A Distinguisher for the Compression Function of SIMD-512

Florian Mendel and Tomislav Nad

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria.
Tomislav.Nad@iaik.tugraz.at

Abstract. SIMD is one of the round 2 candidates of the public SHA-3 competition hosted by NIST. It was designed by Leurent *et al.*. In this paper, we present a distinguisher attack on the compression function of SIMD-512. By linearizing the compression function we construct a linear code. Using techniques from coding theory to search for low Hamming weight codewords, we can find differential characteristics with low Hamming weight (and hence high probability). In the attack the differences are introduced only in the IV. Such a characteristic is the base for our distinguisher, which can distinguish the compression function of SIMD-512 from random with a complexity of $5 \cdot 2^{425.28}$ compression function calls. Furthermore, we can distinguish the output transformation of SIMD-512 from random with a complexity of about $22 \cdot 2^{425.28}$ compression function calls. So far this is the first cryptanalytic result for the SIMD hash function.

Keywords: SHA-3 candidate, SIMD, cryptanalysis, distinguisher.

1 Introduction

Recently, the NIST hash function competition [13] has started. In this public competition to find an alternative hash function to replace the SHA-1 and SHA-2 hash functions, many new designs have been proposed. In November 2008, round one has started and in total 51 out of 64 submissions have been accepted. Recently, the 14 round 2 candidates were announced. SIMD, designed by Leurent *et al.* [9], is one of them. It is an iterative hash function based on the Merkle-Damgård design principle [6,12]. It is a wide-pipe design [10] producing a hash value up to 512 bits, denoted by SIMD-n, where n is the output length. For the remainder of this paper wherever we mention SIMD we refer to SIMD-512. The design of the compression function is similar to the MD4 family. Furthermore, there exist several proofs [5,11] for the mode of operation used by SIMD. The designers additionally provide bounds for a large class of differential attacks. Most of the security is based on the message expansion. In this paper, we present a distinguisher attack on the compression function of SIMD-512 with a complexity of $5 \cdot 2^{425.28}$ compression function calls. Including the output transformation we can distinguish the output of SIMD-512 from random with a complexity of about $22 \cdot 2^{425.28}$ compression function calls. The distinguisher is based on a differential characteristic with differences only in the IV. A characteristic with high success probability is found by using techniques from coding theory. By linearizing the compression function we define a linear code where each codeword represents a differential characteristic. Using an algorithm to find low Hamming weight codewords, we found characteristics which lead to the above attack complexity.

Even if we do not attack the whole hash function, we show unexpected non-random properties of the SIMD-512 compression function. However, our attack does not invalidate the security claims of the designers, since most of the security comes from the message expansion, but note that the non-randomness of the compression function of SIMD effects the applicability of the proofs for the mode of operation build upon it.

The structure of this paper is as follows. A short description of SIMD is given in Section 2. Section 3 gives an overview of the basic attack strategy. Section 4 shows in which way we linearized the compression function of SIMD. Followed by Section 5 containing the description of the techniques from coding theory to find good characteristics. Finally, the distinguisher for full SIMD is presented in Section 6.

2 Description of SIMD

SIMD is an iterative hash function that follows the Merkle-Damgård design. The main component of a Merkle-Damgård hash function is the compression function. In the case of SIMD-512 to compute the hash of a message M, it is first divided into k chunks of 1024 bits. By the use of a message expansion one block is expanded to 8192 bits. Then the compression function is used to compress the message chunks and the internal state. The padding rule to fill the last blocks is known as the Merkle-Damgård strengthening. The initial value of the internal state is called IV and is fixed in the specification of the hash function. The output of the hash function is given by computing a finalization function on the last internal state, which is a truncation for SIMD. The internal state of SIMD contains 32 32-bit words and is therefore twice as large as the output. SIMD consist of 4 rounds where each round consist of 8 steps. The feed-forward consists of four additional steps with the IV as message input. Since we apply a compression function attack independent from the message expansion, we omit the description of the message expansion. For a detailed description of the hash function we refer to [9].

