# Boomerang Distinguisher for the SIMD-512 Compression Function

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.** In this paper, we present a distinguisher for the permutation of SIMD-512 with complexity $2^{226.52}$. We extend the attack to a distinguisher for the compression function with complexity $2^{200.6}$. The attack is based on the application of the boomerang attack for hash functions. Starting from the middle of the compression function we use techniques from coding theory to search for two differential characteristics, one for the backward direction and one for the forward direction to construct a second-order differential. Both characteristics hold with high probability. The direct application of the second-order differential leads to a distinguisher for the permutation. Based on this differential we extend the attack to distinguisher for the compression function.

**Keywords:** SHA-3, SIMD, cryptanalysis, higher-order differentials, hash function, distinguisher

## 1 Introduction

Recently, the NIST hash function competition [21] 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. In December 2009 the 14 round 2 candidates and in December 2010 the final five were announced. During the competition distinguishing attacks on hash functions and their building blocks are getting more attention. In such attacks an adversary utilizes specific properties of a hash function to define a distinguishing property such that one can distinguish the output of a hash function from a random function. Usually, the existence of such properties is not intended by the designers. However, as shown in [4] for wide-pipe designs the impact of distinguishers is limited.

In this paper, we present a distinguisher for the compression function of SIMD-512. SIMD, designed by Leurent *et al.* [13], was submitted to the NIST competition and was one of the second round candidates. It is an iterative hash function based on the Merkle-Damgård design principle [5,18]. It is a wide-pipe design [14] producing a hash value up to 512 bits, denoted by SIMD-$n$, where $n$ is the output length. The design of the compression function is similar to the

D.J. Bernstein and S. Chatterjee (Eds.): INDOCRYPT 2011, LNCS 7107, pp. 255–270, 2011.
The original publication is available at http://www.springerlink.com/content/b620800886557568/

MD4 family. For the remainder of this paper wherever we mention SIMD we refer to SIMD-512.

We will show how one can use the boomerang attack on a hash function to construct a distinguisher with high probability. The first result is a distinguishing attack for the full permutation of SIMD-512 with complexity $\approx 2^{226.52}$. Next we show how this distinguisher can be extended to the full compression function of SIMD-512. with complexity $\approx 2^{200.6}$. The strategy to construct such second order differentials is based on the recently proposed cryptanalysis of reduced SHA-2 [12] and Blake [3].

The structure of this paper is as follows. In Section 2, we recall the basic definitions needed for the attack and give an overview how higher-order differentials can be used to attack hash functions. A short description of SIMD is given in Section 3. Section 4 presents the application on the permutation of SIMD-512. In Section 5, we show how the attack can be extended to the compression function of SIMD-512. Finally, we discuss the results in Section 6.

### 1.1 Related Work

The amount of available cryptanalysis of SIMD is low compared to other candidates. Mendel and Nad presented the first attack on the full SIMD-512 compression function [15]. They used techniques from coding theory to find a differential characteristic that holds with probability $2^{-507}$. Based on this characteristic and the differential multicollision distinguisher introduced by Biryukov *et al.* [1] they constructed a distinguishing attack for the SIMD-512 compression function. Using IV/message modification the attack complexity was reduced to $2^{427}$. The differential path used some unwanted properties in the permutation of SIMD. Therefore, the designers tweaked the hash function by changing the permutations and round constants of SIMD to prevent the attack.

A round reduced version of tweaked SIMD was attacked by Nikolić *et al.* [8]. They presented distinguishers for the compression function of SIMD-512 reduced to 24 round with a linearized message expansion and SIMD-512 reduced to 12 rounds with unmodified message expansion. Both are based on rotational properties of the compression function. The success probabilities for the distinguishers are $2^{-497}$ and $2^{-236}$, respectively.

Later Yu and Wang [27] presented a free-start near-collision attack for SIMD-256 reduced to 20 rounds and for SIMD-512 reduced to 24 rounds. The attack complexities are $2^{107}$ and $2^{208}$, respectively. Furthermore, they showed a distinguisher for the full compression function with complexity $2^{398}$.

Finally, the designers [4] published a free-start distinguisher for the compression function exploiting the existence of symmetric states. Furthermore, they showed that distinguishers without differences in the message have only a minimal impact on the security of the hash function.

Higher-order differentials have been introduced by Lai in [11] and first applied to block ciphers by Knudsen in [10]. The application to stream ciphers was proposed by Dinur and Shamir in [6] and Vielhaber in [23].

