Boomerang and Slide-Rotational Analysis of the SM3 Hash Function

Aleksandar Kircanski and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, Quebec, CANADA

Abstract. SM3 is a hash function designed by Xiaoyun Wang *et al.*, and published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service system. The design of SM3 builds upon the design of the SHA-2 hash function, but introduces additional strengthening features. In this paper, using a higher order differential cryptanalysis approach, we present a practical 4-sum distinguisher against the compression function of SM3 reduced to 32 rounds. In addition, we point out a slide-rotational property of SM3-XOR, which exists due to the fact that constants used in the rounds are not independent.

1 Introduction

In December of 2007, the Chinese National Cryptographic Administration Bureau released the specification of a Trusted Cryptography Module, detailing a cryptoprocessor to be used within the Trusted Computing framework in China. The module specifies a set of cryptographic algorithms that include the SMS4 block cipher, the SM2 asymmetric algorithm and SM3, a new cryptographic hash function designed by Xiaoyun Wang *et al.* [6]. The design of SM3 resembles the design of SHA-2 but includes additional fortifying features such as feeding two message-derived words into each round, as opposed to only one in the case of SHA-2.

The main idea of the second order differential cryptanalysis of hash functions [2, 3] is to use the boomerang technique [14], previously used for block ciphers, whereby the additional freedom to choose the key is exploited by using message modification techniques. Unlike in the context of first order analysis, where message modification is applied on a pair of messages, in second order analysis, this technique is applied to a quartet of values. Generally, the aim of the second order analysis is to find zero-sum quartets, i.e., quartets of input-output function values, for which the four inputs as well as the four outputs sum to zero. In case of a compression function that follows Davies-Meyer mode, a zero-sum can be seen as a second order collision for the compression function [2]. Finally, the zero-sum condition can be considered as an *evasive* property [5]. Such a property is impossible to achieve with a non-negligible probability using oracle accesses to an ideal primitive. Thus, if it can be shown that the property can be satisfied for a particular construction, then it can be used for disproving its indifferentiability claims [4]. Another example of evasive properties in the context of hash functions are rotational properties [7]. Two words are said to be rotational if they are equal up to bit-wise rotation by some

number of positions. If among the outputs for some carefully chosen inputs, the rotational relations hold with probability higher than the corresponding one for ideal function, then a distinguisher can be mounted [8].
In this work, we investigate the security of the SM3 hash function. To the best of our knowledge, this is the first public analysis of SM3. In the first part of the paper, we present a practical algorithm to find a second order collision for 32 rounds reduced version of the SM3 compression function. An interesting feature of our approach is that the two differential paths that are used for the bottom and the top part of the boomerang are not independent, as was required in [2]. This results in seemingly conflicting bit conditions [11] in the early rounds of the bottom part of the boomerang. However, as will be shown by the analysis in this paper, the bit conflict that occurs is recoverable and it can be bypassed by using a long carry propagation on the left and the right face of the boomerang. The long carry propagation that is required to happen in order to resolve the conflicting condition is a relatively low-probability event. However, in our approach, it is ensured by message modification and does not affect the overall probability of the second-order collision search.
In the second part of the paper, we note a slide-rotational property of SM3 and, we analyze the SM3-XOR compression function, which is the SM3 compression function with the addition mod 2^{32} replaced by XOR. In particular, we show that, for SM3-XOR, one can easily construct input-output pairs satisfying a simple rotational property. Such a property exists due to the fact that, unlike in SHA-2, the constants in rounds i, $i+1$, for $i = 0, \ldots, 63$, $i \neq 15$ are computed by bitwise rotation starting from two predefined independent values. Previously, SHA2-XOR was analyzed in [15].

1.1 Related work

In [2, 3], Biryukov *et al.* presented second order analysis of SHA-2 and BLAKE. In particular, for the SHA-2 hash function, a second order collision for its compression function reduced to 46 rounds was computed [2]. The BLAKE hash function reduced to 8 rounds was shown to be suspectable to a second order attack which requires around 2^{242} compression function calls [3]. Sasaki [13] provided a second order collision for the compression function of the 5-pass HAVAL. A distinguisher for 32-round Skein-256 [10] requiring 2^{114} compression function calls was presented by Leurent *et al.* Rotational cryptanalysis was introduced by Khovratovich *et al* [7]. The SHA2-XOR compression function was analyzed by Yoshida *et al.* [15], where it was shown that an iterative differential can be used to detect non-randomness for up to 31 rounds of SHA2-XOR. The probability of the 31-round iterative differential for SHA2-XOR is 2^{-246} whereas for a random function the corresponding probability should be 2^{-256}. This allowed an attack against 32-round SHACAL-2-XOR and also a pseudo-collision attack for SHA2-XOR reduced to 34 rounds.
The rest of the paper is organized as follows. The relevant specifications of the SM3 hash function are briefly reviewed in the next section. In Section 3, relevant background on higher order analysis of hash functions and the notation used throughout the paper are given. Our second order attack against a reduced-round SM3 compression function is described in Section 4. The slide-rotational property of SM3 is discussed in Section 5. Finally, our conclusion is given in Section 6.

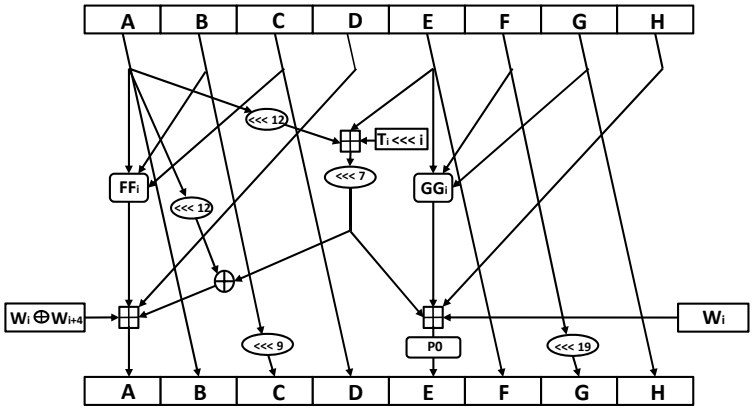

Fig. 1. One round of the SM3 hash function

2 Specifications of the SM3 hash function

SM3 is a Merkle-Damgård construction that processes 512-bit input message blocks and returns a 256-bit hash value. Before hashing, the message of length l is padded by a bit set to 1, followed by k bits set to 0, where k is the smallest integer such that $l + 1 + k = 448 \bmod 512$. Finally, the remaining 64 bits are set to the value of l in the binary form. SM3 consists of two parts: the message expansion and the state update function (see Fig. 1). Below, we describe the two parts. The auxiliary functions, P_0 and P_1, both operating on 32-bit words are used in the specifications and are defined by:

$$P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$$
$$P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23).$$