2.1 SIMD Step Function

The core part of SIMD is the step function of the state update. Figure 1 illustrates the step function at step t. The state update consists of eight step functions in parallel. To make the step function dependent from each other,

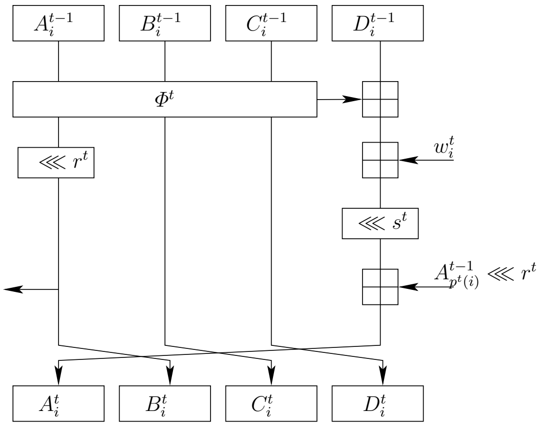

Fig. 1. Update function of SIMD at step t. $i = 0, \cdots, 7$.

$(A_{p^t(i)}^{t-1} \lll r^t)$ is included in a modular addition, where $p^t(i)$ is a permutation, which is different for each step.

Equation (1) is the formal definition of the step function, where $\boxplus$ denotes the addition modulo 2^{32}.

$$\begin{aligned} A_i^t &= (D_i^{t-1} \boxplus w_i^t \boxplus \Phi(A_i^{t-1}.B_i^{t-1}, C_i^{t-1})) \lll s^t \boxplus (A_{p^t(i)}^{t-1} \lll r^t) \\ B_i^t &= A_i^{t-1} \lll r^t \\ C_i^t &= B_i^{t-1} \\ D_i^t &= C_i^{t-1} \end{aligned} \tag{1}$$

The permutation p is separated in 4 different permutations:

$$p^0(x) = \begin{cases} x+1 \pmod 8, & \text{if } x = 0 \pmod 2 \\ x-1 \pmod 8, & \text{otherwise} \end{cases}$$

$$p^1(x) = \begin{cases} x+2 \pmod 8, & \text{if } x = 0 \pmod 4 \text{ or } x = 1 \pmod 4 \\ x-2 \pmod 8, & \text{otherwise} \end{cases}$$

$$p^2(x) = 7 - x \pmod 8$$
$$p^3(x) = x + 4 \pmod 8$$

The permutation used at step t is $p^{t \bmod 4}$. As mentioned before, the 32 steps of SIMD are divided into 4 rounds, each consisting of 8 steps. The boolean function Φ and the rotation constants (s and r) for a round are given in Table 1. In Table 2 the rotation constants for each round are given. The feed-forward consist of four steps using the same step function. Table 3 lists the used Boolean function and the rotation constants for the feed-forward.

Table 1. Φ and rotation constants for a round.

step	$\boldsymbol{\Phi}$	$\boldsymbol{r}$	$\boldsymbol{s}$
0	IF	π_0	π_1
1	IF	π_1	π_2
2	IF	π_2	π_3
3	IF	π_3	π_0
4	MAJ	π_0	π_1
5	MAJ	π_1	π_2
6	MAJ	π_2	π_3
7	MAJ	π_3	π_0

Table 2. Rotation constants for each round.

round	$\boldsymbol{\pi_0}$	$\boldsymbol{\pi_1}$	$\boldsymbol{\pi_2}$	$\boldsymbol{\pi_3}$
0	3	20	14	27
1	26	4	23	11
2	19	28	7	22
3	15	5	29	9

Table 3. Φ and rotation constants for the feed-forward of SIMD

step	$\boldsymbol{\Phi}$	$\boldsymbol{r}$	$\boldsymbol{s}$
0	IF	15	5
1	IF	5	29
2	IF	29	9
3	IF	9	15

3 The Basic Attack Strategy

In this section, we briefly describe the attack strategy to construct a distinguisher for the compression function. The attack can be summarized as follows:

1. Find a differential characteristic for the compression function of SIMD with differences in the IV, which holds with high probability.
2. Use message modification technique to increase the probability.

