Collision Attack on the Hamsi-256 Compression Function

Mario Lamberger[1], Florian Mendel[2], and Vincent Rijmen[2]

[1] NXP Semiconductors, Austria
[2] Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Belgium

Abstract. Hamsi-256 is a cryptographic hash functions submitted by Küçük to the NIST SHA-3 competition in 2008. It was selected by NIST as one of the 14 round 2 candidates in 2009. Even though Hamsi-256 did not make it to the final round in 2010 it is still an interesting target for cryptanalysts. Since Hamsi-256 has been proposed, it received a great deal of cryptanalysis. Besides the second-preimage attacks on the hash function, most cryptanalysis focused on non-random properties of the compression function or output transformation of Hamsi-256. Interestingly, the collision resistance of the hash or compression function got much less attention. In this paper, we present a collision attack on the Hamsi-256 compression function with a complexity of about $2^{124.1}$.

Keywords: hash function, differential cryptanalysis, collision attack

1 Introduction

In recent years, significant advances in the field of hash function research have been made which had a formative influence on the landscape of hash functions. Especially the work on MD5 and SHA-1 [18,19] has convinced many cryptographers that these widely deployed hash functions can no longer be considered secure. As a consequence, researchers are evaluating alternative hash functions in the SHA-3 initiative organized by NIST [15]. The goal is to find a hash function which is fast and still secure within the next few decades.

Many new and interesting hash functions have been proposed. Hamsi-256 [11], proposed by Küçük, was one of the 64 submissions to the SHA-3 competition from which 51 submissions were selected for the first round in 2008 and 14 of them advanced to the second round in 2009. Hamsi-256 was one of them. Even though Hamsi-256 was not selected as one of the five finalists in 2010, mainly because of the second-preimage attacks, it is still an interesting target for cryptanalysts. In this work, we focus on the collision resistance on the Hamsi-256 compression function, which in turn gives new insights in the collision resistance of the hash function.

Previous Analysis. Hamsi-256 received a great deal of cryptanalysis during the ongoing SHA-3 competition. However, the only analysis of the Hamsi-256

S. Galbraith and M. Nandi (Eds.): INDOCRYPT 2012, LNCS 7668, pp. 156–171, 2012.
The original publication is available at http://dx.doi.org/10.1007/978-3-642-34931-7_10

hash function itself is due to Dinur and Shamir [7,8] and Fuhr [9]. Both attacks are algebraic attacks targeting the second-preimage security of the hash function. The attacks are based on the observation that it is sufficient to show that one of the hashed output bits is wrong to discard a possible second-preimage. Since the output bits of the compression function of Hamsi-256 can be described by low degree polynomials, it is faster to compute a small number of output bits by a fast polynomial evaluation technique than with the original algorithm. The results are second-preimage attacks on Hamsi-256 with a complexity of about 2^{247} and $2^{251.3}$, respectively. But then again, one still needs to test 2^{256} inputs to find a second preimage as in the generic case. In other words, the attacks are a clever way to speed up brute force search. We want to note that in a similar way also the complexity of a generic collision search for the compression function of Hamsi-256 can be improved, resulting in an attack complexity of 2^{125} [7]. Moreover, in [12] Küçük showed a collision attack for a simplified version of the Hamsi-256 compression function, ignoring the message expansion.

Most other attacks published so far are differential attacks targeting the compression function or output transformation of Hamsi-256. Practical near-collisions for the compression function have been shown in [1,13,16,17] and a distinguisher for the compression function has been presented in [5]. Furthermore, non-random properties for the underlying permutation of the Hamsi-256 compression function and output transformation have been demonstrated in [1,4] and a distinguisher for the output transformation of Hamsi-256 has been described in [1].

Our Contribution. In this work, we present a collision attack for the Hamsi-256 compression function. Our collision attack is based on the attack of Çalik and Turan [5] and has a complexity of about $2^{124.1}$ compression function evaluations. The main idea of the attack is very simple. We extend the approach of Çalik and Turan, which was originally used to show non-random properties in the compression function. This is then used to fix some output bits of the compression function to a predefined value faster than in the generic case. Finally, we use a birthday attack on the remaining bits to construct a collision for the compression function of Hamsi-256. Even though the complexity of the attack is very high, namely $2^{124.1}$ compression function evaluations, it demonstrates that the compression function of Hamsi-256 is not collision resistant and gives new insights in the security of Hamsi-256. However, it has to be noted that the attack cannot be extended to the hash function.

Outline. The remainder of the paper is organized as follows. In Section 2, we give a short description of the Hamsi-256 compression function. In Section 3, we describe the distinguishing attack of Çalik and Turan on Hamsi-256, since it is the basis for our collision attack. We present our new attack strategy in Section 4 and apply it to the Hamsi-256 compression function in Section 5. Finally, we conclude in Section 6.

