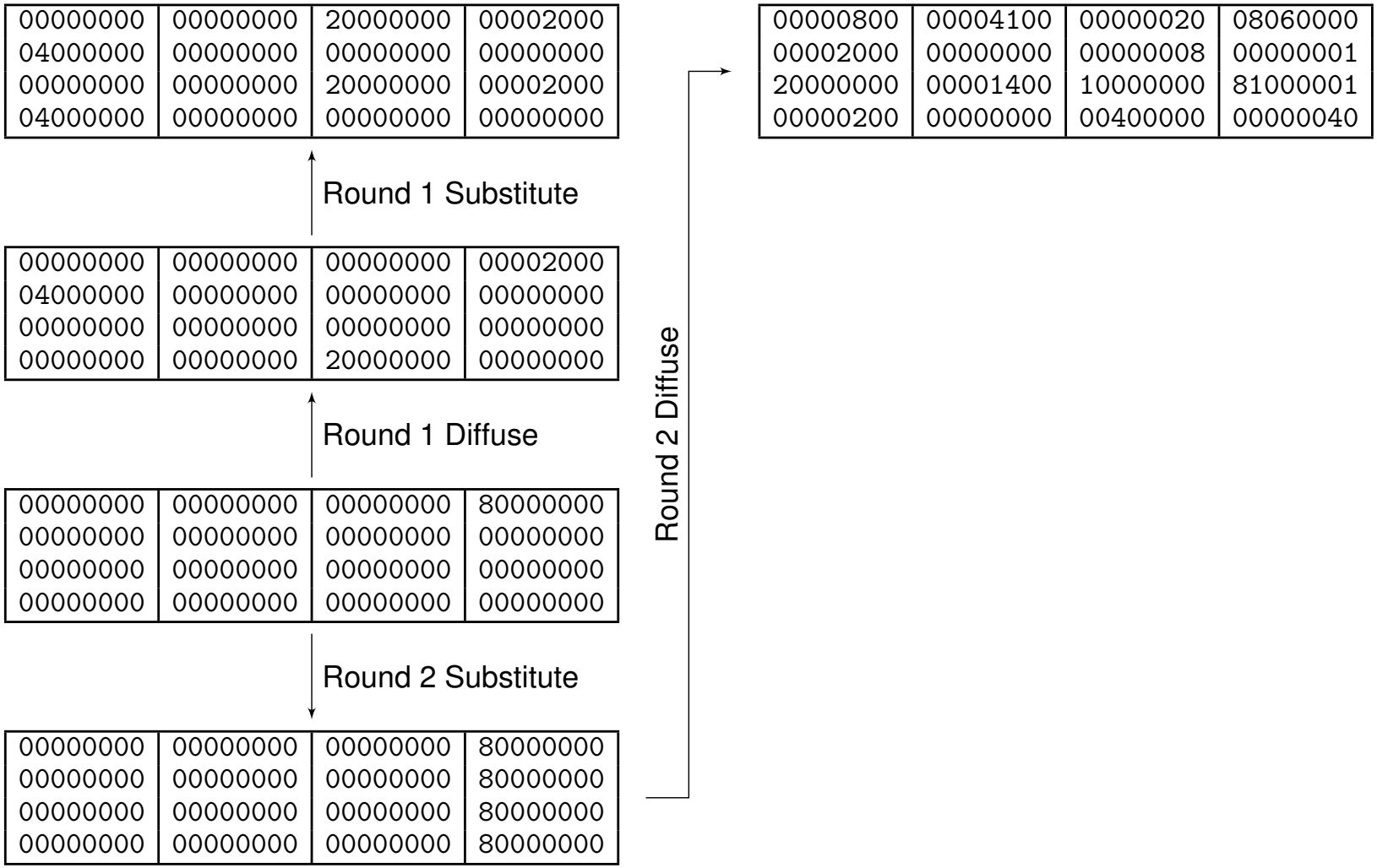
00000000	00000000	00000000	80000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