Message expansion: The input here is the 512 message block represented as 16 32-bit words, $M_0, \ldots, M_{15}$. It is expanded to 68 32-bit words by letting $W_i = M_i$ for $0 \leq i < 16$ and

$$W_i = P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6} \qquad (1)$$

for $16 \leq i < 68$. Another expanded message array used in SM3 is W'_i, $0 \leq i < 64$, defined by

$$W'_i = W_i \oplus W_{i+4}$$

State update transformation: In SM3, the state update starts from the fixed initial value of 8 32-bit words [6] and updates them in 64 rounds. Let A, B, C, D, E, F, G and H denote the inner state registers. As shown in Fig.

1, the j-th round transformation is given by

$$\begin{aligned} SS1 &= ((A \lll 12) + E + (T_j \lll j)) \lll 7, \\ SS2 &= SS1 \oplus (A \lll 12) \\ TT1 &= FF_j(A,B,C) + D + SS2 + W'_j \\ TT2 &= GG_j(E,F,G) + H + SS1 + W_j \\ D = C,\ \ C &= B \lll 9,\ \ B = A,\ \ A = TT1 \\ H = G,\ \ G &= F \lll 19,\ \ F = E,\ \ E = P_0(TT2) \end{aligned} \tag{2}$$

where the functions FF_j and GG_j are defined by

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \le j \le 15 \\ (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z) & 16 \le j < 64 \end{cases}$$

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \le j \le 15 \\ (X \wedge Y) \vee (\neg X \wedge Z) & 16 \le j < 64 \end{cases}$$

The round constants are $T_j = 0x79cc4519$ for $j \in \{0, \ldots, 15\}$ and $T_j = 0x7a879d8a$, for $j \in \{16, \ldots, 63\}$.

Comparison with SHA-2: The major difference between SHA-2 and SM3 is that in each round of SM3, two expanded message words are fed to the inner state, as opposed to just one in SHA-2. Also, the maximal distance between taps in the message expansion mechanism in SM3 is 4, whereas in SHA-2, it is 8. Another difference is that while addition modulo 2^{32} is used in the message expansion and the feedforward mechanisms in case of SHA-2, only XOR is used in SM3. Finally, one round of the SM3 hash function contains 8 mod 2^{32} additions, as opposed to 7 such additions in the case of SHA-2.

3 Background and notation

The following notation is used throughout the paper:

- $x^{(b)}$: the b^{th} bit of an n-bit word x
- $x^{(c \cdots b)}$: the word $x^{(c)}x^{(c-1)} \cdots x^{(b)}$
- e_i: an n-bit unit vector with 1 in the i^{th} bit position
- $\overline{x}$: the bit-wise complemented word (or bit) corresponding to x
- W_j^i, $1 \le i \le 4$, $0 \le j \le 63$: expanded message words, where i denotes the boomerang branch. More precisely, $i = 1, 2$ signify the left and right branches on the front face and $i = 3, 4$ signify the left and right branches on the back face of the boomerang
- h_j, $0 \le j \le 63$: the 256-bit compression function inner state after j rounds (e.g., h_0 is the state before any round has been executed)
- i-th round: the transformation that maps h_i into h_{i+1}
- h_j^i, $1 \le i \le 4$, $0 \le j \le 63$: the 256-bit inner state after j rounds at the i-th boomerang branch, where h_i^1 and h_i^2 correspond to the front face branches and h_i^3 and h_i^4 correspond to the back face branches.

3.1 Higher-order analysis of hash functions

In this section, the main idea of how to use the boomerang attack in the context of compression functions is provided. The goal of this attack on the function f is to find a quartet (x_0, x_1, x_2, x_3) such that

$$\begin{aligned} x_0 \oplus x_1 \oplus x_2 \oplus x_3 &= 0 \\ f(x_0) \oplus f(x_1) \oplus f(x_2) \oplus f(x_3) &= 0 \end{aligned} \tag{3}$$

which is called a *zero-sum* or equivalently, a *second-order collision*. For a detailed exposition of these notions, plese refer to Appendix A. The strategies to

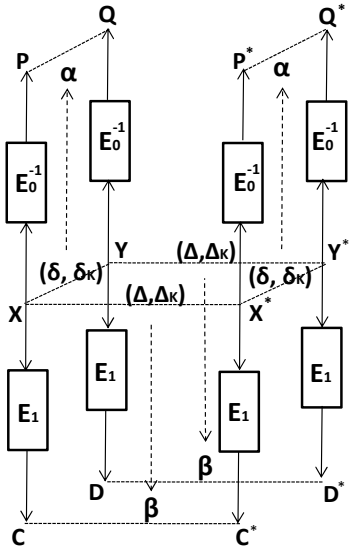

Fig. 2. Boomerang attack against a compression function

construct a second order collision previously applied to SHA-2 and BLAKE [2, 3] varied to some degree. Here, we review the approach used in [2]. The general idea is to construct a quartet that forms boomerang structure [14] for a block cipher in the Davis-Meyer mode. The differentials used in the boomerang are related key differentials, where the secret key of the block cipher corresponds to the message block in the case of a compression function. The encryption function is divided into two parts, $E_1 \circ E_0$. As shown in Fig. 2, for the bottom part of the boomerang, a related-key differential $(\Delta, \Delta_K) \rightarrow \beta$ for E_1 with probability q is constructed. Similarly, another related-key differential $(\delta, \delta_K) \rightarrow \alpha$ with probability p is used for E_0^{-1}. Then, an attempt to randomly satisfy the differentials in the boomerang structure, with probability p^2q^2 would proceed as follows:

- Randomly choose X, the inner state in the middle of the hash function execution, representing the input to E_1 (and the output of E_0). Let $X^* = X \oplus \Delta$, $Y = X \oplus \delta$ and $Y^* = X \oplus \Delta \oplus \delta$.