2 Description of Hamsi-256

Hamsi-256 is a cryptographic hash function proposed by Küçük [11] which has been submitted to the SHA-3 competition in 2008. It is an iterated hash function based on the Merkle-Damgård design principle [6,14] and produces a 256-bit hash value. Like most hash functions, Hamsi-256 iterates a compression function f to compute the hash value. It takes a 32-bit message block M_i and a 256-bit chaining value h_{i-1} as input and outputs a 256-bit chaining value h_i. In the following, we give a brief overview of the compression function of Hamsi-256 (see Figure 1).

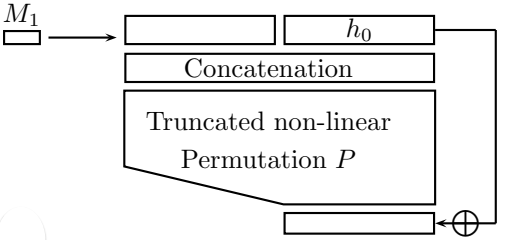

Fig. 1. The compression function of Hamsi-256 [11].

Message Expansion. The message expansion of Hamsi-256 uses a linear code to expand the 32-bit message word M_i into eight 32-bit words $m_0, m_1, \ldots, m_7$, i.e. 256 bits. The $(128, 16, 70)$ linear code defined over $\mathbb{F}_4$ used in the message expansion ensures that any difference in the message word M_i will lead to differences in at least 70 of the 128 columns of the initial state. For a detailed description of the message expansion we refer to [11].

Concatenation. The 256-bit expanded message $(m_0, \cdots, m_7)$ and the 256-bit chaining value $h_{i-1} = (c_0, \ldots, c_7)$ are concatenated to form the 512-bit initial state (see Figure 2). We want to note that the initial state can be considered as both a 4×4 matrix of 32-bit words and 128 columns each consisting of 4 bits.

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. The initial state of the compression function after concatenation.

Non-linear Permutation. The non-linear permutation P used in the Hamsi-256 compression function is composed of 3 rounds. In each round the round transformation updates the state by means of a sequence of transformations:

Addition of Constants Predefined constants and a round counter is xored to the whole state. For the value of the constants we refer to [11].

Substitution Each of the 128 columns of the state (each 4 bits) is updated by the 4-bit s-box of the block cipher Serpent [2].

Diffusion The linear transformation L of the block cipher Serpent, which accepts four 32-bit input words, and outputs four 32-bit words is applied to the four independent diagonals of the state.

A detailed description of the s-box and the linear transformation L is given in [11] and the differential properties of the s-box and linear transformation of Hamsi-256 have been studied for instance in [1] among others.

Truncation and Feed-Forward. Finally, after the application of the permutation P the second and fourth rows of the state are discarded and the initial chaining value is xored to the truncated state resulting in the initial chaining value for the next iteration or the input to the output transformation to compute the final hash value. For the description of the output transformation of Hamsi-256 we refer to [11].

3 Attack of Çalik and Turan

In this section, we briefly describe the distinguishing attack of Çalik and Turan [5] on the Hamsi-256 compression function, since our collision attack builds upon it. The attack is a differential attack exploiting the fact that for a given input difference not all the output bits of the compression function are affected. This results in a distinguisher for the compression function. Since the message expansion of Hamsi-256 uses a $(128, 16, 70)$ linear code, any difference in the message will lead to differences in at least 70 of the 128 columns of the initial state. In other words, at least 70 s-boxes will be active in the initial state. Therefore, Çalik and Turan consider only differences in the chaining value in their analysis. Furthermore, they restrict themselves to differences in only one column, i.e. one active s-box, in the initial state. Since each column of the initial state contains two bits of the chaining value (see Figure 2), three non-zero differences can be injected to a column: these differences can be $2_x, 8_x$ or a_x for columns 0-63 and $1_x, 4_x$ or 5_x for columns 64-127.

To find the output bits that are not affected by one of the $3 \cdot 128$ possible input differences the authors trace the differences through the round transformations of Hamsi-256 and mark all the bits of the internal state that could have a difference. If there are any unmarked bits in the state after 3 rounds, then one knows that these bits are not affected by the initial difference. However, due to the feed-forward, some of these unaffected bit positions may coincide with the difference in

the initial state, resulting in bits that will always change. However, in the paper the authors do not make a distinction between these two cases (and neither will we) and use for both cases the term unaffected bits.

As noted by Çalik and Turan the number of unaffected output bits depends on the Hamming weight of the difference in the initial state. For the differences with Hamming weight 2, i.e. $\mathtt{5}_x$ and $\mathtt{a}_x$, the number of unaffected bits is higher. The reason for this is that differences with Hamming weight 1, i.e. $\mathtt{1}_x, \mathtt{2}_x, \mathtt{4}_x$ and $\mathtt{8}_x$, will lead to a difference with Hamming weight at least 2 at the output of the s-box in round 1, whereas a difference of Hamming weight 2 can lead to a difference with Hamming weight 1, resulting in a sparser difference at the output of round 1. Hence, it is not surprising that one could find 64 solutions with at least one unaffected output bit using an initial difference with Hamming weight 2, cf. [5, Table 4], while no solution could be found using an initial difference with Hamming weight 1. An example with two unaffected output bits and an initial difference with Hamming weight 2 is given in Figure 3.