Recently, Lamberger and Mendel [12] showed how higher-order differentials can be used to attack SHA-256 and presented a distinguisher for 46 The attack stands between the *boomerang attack* and the *inside-out* attack which were both introduced by Wagner in the cryptanalysis of block ciphers [24]. A previous application of the boomerang attack to hash functions is due to Joux and Peyrin [9], who used the boomerang attack as a neutral bits tool to speed-up existing collision attacks. Another similar attack strategy for hash functions is the *rebound attack* introduced by Mendel *et al.* [17] and its extensions [7,16]. Furthermore, Biryukov et al. [3] presented a boomerang attack on the SHA-3 finalist Blake resulting in a distinguisher for 7 rounds of the Blake-32 compression function with a complexity of $2^{232}$.

**Notation.** For the remainder of this paper we use the notation presented in Table 1.

**Table 1.** Notation

| notation | description |
|---|---|
| $\neg X$ | inversion of X |
| $X \oplus Y$ | bit-wise XOR of X and Y |
| $X + Y$ | modular addition of X and Y |
| $X \lll n$ | bit-rotation of $X$ by $n$ positions to the left |
| $X \ggg n$ | bit-rotation of $X$ by $n$ positions to the right |
| $X \ll n$ | bit-shift of $X$ by $n$ positions to the left |
| $X \gg n$ | bit-shift of $X$ by $n$ positions to the right |

## 2 Higher-order Differentials and Hash Function

In order to find a distinguishing property we construct a second order differential collision for the compression function. In this section we recall the basic definitions and give a high level description of the attack strategy.

While a standard differential attack exploits the propagation of the difference between a pair of inputs to the corresponding outputs, a higher-order differential attack exploits the propagation of the difference between differences. Higher-order differential cryptanalysis was introduced by Lai in [11] and subsequently applied to block ciphers by Knudsen in [10]. We recall the basic definitions that we will need in the subsequent sections.

**Definition 1.** *Let $(S, +)$ and $(T, +)$ be Abelian groups. For a function $F : S \to T$, the derivative at a point $a \in S$ is defined as*

$$\Delta_a F(x) = F(x + a) - F(x)\,. \tag{1}$$

*The i-th derivative of $F$ at $(a_1, a_2, \ldots, a_i)$ is then recursively defined as*

$$\Delta^{(i)}_{a_1,\ldots,a_i} F(x) = \Delta_{a_i}(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} F(x)) \,. \tag{2}$$

When applying differential cryptanalysis to a hash function, a collision for the hash function corresponds to a pair of inputs with output difference zero. Similarly, when using higher-order differentials we define a higher-order differential collision for a function $F$ as follows.

**Definition 2.** *An i-th order differential collision for a function $F$ is an $i$-tuple $(a_1, a_2, \ldots, a_i)$ together with a value $x_0$ such that*

$$\Delta^{(i)}_{a_1,\ldots,a_i} F(x_0) = 0 \,. \tag{3}$$

Note that the common definition of a collision for hash functions corresponds to a higher-order differential collision of order $i = 1$.

From (3) we see that we can freely choose $i + 1$ of the input parameters, i.e. $x_0$ and $a_1, \ldots, a_i$, which then fix the remaining input. Hence, the expected number of solutions to (3) is one after choosing $2^{n/(i+1)}$ values for the inputs and the query complexity is:

$$\approx 2^{n/(i+1)} \tag{4}$$

In the following, we will only consider the case $i = 2$ for which the query complexity of the attack is $2^{n/3}$.

In order to construct a second-order differential collision for the function $F$, we use a strategy recently proposed in cryptanalysis of reduced SHA-2 in [12]. The idea of the attack is quite simple. Assume we are given two differentials for $F_0$ and $F_1$ with $F = F_1 \circ F_0$, where one holds in the forward direction and one in the backward direction. To be more precise, we have

$$F_0^{-1}(y + \beta) - F_0^{-1}(y) = \alpha$$

and

$$F_1(y + \gamma) - F_1(y) = \delta$$

where the differential in $F_0^{-1}$ holds with probability $p_0$ and in $F_1$ holds with probability $p_1$. Using these two differentials, we can now construct a second order differential collision for $F$. This can be summarized as follows (see also Figure 1).