To find a good characteristic for the compression function, we use a linearized model of it. Finding a characteristic in a linear code is not difficult. Since the security of SIMD is heavily based on the message expansion, we concentrate on characteristics with differences only in the IV. The probability that the characteristic holds in the original compression function is related to the Hamming weight of the characteristic. In general, a differential characteristic with low Hamming weight has a higher probability than one with a high Hamming weight. Finding a characteristic with high probability (low Hamming weight) is related to finding a low weight word in linear codes. Therefore, we use the probabilistic algorithm from Canteaut and Chabaud [3] to find a good characteristic for the compression function of SIMD. It has been shown in the past, for instance the cryptanalysis of SHA-0 [4], SHA-1 [14] or EnRUPT [8] that this technique works well for finding differential characteristics with low Hamming weight. Furthermore, we can improve the probability of the characteristic using message modification, which was introduced by Wang *et al.* in [16].

4 Linearization of SIMD

Since we have only differences in the IV, we can omit the message expansion and assume that the message words have no differences. The step function (1) is the only part of SIMD which has to be linearized. The nonlinear parts of this function are the modular additions and the Boolean function Φ. In the attack, we replace all modular addition by XORs. The function Φ depends on the current step and is either the IF function or the MAJ function. To have a good approximation for those, we have to take a closer look on the differential behavior of them.

4.1 Differential Behavior of IF and MAJ

The differential behavior of IF and MAJ is already discussed in [7]. IF and MAJ have three inputs. Table 4 shows the differential propagation of the Boolean functions regarding XOR-differences.

Since we aim for a low weight characteristic, we replace the Boolean function Φ with the 0-function, *i.e.* we block each input difference in Φ, no matter if IF or MAJ is used. This has probability 1/2 in most cases. One can see that there is exactly one input difference for IF and one for MAJ where the output difference is always one. We discard characteristics with such properties, except in the feed-forward. There we manually correct the characteristic, resulting in a

Table 4. Differential Propagation of IF and MAJ

Δx	Δy	Δz	ΔIF	ΔMAJ
0	0	0	0	0
0	0	1	$x \oplus 1$	$x \oplus y$
0	1	0	x	$x \oplus z$
0	1	1	1	$y \oplus z \oplus 1$
1	0	0	$y \oplus z$	$y \oplus z$
1	0	1	$x \oplus y \oplus z$	$x \oplus z \oplus 1$
1	1	0	$x \oplus y \oplus z \oplus 1$	$x \oplus y \oplus 1$
1	1	1	$y \oplus z \oplus 1$	1

slightly higher Hamming weight. Furthermore, we use the non-linearity of the IF function in the feed-forward to decrease the Hamming weight significantly (see Section 6.2).

Finally, the linearized step function looks as follows:

$$\begin{aligned}
A_i^t &= (D_i^{t-1} \oplus w_i^t \oplus 0) \lll s^t \oplus (A_{p^t(i)}^{t-1} \lll r^t) \\
B_i^t &= A_i^{t-1} \lll r^t \\
C_i^t &= B_i^{t-1} \\
D_i^t &= C_i^{t-1}
\end{aligned} \tag{2}$$

Note that for the feed-forward w_i^t is equal to one word of the IV.