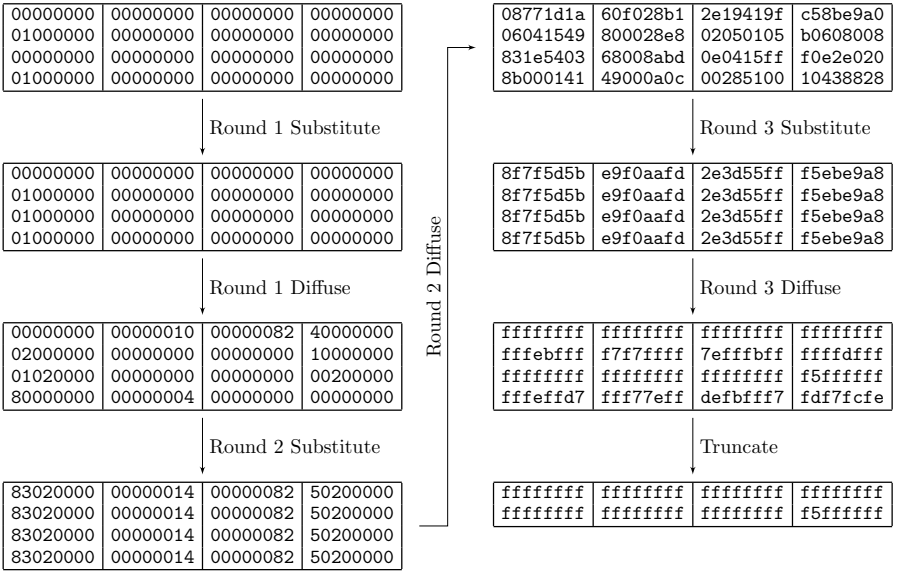

Fig. 3. Propagation of affected bits for the initial difference $\mathtt{a}_x$ inserted to column 7 of the initial state [5].

Another factor that influences the number of unaffected output bits are the values of the message bits in the column with the initial difference. By imposing some additional restrictions on the message bits one could observe more unaffected output bits. The reason for this is that for certain choices of the message bits the Hamming weight of the difference at the output of the s-box might be

smaller. This results in a sparser difference at the output of round 1. By choosing the message bits carefully, Çalik and Turan could find 8 solutions with two unaffected bits at the output using an initial difference with Hamming weight 1, cf. [5, Table 3]. Furthermore, also for an initial difference with Hamming weight 2 the number of solutions as well as the number of unaffected bits can be increased by choosing the message bits accordingly. For instance, in the example given in Figure 3 the number of unaffected output bits can be increased to 9 bits by setting the message bits in column 7 to 0_x. By additionally also imposing conditions on the chaining bits in column 7, i.e. 0_x or 3_x, the number of unaffected bits can be increased to 62.

Based on these differential properties of the compression function Çalik and Turan describe several attacks on Hamsi-256 in the original paper. First, they show a distinguishing attack on the compression function that needs only a few compression function evaluations. Then, they present a message-recovery attack for the compression function with a complexity of $2^{10.5}$ and a pseudo-preimage attack with complexity of about $2^{254.25}$. For a detailed description of these attacks we refer to [5].

In this work, we will use these differential properties of the compression function to show a collision attack for the Hamsi-256 compression function. We describe the basic idea of the attack in the next section.

4 Basic Attack Strategy

In this section, we present the basic attack strategy employed by our new attack to construct collisions in the compression function of Hamsi-256. It is based on the concept of neutral bits [3] and auxiliary differentials [10], which were originally used to speed up differential collision attacks on hash functions.

The main idea of our attack is quite simple. Assume we can find a distinct input differences (in the following referred to as auxiliary differentials), where each difference only affects a few output bits of the compression function. Further, we assume that there exists at least b output bits (for the sake of simplicity say bits $0, 1, \ldots, b-1$) that are not affected by any of these a auxiliary differences. Then this can be used to find 2^a partial preimages for these b output bits of the compression function with a complexity of about $2^a + 2^b$, compared to the generic case of 2^{a+b} compression function evaluations.

Let $v_1, \ldots, v_a$ denote the a auxiliary differentials not affecting the first b output bits of the compression function and assume for the sake of simplicity that we want to fix these b output bits to 0. Then the attack can be summarized as follows.

1. Choose random values for the chaining input h_{i-1} and the message input m_i. If necessary fulfilling all conditions imposed by the a auxiliary differentials $v_1, \ldots, v_a$.
2. Compute the output of the compression function h_i and check if the b output bits of h_i are correct

- If all b bits are 0 then continue with step 3
- else go back to step 1

3. Use the a auxiliary differentials $v_1, \ldots, v_a$ to generate 2^a additional solutions where the first b output bits of the compression function are also 0

$$h_i^{(\underline{d})} = h_i \oplus \bigoplus_{j=1}^{a} d_j \cdot v_j$$

with $\underline{d} = (d_1, \ldots, d_a) \in \{0,1\}^a$.