- Compute backward from X, X^*, Y, Y^* using E_0^{-1} to obtain P, P^*, Q, Q^*, using keys K, $K \oplus \Delta_K$, $K \oplus \delta_K$, $K \oplus \delta_K \oplus \Delta_K$, respectively.
- Compute forward from X, X^*, Y, Y^* using E_1 to obtain C, C^*, D, D^* using keys K, $K \oplus \Delta_K$, $K \oplus \delta_K$, $K \oplus \delta_K \oplus \Delta_K$, respectively.
- Verify whether $C \oplus C^* = D \oplus D^*$ and $P \oplus Q = P^* \oplus Q^*$.

If the last condition is satisfied, a zero-sum quartet is found for the encryption function in Davis-Meyer mode, since $P \oplus Q \oplus P^* \oplus Q^* = 0$ and also $(C \oplus P) \oplus (C^* \oplus P^*) \oplus (D \oplus Q) \oplus (D^* \oplus Q^*) = 0$.
To improve the efficiency of the above process, instead of trying to satisfy the boomerang randomly, message modification can be used for some of the differential paths in the boomerang. For example, in [2], message modification is applied to satisfy the middle part of the boomerang, i.e., to satisfy the two differentials paths of the function E_1. The other paths in the boomerang are satisfied randomly.

4 Zero-sum for reduced-round SM3

Here, a method for finding a zero-sum for the 32-round SM3 compression function is detailed. An example for the found zero-sum for the 32-round reduced SM3 compression function is given in Table 3.

4.1 Choosing the differential paths

In what follows, the backward and the forward differential paths used in the boomerang are provided and we explain why the two chosen paths are favorable. The 32-round block cipher used in the Davis-Meyer mode in the SM3 compression function is decomposed into $E_1 \circ E_0$. The function E_0 consists of rounds $r = 0, \ldots, 14$ and the function E_1 consists of rounds $15, \ldots, 31$. The forward and backward differential paths for E_1 and E_0 used in the attack are given at the end of the Appendix (Tables 1 and 2, respectively). For example, the last row in Table 1 denotes that there is no active bits between the two inner states representing the output of round 14.
The paths have been found by linearizing the compression function and then applying CodingTool [12], a tool for effective search for low Hamming weight codewords of a given linear code. The linearization amounted to replacing addition mod 2^{32} by XOR and the functions FF_i and GG_i, $16 \leq i < 64$ by zero functions. Functions FF_i and GG_i, $0 \leq i < 16$, have been left unchanged, since they are already linear. The input to CodingTool is a generating matrix of a linear code and the output is a low Hamming-weight codeword. Here, the linear code in question is the linear mapping from the message to the concatenated bit-vectors representing consecutive inner states of the compression function. The matrix describing this mapping, i.e., the generating matrix of the linear code, is obtained by applying the linearized compression function to the unit vectors. Then, the matrix is fed to the low Hamming weight codeword search algorithm. The search is done for both functions E_0^{-1} and E_1.
The probabilities for the provided differentials are obtained by multiplying the round probabilities provided in Tables 1 and 2. As shown in the tables, the overall probabilities for E_0^{-1} and E_1 paths are 2^{-25} and 2^{-57}, respectively. Therefore, assuming that the events of satisfying the two differential paths are

independent, by using a naive search, the probability of finding a quartet that yields a zero-sum would be $2^{-2\times(25+57)} = 2^{-164}$. Instead of applying a naive search, message modification allows a major improvement to this complexity. The differential path for E_1 in rounds $15-19$ is satisfied by using message modification and the rest, that is, rounds $0-4$ and also round 31, is satisfied by a random search. Then, the search complexity drops to $2^{2\times(25+2)} = 2^{54}$.
To clarify the advantage of the paths given in Tables 1 and 2, first we note that the Hamming weight of the two paths does not change if the paths are rotated by some fixed number of bit-positions. Therefore, the rotation amount is a free parameter that can be chosen to maximize the probability of success. As for the backward path shown in Table 1, the choice of the rotation amount is simply due to the fact that the number of active most significant bits is maximized, which improves the differential probability, given that the active most significant bits do not affect such probability.
As for the rotational amount used for the forward differential, the number of most significant bits in rounds $15-19$ is less important, because in these rounds, the path is satisfied by message modification and there is no need to minimize the path probability by forcing bits to be on the most significant bit position. What matters for the forward differential is that one of the active bits in round 31 would correspond to the most significant bit since, as explained above, this condition is satisfied by a random search, as is the case with the particular paths in Table 2. This reduces the exhaustive search by more than a factor of 2 and when the alternative paths for the path in round 31 are taken into account, the reduction is by a factor of around 3. This reduction is relatively significant, since our goal is to find a zero-sum efficiently with a practical complexity.
In addition to the path given in Table 2, it can be easily verified that there exist two more paths obtained by rotating the one in Table 2 such that, in round 31, one of the active bits is the most significant bit. These two paths are obtained using rotation to the left by 17 and 9 positions. However, each of the three paths have conflicting bit conditions in rounds $15-16$ with respect to the backward differential, including the path given in Table 2. In the next subsection, we show that the conflict between the backward path and the forward path given in Table 2 is recoverable, i.e., it can be bypassed by message modification, which is the reason why we focus on this path.

4.2 Message modification and the conflicting bits

In this section, we provide some details on the message modification technique in the context of the boomerang, i.e., where the message modification is performed on a quartet of values, instead of on a pair of values. The focus is put on resolving the particular conflict between the bit-conditions on the two faces of boomerang that occurs in our SM3 analysis. A simple general tool for bypassing such conflicts, in the case when this is possible, is provided.
The message modification procedure that satisfies rounds $15-19$ of the forward differential on the front and the back face of the boomerang proceeds as follows. Here, by message modification, we also assume the modification of the middle inner state registers. Following the notation specified in Section 3, let h_{15}^1 and h_{15}^2 denote the inner states satisfying the difference specified by the first row of Table 2. In the back face of the boomerang, the corresponding

inner states are $h_{15}^3 = h_{15}^1$ and $h_{15}^4 = h_{15}^2$. The goal is to have both the front face and back face differences propagate according to Table 2.