↓ Round 2 Substitute

00000000	00000000	00000000	80000000
00000000	00000000	00000000	80000000
00000000	00000000	00000000	80000000
00000000	00000000	00000000	80000000

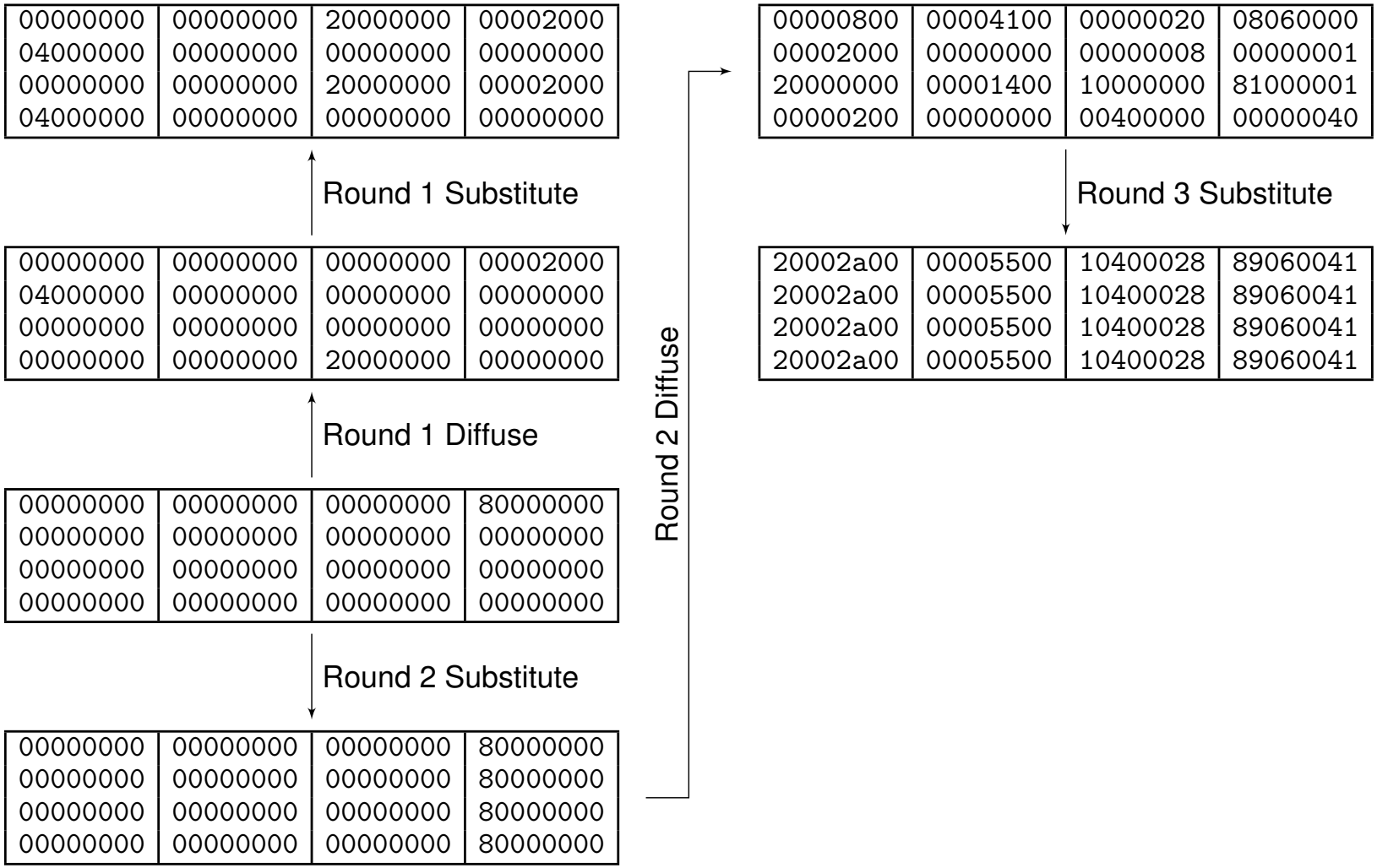
Improving the Attack

3 active columns: 5, 66, 114



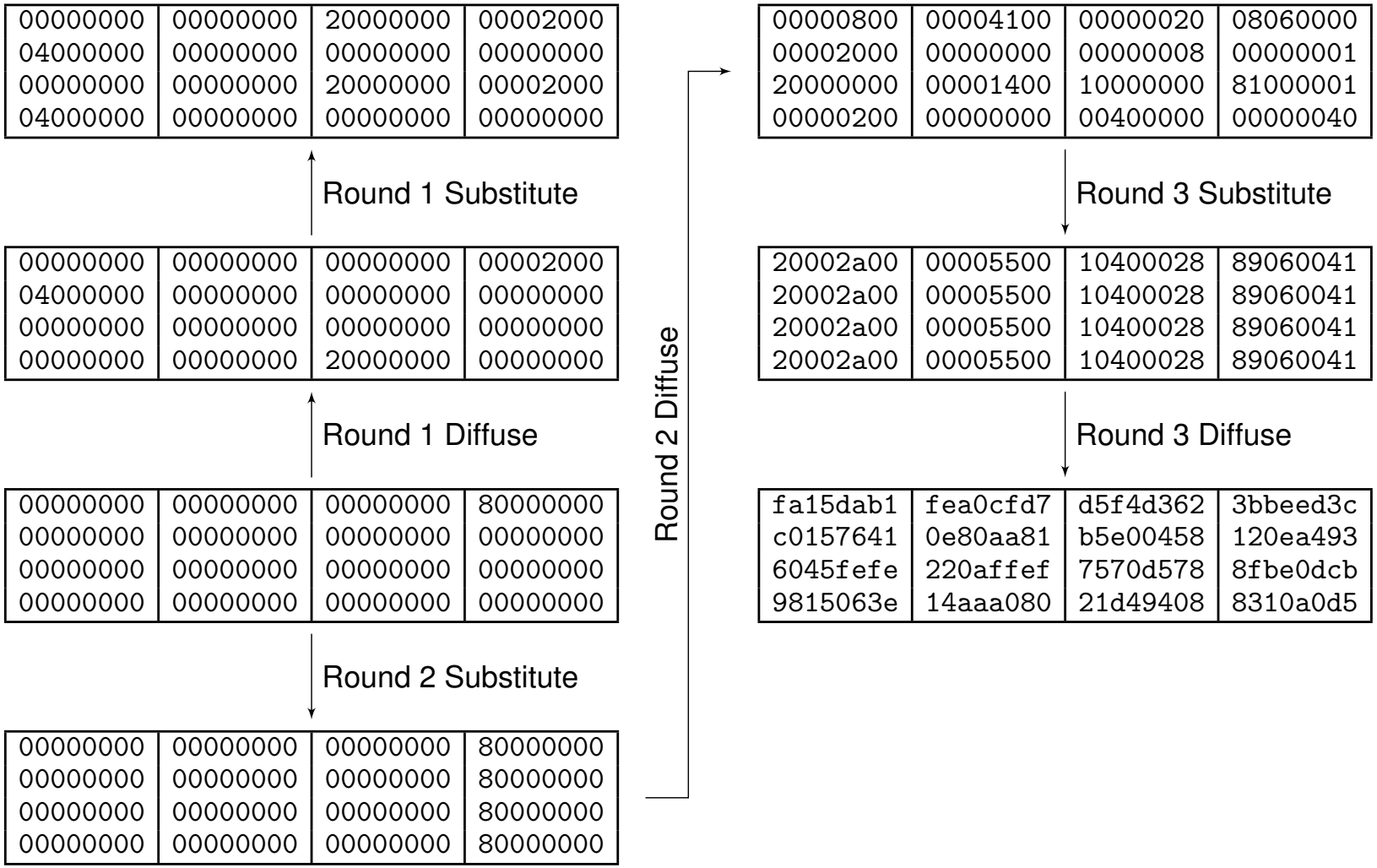
Improving the Attack

3 active columns: 5, 66, 114



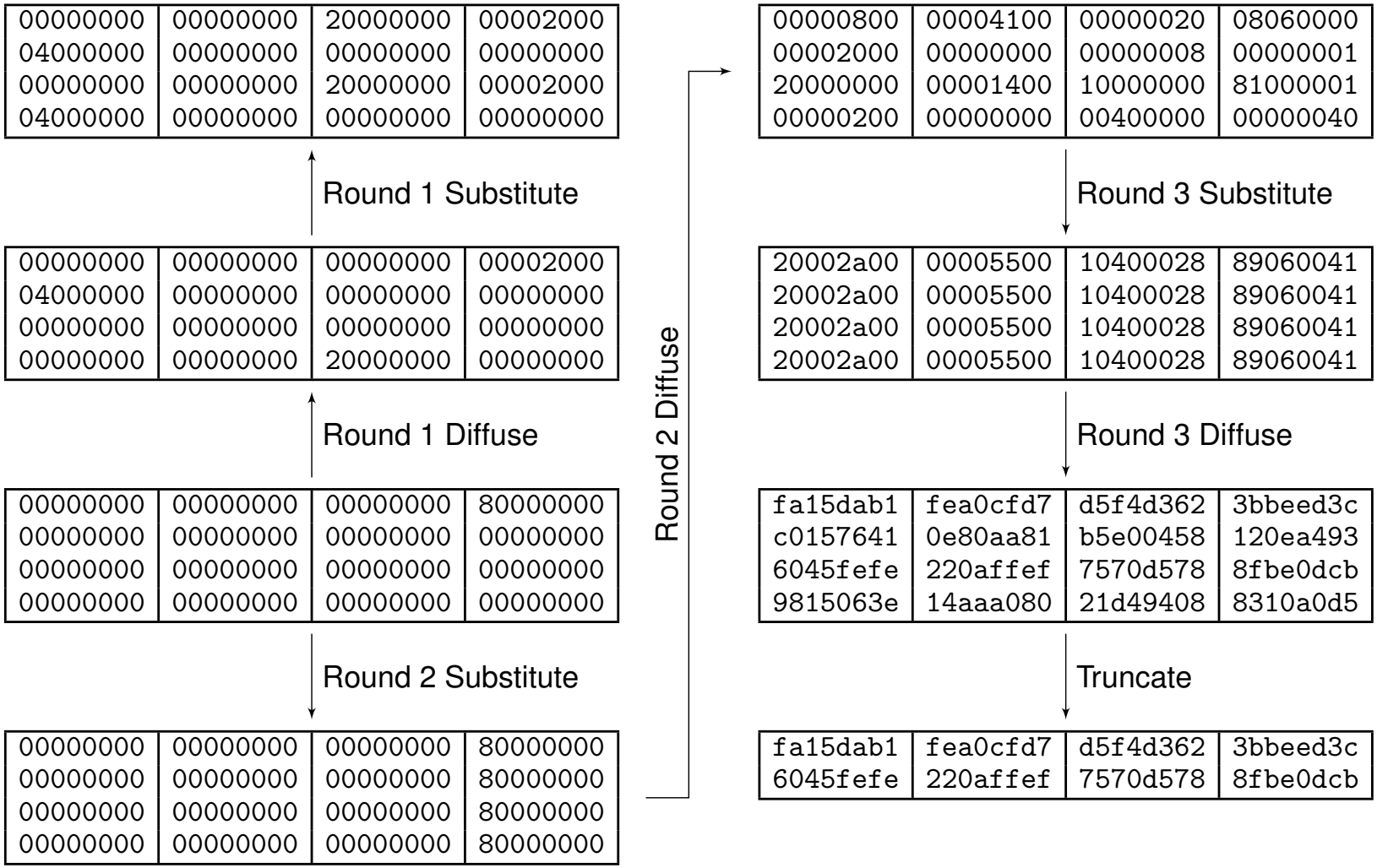
Improving the Attack

3 active columns: 5, 66, 114



Improving the Attack

3 active columns: 5, 66, 114



Improving the Attack

- We found a S with 192 candidates having only a single bit difference at the input of round 2
  - 2-7 active columns at the input of round 1
  - 3-13 conditions on the message bits
  - 2-7 conditions on the chaining bits
  - 100-130 unaffected output bits (average 110)

Improving the Attack

- We found a S with 192 candidates having only a single bit difference at the input of round 2
