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

Jean-Philippe Aumasson[1], Emilia Käsper[2], Lars Ramkilde Knudsen[3], Krystian Matusiewicz[4], Rune Ødegård[5], Thomas Peyrin[6], and Martin Schläffer[7]

[1] Nagravision SA, Cheseaux, Switzerland
[2] Katholieke Universiteit Leuven, ESAT-COSIC, Belgium
[3] Department of Mathematics, Technical University of Denmark
[4] Institute of Mathematics and Computer Science, Wroclaw University of Technology
[5] Centre for Quantifiable Quality of Service in Communication Systems at the Norwegian University of Science and Technology
[6] Ingenico, France
[7] IAIK, TU Graz, Austria

**Abstract.** Hamsi is one of 14 remaining candidates in NIST's Hash Competition for the future hash standard SHA-3. Until now, little analysis has been published on its resistance to differential cryptanalysis, the main technique used to attack hash functions. We present a study of Hamsi's resistance to differential and higher-order differential cryptanalysis, with focus on the 256-bit version of Hamsi. Our main results are efficient distinguishers and near-collisions for its full (3-round) compression function, and distinguishers for its full (6-round) finalization function, indicating that Hamsi's building blocks do not behave ideally.

**Keywords:** hash functions, differential cryptanalysis, SHA-3.

## 1 Introduction

Hash functions are one of the most ubiquitous primitives in cryptography, with digital signatures and integrity checks as their main applications. Collision attacks on the deployed standards MD5 and SHA-1 [18, 19, 20, 21] have weakened the confidence in the MD family of hash functions. Hence, the US Institute of Standards and Technology (NIST) launched a public competition to develop a future SHA-3 standard [13].

The hash function Hamsi [8] is one of 64 designs submitted to NIST in fall 2008. Hamsi is also among the 14 submissions selected for the second round of the competition in July 2009 as one of the few submissions with no major weaknesses detected thus far. While Hamsi reuses the round components of the Serpent block cipher [5], its larger block size and different round structure make existing cryptanalytic results on Serpent hardly useful in its security analysis.

So far, little research has been published on the resistance of Hamsi to common cryptanalytic attacks: in a work independent from ours, Çalık and Turan studied

R. Steinfeld and P. Hawkes (Eds.): ACISP 2010, LNCS 6168, pp. 87–103, 2010.

differential properties of Hamsi-256, and presented message-recovery and pseudo-second-preimage attacks. Near collisions were studied by Nikolić [12] and Wang et al. [17], as discussed in Section 4.3.

We study the resistance of Hamsi to differential and higher-order differential cryptanalysis, with focus on the 256-bit version Hamsi-256. In Section 3, we show by higher-order analysis that the 3-round compression function of Hamsi-256 does not achieve maximal degree. This is demonstrated by showing that the output of certain related chaining values (with fixed message word) or related message words (with fixed chaining value) sums to zero with a high probability.

In Sections 4 and 5, we focus on differential cryptanalysis and construct high-probability differential paths for the 3-round compression function as well as the full 6-round output transformation. The former gives near-collisions on $(256-25)$ bits of the compression function output, with only six differences in the input chaining value. Section 4 describes a technique for building low-weight, high-probability differential paths for Hamsi. Finally, Section 5 presents differential paths for six rounds of Hamsi-256 that show that the output transformation of Hamsi-256 does not behave ideally.

## 2 Description of Hamsi-256

This section describes the hash function Hamsi-256, henceforth just called Hamsi. We refer to [8] for a complete specification.

### 2.1 High-Level Structure

Like most hash functions, Hamsi builds on a finite-domain *compression function*, which is used to process arbitrary-length messages through the use of a *domain extender* (or operation mode). The compression function of Hamsi can be divided into four operations:

| | |
|---|---|
| Message expansion | $E : \{0,1\}^{32} \to \{0,1\}^{256}$ |
| Concatenation | $C : \{0,1\}^{256} \times \{0,1\}^{256} \to \{0,1\}^{512}$ |
| Non-linear permutations | $P, P_f : \{0,1\}^{512} \to \{0,1\}^{512}$ |
| Truncation | $T : \{0,1\}^{512} \to \{0,1\}^{256}$ |

The message $M$ to hash is appropriately padded and split into $\ell$ blocks of 32 bits: $M_1, \ldots, M_\ell$. Each block is iteratively processed by the compression function, which operates on a 512-bit internal state viewed as a 4×4 matrix of 32-bit words.

Figure 1 depicts an iteration of the compression function $H$ (or $H_f$). Starting from the predefined initial value (IV) $h_0$, Hamsi iteratively computes the digest $h$ of $M$ as follows:

$$h_i = H(h_{i-1}, M_i) = (T \circ P \circ C(E(M_i), h_{i-1})) \oplus h_{i-1} \quad \text{for} \quad 0 < i < \ell \ ,$$
$$h = H_f(h_{\ell-1}, M_\ell) = (T \circ P_f \circ C(E(M_\ell), h_{\ell-1})) \oplus h_{\ell-1} \ .$$

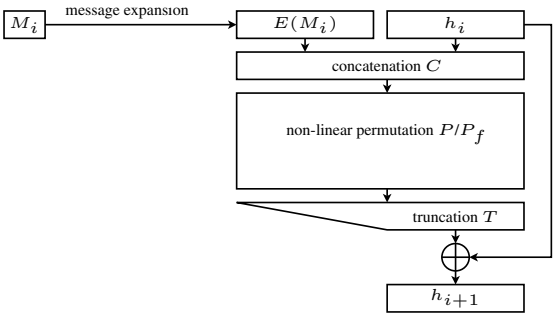


**Fig. 1.** Domain extension algorithm of Hamsi

### 2.2 Internals of the Compression Function of Hamsi

**Message expansion.** The message expansion of Hamsi uses a linear code to expand a 32-bit word into eight words (that is, 256 bits). We write an expanded $M_i$ as $(m_0, \ldots, m_7)$. Thus, the $m_j$'s are defined as the product of a multiplication with the generator matrix of the code:

$$E(M_i) = (m_0, \ldots, m_7) = (M_i \times G) \ ,$$

where $G$ can be found in [8].