5 Finding Good Characteristics

As observed by Rijmen and Oswald [15], all differential characteristics for a linearized hash function can be seen as the codewords of a linear code. Our aim is to find good characteristics. Therefore, we have to include each part where differences could decrease the success probability. Let the vector

$$\Delta cv^t := (\Delta A_i^t | \Delta B_i^t | \Delta C_i^t | \Delta D_i^t), \tag{3}$$

for $i = 0, \cdots, 7$ and $cv^t \in \{0,1\}^{1024}$ be the concatenated difference of all chaining values (in bit representation) at step t. Then the vector

$$\Delta dc := (\Delta IV, \Delta cv^1, \cdots, \Delta cv^{36}),$$

where $\Delta dc \in \{0,1\}^{37 \cdot 1024}$, represents the differences in the IV, chaining values after each step and the output of the SIMD compression function, including the feed-forward. Δdc is one codeword of the linear code and therefore a differential characteristic. To construct the generator matrix for the linear code, we proceed as follows:

1. Compute Δdc_j with the input difference $\Delta IV_j = e_j$, where $e_j \in \{0,1\}^{1024}$ is the j-th unit vector.
2. Repeat the computation for $j = 1, \ldots, 1024$.

The resulting systematic generator matrix of the linear code for the linearized SIMD compression function is defined in the following way:

$$G_{1024 \times 37 \cdot 1024} := [I_{1024 \times 1024} | CV], \tag{4}$$

where CV is defined by

$$\begin{pmatrix} \Delta dc_1 \\ \vdots \\ \Delta dc_{1024} \end{pmatrix}.$$

5.1 Reducing the Code Length

Depending on the number of steps, the linear code can get large. If we take a closer look on the dependencies of each chaining value, one can see that only the A_i's are updated at each step and the other values only depend on them. Therefore, we can reduce the code size by only considering the A_i's at each step function. The definition of Δcv^t in Equation (3) changes to

$$\Delta cv^t := (\Delta A_i^t), \tag{5}$$

Following the same procedure above, the resulting generator matrix is much smaller, namely

$$G_{1024 \times 10240} := [I_{1024 \times 1024} | CV]. \tag{6}$$

Therefore, the performance of the search for low Hamming weight codewords is increased.

5.2 Low Weight Search

We implemented the probabilistic algorithm from Canteaut and Chabaud [3] to search for codewords with low Hamming weight and applied some optimizations to speed up the search. This iterative algorithm basically looks for small Hamming weight codewords in a smaller code. Such a codeword is considered as a good candidate for a low Hamming weight codeword for the whole code. Considering a systematic generator matrix like (6) the algorithm randomly selects σ columns of it and split the selection in two submatrices of equal size. By computing all linear combination of p rows (usually 2 or 3) for each submatrix and storing their weight, the algorithm searches for a collision of both weights which allow to search for codewords of $2p$. Then two randomly selected columns are interchanged, followed by one Gaussian elimination step. This procedure is repeated until a sufficiently small Hamming weight was found. Additionally, we check for each codeword if each difference at the input of the Boolean function can be blocked. If it is not possible we discard the codeword. We omit this check in the feed-forward (see Section 4.1).

In the case of the codes originating from the linearized SIMD compression function we found several low weight codewords in less than an hour on a PC.

5.3 Estimating the Probability for a Characteristic

To compute the probability of the found differential characteristic, we have to consider the differences entering the Boolean function Φ and the modular additions.

The Boolean Function Φ. The probability for blocking a difference in one bit at the input of Φ is 1/2 or 0 for some cases, but then the characteristic is discarded (see Section 4.1). Hence, the total probability is determined by the sum of all differences at the input. Note, that differences at the same bit positions are counted only once. The overall probability for step t is defined by 2^{-x}, where x is given by

$$\sum_{i=0}^{7} hw(\Delta A_i^{t-1} \vee \Delta B_i^{t-1} \vee \Delta C_i^{t-1})$$

and $hw(\cdot)$ is the bit-wise Hamming weight of a 32-bit word.

The Modular Additions. Consider the additions (7) from the step function (1).

$$(\Delta D_i^{t-1} \boxplus \Delta w_i^t) \lll s^t \boxplus (\Delta A_{p^t(i)}^{t-1} \lll r^t) \tag{7}$$

We could consider each modular addition separately and prevent a carry for each bit difference, but this would result in a rather conservative approximation. Therefore, we want to give a more detailed analysis. By allowing carries in the first addition, we can compensate them at the second addition. However, this is not that easy, because of the rotation after the first modular addition.