Note that we need to repeat step 1-2 about 2^b times to find a correct h_i, resulting in a complexity of about 2^b compression function evaluations to finish step 1 and 2. Since step 3 has a complexity of 2^a, the final complexity of the attack is $2^a + 2^b$ compression function evaluations. But then again, we found $1 + 2^a$ partial preimages where the first b output bits $(0, 1, \ldots, b-1)$ are 0 with complexity of $2^a + 2^b$. This can now be used to construct a collision for the compression function faster than in the generic case.

Since we can find about 2^a partial preimages for the first b with a complexity of $2^a + 2^b$ (instead of 2^{a+b}) we can combine this with a birthday attack to find a collision for the compression function faster than in the generic case. By repeating the attack about t times with $t = 2^{(n-b)/2-a}$ we get $t \cdot 2^a = 2^{(n-b)/2}$ outputs where the first b bits collide and due to the birthday paradox we expect to find at least one pair of outputs where also the remaining $n - b$ bits collide. The result is a collision attack on the compression function with a complexity of about $t \cdot (2^a + 2^b) = 2^{n/2} \cdot (2^{-b/2} + 2^{b/2-a})$ compression function evaluations.

Clearly the complexity of the attack depends the value of a and b. For the above computed complexity we can easily observe that the value

$$2^{-b/2} + 2^{b/2-a} = \frac{2^{b-a} + 1}{2^{b/2}}$$

is minimized if in the numerator we have $a \geq b$, and in the denominator we have b as large as possible, so basically $a = b$. The main question is now which values of a and b we can expect in our attack. On the one hand this number depends on the size of the set S containing all auxiliary differentials, and on the other hand on the number of affected output bits of each of these auxiliary differentials in the set. Moreover, also the number of conditions imposed on the message bits by the auxiliary differentials might be a limiting factor, since this is only 32 bits in the case of the Hamsi-256 compression function.

4.1 Probabilistic Considerations

In the following, we denote by $[n]$ the integer interval $\{1, 2, \ldots, n\}$, by $2^{[n]}$ we mean the set of all subsets of $[n]$ and by $\binom{[n]}{k}$ all subsets of $[n]$ of size k. We are looking at subsets of $\binom{[n]}{k}$ because on average, the number of unaffected output

bits is $\approx k$ in our applications. Furthermore, we assume that the auxiliary differentials are independent and that the unaffected bits of each auxiliary differential are randomly distributed.

We want to investigate the probability $P(N, n, k, b)$ that a set $S \subseteq \binom{[n]}{k}$ of size N of auxiliary differentials contains a subset S' of size a such that the elements $s_i \in S'$ satisfying

$$\left| \bigcap_{i=1}^{a} s_i \right| \geq b\,. \tag{1}$$

Now we have

$$P(N, n, k, b) \leq \binom{N}{a} \cdot \sum_{j=b}^{k} P_r(n, k, j) \tag{2}$$

where $P_r(n, k, j)$ denotes the probability that a randomly chosen subsets $s_i \in \binom{[n]}{k}$ satisfy $|\bigcap_{i=1}^{a} s_i| = j$. The exact distribution is hard to compute, however we can come up with the following approximation. Since each set s_i has size k, we assume that a randomly chosen element $e \in [n]$ is contained in s_i with probability k/n. Thus, the probability for a randomly chosen element $e \in [n]$ to be contained in $\bigcap_{i=1}^{a} s_i$ is $p = (k/n)^a$. From this we deduce that

$$\sum_{j=b}^{k} P_r(n, k, j) = \sum_{j=b}^{k} \binom{n}{j} p^j (1-p)^{n-j}$$

which in turn leads to

$$P(N, n, k, r) \leq \binom{N}{a} \cdot \sum_{j=b}^{k} \binom{n}{j} p^j (1-p)^{n-j} \tag{3}$$

5 Application to Hamsi-256

In this section, we will discuss the application of the attack strategy described in the previous section to the Hamsi-256 compression function. Therefore, we first need to find a set S of auxiliary differentials that only affect a few output bits of the compression function of Hamsi-256. Then we need to find a subset of a auxiliary differentials not affecting the same b output bits (for large values of a and b). To construct the set S it seems natural to use the same differentials as Çalik and Turan in their distinguishing attack described in Section 3. However, since we are aiming for a large value of a and b, in order to increase the effectiveness of the attack, we are only interested in auxiliary differentials where the number of unaffected output bits is large. This already rules out all auxiliary differentials with an initial difference of Hamming weight 1. For these cases the maximum number of unaffected output bits is at most 2 (see Section 3). However, auxiliary differentials with an initial difference of Hamming weight 2 might be a good choice, in particular since the number of unaffected output bits

can be increased to up to 62 bits by imposing some additional conditions on the chaining and message bits (see Section 3).

In total we found 198 auxiliary differentials with an initial difference of Hamming weight 2. As shown in Figure 5 in the appendix the number of unaffected output bits is on average 30. Note that since we are interested in large values of a and b, we only considered auxiliary differentials where the number of unaffected output bits is at least 10. Now assuming that these 198 auxiliary differentials are independent and randomly distributed, we can use (3) with $k = 30$ to estimate $a, b \approx 4$ that can be used in our collision attack on the Hamsi-256 compression function.