**Concatenation.** The concatenation function $C$ forms a 512-bit internal state from the 256-bit expanded message $(m_0, \ldots, m_7)$ and the 256-bit incoming chaining value $h_i = (c_0, \ldots, c_7)$ (Figure 2):

$$C(m_0, \ldots, m_7, c_0, \ldots, c_7) = (m_0, m_1, c_0, c_1, c_2, c_3, m_2, m_3, m_4, m_5, c_4, c_5, c_6, c_7, m_6, m_7),$$

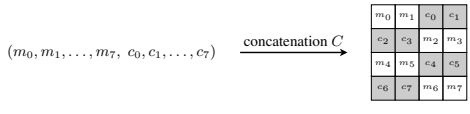


**Fig. 2.** Concatenation of expanded message words $m_0, \ldots, m_7$ and chaining value words $c_0, \ldots, c_7$ in Hamsi

**Truncation.** The truncation function $T$ selects eight 32-bit words among the 16 from the internal state to form the new chaining value after feedforward (Figure 3):

$$T(s_0, s_1, s_2, \ldots, s_{14}, s_{15}) = (s_0, s_1, s_2, s_3, s_8, s_9, s_{10}, s_{11}) \ .$$

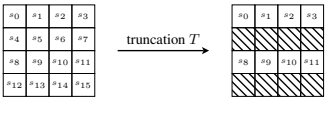


**Fig. 3.** Truncation selects eight out of 16 words of the internal state

**Permutations.** Finally, we describe the permutations $P$ and $P_f$. They only differ in the number of rounds (three for $P$ and six for $P_f$)[1] and in the round constants. The round function is composed of three layers. First, constants and a counter are XORed to the whole internal state. Then there is a substitution layer, followed by a linear layer.

The substitution layer uses one 4-bit Sbox of the block cipher Serpent [5], in a bitsliced way. That is, four bits, one from each of the four 32-words of the same column in the 4×4 internal state matrix are first extracted and then replaced after application of the Sbox. We denote $s_i^j$ the $j$-th bit of the internal state word $s_i$. The substitution layer can be described as follows, for $0 \leq j \leq 31$ and $0 \leq i \leq 3$:

$$(s_i^j, s_{i+4}^j, s_{i+8}^j, s_{i+12}^j) := S(s_i^j, s_{i+4}^j, s_{i+8}^j, s_{i+12}^j) \ ,$$

where $S$ is the 4×4 Sbox given in Table 7 (Appendix A).

The linear diffusion layer applies the Serpent linear transform $L : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ to each of the four diagonals of the state, as follows:

$$\begin{aligned}(s_0, s_5, s_{10}, s_{15}) &:= L(s_0, s_5, s_{10}, s_{15}) \\ (s_1, s_6, s_{11}, s_{12}) &:= L(s_1, s_6, s_{11}, s_{12}) \\ (s_2, s_7, s_8, s_{13}) &:= L(s_2, s_7, s_8, s_{13}) \\ (s_3, s_4, s_9, s_{14}) &:= L(s_3, s_4, s_9, s_{14}) \ .\end{aligned}$$

The algorithm below (read column by column) describes the linear transform $L$ on input $(a, b, c, d)$, with $x \lll k$ denoting the left bit rotation of $k$ positions on the word $x$ and $x \ll k$ denoting the left bit shift of $k$ positions on the word $x$.

$$\begin{array}{l|l} a := a \lll 13 & d := d \lll 7 \\ c := c \lll 3 & a := a \oplus b \oplus d \\ b := a \oplus b \oplus c & c := (b \ll 7) \oplus c \oplus d \\ d := (a \ll 3) \oplus c \oplus d & a := a \lll 5 \\ b := b \lll 1 & c := c \lll 22 \end{array}$$

## 3 Higher-Order Differential Analysis

This section reports on properties of Hamsi related to higher-order derivatives. After some definitions, we present upper bounds on the algebraic degree of Hamsi's compression function and show how to exploit them to find "$k$-sums" and "zero-sums". This illustrates the fact that, due to its low algebraic degree, the compression function of Hamsi does not behave ideally.

### 3.1 Definitions

**Higher-order derivatives.** Higher-order differential analysis [7,10] of cryptographic algorithms generalizes the notion of differential cryptanalysis by

[1] While 6 rounds remains the official parameter, the designer has suggested 8 rounds as a conservative alternative. Our results indicate that moving to 8 rounds may be a necessary precaution.

considering derivatives of order two or more. It is based on the basic observation that for a function $f$ with algebraic degree $s \geq 1$, the degree of a $d$th-order derivative of $f$ is at most $(s-d)$, where $s \geq d$. Consequently, an $s$th-order derivative of $f$ is a constant and an $(s+1)$st-order derivative of $f$ is zero, which directly gives a $2^{s+1}$-sum for $f$.

In the following we consider derivatives of functions with domain $\{0,1\}^n$, $n \geq 1$ and range $\{0,1\}$. Note that a (certain type of) $d$-th order derivative is then the XOR of $2^d$ values of the function for the $2^d$ choices of $d$ input bits.