First we take a look at the following addition:

$$\Delta D_i^{t-1} \boxplus \Delta w_i^t.$$

If we have a difference at the same bit position, we can cancel them out with probability 1/2. The overall probability to cancel out such differences for step t is 2^{-y}, where y is defined by

$$\sum_{i=0}^{7} hw(\Delta D_i^{t-1} \wedge \Delta w_i^t).$$

Note that $\Delta w_i^t \neq 0$ only for the feed-forward. If there is only a difference in one input of the modular addition (bit-wise), we allow carries. However, we do not want that the carry expansion is destroyed, due to the rotation to left by s^t bits, since we cannot compensate this in the second addition. To take care of this problem we have to consider two cases.

Let be l_j the bit position of the j-th difference in ΔD_i^{t-1} before the rotation, l'_j after the rotation and $d_{\mathtt{MSB}}(l_j)$ ($d_{\mathtt{MSB}}(l'_j)$) the distance of l_j (l'_j) to the most significant bit (MSB). The first case is $d_{\mathtt{MSB}}(l_j) < s^t$, *i.e.* the difference is rotated over the MSB. Therefore, we have to ensure that the carry expands at most to the MSB from the position of the difference *before* the rotation. The probability for that is

$$1 - 2^{-d_{\texttt{MSB}}(l_j)}.$$

The second case considers $d_{\texttt{MSB}}(l_j) \geq s^t$, *i.e.* the difference is not rotated over the MSB. In this case we have to ensure that the carry expands at most to the MSB from the position of the difference *after* the rotation. The probability for that is

$$1 - 2^{-d_{\texttt{MSB}}(l'_j)}.$$

This differentiation has to be done for each difference in ΔD_i^{t-1}. The overall probability is given by the product of all single probabilities.

In the last modular addition

$$(\Delta D_i^{t-1} \lll s^t) \boxplus (\Delta A_{p^t(i)}^{t-1} \lll r^t)$$

we first cancel out differences at the same bit positions of both variables with probability 1/2 for each such difference. In the last step we compensate the carries from the first addition with the same probability. Finally, the overall success probability for the second modular addition is 2^{-z}, where z is defined as follows:

$$\sum_{i=0}^{7} hw(\Delta D_i^{t-1} \lll s^t \vee \Delta A_{p^t(i)}^{t-1} \lll r^t).$$

Note, that we ignore differences in the MSB for these calculations, which results in a small improvement.

Message Modification. To improve the success probability of the differential characteristic we use message modification. We have the freedom to choosing the actual values of the IV and the message words. Regarding the message words, we assume that we can increase the success probability in the first 4 steps to 1. Since one message block in SIMD has 1024 bit and is expanded to 8192, we can at least choose the first 32 expanded message words w, but not completely arbitrary. The message modification for the first 4 steps results in a significant improvement of the overall success probability, since this probability is low in these steps. However, the message expansion needs to be studied in more detail to get a good view on the security of SIMD. It might be possible to improve the attack by using more sophisticated message modification techniques.

6 Distinguisher for Full SIMD

In this section, we present a distinguisher for the full (32 steps and feed-forward) compression function of SIMD. It is based on the differential multicollision distinguisher introduced by Biryukov *et al.* [1] and high probability differential characteristics for the compression function of SIMD. This characteristic was found by using the techniques described in the previous section. Before describing the differential characteristic in detail, we first have to discuss the setting we use to show non-randomness in the compression function of SIMD.

6.1 Differential q-multicollision

The notion of differential q-multicollision was introduced by Biryukov *et al.* in the cryptanalysis of AES-256. They show that differential q-multicollision can be found for AES-256 with a complexity of $q \cdot 2^{67}$, while for an ideal cipher an adversary needs at least

$$O(q \cdot 2^{\frac{q-2}{q+2} \cdot n}) \tag{8}$$

time. Note that in [1] the attack is described for a block cipher. However, it can be easily adapted for a random function. Below we repeat the basic definition and lemma, we need for the distinguishing attack for the compression function of SIMD.