However, the auxiliary differentials in the set S are not independent nor randomly distributed. By doing a brute-force search we found a solution with $a = b = 6$ resulting in a collision attack on the Hamsi-256 compression function with complexity of about 2^{126} compression function evaluations. The six auxiliary differentials including the necessary conditions on the chaining and message bits are given in Table 1 and the affected output bits for these six auxiliary differentials are shown in Table 4 in the appendix. Note that the 10 conditions imposed by the six auxiliary differentials on the message bits can be fulfilled by solving a set of linear equations. This is due to the fact that the message expansion of Hamsi-256 is linear.

Table 1. The six auxiliary differentials used in our attack including all conditions on the chaining and message bits. Note that none of them affects the output bits 131, 200, 201, 202, 237, 238.

	i					
	1	2	3	4	5	6
column	1	2	38	39	110	111
difference	$\mathtt{a}_x$	$\mathtt{a}_x$	$\mathtt{a}_x$	$\mathtt{a}_x$	$\mathtt{5}_x$	$\mathtt{5}_x$
message bits	$\mathtt{2}_x$	$\mathtt{2}_x$	$\mathtt{3}_x$	$\mathtt{1}_x$	$\mathtt{2}_x, \mathtt{3}_x$	$\mathtt{2}_x, \mathtt{3}_x$
chaining bits	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{0}_x, \mathtt{3}_x$	$\mathtt{1}_x, \mathtt{2}_x$

5.1 Improving the Attack

To improve the attack described above we need to find a subset of size a of auxiliary differentials not affecting the same b output bits for larger values of a and b. Therefore, we need to find a set S of auxiliary differentials, where the number of unaffected output bits is larger than 30 on average. To find such auxiliary differentials we need to consider initial differences with more than only one active column at the input of the first round. This significantly increases the search space, but also the complexity to generate the set S. However, since we are interested in auxiliary differentials which are affecting only a few output bits, the search space and hence the complexity can be reduced by only considering initial differences leading to a sparse difference at the input of round 2. To be

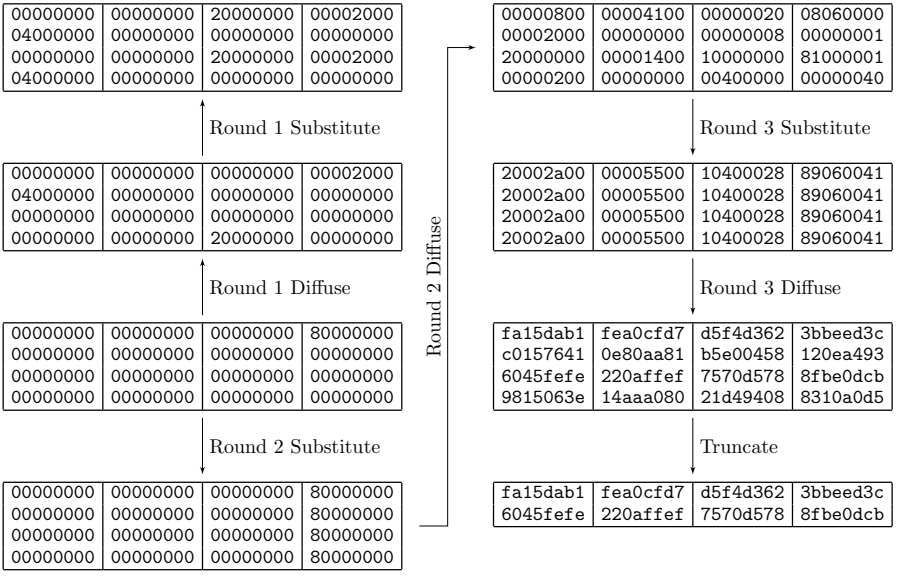

Fig. 4. Propagation of affected bits for the initial difference $\mathbf{1}_x$ inserted to column 96 at the input of round 2. Note that some conditions on the chaining and message bits in the column 5, 66 and 114 have to be fulfilled in order to guarantee that there will be only a single bit difference in the column 96 at the input of round 2.

more precise, we restrict ourselves to initial differences resulting in a single bit difference (one active column) at the input of round 2. One example of such an auxiliary differential with only a single bit difference in column 96 at the input of round 2 is given in Figure 4.

Considering only auxiliary differentials resulting in a singe bit difference at the input of round 2 has several advantages. First of all since only one column is active at the input of round 2, this results in sparse auxiliary differentials affecting only a few output bits. Second, due to the fact that the difference at the input of round 2 has Hamming weight 1, this results in at most 7 active columns at the input of round 1. Note that a difference with a higher Hamming weight would result in more active columns, complicating the attack. Moreover, a disadvantage is that the number of conditions on the chaining and message bits, that need to be fulfilled to guarantee that the auxiliary differential holds, would increase. For example the auxiliary differential given in Figure 4 needs 3 conditions on the chaining bits and 5 conditions on the message bits (see Table 2).