***k*-sums.** The $k$-sum problem is, given $k$ lists of random $n$-bit values (for example, $k$ distinct instances of a compression function $f_1, \ldots, f_k$) , to find one value from each list such that the sum of the $k$ values is zero. The case $k = 2$ is essentially the collision problem.

The $k$-sum problem can be solved in polynomial time (using the XHASH attack [2]) when $k \geq n$. However, the problem is believed to be hard for small $k$. The standard method for the $k$-sum problem with small $k$ is Wagner's "generalized birthday" method, which requires time and space $O(k2^{n/(1+\log k)})$ [16] (see also [3]).

Henceforth, we consider the problem of finding $k$ values whose images by a same function $f$ sum to zero. Note that if $f$ has degree $s < (n-1)$, then a $2^{s+1}$-sum can be found by returning the values corresponding to a $(s+1)$st order derivative.

An example of application of $k$-sums is to forge message authentication codes (MACs). Let $H$ he a hash function and consider the "prefix-MAC" construction defined as $\mathrm{MAC}_K(m) = \mathrm{trunc}\,(H(K\|m))$, where trunc is a function removing some bits of the hash output to combat length extension attacks. Assume we know messages $m_1, \ldots, m_k$ such that the probability

$$\Pr_K \left[ \bigoplus_{i=1}^{k} H(K\|m_i) = 0 \right] = p$$

is nonzero. Then by querying $\mathrm{MAC}_K$ with $m_1, \ldots, m_{k-1}$ we can determine $\mathrm{MAC}_K(m_k)$ with probability $p$ and thus break the existential forgery of MAC.

This can be generalized to messages whose MAC tags sum to any fixed value, to other MAC constructions, etc. For example, one may fix a message and forge the MAC $H_K(m)$ where $K$ is the IV of $H$ by making related-key queries.

**Zero-sums.** We define the zero-sum problem as a particular case of the $k$-sum problem: given a function $f$, find distinct values that sum to zero such that their images by $f$ also sum to zero.

Both the XHASH attack [2] and Wagner's generalized birthday [16] can be adapted to find zero-sums. These methods are generic, and are probabilistic algorithms whose failure probability can be made exponentially small.

### 3.2 On the Degree of the Compression Function

**Simple bounds.** The only nonlinear component of Hamsi's compression function is the layer of 4×4 Sboxes. One round thus has degree three (see [14] for

explicit expressions of the Sboxes used), so $N$ rounds have degree at most $3^N$, with respect to any choice of variables.

If variables are chosen in $c_0, \ldots, c_3$ only, or in $c_4, \ldots, c_7$ only, then they are all in distinct slices and thus go into distinct Sboxes in the first round. Hence, the first round is linear and after $N$ rounds, the degree is at most $3^{N-1}$. This means that the degree is at most 81 after five rounds, and that at least six rounds are necessary to reach maximal degree. In particular, the 3-round compression function has degree at most 9 with respect to choices of 128 variables in distinct slices, which distinguishes it from a randomly chosen function (whose degree would be below 9 with negligible probability).

**Case of four variables.** If four variables are chosen in the LSB's of $c_0, \ldots, c_3$, after the first application of the Sbox, all the LSB's of a given word depend on the bit varied in the corresponding column. Since only one bit is varied per column, the degree of equations corresponding to LSB's are of degree 1. Then, the linear function $L(a, b, c, d)$ is applied to each column, and we can determine, for a given bit of the state, whether it depends on the single variable of its diagonal. Based on this, we can determine whether a given 4-bit slice depends on 1, 2, 3, or 4 of the variables.

A simple computer-assisted analysis revealed that each slice depends on only one variable. Therefore, the (3-round) compression function of Hamsi always has degree 3 with respect to four variables in the first four LSB's, for any values of the other bits. Ideally, the function should have degree 4 with probability 1/2, over the choice of the other input bits.

### 3.3 Finding $k$-sums for the Compression Function

For randomly chosen 256-bit values, finding 4-sums for the compression function of Hamsi requires an effort of complexity approximately $4 \cdot 2^{256/3} \approx 2^{87}$, using the generalized birthday method. Below we show efficient methods to find 16-, 8-, and 4-sums.

**16-sums.** Recall the above observation that three rounds have degree at most 3 with respect to a certain choice of four variables. This observation can directly be used to find 16-sums, without any computation. Based on empirical observations, we discovered that we can do better, as presented below.

**8-sums.** Choose a random value of one 256-bit chaining value, then select seven other chaining values, which are different from the first one only in the LSB's of the first three 32-bit words. Denote these chaining values by $h^0, \ldots, h^7$. Choose a random 32-bit message block $M$, then compute $\sum_{i=0}^{7} H(h^i, M)$, In 1 000 000 such tests, the above sum was zero in 1458 cases (whereas for a random mapping, the probability to obtain zero is negligible). This indicates that there are 3rd-order derivatives with the value zero (or 8-sums) of a high probability for the compression function of Hamsi. It is very likely that one can identify other 3rd-order derivatives of higher probabilities (our search was limited).

**4-sums.** We found 2nd-order derivatives with value zero, that is, 4-sums. One example is when one chaining value is the IV of Hamsi specified in [9], and where the three others differ only in two LSB's of the second words; the XOR of the four outputs is the all-zero string (note that the four inputs also sum to zero, thus this is also a zero-sum).

Via an exhaustive search over all $2^{32}$ message words, we identified 70 messages for which the above four chaining values lead to a 4-sum. We also found 4-sums for the IV given in [8], for 86 values of the 32-bit message block. Although complete analytical justification of these observations remains to be found, the results of these observations strongly differ from what one obtains for a random mapping (for which a 2nd-order derivative is zero with negligible probability).