1. Choose a random value for $X$ and compute $X^* = X + \beta$, $Y = X + \gamma$, and $Y^* = X^* + \gamma$.
2. Compute backward from $X, X^*, Y, Y^*$ using $F_0^{-1}$ to obtain $P, P^*, Q, Q^*$.
3. Compute forward from $X, X^*, Y, Y^*$ using $F_1$ to obtain $R, R^*, S, S^*$.
4. Check if $P^* - P = Q^* - Q$ and $S - R = S^* - R^*$ is fulfilled.

Since

$$P^* - P = Q^* - Q = \alpha, \;\; \text{resp.} \;\; S - R = S^* - R^* = \delta, \tag{5}$$

will hold with probability at least $p_0^2$ in the backward direction, resp. $p_1^2$ in the forward direction and assuming that the differentials are independent the attack succeeds with a probability of $p_0^2 \cdot p_1^2$. Hence, the expected number of solutions to (5) is 1, if we repeat the attack about $1/(p_0^2 \cdot p_1^2)$ times.

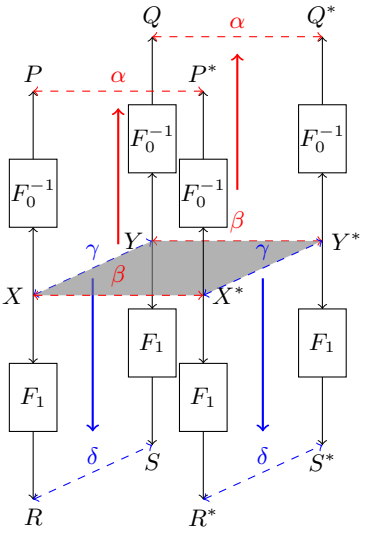


**Fig. 1.** Schematic view of the attack.

## 3 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 chaining value as message input. Since we inject differences only in the state variables and not in the message, our attack is independent from the message expansion and works for any given message. Therefore, we omit the description of the message expansion. For a detailed description of the hash function we refer to [13].

### 3.1 SIMD Step Function

The core part of SIMD is the step function of the state update. Figure 2 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,

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

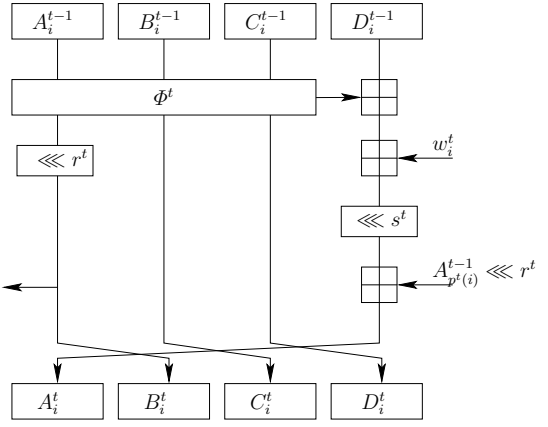


**Fig. 2.** Update function of SIMD at step $t$. $i = 0, \cdots, 7$.

Equation (6) is the formal definition of the step function, where '+' denotes the addition modulo $2^{32}$.

$$\begin{aligned} A_i^t &= (D_i^{t-1} + w_i^t + \Phi(A_i^{t-1}.B_i^{t-1}, C_i^{t-1})) \lll s^t + (A_{p^t(i)}^{t-1} \lll r^t) \\ B_i^t &= A_i^{t-1} \lll r^t \\ C_i^t &= B_i^{t-1} \\ D_i^t &= C_i^{t-1} \end{aligned} \tag{6}$$

The permutation $p$ for SIMD-512 is given by:

$$\begin{aligned} p^0(x) &= x \oplus 1 \\ p^1(x) &= x \oplus 6 \\ p^2(x) &= x \oplus 2 \\ p^3(x) &= x \oplus 3 \\ p^4(x) &= x \oplus 5 \\ p^5(x) &= x \oplus 7 \\ p^6(x) &= x \oplus 4 \end{aligned}$$

The permutation used at step $t$ is $p^{t \mod 7}$. As mentioned before, the 32 steps of SIMD are divided into 4 rounds, each consisting of 8 steps. The boolean function $\Phi$ and the rotation constants ($s$ and $r$) for a round are given in Table 2. The

**Table 2.** $\Phi$ and rotation constants for a round.

| step | $\Phi$ | $r$ | $s$ |
|---|---|---|---|
| 0 | IF | $\pi_0$ | $\pi_1$ |
| 1 | IF | $\pi_1$ | $\pi_2$ |
| 2 | IF | $\pi_2$ | $\pi_3$ |
| 3 | IF | $\pi_3$ | $\pi_0$ |
| 4 | MAJ | $\pi_0$ | $\pi_1$ |
| 5 | MAJ | $\pi_1$ | $\pi_2$ |
| 6 | MAJ | $\pi_2$ | $\pi_3$ |
| 7 | MAJ | $\pi_3$ | $\pi_0$ |

