Finding Collisions for Round-Reduced SM3

Florian Mendel, Tomislav Nad, and Martin Schläffer

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 work, we provide the first security analysis of reduced SM3 regarding its collision resistance. SM3 is a Chinese hash function standard published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems and hence, might be used in several cryptographic applications in China. So far only few results have been published for the SM3 hash function. Since the design of SM3 is very similar to the MD4 family of hash functions and in particular to SHA-2, a revaluation of the security of SM3 regarding collision resistance is important taking into account recent advances in the cryptanalysis of SHA-2. In this paper, we extend the methods used in the recent collision attacks on SHA-2 and show how the techniques can be effectively applied to SM3. Our results are a collision attack on the hash function for 20 out of 64 steps and a free-start collision attack for 24 steps of SM3, both with practical complexity.

Keywords: hash functions, cryptanalysis, collisions, free-start collisions.

1 Introduction

A cryptographic hash function H maps a message M of arbitrary length to a fixed-length hash value h. Informally, a cryptographic hash function has to fulfill the following security requirements:

- *Collision resistance:* it is practically infeasible to find two messages M and M^*, with $M^* \neq M$, such that $H(M) = H(M^*)$.
- *Second preimage resistance:* for a given message M, it is practically infeasible to find a second message $M^* \neq M$ such that $H(M) = H(M^*)$.
- *Preimage resistance:* for a given hash value h, it is practically infeasible to find a message M such that $H(M) = h$.

The resistance of a hash function to collision and (second) preimage attacks depends in the first place on the length n of the hash value. Regardless of how a hash function is designed, an adversary will always be able to find preimages or second preimages after trying out about 2^n different messages. Finding collisions requires a much smaller number of trials: about $2^{n/2}$ due to the birthday paradox. If the internal structure of a particular hash function allows collisions or (second) preimages to be found more efficiently than what could be expected based on its

E. Dawson (Ed.): RSA 2013, LNCS 7779, pp. 174–188, 2013.

hash length, then the function is considered to be broken. For a formal treatment of the security properties of cryptographic hash functions we refer to [10, 11].

Most cryptanalytic results on hash functions focus on collision attacks. In the last years collisions have been shown for many commonly used hash functions. In particular, the collision attacks of Wang et al. [13, 14] on MD5 and SHA-1 have convinced many cryptographers that these widely deployed hash functions can no longer be considered secure. As a consequence, NIST proposed the transition from SHA-1 to the SHA-2 family and many companies and organization are migrating to SHA-2. Furthermore, researchers are evaluating alternative hash functions in the SHA-3 initiative organized by NIST [9] to find a new hash function standard.

In this work, we analyze the Chinese hash function standard SM3. SM3 was designed by Wang et al. [1] and is published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems. The amount of cryptanalytic results on SM3 is low compared to other hash function standards. Kircanski et al. [4] presented a distinguisher for the compression function of SM3 up to 35 steps with complexity $2^{117.1}$. Moreover, Zou et al. [15] presented a preimage attack on 30 steps of SM3 with complexity of 2^{249}.

The design of SM3 is very similar to the MD4 family in particular SHA-2. New collision attacks on SHA-2 and similar hash functions have been shown [2, 5–8] recently. The attacks have in common that they are all of practical complexity and are based on automatic search algorithms to find complex differential characteristics.

In this paper, we develop the methods by Mendel et al. for SHA-256 [7] further and apply them on SM3. We show how the technique can be effectively applied to SM3. Furthermore, we present a collision for 20 steps and a free-start collision for 24 steps of SM3. These are the first collision attacks on the step-reduced SM3 hash and compression function.

The remainder of this paper is structured as follows. A description of the hash function is given in Section 2. In Section 3 we describe the basic attack strategy. In Section 4 we show how we can find differential characteristics and conforming message pairs for SM3. Finally, we present a collision and free-start-collision for step-reduced SM3 in Section 5 and conclude in Section 6.