***k*-sums for fixed chaining value.** Here we report on the case where the chaining value is fixed and where only the message block is varied. The outputs of the compression function in this case has a much higher algebraic degree.

Consider $h_0$, the IV specified in [9], and $2^{19}$ values of the 32-bit message block obtained by varying the first and second bytes, and the three least significant bits of the third byte. The remaining bits can be fixed to arbitrary values. Denoting these message words by $m_0, \ldots, m_{2^{19}-1}$, we have:

$$\bigoplus_{i=0}^{2^{19}-1} H(h_0, m_i) = 0 \ .$$

This observation holds for any initial chaining variable. Here we obtain zero because we perform a 19th-order derivative of a function of degree 18 only. Indeed, in the first round at most two bit variables enter a same Sbox, hence the degree of the first round is 2. Since the two subsequent rounds have degree 3 each, the three rounds have degree $2 \times 3 \times 3 = 18$.

Note that if $P_f$ is replaced by $P$ in Hamsi's domain extender, then the above observation can be used to forge MAC's (cf. Section 3.1), which shows that the extended 6-round output transformation is necessary, and cannot be removed without compromising the security of Hamsi.

### 3.4 Finding Zero-Sums for the Output Permutation

We describe a dedicated method to find large zero-sums for the 6-round permutation of the finalization function of Hamsi(we stress that it only applies to the internal permutation and not to the finalization as a whole, for it puts no restriction on the initial state). Contrary to Wagner's and the XHASH methods, it is deterministic rather than probabilistic, and needs to evaluate (and to know) only half the function.

In the spirit of [15, §9], we present an "inside-out" technique that exploits the fact that two *halves* of Hamsi's permutation have low algebraic degree. This differs from our method for finding $k$-sums which exploited the low degree of the full permutation. The attack works as follows:

1. Choose an arbitrary value for the state of Hamsi's permutation after three rounds.
2. Choose 28 distinct bits of the state.
3. Compute the $2^{28}$ initial states obtained by varying these bits and inverting the first three rounds of the permutation.

We obtain $2^{28}$ values that sum to zero, since their sum is the 28th-order derivative with respect to three inverse rounds. Their images also sum to zero, since they are the 28th-order derivative with respect to three forwards rounds (although the images are unknown, and need not be computed).

The method works whenever a function can be written as the composition of two low-degree functions. As explained in [4], the proposed technique is slightly more efficient than previous methods, for finding (here) zero-sums of $2^{28}$ elements.

## 4 First Order Differential Analysis

In this section, we analyze the differential properties of the Hamsi round transformations and show how to find high-probability differential paths for up to six rounds. Since we use XOR differences in our analysis, the differential propagation is deterministic in the message expansion and in the linear layer based on the $L$ transform. However, the propagation of differences through the Sbox layer is probabilistic and depends on the actual values of the input. To maximize the differential probability of a differential path, we try to minimize the number of active Sboxes during the path search.

### 4.1 Differential Properties of the Sbox

The differential distribution table (DDT) of the 4-bit Hamsi Sbox $S$ is given in Table 8 (Appendix A). Note that about half the differential transitions are impossible. The probabilities of the non-zero differentials are either $2^{-2}$ or $2^{-3}$. In our approach, besides minimizing the number of active Sboxes, we thus try to minimize the number of probability-$2^{-3}$ differentials.

### 4.2 Differential Properties of the Linear Transform $L$

The linear transform $L$ has on average good diffusion properties, that is, a few differences in the input lead to many differences in the output. Additionally, each bit of $L$ contributes to one of the 128 Sboxes in each round. To minimize the number of active Sboxes, we thus need to minimize number of differences in $L$. The Hamming weight (HW) of a difference is a good heuristic to measure the quality of a differential path. In the following, we first analyze the difference propagation through the linear layer for differences with HW one.

If we introduce a single input difference at bit position $i$ in one input word, the HW of the output differences depends on the position and word of the input difference. In Table 1 and Table 2 give the HW of the output difference for each of the 128 single bit input differences.

We observe that for some specific words and bit positions, the resulting HW can be quite small. This happens if one or more differences are removed by the shift operation. More specifically, the branch number of $L$ is only 3, so certain 1-bit input differences lead to only a 2-bit output difference, and vice versa. Table 1 and Table 2 show the worst case of diffusion, that is, the output HW for a multiple-bit input difference can be upper bounded by summing the corresponding table entries. However, when inserting many differences in several input words, some bit differences might erase each other, thus lowering the overall HW.

**Table 1.** Hamming weight of output differences if a single difference is introduced at one input word of the 128-bit linear transformation $(a', b', c', d') = L(a, b, c, d)$ of Hamsi in *forward* direction. The total and word-wise Hamming weight of the output difference is given depending on the bit position $i$ and input word of the input difference.

| Difference in input word | Position $i$ of input difference | Total HW of output diff. | HW of output diff. in $a'$ | $b'$ | $c'$ | $d'$ | Conditions (mod 32) |
|---|---|---|---|---|---|---|---|
| $a$ | 16,17 | 3 | 2 | 1 | - | - | $i+13 > 28, i+14 > 24$ |
| | 18 | 4 | 2 | 1 | 1 | - | $i+13 > 28, i+14 \leq 24$ |
| | 11...15 | 6 | 3 | 1 | 1 | 1 | $i+13 \leq 28, i+14 > 24$ |
| | else | 7 | 3 | 1 | 2 | 1 | $i+13 \leq 28, i+14 \leq 24$ |
| $b$ | 24...30 | 2 | 1 | 1 | - | - | $i+1 > 24$ |
| | else | 3 | 1 | 1 | 1 | - | $i+1 \leq 24$ |
| $c$ | 21...27 | 6 | 2 | 1 | 2 | 1 | $i+4 > 24$ |
| | else | 7 | 2 | 1 | 3 | 1 | $i+4 \leq 24$ |
| $d$ | | 3 | 1 | - | 1 | 1 | |