  - 2-7 active columns at the input of round 1
  - 3-13 conditions on the message bits
  - 2-7 conditions on the chaining bits
  - 100-130 unaffected output bits (average 110)
- We found a S' with $a = 10$ and $b = 9$

Improving the Attack

- We found a S with 192 candidates having only a single bit difference at the input of round 2
  - 2-7 active columns at the input of round 1
  - 3-13 conditions on the message bits
  - 2-7 conditions on the chaining bits
  - 100-130 unaffected output bits (average 110)
- We found a S' with $a = 10$ and $b = 9$

$\Rightarrow$ Collision attack with complexity of $2^{124.1}$

Outline

1 Description of Hamsi-256

2 Existing Attacks

3 (New) Improved Attack

4 Future Work

Future Work

- Improve the attack
  - Find a larger subset S' (with $a, b > 10$)
  - Find input differences that need less degrees of freedom
  - Consider other linear relations at the output (e.g. pairs of bits)

Future Work

- Improve the attack
  - Find a larger subset S' (with $a, b > 10$)
  - Find input differences that need less degrees of freedom
  - Consider other linear relations at the output (e.g. pairs of bits)
- Extend the attack to more rounds (output transformation)

Future Work

- Improve the attack
  - Find a larger subset S' (with $a, b > 10$)
  - Find input differences that need less degrees of freedom
  - Consider other linear relations at the output (e.g. pairs of bits)
- Extend the attack to more rounds (output transformation)
- What about linear cryptanalysis?

References I

J.-P. Aumasson, E. Käsper, L. R. Knudsen, K. Matusiewicz, R. S. Ødegård, T. Peyrin, and M. Schläffer.

Distinguishers for the Compression Function and Output Transformation of Hamsi-256.

In R. Steinfeld and P. Hawkes, editors, *ACISP*, volume 6168 of *LNCS*, pages 87–103. Springer, 2010.

C. Boura and A. Canteaut.

Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-*f* and Hamsi-256.

In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *LNCS*, pages 1–17. Springer, 2010.

Çağdaş Çalık and M. S. Turan.

Message Recovery and Pseudo-preimage Attacks on the Compression Function of Hamsi-256.

In M. Abdalla and P. S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *LNCS*, pages 205–221. Springer, 2010.

References II

I. Dinur and A. Shamir.

An Improved Algebraic Attack on Hamsi-256.

In A. Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 88–106. Springer, 2011.

T. Fuhr.

Finding Second Preimages of Short Messages for Hamsi-256.

In M. Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 20–37. Springer, 2010.

Ö. Küçük.

The Hash Function Hamsi.

Submission to NIST, 2008.

`http://homes.esat.kuleuven.be/~okucuk/hamsi/`.

Y. Li and A. Wang.

Using genetic algorithm to find near collisions for the compress function of Hamsi-256.

In *BIC-TA*, pages 826–829. IEEE, 2010.