The modification procedure can start from h_{15}^1. The message words W_{15}^1, W_{15}^2 and the inner states h_{15}^1, h_{15}^2 are modified so that the difference on the front face between h_{15}^1 and h_{15}^2 propagates according to Table 2. Now, if the bit-conditions for controlling the propagation between h_{15}^1 and h_{15}^3 do not conflict with the bit-conditions for h_{15}^1 and h_{15}^2, the message modification procedure can be applied again, but this time on h_{15}^1 and h_{15}^3. Then, clearly, the difference between h_{15}^3 and h_{15}^4 propagates in the exact same way as the difference between h_{15}^1 and h_{15}^2 and the goal is fulfilled. Also, due to the boomerang property, the difference between h_{15}^2 and h_{15}^4 propagates in the same way as h_{15}^1 and h_{15}^3.

However, the conflicting bit condition occurs due to the backward and forward paths in Tables 1 and 2. The conflicting condition and how it is resolved is explained below and also visualized on Fig. 3, where rounds 15 and 16 are shown. As depicted in the figure, the front-side and back-side differences (due to the forward path) are denoted by Δ and the left-side and right-side differences (due to the backward path) by δ. The conflict arises when one attempts to force the bit 22 difference coming from D_{15} not propagate to more than one bit in both front-side and back-side of A_{16}. The problem is that bit 22 is also active on the left-side and the front-side, as shown in Fig. 3 beside the $W_{15} \oplus W_{19}$ word. In particular, due to the message difference specified by the backward differential, we have

$$(W_{15}^1 \oplus W_{19}^1) \oplus (W_{15}^3 \oplus W_{19}^3) = e_{14} \oplus e_{22} \oplus e_{31} \tag{4}$$

At the same time, due to (2), for the active bit 22 of D_{15} to propagate only to a 1-bit difference in A_{16} in both the front and the back face of the boomerang, bits 22 of both

$$\alpha^1 = FF_{15}(A_{15}^1, B_{15}^1, C_{15}^1) + SS2_{15}^1 + (W_{15}^1 \oplus W_{19}^1)$$
$$\alpha^3 = FF_{15}(A_{15}^3, B_{15}^3, C_{15}^3) + SS2_{15}^3 + (W_{15}^3 \oplus W_{19}^3)$$

have to be fixed to the bit value $b = 0$ if there is no carry generated at bit position 21 in neither of the two additions

$$A_{16}^1 = \alpha^1 + D_{15}^1 \tag{5}$$
$$A_{16}^3 = \alpha^3 + D_{15}^3 \tag{6}$$

If, however, there is a carry generated at position 21 in both (5) and (6), then the above bit b needs to be fixed to 1. Thus, under the assumption that the carry is either generated at position 21 in both (5) and (6), or not generated in neither of these two additions, both the front and the back face of the boomerang cannot be satisfied, since bit 22 is active in (4).

Thus, to bypass the conflicting bit, the fact that, in (4), bit 14 is also active should be utilized. Then, by using an 8-bit long carry propagation from bit 14 to bit 22 on the left and the right face of the boomerang, the existence of carry at bit 21 can be ensured in exactly one of the additions (5) and (6), which cancels out the activation of bit 22 on the left and the right face of the boomerang.

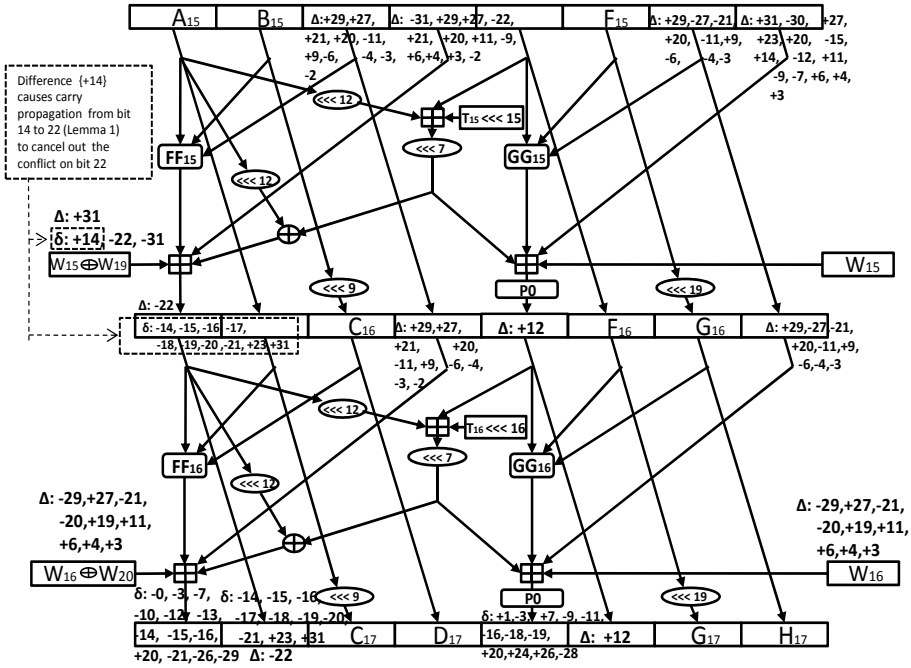

Fig. 3. Resolving the conflicting bit condition in round 16

Next, a simple lemma that ensures long carry propagation for the purpose of deactivating a particular bit is provided. The lemma can be used during the message modification process, i.e., whenever a deactivation of a bit by a carry chain is needed. Consider sums of n-bit words $X + S$ and $X' + S$ and suppose that bits k and l, where $k < l$ are active in X. The lemma specifies how to perform message modification on X so that the bit l in $X + S$ and $X' + S$ remains inactive.

Lemma 1 *Let X, X', and S be n-bit words. Also, let $0 \leq k < l \leq n$ and $X \oplus X' = e_l \oplus e_k$. If*

$$2^k \leq X^{(k-1..0)} + S^{(k..0)} < 2^{k+1} \tag{7}$$

$$X^{(k)} = \overline{X^{(l)}} \tag{8}$$

$$X^{(l-1..k+1)} = \overline{S^{(l-1..k+1)}} \tag{9}$$

then

$$(X + S) \oplus (X' + S) = e_{l-1} \oplus \ldots \oplus e_{k+1} \oplus e_k. \tag{10}$$

The proof is provided in Appendix B. To resolve the round 15 bit conflict explained above, the Lemma can be applied by letting $X = (W_{15}^1 \oplus W_{19}^1)$,