The best way to find the auxiliary differentials is to start with a single bit difference at the input of round 2 and then compute backward. Due to the properties of the linear layer, one will get 2 to 7 active columns at the input of

Table 2. Detailed information for the auxiliary differential used in the example given in Figure 4.

column	5	66	114
difference	$\mathtt{a}_x$	$\mathtt{5}_x$	$\mathtt{5}_x$
message bits	$\mathtt{0}_x$	$\mathtt{0}_x$	$\mathtt{0}_x, \mathtt{1}_x$
chaining bits	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{1}_x, \mathtt{2}_x$	$\mathtt{1}_x, \mathtt{2}_x$

round 1, resulting in 2-7 conditions on the chaining bits and 3-13 conditions on the message bits. We want to note that since the message input of Hamsi-256 is only 32 bits the increased number of message bit conditions might be the limiting factor for the attack. In Table 3 we list all the auxiliary differentials that result in a single bit difference at the input of round 2. As can be seen in the table the number of unaffected output bits for all the 192 auxiliary differentials is in the range between 100 and 130 bits with an average of 110 bits as shown in Figure 6 in the appendix.

Table 3. List of all possible auxiliary differentials with only a single bit difference at the input of round 2.

input round 2		output	input	message
difference	column	# unaffected bits	# active columns	# conditions
$\mathtt{1}_x$	$98, \ldots, 100$	102–103	2	3
	$96, 97; 101, \ldots, 127$	100–125	3	5
$\mathtt{2}_x$	$29, \ldots, 31$	111–116	3	5
	$25, \ldots, 28$	113–114	4	7
	$0, \ldots, 24$	107–129	7	13
$\mathtt{4}_x$	$32, \ldots, 95$	100–125	3	6
$\mathtt{8}_x$	$61, \ldots, 63; 93, \ldots, 95$	111–116	4	7
	$32, \ldots, 60; 64, \ldots, 62$	107–129	5	9

Assuming again that the 192 auxiliary differentials are independent and randomly distributed and the number of unaffected output bits is 110. Then, as above from (3) we would expect to find a solution with $a, b \approx 10$.

This is very close to our result, by using a brute-force search we found $a = 10$ auxiliary differentials not affecting the same $b = 9$ output bits resulting in an attack complexity of about $2^{124.1}$ compression function evaluations. The 10 auxiliary differentials and the list of the affected output bits for each auxiliary differential are given in Table 5 and Table 6 in the appendix.

However, the results of the attack described in the previous section would suggest that there might solutions for larger values of a and b than estimated by (3), since the auxiliary differentials are not independent nor random. Indeed, if we ignore the conditions imposed by the auxiliary differentials on the message bits, then we could find $a = 14$ auxiliary differentials not affecting the same

$b = 14$ output bits, however we were not able to find a confirming message input. The reason for this is that we need to fulfill on average about 7 conditions on the message bits per auxiliary differential, while the message input of Hamsi-256 is only 32 bits. Note that for the attack described in the previous section, we had in total only 10 message bit conditions for the 6 auxiliary differentials all together.

6 Conclusion

In this work, we have analyzed the Hamsi-256 compression function with respect to its collision resistance. By exploiting non-random properties of the compression function we could show a collision attack with a complexity of about $2^{124.1}$. The attack is an extension of the distinguishing attack of Çalik and Turan combined with the idea of neutral bits and auxiliary differentials originally used to speed up existing differential collision attacks. Even though the complexity of our attack is very high and close the the generic case it gives some new insights in the security of Hamsi-256.

Acknowledgments. This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. In addition, this work was supported by the Research Fund KU Leuven, OT/08/027.

References

1. Aumasson, J.P., Käsper, E., Knudsen, L.R., Matusiewicz, K., Ødegård, R.S., Peyrin, T., Schläffer, M.: Distinguishers for the Compression Function and Output Transformation of Hamsi-256. In: Steinfeld, R., Hawkes, P. (eds.) ACISP. LNCS, vol. 6168, pp. 87–103. Springer (2010)
2. Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A New Block Cipher Proposal. In: Vaudenay, S. (ed.) FSE. LNCS, vol. 1372, pp. 222–238. Springer (1998)
3. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M.K. (ed.) CRYPTO. LNCS, vol. 3152, pp. 290–305. Springer (2004)
4. Boura, C., Canteaut, A.: Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography. LNCS, vol. 6544, pp. 1–17. Springer (2010)
5. Çagdas Çalik, Turan, M.S.: Message Recovery and Pseudo-preimage Attacks on the Compression Function of Hamsi-256. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT. LNCS, vol. 6212, pp. 205–221. Springer (2010)
6. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO. LNCS, vol. 435, pp. 416–427. Springer (1989)
7. Dinur, I., Shamir, A.: An Improved Algebraic Attack on Hamsi-256. Cryptology ePrint Archive, Report 2010/602 (2010), `http://eprint.iacr.org/`
8. Dinur, I., Shamir, A.: An Improved Algebraic Attack on Hamsi-256. In: Joux, A. (ed.) FSE. LNCS, vol. 6733, pp. 88–106. Springer (2011)