Boolean functions IF and MAJ are defined as follows:

$$f_{IF}(x, y, z) = (x \wedge y)|(\neg x \wedge z)$$
$$f_{IF}(x, y, z) = (x \wedge y)|(x \wedge z)|(y \wedge z).$$

In Table 3 the rotation constants for each round are given. The feed-forward

**Table 3.** Rotation constants for each round.

| round | $\pi_0$ | $\pi_1$ | $\pi_2$ | $\pi_3$ |
|---|---|---|---|---|
| 0 | 3 | 23 | 17 | 27 |
| 1 | 28 | 19 | 22 | 7 |
| 2 | 29 | 9 | 15 | 5 |
| 3 | 4 | 13 | 10 | 25 |

consist of four steps using the same step function. Table 4 lists the used Boolean function and the rotation constants for the feed-forward. In the feed-forward the

**Table 4.** $\Phi$ and rotation constants for the feed-forward of SIMD

| step | $\Phi$ | $r$ | $s$ |
|---|---|---|---|
| 0 | IF | 4 | 13 |
| 1 | IF | 13 | 10 |
| 2 | IF | 10 | 25 |
| 3 | IF | 25 | 4 |

chaining value is used as message input. In the first step $A_i^0$, in the second step $B_i^0$, in the third $C_i^0$ and in the fourth $D_i^0$ for $i = 0, \ldots, 7$ are used.

# 4 Application on SIMD-512

In this section we will show how to construct a second order differential collision which suits as a distinguishing property for the full permutation (compression function without feed-forward) of SIMD-512. For the permutation of SIMD-512 the attack strategy can be directly applied using a good differential characteristic for the forward and backward direction. We show how we construct such differential characteristic and compute the complexities. In contrast to the attack on SHA-256 [12], where the second-order collision for the internal block cipher immediately transfers to the compression function, we need to overcome the feed-forward which performs 4 additional steps with the chaining value as message input. In Section 5 we show how the attack can be extended to the compression function using a weaker attack scenario.

## 4.1 Searching for Characteristics

A common approach to construct differential characteristics, which have a high probability, is to use a linearized approximation of the attacked hash function. As observed by Rijmen and Oswald [22], all differential characteristics for a linearized hash function can be seen as the codewords of a linear code. To find good differential characteristics we used the same technique as Mendel and Nad in the cryptanalysis of the first version of SIMD [15]. The procedure can be described in the following way:

- Linearize the step function of SIMD, i.e. replace all nonlinear operations with linear ones.
- Construct a generator matrix.
- Use a probabilistic algorithm from coding theory to search for codewords with low Hamming weight.

The nonlinear parts of the step function are the modular additions and the Boolean function IF and MAJ. In the attack, we replace all modular additions by XORs. Since we aim for a characteristic with low Hamming weight, we replace the Boolean functions with the 0-function, *i.e.* we block each input difference in $\Phi$ such that the output difference is always zero. This has probability $1/2$ in most cases. Note that there is exactly one input difference for IF and one for MAJ where the output difference is always one. Such characteristics are discarded.

For the search we used the *CodingTool Library* [20], which is an open-source implementation of the needed coding theoretic algorithms and data structures. We searched for good differential characteristics for the backward and forward direction with no differences in the message. Moreover, we also searched for a good starting step. One would expect that starting from the exact middle (round 16) would result in the best probability, but it turns out that moving the starting step two steps further, results in a better overall probability.

**Differential Characteristics.** The complete differential characteristics are given in Appendix A. To describe the differential characteristics we used signed-bit differences introduced by Wang *et al.* [26] in the cryptanalysis of MD5. The advantage of using signed-bit differences is that there exists a unique mapping to both XOR and modular differences.

The characteristic for the backward direction consists of the first 18 steps of the permutation and has Hamming weight 72. The characteristic for the forward direction consists of the last 14 steps of the permutation and has Hamming weight 52.

To estimate the success probability of each characteristic we used the same heuristic as in [15]. The probability for blocking a difference in one bit at the input of IF or MAJ is 1/2 or 0 for some cases, but then the characteristic is discarded. Hence, the total probability is determined by the sum of all differences at the input. Differences at the same bit positions are counted only once. For the modular additions carries are not prevented for each bit difference. By allowing carries in the first addition, one can compensate them at the second addition. However, the rotation after the first modular addition needs to be considered. Therefore, the probability in this part is slightly decreased, but results in a overall increase. Table 5 summarizes the overall probability of each characteristic.