$X' = (W_{15}^3 \oplus W_{19}^3)$, $S = (FF_{15}(A_{15}^1, B_{15}^1, C_{15}^1) + SS2_{15}^1 + D_{15}^1)$, $k = 14$, $l = 22$, $n = 32$, where the active bit 31 in $(W_{15}^1 \oplus W_{19}^1)$ can be ignored since it is on the most significant position. Then, $X = (W_{15}^1 \oplus W_{19}^1)$ is modified to satisfy requirements (7)-(9). This message modification is done by only modifying a subset of bits $\{21, 20 \ldots, 0\}$ of W_{19}^1. Now, on the front face of the boomerang, A_{16}^1 and A_{16}^2 will have a signed difference of -22, while, at the back face of the boomerang, A_{16}^3 and A_{16}^4 will have the same signed difference, as specified by the forward differential, i.e., the conflicting bit condition has been bypassed. As can be verified, this is the only conflicting condition in rounds 15 and 16. The application of Lemma 1 can also be seen as a way to increase the probability of satisfying the paths in the boomerang by 2^8, since the event of the carry propagation takes place with probability of around 2^{-8}. As for the conflicting conditions in rounds $17 - 19$, they are resolved by repeated applications of Lemma 1. The result of the message modification procedure described above is a quartet that satisfies the differences in rounds $15-19$. The quartet is used as input for the next stage of the zero-sum search procedure.

4.3 Searching for the zero-sum

After the differences in rounds $15-19$ have been satisfied, the remaining paths in the boomerang are satisfied randomly. The corresponding search procedure is facilitated due to the existence of many neutral bits with respect to the majority of already satisfied conditions. Namely, all the bits in words $W_8, \ldots, W_{14}$ are neutral with respect to the rounds $15 - 19$. The search proceeds by randomly satisfying the remaining conditions, i.e., the path given in Table 1 and also the last round of the path in Table 2. Although the nominal probability is as low as $2^{2\times(25+2)} = 2^{-54}$, this probability does not take into account the alternative differential paths that are similar to the ones specified above. Due to these additional paths, the actual probability is much larger, as was confirmed by executing the search procedure. Similar observation have been reported in for example in [3, 10].

The zero-sum for 32 rounds of the SM3 compression function, given in Table 3, was found after around 20 days of computation using 4 workstations, each with four 2.4 GHz Dual-Core AMD Opteron processors.

A natural way to extend the attack to more rounds would be to increase the number of rounds on which the message modification is performed. Namely, it should be noted that the consequence of the neutral bits described above is the fact that the complexities to satisfy the bottom and the top differential add up and do not multiply, similar as in [2]. Thus, one can imagine extending the middle rounds so that the message modification procedure terminates after a practical amount of computation. However, caution should be exercised when estimating the number of rounds that can be added since the number of conflicting bits grows quickly which consequently increases the number of necessary applications of Lemma 1. According to our experience, satisfying the conditions in the boomerang becomes increasingly difficult without an automated systematic procedure as the number of conflicting conditions grows. Thus, extending the number of rounds to be dealt with by an automated message modification procedure is the goal of our future research. Such a procedure would also be relevant for the SHA-2 boomerang analysis [2].

5 A slide-rotational property of SM3-XOR

As mentioned in Section 1.1, the SHA2-XOR compression function was previously studied by Yoshida *et al.* [15]. In this section, we show that, in the case of the full SM3-XOR, pairs satisfying a certain rotational relation can be easily generated. An example of such a pair for the SM3-XOR is provided in Table 4. The possibility of practical generation of such evasive [5] SM3-XOR pairs demonstrates the existence of a non-trivial property which is not known to exist in SHA2-XOR.

The above mentioned property exists due to the fact that the constants over the 64 rounds of SM3 are related. According to the SM3 specification, in rounds $j \in \{0, \ldots, 15\}$, one constant rotated by j is utilized, whereas the other constant rotated by j is used in rounds $j \in \{16, \ldots, 63\}$. Since operations like XOR, FF_i, GG_i, $0 \leq i < 64$, that are used in the SM3-XOR round function preserve the rotational property, it is natural to attempt a rotational attack, as provided below. We note that if instead of SM3-XOR, the original SM3 compression function is used, the addition mod 2^{32} transforms the attack into a probabilistic one, as outlined below. Due to the high number of additions per round, it appears difficult to exploit this rotational property directly and therefore the security of the SM3 compression function, at this stage of analysis, does not seem to be directly affected.

Two 32-bit words X, Y are said to be *rotational* if $X = Y \lll n$. Let messages W and W^* satisfy $W_1^* = W_0 \lll 1, W_2^* = W_1 \lll 1, \ldots, W_{16}^* = W_{15} \lll 1$. Below, a procedure for the instant generation of pairs v, v^* such that

$$\begin{aligned} v_1^* = v_0 \lll 1, v_2^* = v_1 \lll 8, v_3^* = v_2 \lll 1 \\ v_5^* = v_4 \lll 1, v_6^* = v_5 \lll 18, v_7^* = v_6 \lll 1 \\ V_1^* = V_0 \lll 1, V_2^* = V_1 \lll 8, V_3^* = V_2 \lll 1 \\ V_5^* = V_4 \lll 1, V_6^* = V_5 \lll 18, V_7^* = V_6 \lll 1 \end{aligned} \tag{11}$$

is provided, where $V = \text{SM3-XOR}(v, W)$, $V^* = \text{SM3-XOR}(v^*, W^*)$ and v_i, V_i for $0 \leq i \leq 7$ denote i-th 32-bit word in the v and V, respectively. For a random function, a random (v, W), (v^*, W^*) satisfying the above constraints will yield the corresponding V and V^* with probability $2^{-6\times 32} = 2^{-192}$, since (11) imposes 6 32-bit conditions on V, V^*.

5.1 Constructing a slide-rotational pair

We start by the following observations:

- The slide rotational messages expand to slide-rotational expanded messages with probability 1. In particular, fix $W_0, \ldots, W_{15}$ and let

$$W_1^* = W_0 \lll 1, W_2^* = W_1 \lll 1, \ldots, W_{16}^* = W_{15} \lll 1 \tag{12}$$

  After expanding both W and W^*, we have $W_{i+1}^* = W_i \lll 1$, for $i = \{0, 1, \ldots, 62\}$ and also $W_{i+1}^{'*} = W_i' \lll 1$, for $i = \{0, 1, \ldots, 66\}$.
- We recall that T_i, $0 \leq i \leq 63$ are the round constants. If we have

$$W_{i+1}^* = W_i \lll 1, W_{i+1}^{'*} = W_i' \lll 1, T_{i+1} = T_i \lll 1 \tag{13}$$

$$A_{i+1}^* = A_i \lll 1, B_{i+1}^* = B_i \lll 1, \ldots, H_{i+1}^* = H_i \lll 1 \tag{14}$$

  for $i = k$, then (14) will also hold for $i = k + 1$, where $k = 0, \ldots, 62$.