**Table 2.** Hamming weight of input differences if a single difference is introduced at one output word of the 128-bit linear transformation $(a', b', c', d') = L(a, b, c, d)$ of Hamsi in *backward* direction. The total and word-wise Hamming weight of the input difference is given depending on the bit position $i$ and output word of the output difference.

| Difference in output word | Position $i$ of output difference | Total HW of input diff. | HW of input diff. in $a$ | $b$ | $c$ | $d$ | Conditions (mod 32) |
|---|---|---|---|---|---|---|---|
| $a'$ | 2...4 | 2 | 1 | 1 | - | - | $i+27 > 28$ |
| | else | 3 | 1 | 1 | - | 1 | $i+27 \leq 28$ |
| $b'$ | 28...31 | 3 | 1 | 2 | - | - | $i > 28, i > 24$ |
| | 25...28 | 4 | 1 | 2 | - | 1 | $i \leq 28, i > 24$ |
| | never | 6 | 1 | 3 | 1 | 1 | $i > 28, i \leq 24$ |
| | else | 7 | 1 | 3 | 1 | 2 | $i \leq 28, i \leq 24$ |
| $c'$ | | 3 | - | 1 | 1 | 1 | |
| $d'$ | 29...31 | 4 | 1 | - | 1 | 2 | $i > 28$ |
| | else | 5 | 1 | - | 1 | 3 | $i \leq 28$ |

### 4.3 Near-Collisions for the Compression Function

Using our observations on the differential properties of Hamsi's Sbox and linear transform, we first searched manually for high-probability paths leading to near-collisions for the compression function, given some difference in the chaining value.

Previous work by Nikolic reported near collisions [12] on $(256 - 25)$ bits with 14 differences in the chaining value; work by Wang et al. reported [17] near collisions on $(256 - 23)$ bits with 16 differences. Below we present near collisions on $(256 - 25)$ bits with only six differences in the chaining value, using the differential path in Table 3.

**Table 3.** Differential path for three rounds of Hamsi with probability $2^{-26}$

| It. | Sbox input | Sbox output | Prob. |
|---|---|---|---|
| 1 | 00000000 00000000 00020000 00000002<br>00004000 00000000 00000000 00000000<br>00000000 00000000 00020000 00000002<br>00004000 00000000 00000000 00000000 | 00000000 00000000 00000000 00000002<br>00004000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00020000 00000000 | 8 |
| 2 | 00000000 00000000 00000000 00080000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000 | 00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00080000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00080000 | 3 |
| 3 | 80000000 00000000 02000000 00000000<br>00000000 00000000 00000000 00100000<br>00020000 00000000 00010000 00000000<br>00000000 00000000 00000000 04000000 | 00000000 00000000 00000000 04100000<br>80020000 00000000 02010000 04100000<br>00020000 00000000 00000000 00000000<br>80000000 00000000 02010000 00000000 | 15 |
| End | 00000000 80400800 00000000 10C130C0<br>00040105 00000000 04020000 08000000<br>00020400 A040A0A2 00000000 10004000<br>00000040 08000000 00820801 00000000 | | |

The differential path in Table 3 is followed with probability $2^{-26}$ under standard uniformity and independence assumptions. However, for the IV defined in [9] the path is followed with probability $2^{-23}$. This is because of the condition put by the two fixed bits in each Sbox. These probabilities were verified experimentally.

Finally, note that the near collisions also result in other 4-sums: for example, for the IV $h_0$ specified in [9], the IV $h_1$ obtained by applying the weight-6 initial difference in Table 3, and the message $M_1$=`C33BE456` and $M_2$=`C8D1B855`, we have:

1. A near collision between $H(h_0, M_1)$ and $H(h_1, M_1)$.
2. A near collision between $H(h_0, M_2)$ and $H(h_1, M_2)$.
3. A 4-sum $H(h_0, M_1) \oplus H(h_1, M_1) \oplus H(h_0, M_2) \oplus H(h_1, M_2) = 0$.

For inputs of an "ideal" function, the latter equality is unlikely to hold with probability $2^{-23}$, but rather with probability close to $2^{-256}$.

In the following, we automate our search for high-probability differential paths. Our heuristic algorithm, described in the next section, produced good differential paths for up to six rounds of Hamsi.

### 4.4 Automated Differential Path Search

As before, we search for differential paths with some difference in the input and output chaining value, and no difference in the input message. The resulting

6-round paths allow us to distinguish the output transform from random, as shown in Sect. 5.4.

Our primary heuristic is to minimise the HW of the differences in each round. To achieve that goal, we start with a very low HW (1 or 2 bit) difference in the middle of the path (at the start of round 3 for a 6-round search) and let the difference spread in both forward and backward directions. Additionally, we try to maximise the transition probabilities and randomize the search.

More precisely, our automated differential path starts from the input of the Sbox layer in round 3, forcing a 1-bit or 2-bit input difference on only one Sbox position $i$ (among the 128 possible bit positions). We then choose one of the best differential transitions through the forward application of the Sbox and apply the linear layer on this new internal state. By best Sbox transitions, we mean the transitions that lead to a low HW after the application of the linear layer. To keep the search complexity feasible, we apply the $L$-layer to each active S-box separately and use the sum of the HWs as an estimate of the total output HW at the end of each round. Since the path is sparse, the sum of HWs proves to be a good heuristic. We continue picking the best differential transitions for all the active Sbox positions until the end of the fifth round of the output function of Hamsi. As the final output HW of the difference does not influence the path complexity, we optimise for transition probabilities in the last round, and pick the most probable differential Sbox transitions (not the ones minimizing the HW). Finally, we apply the very last linear layer to obtain the full path.