**Table 5.** Summary of the success probabilities.

| characteristic | Hamming weight | probability |
|---|---|---|
| backward | 72 | $2^{-72.04}$ |
| forward | 52 | $2^{-51.4}$ |

### 4.2 Independency of the Characteristics

The assumption on independent characteristics is quite strong (cf. [19]). Nevertheless, one can check this property easily for few steps in both directions, which was done for the presented characteristics. Furthermore, the used characteristics have a low Hamming weight, which makes it very unlikely that they interfere with each other.

### 4.3 Complexity of the Attack

As described in Section 2 the generic complexity for the attack is $2^{n/3}$. For the SIMD compression function $n$ is 1024 bits. Hence, the generic complexity is $\approx 2^{342}$. The total complexity of the attack based on the presented characteristic is $(2^{72.04} \cdot 2^{51.4})^2 \approx 2^{247}$ which can be improved by ignoring conditions at the end. As was already observed by Wang *et al.* [25] in the cryptanalysis

of SHA-1 conditions resulting from the modular addition in the last steps of the differential characteristic can be ignored, due to the fact that carries can be ignored since the modular difference at the output stays the same. This reduces the complexity by a factor $2^{8.24}$ in the backward direction and $2^2$ in the forward direction which improves the overall complexity by a factor of $2^{2 \cdot 10.24}$ resulting in $2^{226.52}$.

*Remark:* Note that we also have the freedom to choose the actual values for the state (at the beginning of each characteristic) and for the message. Message/chaining input modification can be used to improve the attack complexities further.

## 5 Extending the Attack to the Compression Function

In contrast to SHA-2 it is not easy to extend the second-order differential collision to the compression function since the feed-forward of SIMD is non-linear. However, the first step of the feed-forward is almost linear and therefore we can show non-random properties in the output of the state variables $D_i$ for $i = 0, \ldots, 7$.

In the feed-forward 4 additional steps with the initial value as message input are performed. This destroys the distinguishing property at the output of the permutation. However, the values of $D_i^{36}$ for $i = 0, \ldots, 7$ (output of the feed-forward) are determined already in the first step of the feed-forward and not modified in the other three steps. By considering only $D_i^{36}$ for $i = 0, \ldots, 7$ and accordingly only $A_i^0$ for $i = 0, \ldots, 7$ of the initial value the attack complexity is only slightly increased. Consequently, the dimension of the input and output space for the distinguisher is reduced to 256 bits ($8 \cdot 32$). However, by fixing the differences in the rectangle in the middle of the second-order differential characteristic one can construct a distinguisher for the compression function.

### 5.1 Distinguisher for the Compression Function

For the feed-forward of SIMD we extend the scheme shown in Figure 1 to the one shown in Figure 3. The function $F_2$ takes two inputs, namely the state of the last step and the chaining value. As mentioned before we consider only $A_i^0$ in the initial value and $D_i^{36}$ at the output which is denoted by the quartets $\{P_{A_i}, P^*_{A_i}, Q_{A_i}, Q^*_{A_i}\}$ and $\{\tilde{R}_{D_i}, \tilde{R^*}_{D_i}, \tilde{S}_{D_i}, \tilde{S^*}_{D_i}\}$, respectively.

So far we have considered the inputs $X$, $\beta$ and $\gamma$ to be unrelated. Due to the way we build the second-order collisions, we can see that they are the inputs to a rectangle, hence they are related in the middle of the rectangle (gray layer in Figure 3). Therefore, we can extend the attacks by fixing $\beta$ and $\gamma$, since the complexity of the generic case for this type of attacks is $2^n$ (or $2^t$) [3]. Since we show non-randomness only in part of the output, namely $D_i$ for $i = 0, \ldots, 7$, the generic complexity of the attack becomes $2^t = 2^{8 \cdot 32} = 2^{256}$. Hence, by using the second-order differential characteristic from Section 4.1 one can construct a

distinguisher for the compression function of SIMD. Note that the distinguisher becomes even more powerful if the attacker can find several of the above quartets with the same difference.