The observations above suggest that sliding can be introduced, as depicted in Fig. 4. Namely, consider randomly initializing W and letting W^* satisfy

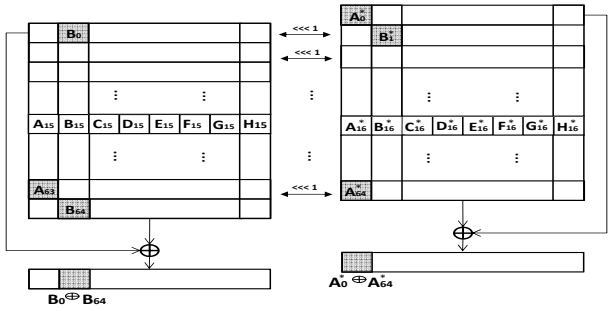

Fig. 4. The slide-rotational attack against SM3-XOR

(12). Moreover, $A_0, B_0 \ldots, H_0$ is chosen randomly and the inner state registers after the first round in the second instance of the hash function are initialized according to (14). Then, until round 15, due to (13), the rotational property in the inner state registers will be preserved. Once the two instances reach rounds 15 and 16, respectively, a different round transformation is applied in the two instances and the rotational property may discontinue. This problem is bypassed by starting from the middle, i.e., by populating the inner states entering the critical rounds 15 and 16 (see Fig. 4). The details of how the critical rounds are bypassed are provided in Appendix C and an example of a rotational pair satisfying (11) is given in Table 4.

When instead of SM3-XOR, the SM3 compression function is considered, this property turns into a probabilistic one. Following [7], if $p_r = P[(x \lll r)+(y < \ll r) = (x+y) \lll r]$ where x and y are 32-bit words, then $p_1 = 2^{-1.415}$. Since there exists 8 additions in one SM3 round, the probability that one round and its corresponding slided round will preserve the rotational property is given by $(p_1)^8 = 2^{-11.320}$ [7].

6 Conclusion

In this paper, a second order collision for the SM3 compression function reduced to 32 rounds is presented. The top and the bottom differentials, used in the boomerang, impose seemingly conflicting conditions. The novelty of our method is that these conditions are resolved during message modification by using long carry propagation on the left and right face of the boomerang. In the second part of the paper, a slide-rotational property of SM3-XOR function is exposed and an example of a slide-rotational pair for SM3-XOR compression function is given.

References

1. Aumasson, J.-P. Zero-sum distinguishers, Rump session talk at CHES, 2009. http://131002.net/data/talks/zerosum_rump.pdf.
2. Biryukov, A., Lamberger, M., Mendel, F., and Nikolić, I. Second-order differential collisions for reduced SHA-256. In *ASIACRYPT* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 270–287.
3. Biryukov, A., Nikolić, I., and Roy, A. Boomerang attacks on BLAKE-32. In *FSE* (2011), A. Joux, Ed., vol. 6733 of *Lecture Notes in Computer Science*, Springer, pp. 218–237.
4. Coron, J.-S., Patarin, J., and Seurin, Y. The random oracle model and the ideal cipher model are equivalent. In *CRYPTO* (2008), D. Wagner, Ed., vol. 5157 of *Lecture Notes in Computer Science*, Springer, pp. 1–20.
5. Gilbert, H., and Peyrin, T. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 365–383.
6. Internet Engineering Task Force. RFC: SM3 hash function, October, 2011. `tools.ietf.org/html/shen-sm3-hash-00`.
7. Khovratovich, D., and Nikolić, I. Rotational cryptanalysis of ARX. In *FSE* (2010), S. Hong and T. Iwata, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 333–346.
8. Khovratovich, D., Nikolić, I., and Rechberger, C. Rotational Rebound Attacks on Reduced Skein. In *ASIACRYPT* (2010), M. Abe, Ed., vol. 6477 of *Lecture Notes in Computer Science*, Springer, pp. 1–19.
9. Lai, X. Higher order derivatives and differential cryptanalysis. In Blahut, R., Costello Jr., D., Maurer, U., Mittelholzer, T. (eds.). *Communications and Cryptography* (1992), 227–233. Kluwer.
10. Leurent, G., and Roy, A. Boomerang attacks on hash function using auxiliary differentials. In *CT-RSA* (2012), O. Dunkelman, Ed., vol. 7178 of *Lecture Notes in Computer Science*, Springer, pp. 215–230.
11. Murphy, S. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory 57*, 4 (2011), 2517–2521.
12. Nad, T. The CodingTool Library, 2010. Available at: http://www.iaik.tugraz.at/content/research/krypto/codingtool/.
13. Sasaki, Y. Boomerang distinguishers on MD4-based hash functions: First practical results on full 5-pass HAVAL. In *Selected Areas of Cryptography* (2011), A. Miri and S. Vaudenay, Eds., Lecture Notes in Computer Science, Springer, *to appear*.
14. Wagner, D. The boomerang attack. In *FSE* (1999), L. R. Knudsen, Ed., vol. 1636 of *Lecture Notes in Computer Science*, Springer, pp. 156–170.
15. Yoshida, H., and Biryukov, A. Analysis of a SHA-256 variant. In *Selected Areas in Cryptography* (2005), B. Preneel and S. E. Tavares, Eds., vol. 3897 of *Lecture Notes in Computer Science*, Springer, pp. 245–260.

A Higher-order analysis of hash functions

Here, we review higher-order analysis of hash functions. In particular, the two equivalent notions of second order collisions and zero-sums are defined.

Then, the main idea of how to use the boomerang attack in the context of a compression function is provided. First, the notion of higher order differentials [9] is recalled.