Definition 1. *A set of two differences and q pairs*

$$\{\Delta IV, \Delta M; (IV_1, M_1), (IV_2, M_2), \cdots, (IV_q, M_q)\}$$

is called a differential q-multicollision for $f_{IV}(\cdot)$ if

$$\begin{aligned} f_{IV_1}(M_1) \oplus f_{IV_1 \oplus \Delta IV}(M_1 \oplus \Delta M) &= f_{IV_2}(M_2) \oplus f_{IV_2 \oplus \Delta IV}(M_2 \oplus \Delta M) \\ &= \cdots = f_{IV_q}(M_q) \oplus f_{IV_q \oplus \Delta IV}(M_q \oplus \Delta M). \end{aligned}$$

In the case of SIMD, f is the compression function and ΔM is equal 0.

Lemma 1. *To construct a differential q-multicollision for an ideal function with an n-bit output an adversary needs at least $O(q \cdot 2^{\frac{q-2}{q+2} \cdot n})$ queries on the average.*

The proof for Lemma 1 works similar as in [1] for an ideal cipher.

In this section, we show how to find a differential q-multicollision for the SIMD compression function with a complexity of about $q \cdot 2^{425.28}$ instead of the expected

$$q \cdot 2^{\frac{q-2}{q+2} \cdot 1024}.$$

This is described in detail in the subsequent sections.

6.2 The Differential Characteristic

We have found several characteristics with low Hamming weight. The best ones have a weight of 504 in all chaining variables. We can further reduce the weight by using the non-linearity of the IF function in the feed-forward. If we do not block all input differences in the Boolean function, we can cancel out additional differences, which results in a lower Hamming weight for the subsequent steps. Thus, the overall success probability of the characteristic is increased. In that way we can improve the characteristics to a weight of 486. By a detailed analysis (see Section 5.3) we determine the success probability of the characteristics with $\approx 2^{-507.34}$ without message modification. If we use additionally message modification as described in Section 5.3, we increase the probability to $\approx 2^{-425.28}$.

Table 8 presents one of the differential characteristics with weight 486. Due to space restriction we do not show the complete characteristic but the differences in the IV, which is enough to reconstruct the whole differential path. In Appendix A the characteristic in the steps of the feed-forward, including the above modifications, is given.

Table 5. Differences in the IV.

i	A_i^0	B_i^0	C_i^0	D_i^0
0	00000000	00000000	00000000	00000000
1	00000000	00000000	00000000	00000000
2	00000000	00000000	00000000	00000000
3	00000000	104804a0	00000000	00000000
4	00000000	00000000	050e0010	00000000
5	00000000	00000000	00000000	00000000
6	00000000	00000000	00000000	68801201
7	04004400	00000000	00000000	00000000

Table 6 splits the probability estimation into rounds and steps (for readability the probabilities are given in $\log_2$).

Table 6. Probabilities in $\log_2$ for each round and step.

round \ **step**	0	1	2	3	4	5	6	7
0	−23.85	−23.03	−19.09	−16.19	−15.12	−12.09	−9	−8.03
1	−7.09	−5	−4	−4	−3	−2	−2	−2
2	−1	−1	−1	−2	−3	−4	−4	−3
3	−4.19	−6	−9	−12	−16	−19.42	−19	−23.30
feed-forward	−31.05	−46.09	−69.46	−77.34				

The characteristic in Table 8 leads to a guaranteed difference in one bit at the output of Φ in the third step of the feed-forward. By correcting this manually, the success probability is slightly decreased, which is already included in the overall probability.