To summarize, the algorithm works as follows:

1. Use the differential from Section 4.1
2. Choose a random value for $X$ and compute $X^* = X + \beta$, $Y = X + \gamma$, and $Y^* = X^* + \gamma$.
3. Compute backward from $X, X^*, Y, Y^*$ using $F_0^{-1}$ to obtain $P_{A_i}, P^*_{A_i}, Q_{A_i}, Q^*_{A_i}$.
4. Compute forward from $X, X^*, Y, Y^*$ using $F_1$ and $F_2$ to obtain $\tilde{R}_{D_i}, \tilde{R^*}_{D_i}, \tilde{S}_{D_i}, \tilde{S^*}_{D_i}$.
5. Check if $P^*_{A_i} - P_{A_i} = Q^*_{A_i} - Q_{A_i}$ and $\tilde{S}_{D_i} - \tilde{R}_{D_i} = \tilde{S^*}_{D_i} - \tilde{R^*}_{D_i}$ and therefore $P^*_{A_i} - P_{A_i} - Q^*_{A_i} + Q_{A_i} + \tilde{S}_{D_i} - \tilde{R}_{D_i} - \tilde{S^*}_{D_i} + \tilde{R^*}_{D_i} = 0$ is fulfilled.

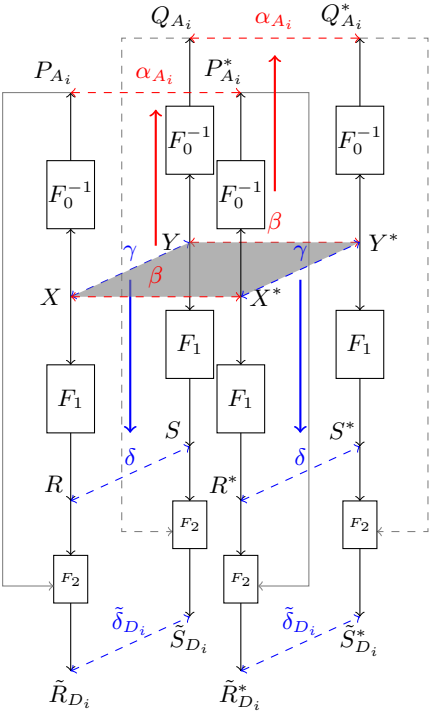


**Fig. 3.** Extending the attack to the compression function.

### 5.2 Complexity of the Attack

As mentioned before the attack complexity is increased slightly by the feed-forward. In fact using the backward and forward characteristics from Table 6

and Table 7 the additional costs are negligible. In backward direction we have at the end only a difference in $\Delta A_6^{-1}$ which needs to be considered. This difference is rotated to the left by $s$ bits. In the forward direction we have differences in $\Delta B_0^{31}$ and $\Delta A_3^{31}$. Both are input to the Boolean IF function. Blocking each difference at the input of the IF function costs $2^2$ for both differences. Additionally, $\Delta A_3^{31}$ is used to compute $\Delta A_6^{32}$ in the following way:

$$\Delta A_6^{32} = (\Delta D_6^{31} + \Delta A_6^{-1} + IF(\Delta A_6^{31}, \Delta B_6^{31}, \Delta C_6^{31})) \lll s^{32} + (\Delta A_3^{31} \lll r^{32}) \quad (7)$$

In Equation (7) only $\Delta A_6^{-1}$ and $\Delta A_3^{31}$ have differences. Only the rotation to the left by $s^{32}$ bits adds a complexity about $2^1$ [15].

Finally, we can ignore the costs of the last three steps in the backward ($2^{8.24+7+5}$) and forward ($2^{1+1.4+2}$) direction since we only consider the state variables $A_i$ and $D_i$ for $i = 0, \ldots, 7$ respectively. The differences in these variables do not change in the last three steps. Therefore, the total complexity is $(2^{72.04-20.24} \cdot 2^{51.4-4.4})^2 \cdot 2^1 \cdot 2^2 \approx 2^{200.6}$.

Hence, one can distinguish the compression function of SIMD from a random function with a complexity of about $2^{200.6}$. Note that the generic complexity for this attack is $2^{256}$.

## 6 Conclusions and Discussion

In this paper, we present a distinguisher for the full permutation of SIMD-512 by an application of the boomerang attack on hash functions. Starting from the middle of the compression function we used techniques from coding theory to search for two differential characteristics, one for the backward direction and one for the forward direction, which hold with high probability. Then we construct a second-order differential and define a distinguishing property such that we can distinguish the permutation from a random permutation with a complexity of $2^{226.52}$.