The backward computation is done analogously in the middle rounds, applying the linear layer backward and picking the best backward differential transitions for all active Sboxes. In the first round (the last round when computing backward) we impose additional restrictions in order to fulfill constraints on the message expansion.

As we force no difference in the message input of the compression function, we expect the 256-bit expanded message word to contain no difference at all. Hence, in the first round we only allow Sbox transitions where the difference in the expanded message bits is zero. Note that the probabilities of the first-round transitions do not affect the complexity of the path, as long as they are different from 0. Indeed, in the first round we can use the freedom of the chaining input to fulfill the conditions on the Sboxes and we expect the complexity cost of this first round to be negligible.

In order to increase our chances to obtain a good trail, we randomized the search with several parameters. First, we randomized the first 1-bit or 2-bit perturbation introduction in the output of round 3, as well as its position $i$ among the 128 Sbox locations. Furthermore, we are also randomizing the Sbox transitions when several candidates are equally good. Finally, another improvement has been incorporated in our implementation: after having found a potentially interesting 6-round candidate, we recompute the forward search by allowing more differential transitions through the Sbox. Said in other words, after having placed ourselves in an interesting differential paths subspace, we look in the neighborhood if better ones exist.

Our heuristic search revealed that after three rounds in both backward and forward directions, the diffusion of Hamsi is not sufficient to avoid high-probability differential paths and we can find a differential path with a rather low total HW and good probability. We were able to construct a 6-round differential path with a relatively high probability, which is used to distinguish the the whole Hamsi output transformation in the following section.

## 5 Non-randomness of the Ouput Transformation

### 5.1 The Differential Path

The best 6-round path produced by our randomized search program is depicted in Table 4. We can find an input pair (chaining values and messages) conforming to this path with a probability of $2^{-206}$. Note that in the first round we have a probability of $2^{-58}$ for a random message and a random chaining value. However, we can fix a suitable message (see below), and choose a valid chaining value bit-by-bit such that the desired output difference is guaranteed. This means that we can find a conforming input pair to the differential path with a complexity of about $2^{148}$.

### 5.2 First Round and Message Expansion

In the first iteration, active S-boxes impose conditions on the expanded message: for a given non-zero Sbox differential, only one or two pairs of values of the corresponding two expanded message bits are possible. Since we have only 32 degrees of freedom in the message, we need to keep the number of active Sboxes in the first round low. To improve the probability of finding a suitable message candidate, we can vary the differences in the chaining values, whenever several input differences lead to the same output difference of the first Sbox layer. These relaxable differential Sbox transitions are listed in Table 5. In our path, five of the 23 active Sboxes of the first iteration are relaxable. In total, we have only nine Sboxes with two constraints on the message bits; 12 Sboxes with one constraint on the message; and two S-Boxes with a "half" constraint on the message (three of four bit pairs are possible). Therefore, we expect to find $2^{32-2\times 9-12} \cdot \left(\frac{3}{4}\right)^2 \approx 2$ messages satisfying the relaxed first round differential. In practice, we found one such message using the constants of permutation $P$ and three messages using the constants of the output permutation $P_f$ (see the full version of this paper for an example [1]). Note that finding conforming message words can be done in $2^{32}$ by exhaustive search. The complexity to find chaining values such that the first four rounds of the path are satisfied is about $2^{25}$, since we can fulfill the conditions in the first round deterministically.

### 5.3 Last Round and Truncation

In order to improve the probability of the differential path, we consider truncated differentials in the last application of the Sbox. Namely, we relax the Sbox transitions by fixing some bits in the output difference, while letting the remaining

**Table 4.** Differential path for six rounds of Hamsi with probability $2^{-148}$

| It. | Sbox input | Sbox output | Prob. |
|---|---|---|---|
| start | | 00000000 00000000 84004880 4081C400<br>2C020018 000045C0 00000000 00000000<br>00000000 00000000 84024880 4081C400<br>28020018 000045C0 00000000 00000000 | |
| 1 | 00000000 00000000 84004880 4081C400<br>2C020018 000045C0 00000000 00000000<br>00000000 00000000 84024880 4081C400<br>28020018 000045C0 00000000 00000000 | 04000000 00000000 04000000 40818000<br>28020018 000040C0 04020000 00000000<br>00000018 00004100 00000800 00804000<br>04020000 000004C0 80024880 00004400 | (58) |
| 2 | 00000000 00000000 00000000 00010000<br>30000010 00000080 00000000 00000080<br>30000010 00000080 00000000 00010080<br>00000000 00000000 00000000 00000000 | 00000000 00000000 00000000 00010000<br>30000000 00000000 00000000 00000080<br>00000010 00000000 00000000 00000000<br>00000000 00000080 00000000 00000000 | 17 |
| 3 | 00000000 00000000 00000000 00000000<br>20000000 00000000 00000000 00000000<br>20000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000 | 00000000 00000000 00000000 00000000<br>20000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000 | 3 |
| 4 | 00000000 00000000 00000000 00000008<br>40000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000000 | 40000000 00000000 00000000 00000000<br>40000000 00000000 00000000 00000008<br>00000000 00000000 00000000 00000000<br>00000000 00000000 00000000 00000008 | 5 |
| 5 | 04038000 00000000 00000200 00000010<br>80000000 00001000 00000000 00000010<br>00000002 00000000 00000a01 00000000<br>00000000 00000000 00000000 00200400 | 80000000 00001000 00000000 00200410<br>04038002 00001000 00000801 00000000<br>00000000 00000000 00000000 00000000<br>84038002 00000000 00000a01 00200400 | 33 |
| 6 | 08420002 F8022900 00000000 30821140<br>0903000C 00000000 04001002 00000000<br>00000000 A0A26145 00041080 12807200<br>01C0014A 00000000 08051082 10420000 | 08830144 A0022100 0C051080 10C01000<br>0181014C 58A04845 0C051082 22406340<br>01800148 58A04845 08011002 22406340<br>00400002 58000800 00040080 20020140 | 90 |
| End | CD9F7546 362513EA 56FE147F 85F6B1E1<br>8D0682FD F100928A B44C3D06 18A0D101<br>B8871BEA 70315A82 4819C14B 26257026<br>A1DD0199 40072022 8329356A A744E830 | | |