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

$$\Delta_a f(x) = f(x+a) - f(x)$$

The i-th derivative of f at $(a_1, \ldots, a_i)$ is then defined by

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

In case of a function $f: F_2^m \to F_2^n$, we have (Proposition 3, [9]):

Lemma 2 *Let $L[a_1, \ldots, a_i]$ be the list of all 2^i possible linear combinations of $a_1, a_2, \ldots, a_i$. Then,*

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \bigoplus_{c \in L[a_1,\ldots,a_i]} f(x \oplus c)$$

As defined in [2], an i-th order differential collision for f is an i-tuple $(a_1, \ldots, a_i)$, together with a value x such that

$$\Delta_{(a_1,\ldots,a_i)} f(x) = 0$$

As argued in [2], since the $i+1$ input parameters $a_1, \ldots, a_i$ and x can be chosen freely, the query complexity of finding an i-th order collision is $2^{n/(i+1)}$, where n denotes the bit-size of the output of the function f. Here, the query complexity denotes the number of queries made to the f function oracle. Thus, the query complexity of finding a second order collision for the function f, i.e., values x, a_1 and a_2, such that

$$f(x \oplus a_1 \oplus a_2) \oplus f(x \oplus a_1) \oplus f(x \oplus a_2) \oplus f(x) = 0 \tag{15}$$

is $2^{n/3}$. As for the computational complexity, which would include evaluating f around $2^{n/3}$ times and finding, among the outputs, a quartet that sums to 0, no algorithm with complexity better than $2^{n/2}$ is known. It should be noted that in [13], the boomerang technique is used to find a zero-sum quartet of inputs x_0, x_1, x_2, x_3, introduced in [1], such that

$$\begin{aligned} x_0 \oplus x_1 \oplus x_2 \oplus x_3 &= 0 \\ f(x_0) \oplus f(x_1) \oplus f(x_2) \oplus f(x_3) &= 0 \end{aligned} \tag{16}$$

It is easy to verify that a zero-sum quartet and second order collision notions are equivalent. For example, given a zero-sum quartet, it suffices to put $x = x_0$, $a_1 = x_0 \oplus x_1$, $a_2 = x_0 \oplus x_2$ to have (15) satisfied.

B Proof of Lemma 1

Due to (7) and the fact that $X^{(k)} \neq X'^{(k)}$, exactly one of the values in $\{X^{(k..0)} + S^{(k..0)}, X'^{(k..0)} + S^{(k..0)}\}$ will have a carry propagation from bit position k to $k+1$. Therefore, using (9), it is clear that $(X+S)^{(l-1..k)} \oplus (X'+S)^{(l-1..k)} = e_{l-1} \oplus \ldots \oplus e_{k+1} \oplus e_k$. Since $X^{(l)} \neq X'^{(l)}$, $X+S$ and $X'+S$ will be equal on bit l. Finally, from (8), $(X+S)^{(m)} = (X'+S)^{(m)}$ for $m > l$, $m < n$, which completes the proof.

C Bypassing rounds 15 and 16

In this appendix, we explain how to bypass the critical rounds 15 and 16 which may discontinue the rotational property. That way, it is possible to have the slide property hold over all rounds of the two hash function instances. The idea is to start by populating the inner states entering the critical rounds 15 and 16 (see Fig. 4). In particular, a rotational pair $(A_{15}, \ldots, H_{15})$, $(A^*_{16}, \ldots, H^*_{16})$ is carefully chosen so that $(A_{16}, \ldots, H_{16})$ and $(A^*_{17}, \ldots, H^*_{17})$ satisfy relation (14). It should be noted that the rotational property may be destroyed only between A_{16} and A^*_{17} and between E_{16} and E^*_{17}, since the other registers go through identical rotational-preserving transformations in round 15 and round 16. As for A_{16} and A^*_{17}, for the purpose of tracking the possible rotational disturbance between the two registers, the equation to compute these two registers can be rewritten as

$$A_{16} = FF_{15}(A_{15}, B_{15}, C_{15}) \oplus (T_{15} \lll 22) \oplus \alpha \tag{17}$$

$$A^*_{17} = FF_{16}(A^*_{16}, B^*_{16}, C^*_{16}) \oplus (T_{16} \lll 23) \oplus \alpha^* \tag{18}$$

where $\alpha = D_{15} \oplus W_{15} \oplus W_{19} \oplus (((A_{15} \lll 12) \oplus E_{15}) \lll 7) \oplus (A_{15} \lll 12)$ and $\alpha^* = D^*_{16} \oplus W^*_{16} \oplus W^*_{20} \oplus (((A^*_{16} \lll 12) \oplus E^*_{16}) \lll 7) \oplus (A^*_{16} \lll 12)$. Since (14) and (13) hold for $i = 15$, $\alpha^* = \alpha \lll 1$. Therefore, to have A_{16} and A^*_{17} be a rotational pair, it suffices to make $FF_{15}(A_{15}, B_{15}, C_{15}) \oplus (T_{15} \lll 22)$ and $FF_{16}(A^*_{16}, B^*_{16}, C^*_{16}) \oplus (T_{16} \lll 23)$ satisfy the rotational property. After expressing A^*_{16}, B^*_{16}, C^*_{16} in terms of A_{15}, B_{15}, C_{15} and using that FF_{15} and FF_{16} preserve the rotational property, the condition can be expressed in terms of A_{15}, B_{15}, C_{15} as follows:

$$FF_{15}(A_{15}, B_{15}, C_{15}) \oplus FF_{16}(A_{15}, B_{15}, C_{15}) = (T_{15} \oplus T_{16}) \lll 22 \tag{19}$$

When applied on 1-bit values X, Y and Z, the equation $FF_{15}(X, Y, Z) \oplus FF_{16}(X, Y, Z) = 0$ is satisfied for 2 out of 8 (X, Y, Z) values. Since the Hamming weight of the right-hand side of (19) is equal to 14, the number of solutions to the equation is $2^{18} \times 6^{14} = 2^{32} \times 3^{14}$. As for preserving the rotational property between E_{16} and E^*_{17}, developing the registers as in (17) and then forming the equation of the form (19) yields that the number of solutions E_{15}, F_{15} and G_{15} is $4^{32} = 2^{64}$. Therefore, the number of solutions for $(A_{15}, \ldots, H_{15})$ that pass the disturbance in rounds 15 and 16 is $2^{32} \times 3^{14} \times 2^{64} \times 2^{64} \approx 2^{182.19}$, since D_{15} and H_{15} are free variables. For such pairs, it follows that relations (11) are satisfied.