Furthermore, we extend the attack to the full compression function of SIMD-512. By fixing the differences in the rectangle we can distinguish the output of the compression function from a random function with a complexity of $2^{200.6}$ compression function evaluations. This is a significant improvement to the current best known distinguisher with complexity $2^{398}$ [27].

However, our attack does not invalidate the security claims of the designers since it seems difficult to extend such an attack to the hash function and most of the security comes from the message expansion. In [4] the designers presented a more detailed analysis of SIMD regarding differential paths without differences in the message and are claiming that such characteristics does not affect the security of the SIMD hash function. Nevertheless, the results presented in this paper show how boomerang like attacks can be effectively used on compression functions. Furthermore, the results contribute to a better understanding of the security margin of SIMD.

**Acknowledgments**

The work in this paper has been supported by the European Commission under contract ICT-2007-216646 (ECRYPT II) and by the Austrian Science Fund (FWF, project P21936).

## References

1. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.
2. Alex Biryukov, Mario Lamberger, Florian Mendel, and Ivica Nikolic. Second-Order Differential Collisions for Reduced SHA-256. In *ASIACRYPT*, 2011. Io appear.
3. Alex Biryukov, Ivica Nikolic, and Arnab Roy. Boomerang Attacks on BLAKE-32. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 218–237. Springer, 2011.
4. Charles Bouillaguet, Pierre-Alain Fouque, and Gatan Leurent. Security Analysis of SIMD. Cryptology ePrint Archive, Report 2010/323, 2010.
5. Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.
6. Itai Dinur and Adi Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.
7. Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *LNCS*, pages 365–383. Springer, 2010.
8. Przemyslaw Sokolowski Ivica Nikolić, Josef Pieprzyk and Ron Steinfeld. Rotational Cryptanalysis of (Modified) Versions of BMW and SIMD. Available online, 2010.
9. Antoine Joux and Thomas Peyrin. Hash Functions and the (Amplified) Boomerang Attack. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *LNCS*, pages 244–263. Springer, 2007.
10. Lars R. Knudsen. Truncated and Higher Order Differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
11. Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard Blahut, Daniel Costello Jr., Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography*, pages 227–233. Kluwer, 1992.
12. Mario Lamberger and Florian Mendel. Higher-Order Differential Attack on Reduced SHA-256. Cryptology ePrint Archive, Report 2011/037, 2011.
13. Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. SIMD Is a Message Digest. Submission to NIST (Round 2), September 2009. Available online: `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html`.
14. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *LNCS*, pages 474–494. Springer, 2005.
15. Florian Mendel and Tomislav Nad. A Distinguisher for the Compression Function of SIMD-512. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *LNCS*, pages 219–232. Springer, 2009.
16. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *LNCS*, pages 16–35. Springer, 2009.

17. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.
18. Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 428–446. Springer, 1989.
19. Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.
20. Tomislav Nad. The CodingTool Library. Workshop on Tools for Cryptanalysis 2010, 2010. `http://www.iaik.tugraz.at/content/research/krypto/codingtool/`.
21. National Institute of Standards and Technology. Cryptographic Hash Algorithm Competition, November 2007. Available online: `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`.
22. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
23. Michael Vielhaber. Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. Cryptology ePrint Archive, Report 2007/413, 2007.
24. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.
25. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
26. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
27. Hongbo Yu and Xiaoyun Wang. Cryptanalysis of the Compression Function of SIMD. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *LNCS*, pages 157–171. Springer, 2011.