9. Fuhr, T.: Finding Second Preimages of Short Messages for Hamsi-256. In: Abe, M. (ed.) ASIACRYPT. LNCS, vol. 6477, pp. 20–37. Springer (2010)
10. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO. LNCS, vol. 4622, pp. 244–263. Springer (2007)
11. Küçük, Ö.: The Hash Function Hamsi. Submission to NIST (updated) (2009)
12. Küçük, Ö.: Design and Analysis of Cryptographic Hash Functions. Ph.D. thesis, KU Leuven (April 2012)
13. Li, Y., Wang, A.: Using genetic algorithm to find near collisions for the compress function of Hamsi-256. In: BIC-TA. pp. 826–829. IEEE (2010)
14. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO. LNCS, vol. 435, pp. 428–446. Springer (1989)
15. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`
16. Nikolic, I.: Near Collisions for the Compression Function of Hamsi-256. CRYPTO rump session (2009)
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), `http://eprint.iacr.org/`
18. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. LNCS, vol. 3621, pp. 17–36. Springer (2005)
19. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 19–35. Springer (2005)

A Supporting Material

Table 4. Affected output bits for the 6 auxiliary differentials used in our collision attack on the Hamsi-256 compression function.

i	output	
	affected bits	# unaffected bits
1	7bbfff3f fe1dfaff febaefdf f7fdf7e7 cfbfdfcf 615fffff 2e1affef f7f1fd7b	53
2	bddfff9f fe0efd7f ff5d77ef fbfefbf3 e5dfefe7 a0afffff 970d7ff7 fbf8febd	56
3	3fbfefbf fbddfff9 ffe0efd7 fff5d77e cfbf8feb 6c7dfefe a21affff 7f70d7ff	58
4	9fdff7df fceebffc fff077eb 7ffaebbf e5dfc7f5 263eff7f d10d7fff bfb86bff	62
5	fbffff3f fe1dffff ffffffff fffff7e7 effffef e1ffffff 3e1fffef fff1fdfb	29
6	fdffff9f ff0effff ffffffff fffffbf3 e7ffffff f0ffffff 9f0ffff7 fff8fefd	30
	ffffffff ffffffff ffffffff ffffffff effffff ffffffff ff1fffff fff9ffff	6

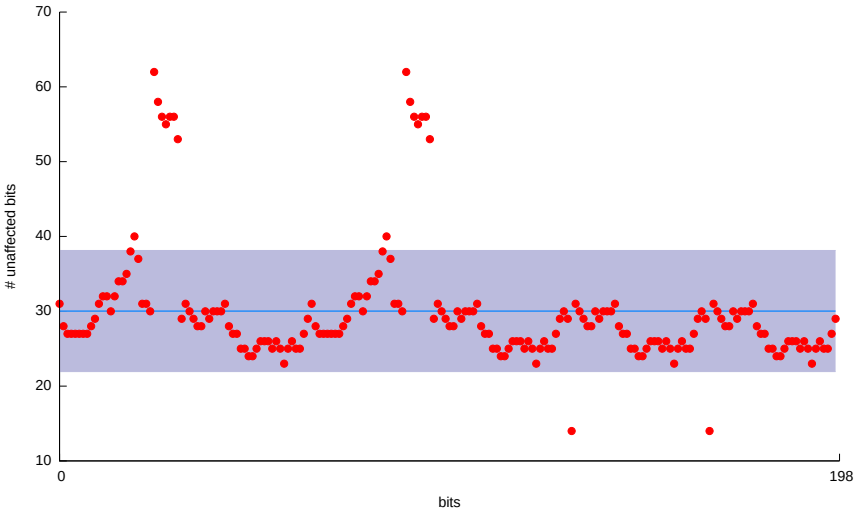

Fig. 5. Number of unaffected bits at the output of the compression function with an initial difference of Hamming weight 2. Note that this does not include auxiliary differentials where the number of unaffected output bits is smaller than 10, since they are not useful for our attack.