bits vary. Since the "a"-bits and "c" bits diffuse faster through the linear layer (see Table 1), we chose to fix these bits in the output of each Sbox. Amongst four different truncated output differences (?0?0, ?0?1, ?1?0 and ?1?1), we chose, for each input, the output difference with the highest probability. Table 6 lists the relaxed input-output transitions for the Sbox. Details of the path used can be found in the full version of the article [1].

Relaxing the Sbox transitions increases the probability of the last round to $2^{-61.8}$, giving a total path complexity $2^{-120.8}$. At the same time, since the "wild card" bits are chosen to have low diffusion, the difference is still fixed in 180 bits of the chaining value. Thus, we obtain a distinguisher by observing the difference in these output bits.

**Table 5.** Relaxable differential transitions for the first round of the Hamsi Sbox. The first table shows the possible input differences that give the same output if 1, 4 and 5 are the only possible Sbox input differences. The second table shows the same possibilities if 2, 8, and 10 are the only possible Sbox input differences. For each underlined transition two message pairs are possible, while for the other transitions only one message pair is possible.

| Desired output | 1 a | 2 b | 3 ab | 4 c | 5 ac | 6 bc | 7 abc | 8 d | 9 ad | 10 bd | 11 abd | 12 cd | 13 acd | 14 bcd | 15 abcd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Possible input | | | | | 1<br>5 | <u>4</u><br>5 | | | | 1<br><u>4</u> | 1<br><u>4</u> | 1<br>5 | | | |

| Desired output | 1 a | 2 b | 3 ab | 4 c | 5 ac | 6 bc | 7 abc | 8 d | 9 ad | 10 bd | 11 abd | 12 cd | 13 acd | 14 bcd | 15 abcd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Possible input | | | <u>2</u><br>8 | | <u>2</u><br>8 | | | | <u>2</u><br>8 | | | | | | <u>2</u><br>8 |

**Table 6.** Relaxed differential transitions for the last round of the Hamsi Sbox. The table shows the chosen set of output differences for each given input difference. Underlined transitions have probability $2^{-2}$, while the other transitions have probability $2^{-3}$.

| input | 1 a | 2 b | 3 ab | 4 c | 5 ac | 6 bc | 7 abc | 8 d | 9 ad | 10 bd | 11 abd | 12 cd | 13 acd | 14 bcd | 15 abcd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| output | 12<br><u>14</u> | <u>3</u><br><u>9</u> | <u>1</u><br>9 | <u>10</u> | <u>1</u><br>3 | 2<br>8 | <u>4</u><br>12 | 5<br><u>7</u><br><u>13</u><br>15 | <u>8</u><br>10 | 2<br>8<br><u>10</u> | <u>1</u><br>9 | <u>11</u> | <u>1</u><br>3 | <u>7</u><br><u>13</u> | <u>2</u><br>10 |
| mask | 11?0 | ?0?1 | ?001 | 1010 | 00?1 | ?0?0 | ?100 | ?1?1 | 10?0 | ?0?0 | ?001 | 1011 | 00?1 | ?1?1 | ?010 |

### 5.4 Distinguishing the Output Transformation

To distinguish the output transformation of Hamsi we use the concept of differential $q$-multicollision introduced by Biryukov et al. in the cryptanalysis of AES-256 [6] and applied to the SHA-3 candidate SIMD in [11]. Originally, differential $q$-multicollision have been applied to a block cipher but can be easily adapted to a random function. A differential $q$-multicollision for a random (compression) function $f(H, M)$ is a set of two differences $\Delta H$, $\Delta M$ and $q$ pairs $(H_1, M_1), (H_2, M_2), \ldots, (H_q, M_q)$ such that:

$$
\begin{aligned}
&f(H_1, M_1) \oplus f(H_1 \oplus \Delta H, M_1 \oplus \Delta M) = \\
&f(H_2, M_2) \oplus f(H_2 \oplus \Delta H, M_2 \oplus \Delta M) = \\
&\ldots \\
&f(H_q, M_q) \oplus f(H_q \oplus \Delta H, M_q \oplus \Delta M)
\end{aligned}
$$

The generic complexity to find differential $q$-multicollision for a random function $f$ with output size $n$ is at least $q \cdot 2^{\frac{q-2}{q+2} \cdot n}$ evaluations of $f$.

In the case of Hamsi-256, the function $f$ is the output transformation, the message difference $\Delta M$ is zero and the output size is $n = 256$. The generic complexity to find differential $q$-multicollision should be $q \cdot 2^{\frac{q-2}{q+2} \cdot 256}$ and we get for $q = 8$ a generic complexity of $2^{156.1}$. Using our differential path of Section 5.1, we get for $q = 8$ a complexity of $8 \cdot 2^{148} = 2^{151}$. Hence, for $q \geq 8$ we can distinguish the output transfomation of Hamsi from a random function, since we expect to find a $q$-multicollision approximately 32 times faster than for an ideal transform.