6.3 The Complexity of the Attack

The differential characteristic described in the previous section can be used to construct a distinguisher for the compression function of SIMD. It is easy to see that by using the differential characteristic q times one can find a differential q-multicollision with a complexity of about $q \cdot 2^{507.34}$ compression function evaluations. Furthermore, by using message modification (see Section 5.3) in the first 4 steps the complexity of the attack can be significantly reduced, resulting in a complexity of about $q \cdot 2^{425.28}$. Note that the generic attack has a complexity of about

$$q \cdot 2^{\frac{q-2}{q+2} \cdot 1024}$$

compression function evaluations. Hence, one can distinguish the compression function of SIMD from a random function with a complexity of about $q \cdot 2^{507.34}$ and $q \cdot 2^{425.28}$ for $q = 6$ and $q = 5$, respectively.

In a similar way as we can distinguish the compression function of SIMD from random, we can also distinguish the output transformation (last iteration of SIMD) from random. While the complexity for constructing a differential q-multicollision for the output transformation using the differential characteristic described in the previous section is the same as before, the complexity of the generic attack has changed, since the output is only 512 instead of 1024 bits in the last iteration due to the truncation at the end. Hence, the complexity of the generic attack is

$$q \cdot 2^{\frac{q-2}{q+2} \cdot 512}.$$

However, by setting $q = 438$ and $q = 22$ for the case with message modification in the first 4 rounds, we can distinguish the output transformation of SIMD from random with a complexity of about $438 \cdot 2^{507.34}$ and $22 \cdot 2^{425.28}$, respectively. Table 7 provides a summary of the complexities for our distinguisher and the generic complexities.

Table 7. Summary of the attack complexities.

	compression function		output transformation	
message modification	**generic**	**our attack**	**generic**	**our attack**
no	$6 \cdot 2^{\frac{4}{8} \cdot 1024}$	$6 \cdot 2^{507.34}$	$438 \cdot 2^{\frac{436}{440} \cdot 512}$	$438 \cdot 2^{507.34}$
yes	$5 \cdot 2^{\frac{3}{7} \cdot 1024}$	$5 \cdot 2^{425.28}$	$22 \cdot 2^{\frac{20}{24} \cdot 512}$	$22 \cdot 2^{425.28}$

7 Conclusions

In this paper, we presented a distinguishing attack on the compression function of SIMD-512. We used techniques from coding theory to search for differential characteristics with low Hamming weight. We have found several characteristics with weight 486. Our attack strategy for the distinguisher is similar to the multicollision distinguisher introduced by Biryukov *et al.* [1]. By using the characteristic with the highest success probability, we are able to construct a distinguisher, which complexity is below the generic bound in [1], even with a still conservative probability estimation. We are able to distinguish the compression function from random with a complexity of $5 \cdot 2^{425.28}$ compression function calls. Including the output transformation the complexities are still below the generic bound, *i.e.* we can distinguish the output transformation of SIMD from random with a complexity of about $22 \cdot 2^{425.28}$ compression function calls.

Even if we do not attack the whole hash function, we show unexpected properties for the SIMD-512 compression function. However, our attack does not invalidate the security claims of the designers, since most of the security comes from the message expansion, but note that the non-randomness of the compression function of SIMD effect the applicability of the proofs for the mode of operation build upon it.

This is the first external cryptanalysis of the SIMD hash function. However, the desigerns have tweaked the design to avoid this attack.

Acknowledgements

The authors wish to thank Gaëtan Leurent for validating our attack, Christian Rechberger, Vincent Rijmen and the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II) and by the Austrian Science Fund (FWF), project P19863.