i	Inner state	$W_i \oplus W_{i+4}$	W_i	Prob
0	A: +22 B: −22 C: −31, −22 D: −31, +29, +27, +22, +21, +20, +11, +9, −6, −4, +3, −2 E: +12 F: −12 G: −31, +12 H: +31, −29, −27, +21, +20, +12, −11, −9, +6, −4, +3	−29, +27, +21, +20, +19, −11, −6, +4, −3	−29, +27, +21, +20 −19, +11, +6, −4, −3	2^{-23}
1	B: +22 C: −31 D: −31, −22 F: +12 G: −31 H: −31, +12			2^{-2}
2	C: +31 D: −31 G: +31 H: −31			1
3	D: +31 H: +31	-31	-31	1
4				
. . .	. . .	. . .	. . .	. . .
15				

Table 1. Backward differential path with probability 2^{-25}

i	Inner state	$W_i \oplus W_{i+4}$	W_i	Prob
15	C: +29, +27, +21, +20, −11, +9, −6, −4, −3, −2 D: −31, +29, +27, −22, +21, +20, +11, −9, +6, +4, +3, −2 G: +29, −27, −21, +20, −11, +9, −6, −4, −3, H: +31, −30, +27, +23, +20, −15, +14, −12, +11, −9, −7, +6, +4, +3	+31		2^{-26}
16	A: −22 D: +29, +27, +21, +20, −11, +9, −6, −4, −3, −2 E: +12 H: +29, −27, −21, +20, −11, +9, −6, −4, −3	−29, +27, −21, −20, +19, +11, +6, +4, +3	−29, +27, +21, −20, +19, +11, +6, +4, +3	2^{-25}
17	B: −22 F: +12			2^{-2}
18	C: −31 G: +31			2^{-2}
19	D: −31 H: +31	−31	−31	1
20				
. . .	. . .	. . .	. . .	. . .
31		−31, −22, +14		2^{-2}
32	A: −31, −22, +14			

Table 2. Forward differential path with probability 2^{-57}

$A^1, B^1, \ldots, H^1$	$0x7a0d7b2f$ $0x776a25d5$ $0xcff768ac$ $0xd2eb20d5$ $0xd2c08d9b$ $0x744d3e5c$ $0xdf04e2ba$ $0x2cd3bb94$
$W_0^1, \ldots, W_{15}^1$	$0x31c2ba4f$ $0x336fa0d6$ $0x94a32431$ $0x9d3caeaa$ $0x814d29d5$ $0xc8ebf2e6$ $0x7a41c51f$ $0x3aa0bedd$ $0xaac4fb81$ $0xd584f8b$ $0x619690c2$ $0xfac9a4d1$ $0x2a28a333$ $0x175fb61c$ $0x6133d9ab$ $0x81e48a5e$
$A^2, B^2, \ldots, H^2$	$0x3c6a7d6d$ $0xbbdf98c0$ $0x5da6c569$ $0x89c62255$ $0xd75bec0e$ $0x6117de9c$ $0xb3f56bd3$ $0x3445d8b4$
$W_0^2, \ldots, W_{15}^2$	$0x2dc2be4f$ $0x33efa256$ $0xbc9b2c69$ $0x3d6cfeaa$ $0x3cea950$ $0x48ebf2e6$ $0x7241ad1f$ $0x32a0bedd$ $0xaac4fb81$ $0xbd083f9b$ $0x618698ca$ $0xfac9a4d1$ $0xaa28a333$ $0x1f5f9e1c$ $0x6133d9ab$ $0x81e48a5e$
$A^3, B^3, \ldots, H^3$	$0x7a4d7b2f$ $0x772a25d5$ $0x4fb768ac$ $0x6a7b1aa9$ $0xd2c09d9b$ $0x744d2e5c$ $0x5f04d2ba$ $0xc493b1cc$
$W_0^3, \ldots, W_{15}^3$	$0x19fab217$ $0x336fa0d6$ $0x94a32431$ $0x1d3caeaa$ $0x814d29d5$ $0xc8ebf2e6$ $0x7a41c51f$ $0x3aa0bedd$ $0xaac4fb81$ $0xd584f8b$ $0x619690c2$ $0xfac9a4d1$ $0x2a28a333$ $0x175fb61c$ $0x6133d9ab$ $0x81e48a5e$
$A^4, B^4, \ldots, H^4$	$0x3c2a7d6d$ $0xbb9f98c0$ $0xdde6c569$ $0x31561829$ $0xd75bfc0e$ $0x6117ce9c$ $0x33f55bd3$ $0xdc05d2ec$
$W_0^4, \ldots, W_{15}^4$	$0x5fab617$ $0x33efa256$ $0xbc9b2c69$ $0xbd6cfeaa$ $0x3cea950$ $0x48ebf2e6$ $0x7241ad1f$ $0x32a0bedd$ $0xaac4fb81$ $0xbd083f9b$ $0x618698ca$ $0xfac9a4d1$ $0xaa28a333$ $0x1f5f9e1c$ $0x6133d9ab$ $0x81e48a5e$

Table 3. An example for a zero-sum for 32 rounds of the SM3 compression function

$A^1, B^1, \ldots, H^1$	$0x565060b7$ $0x125d5655$ $0x285c7653$ $0xeaf5fe1e$ $0xda8bd7dd$ $0xb8bb1904$ $0x43bcaf18$ $0x7cf88895$
$W_0^1, \ldots, W_{15}^1$	$0x8f450bbd$ $0x4a0c9922$ $0x73dd44f8$ $0x9eceaaf8$ $0x33b13e20$ $0xb59d9c33$ $0x6b5a5f23$ $0xc0d2b468$ $0x7a9a1e16$ $0xaff62878$ $0x3fbb01f4$ $0x75278787$ $0xac0b849e$ $0x498f3045$ $0x62687c15$ $0xd3498eb$
$A^2, B^2, \ldots, H^2$	$0x24baacaa$ $0x53285c76$ $0xd5ebfc3d$ $0xdf1ee2a6$ $0x71763209$ $0x2bc610ef$ $0xf9f1112a$ $0xffeb86a4$
$W_0^2, \ldots, W_{15}^2$	$0x7efa7542$ $0x1e8a177b$ $0x94193244$ $0xe7ba89f0$ $0x3d9d55f1$ $0x67627c40$ $0x6b3b3867$ $0xd6b4be46$ $0x81a568d1$ $0xf5343c2c$ $0x5fec50f1$ $0x7f7603e8$ $0xea4f0f0e$ $0x5817093d$ $0x931e608a$ $0xc4d0f82a$

Table 4. An example for a slide-rotational pair for the SM3-XOR compression function