Due to the relaxed conditions, we only fix a truncated difference in 180 output bits and hence, we get $n = 180$. In this case, the generic complexity for $q = 11$ is $q \cdot 2^{\frac{q-2}{q+2} \cdot 180} = 2^{128.1}$. Using the relaxed differential path, we get $q \cdot 2^{120.8} = 2^{124.3}$ and hence, can distinguish the output transfomation of Hamsi from a random function for $q \geq 11$.

## 6 Conclusion

We investigated the resistance of the 256-bit version of the second round SHA-3 candidate Hamsi against differential and higher-order differential attacks.

Using higher-order analysis, we showed that the 3-round compression function of Hamsi has suboptimal algebraic degree. Using this observation, we provided sets of four related IV's such that the outputs of the compression function obtained with a given fixed message sum to zero. We also presented a set of $2^{19}$ message words such that the output chaining values, using any fixed IV, sum to zero. The latter result indicates that the compression function of Hamsi, when seen as a function of message words, does not reach the expected maximal degree 27. As an application, we note that the low degree makes the standalone compression function existentially forgeable in the message authentication setting.

Further, we constructed high-probability differential paths for the 3-round compression function to demonstrate a near-collision on $(256-25)$ bits with only six differences in the input chaining value. We have also developed a technique for building low-weight, high-probability differential paths for more rounds of Hamsi. Our best differential path for six rounds has probability $2^{-148}$, much higher than expected for a random function. Additionally, we gave a truncated differential on 180 output bits with probability $2^{-120.8}$. These are the first results on six rounds of Hamsi, allowing us to distinguish the full output transformation from a random function using differential $q$-multicollisions.

Although none of our findings directly leads to an attack on the hash algorithm, they indicate that the buildings blocks of Hamsi exhibit nonrandom behavior. We expect our work to serve as a starting point for future analysis of Hamsi.

In order to prevent more serious attacks, we recommend increasing the number of rounds in the output transformation as a precaution. While the current specification does not include performance figures for the 8-round alternative, this change is only expected to noticeably affect the speed of hashing short messages.

## Acknowledgements

Emilia Käsper thanks the Computer Laboratory of the University of Cambridge for hosting her.

This work was supported in part by the European Commission through the ICT Programme under Contract ICT-2007-216646 ECRYPT II. Emilia Käsper was also supported by the IAP–Belgian State–Belgian Science Policy BCRYPT and the IBBT (Interdisciplinary institute for BroadBand Technology) of the Flemish Government.

## References

1. Aumasson, J.P., Käsper, E., Knudsen, L.R., Matusiewicz, K., Odegaard, R., Peyrin, T., Schlffer, M.: Differential distinguishers for the compression function and output transformation of Hamsi-256. Cryptology ePrint Archive, Report 2010/091 (2010)
2. Bellare, M., Micciancio, D.: A new paradigm for collision-free hashing: Incrementality at reduced cost. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 163–192. Springer, Heidelberg (1997)
3. Bernstein, D.J.: Better price-performance ratios for generalized birthday attacks. In: SHARCS (2007), http://cr.yp.to/papers.html#genbday
4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Note on zero-sum distinguishers of keccak-f. NIST mailing list (2010), http://keccak.noekeon.org/NoteZeroSum.pdf
5. Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A new block cipher proposal. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 222–238. Springer, Heidelberg (1998)
6. Khovratovich, D., Biryukov, A., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
7. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
8. Kücük, O.: The hash function Hamsi. Submission to NIST (January 2009), http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/HamsiUpdate.zip
9. Kücük, O.: Reference implementation of Hamsi. Submission to NIST (January 2009)
10. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R., Costello Jr., D., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography, pp. 227–233. Kluwer, Dordrecht (1992)
11. Mendel, F., Nad, T.: A distinguisher for the compression function of simd-512. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 219–232. Springer, Heidelberg (2009)
12. Nikolić, I.: Near collisions for the compression function of Hamsi-256. CRYPTO rump session (2009), http://rump2009.cr.yp.to/936779b3afb9b48a404b487d6865091d.pdf
13. NIST: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register Notice. 72(112) (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

14. Singh, B., Alexander, L., Burman, S.: On algebraic relations of Serpent S-boxes. Cryptology ePrint Archive, Report 2009/038 (2009)
15. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
16. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
17. Wang, M., Wang, X., Jia, K., Wang, W.: New pseudo-near-collision attack on reduced-round of Hamsi-256. Cryptology ePrint Archive, Report 2009/484 (2009)
18. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
19. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
20. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
21. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

## A The Sbox of Hamsi

**Table 7.** The Hamsi Sbox in decimal basis

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | 8 | 6 | 7 | 9 | 3 | 12 | 10 | 15 | 13 | 1 | 14 | 4 | 0 | 11 | 5 | 2 |

**Table 8.** The differential distribution table (DDT) of the Hamsi Sbox in decimal basis

| In \ Out | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 4 | 2 |
| 2 | 0 | 0 | 0 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 4 |
| 3 | 0 | 4 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 2 | 2 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 4 | 4 | 0 | 4 | 0 | 0 |
| 5 | 0 | 4 | 0 | 2 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| 6 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 2 |
| 7 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 2 |
| 8 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 |
| 9 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 4 | 2 | 2 | 2 | 2 | 0 | 0 | 0 |
| 10 | 0 | 0 | 2 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 4 | 0 | 0 | 0 | 2 | 0 |
| 11 | 0 | 4 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 2 |
| 12 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 4 | 2 | 0 |
| 13 | 0 | 4 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 |
| 14 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 0 | 4 | 2 | 0 |
| 15 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 2 | 0 | 0 | 0 |