References

1. Alex Biryukov, D.K., Nikolić, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) Crypto. Lecture Notes in Computer Science, vol. 5677, pp. 231–249. Springer (2009)
2. Brassard, G. (ed.): Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, Lecture Notes in Computer Science, vol. 435. Springer (1990)
3. Canteaut, A., Chabaud, F.: A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Transactions on Information Theory 44(1), 367–378 (1998)
4. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1462, pp. 56–71. Springer (1998)
5. Chang, D., Nandi, M.: Improved Indifferentiability Security Analysis of chopMD Hash Function. In: Nyberg, K. (ed.) FSE. Lecture Notes in Computer Science, vol. 5086, pp. 429–443. Springer (2008)
6. Damgård, I.: A Design Principle for Hash Functions. In: Brassard [2], pp. 416–427
7. Daum, M.: Cryptanalysis of Hash Functions of the MD4-Family. Ph.D. thesis, Ruhr-Universität Bochum (May 2005), `http://www.cits.rub.de/imperia/md/content/magnus/dissmd4.pdf`

8. Indesteege, S., Preneel, B.: Practical Collisions for EnRUPT. In: Dunkelman, O. (ed.) FSE. Lecture Notes in Computer Science, vol. 5665, pp. 246–259. Springer (2009)
9. Leurent, G., Bouillaguet, C., Fouque, P.A.: SIMD Is a Message Digest. Submission to NIST (2008), `http://www.di.ens.fr/~leurent/files/SIMD.pdf`
10. Lucks, S.: A Failure-Friendly Design Principle for Hash Functions. In: Roy, B.K. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 3788, pp. 474–494. Springer (2005)
11. Maurer, U.M., Tessaro, S.: Domain Extension of Public Random Functions: Beyond the Birthday Barrier. In: Menezes, A. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 4622, pp. 187–204. Springer (2007)
12. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [2], pp. 428–446
13. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register Notice (November 2007), available online at: `http://csrc.nist.gov`
14. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) IMA Int. Conf. Lecture Notes in Computer Science, vol. 3796, pp. 78–95. Springer (2005)
15. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 3376, pp. 58–71. Springer (2005)
16. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer (2005)

A Differential characteristic for the 4 steps in the feed-forward

Table 8. Differences in the chaining values in the feed-forward.

(t, i)	A_i^t	B_i^t	C_i^t	D_i^t
$(33, 0)$	00000000	00000000	00000000	00000000
$(33, 1)$	00000000	00000000	00000000	00000000
$(33, 2)$	00000000	00000000	00000000	00000000
$(33, 3)$	00000000	00000000	83801001	00000000
$(33, 4)$	00000000	00000000	00000000	0000c008
$(33, 5)$	00000000	00000000	00000000	00000000
$(33, 6)$	84d0c901	00000000	00000000	00000000
$(33, 7)$	80088000	8410c1c0	00000000	00000000
$(34, 0)$	00000000	00000000	00000000	00000000
$(34, 1)$	00000000	00000000	00000000	00000000
$(34, 2)$	00000000	00000000	00000000	00000000
$(34, 3)$	02090094	00000000	00000000	83801001
$(34, 4)$	9a193831	00000000	00000000	00000000
$(34, 5)$	01100010	00000000	00000000	00000000
$(34, 6)$	00000000	9a192030	00000000	00000000
$(34, 7)$	00000000	01100010	8410c1c0	00000000
$(35, 0)$	00000000	00000000	00000000	00000000
$(35, 1)$	00000000	00000000	00000000	00000000
$(35, 2)$	00220002	00000000	00000000	00000000
$(35, 3)$	21620401	80412012	00000000	00000000
$(35, 4)$	8c010008	33432706	00000000	00000000
$(35, 5)$	00000000	00220002	00000000	00000000
$(35, 6)$	00000000	00000000	9a192030	00000000
$(35, 7)$	20000000	00000000	01100010	8410c1c0
$(36, 0)$	02001118	00000000	00000000	00000000
$(36, 1)$	00000000	00000000	00000000	00000000
$(36, 2)$	00000000	44000400	00000000	00000000
$(36, 3)$	00000040	c4080242	80412012	00000000
$(36, 4)$	00000000	02001118	33432706	00000000
$(36, 5)$	00000000	00000000	00220002	00000000
$(36, 6)$	4d00b040	00000000	00000000	9a192030
$(36, 7)$	a4e04042	00000040	00000000	01100010