## A Differential Characteristics for the Forward and Backward Direction

**Table 6.** Backward characteristic. The state at step $-1$ is the chaining value.

| step | state | probability |
|---|---|---|
| $-1$ | $\Delta D_0 : -28, \Delta D_3 : -7, \Delta B_5 : -12, \Delta C_5 : +4, \Delta A_6 : +3, \Delta C_6 : +25, \Delta C_7 : +5, \Delta D_7 : -15$ | |
| 0 | $\Delta A_0 : -19, \Delta A_3 : -30, \Delta C_5 : -12, \Delta D_5 : +4, \Delta B_6 : +6, \Delta D_6 : +25, \Delta D_7 : +5$ | $2^{-8.24}$ |
| 1 | $\Delta B_0 : -10, \Delta B_3 : -21, \Delta D_5 : -12, \Delta C_6 : +6, \Delta A_7 : +22$ | $2^{-7}$ |
| 2 | $\Delta C_0 : -10, \Delta C_3 : -21, \Delta D_6 : +6, \Delta B_7 : +7$ | $2^{-5}$ |
| 3 | $\Delta D_0 : -10, \Delta D_3 : -21, \Delta A_6 : +9, \Delta C_7 : +7$ | $2^{-4}$ |
| 4 | $\Delta A_0 : -1, \Delta B_6 : +12, \Delta D_7 : +7$ | $2^{-4}$ |
| 5 | $\Delta B_0 : -24, \Delta C_6 : +12$ | $2^{-3}$ |
| 6 | $\Delta C_0 : -24, \Delta D_6 : +12$ | $2^{-2}$ |
| 7 | $\Delta D_0 : -24, \Delta A_6 : +15$ | $2^{-2}$ |
| 8 | $\Delta B_6 : +11$ | $2^{-2}$ |
| 9 | $\Delta C_6 : +11$ | $2^{-1}$ |
| 10 | $\Delta D_6 : +11$ | $2^{-1}$ |
| 11 | $\Delta A_6 : +7$ | $2^{-1}$ |
| 12 | $\Delta A_1 : +3, \Delta B_6 : +3$ | $2^{-2}$ |
| 13 | $\Delta B_1 : +22, \Delta A_5 : +22, \Delta C_6 : +3$ | $2^{-3}$ |
| 14 | $\Delta C_1 : +22, \Delta A_4 : +12, \Delta B_5 : +12, \Delta D_6 : +3$ | $2^{-4}$ |
| 15 | $\Delta D_1 : +22, \Delta A_2 : +19, \Delta B_4 : +19, \Delta C_5 : +12, \Delta A_6 : +31$ | $2^{-5.4}$ |
| 16 | $\Delta A_0 : +16, \Delta A_1 : +31, \Delta B_2 : +16, \Delta A_4 : +28, \Delta C_4 : +19, \Delta D_5 : +12, \Delta B_6 : +28$ | $2^{-7.4}$ |
| 17 | $\Delta B_0 : +25, \Delta B_1 : +8, \Delta A_2 : +8, \Delta C_2 : +16, \Delta A_3 : +25, \Delta B_4 : +5, \Delta D_4 : +19, \Delta A_5 : +27, \Delta C_6 : +28, \Delta A_7 : +5$ | $2^{-10}$ |

**Table 7.** Forward characteristic.

| step | state | probability |
|---|---|---|
| 17 | $\Delta B_0 : +24, \Delta D_0 : +32, \Delta C_2 : -29, \Delta D_3 : +1, \Delta C_4 : -7, \Delta A_6 : -23, \Delta B_6 : +14, \Delta B_7 : +3, \Delta C_7 : -13$ | |
| 18 | $\Delta A_0 : +5, \Delta C_0 : +24, \Delta D_2 : -29, \Delta D_4 : -7, \Delta B_6 : -6, \Delta C_6 : +14, \Delta C_7 : +3, \Delta D_7 : -13$ | $2^{-9}$ |
| 19 | $\Delta B_0 : +10, \Delta D_0 : +24, \Delta A_2 : -26, \Delta A_4 : -4, \Delta C_6 : -6, \Delta D_6 : +14, \Delta D_7 : +3$ | $2^{-8}$ |
| 20 | $\Delta C_0 : +10, \Delta B_2 : -23, \Delta B_4 : -1, \Delta D_6 : -6, \Delta A_7 : +12$ | $2^{-7}$ |
| 21 | $\Delta D_0 : +10, \Delta C_2 : -23, \Delta C_4 : -1, \Delta B_7 : +21$ | $2^{-5}$ |
| 22 | $\Delta A_0 : +15, \Delta D_2 : -23, \Delta D_4 : -1, \Delta C_7 : +21$ | $2^{-4}$ |
| 23 | $\Delta B_0 : +20, \Delta A_4 : -30, \Delta D_7 : +21$ | $2^{-4}$ |
| 24 | $\Delta C_0 : +20, \Delta B_4 : -2$ | $2^{-3}$ |
| 25 | $\Delta D_0 : +20, \Delta C_4 : -2$ | $2^{-2}$ |
| 26 | $\Delta A_0 : +13, \Delta D_4 : -2$ | $2^{-2}$ |
| 27 | $\Delta B_0 : +6$ | $2^{-2}$ |
| 28 | $\Delta C_0 : +6$ | $2^{-1}$ |
| 29 | $\Delta D_0 : +6$ | $2^{-1}$ |
| 30 | $\Delta A_0 : +31$ | $2^{-1.4}$ |
| 31 | $\Delta B_0 : +24, \Delta A_3 : +24$ | $2^{-2}$ |