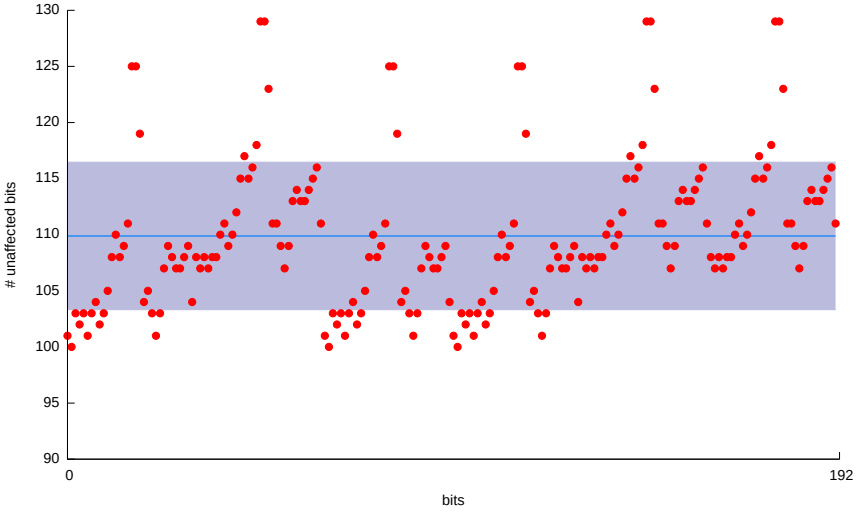

Fig. 6. Number of unaffected bits at the output of the compression function with a single bit difference, i.e. $1_x, 2_x, 4_x$ or 8_x, at the input of round 2.

Table 5. List of auxiliary differentials used in our collision attack on the Hamsi-256 compression function including all conditions on the message and chaining bits. Note that all conditions on the message bits can be fulfilled be setting the 32-bit message block $M_i = \{12, 51, aa, 05\}$ prior to the application of the message expansion.

1	column	3	4	64	112			
	difference	a_x	a_x	5_x	5_x			
	message bits	2_x	0_x	2_x	$0_x, 1_x$			
	chaining bits	$0_x, 3_x$	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$			
2	column	3	10	11	38	67	71	119
	difference	a_x	a_x	a_x	a_x	5_x	5_x	5_x
	message bits	2_x	3_x	3_x	1_x	1_x	1_x	$2_x, 3_x$
	chaining bits	$0_x, 3_x$	$1_x, 2_x$	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$	$0_x, 3_x$	$1_x, 2_x$
3	column	3	38	67				
	difference	a_x	a_x	5_x				
	message bits	2_x	1_x	1_x				
	chaining bits	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$				
4	column	3	64	112				
	difference	a_x	5_x	5_x				
	message bits	2_x	2_x	$0_x, 1_x$				
	chaining bits	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$				
5	column	4	39	68				
	difference	a_x	a_x	5_x				
	message bits	0_x	3_x	2_x				
	chaining bits	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$				
6	column	10	71	119				
	difference	a_x	5_x	5_x				
	message bits	3_x	1_x	$2_x, 3_x$				
	chaining bits	$1_x, 2_x$	$0_x, 3_x$	$1_x, 2_x$				
7	column	11	72	120				
	difference	a_x	5_x	5_x				
	message bits	3_x	2_x	$0_x, 1_x$				
	chaining bits	$0_x, 3_x$	$0_x, 3_x$	$1_x, 2_x$				
8	column	16	42	45	52	90		
	difference	a_x	a_x	a_x	a_x	5_x		
	message bits	0_x	3_x	0_x	2_x	$0_x, 1_x$		
	chaining bits	$1_x, 2_x$	$0_x, 3_x$	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$		
9	column	50	85	114				
	difference	a_x	5_x	5_x				
	message bits	2_x	1_x	2_x				
	chaining bits	$1_x, 2_x$	$0_x, 3_x$	$0_x, 3_x$				
10	column	63	67	89	92	105		
	difference	a_x	5_x	5_x	5_x	5_x		
	message bits	3_x	1_x	2_x	0_x	$2_x, 3_x$		
	chaining bits	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$	$1_x, 2_x$		

Table 6. Affected output bits for the 10 auxiliary differentials used in our improved collision attack on the Hamsi-256 compression function.

i	output	
	affected bits	# unaffected bits
1	8777dda7 1f02ab16 ffd419fa 5abe9a0c 7177c1b9 8008bfde f8415ffd 0e3e1aaf	114
2	4f0eefbb 2c3e8576 f5ffa833 18b57d34 7067ef83 bd18917f ebfe82bf 4a1c7c35	109
3	8316afa6 69e19cf7 d58fc0ae 7eb9f506 8143ab86 6a1c7d70 f7a3022f 6d7fd057	119
4	e8576ac7 fa833f5f 57d34d8b eefbb4f0 8117fbf9 e82bffbf d58355e1 7ef8372e	100
5	418b57d3 b4f08e7b 6ac7e857 3e5c3a83 50a1d5c3 352e3cb8 e9d18117 26bfe82b	125
6	8fc0aad5 bff5067e 16afa69b e1ddf769 a0122ff7 7e1057ff c3af86ab 4c7df06e	108
7	c7e0556a 5ffa833f 8b57d34d f0eebbb4 d00917fb bf082bff e1d7c355 263ef837	109
8	8316afa6 69e19cf7 c583c0ae 7eb9f506 81438b86 6a1c7d70 f7a3022f 6d7fd057	123
9	ea0cfd7f 5f4d362d bbeed3c3 a15d8b1f a0affeff 160d5787 fb60dcb9 245fefe6	102
10	ea0cfd7f 5f4d062d bbeed3c3 a15d8b0f a0affeff 160d5787 fb60dcb9 245fef46	107
	efffffff ffffbfff ffffffff ffffffff f1ffffff ff3fffff ffffdfff 7fffffff	9