2 Description of SM3

SM3 is an iterated hash function that processes 512-bit input message blocks and produces a 256-bit hash value. In the following, we briefly describe the hash function. It basically consists of two parts: the message expansion and the state update transformation. A detailed description of the hash function is given in [1].

2.1 Message Expansion

The message expansion of SM3 is linear in $GF(2)$. It splits the 512-bit message block into 16 words M_i, $i = 0, \ldots, 15$, and expands them into 68 expanded message words W_i and 64 expanded message words W_i' as follows:

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_0(W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15) \oplus W_{i-13} \lll 7 \oplus W_{i-6} & 16 \le i < 68 \end{cases}$$

and

$$W'_i = W_i \oplus W_{i+4} \quad 0 \le i < 64 \ .$$

The functions $\sigma_0(X)$ is given by

$$\sigma_0(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$$

2.2 State Update Transformation

The state update transformation starts from a (fixed) initial value IV of eight 32-bit words and updates them in 64 steps. In each step the 32-bit words W_i and W'_i are used to update the eight state variables $A_{i-1}, B_{i-1}, \ldots, H_{i-1}$.

$$\begin{aligned} T_1 &= (A_{i-1} \lll 12 + E_{i-1} + K_i) \lll 7 \\ T_2 &= H_{i-1} + f_0(E_{i-1}, F_{i-1}, G_{i-1}) + T_1 + W_i \\ A_i &= D_{i-1} + f_1(A_{i-1}, B_{i-1}, C_{i-1}) + (T_1 \oplus A_{i-1} \lll 12) + W'_i \\ E_i &= \Sigma_0(T_2) \\ B_i &= A_{i-1} \\ C_i &= B_{i-1} \lll 9 \\ D_i &= C_{i-1} \\ F_i &= E_{i-1} \\ G_i &= F_{i-1} \lll 19 \\ H_i &= G_{i-1} \end{aligned} \tag{1}$$

For the definition of the step constants K_i we refer to [1]. The bitwise Boolean functions f_0 and f_1 are different for each step. In the first 16 steps f_{XOR} is used for both f_0 and f_1. After step 16 f_0 is f_{IF} and f_1 is f_{MAJ}.

$$\begin{aligned} f_{XOR}(X,Y,Z) &= X \oplus Y \oplus Z \\ f_{IF}(X,Y,Z) &= XY \oplus XZ \oplus Z \\ f_{MAJ}(X,Y,Z) &= XY \oplus YZ \oplus XZ \end{aligned} \tag{2}$$

The linear function Σ_0 is defined as follows:

$$\Sigma_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17) \tag{3}$$

After the last step of the state update transformation, the initial values are XORed to the output values of the last four steps (Davies-Meyer construction). The result is the final hash value or the initial value for the next message block.

3 Basic Attack Strategy

In the following, we first give a brief overview of the attack strategy used in the recent collision attacks on the MD4-family of hash functions [12, 14]. The high-level strategy can be summarized as follows:

1. Find a characteristic for the hash function that holds with high probability.
2. Use message modification techniques to fulfill conditions imposed by the characteristic. This increases the probability of the characteristic.
3. Use random trials to find values for the remaining free message bits such that the message follows the characteristic.

The most difficult and important part of the attack is to find a good differential characteristic. The second important part of the attack is to find conforming inputs for the differential characteristic. For both parts we used the technique of the recent attack on SHA-2 [7].

4 Automatic Search Tool

The collision attack on SHA-2 [7] can be summarized as follows:

1. Determine a starting point for the search which results in an attack on a large number of steps. The resulting start characteristic should span over few steps and only some message words should contain differences.
2. Use an automated search tool to find a differential characteristic for the unrestricted intermediate steps including the message expansion.
3. Continue the search to find a conforming message pair. If no message pair can be found, adjust the differential characteristic accordingly.

Due to the linearity of the message expansion, finding a good starting point is rather simple. The most difficult and important part of the attack is to find a good differential characteristic. Due to the increased complexity of SM3 compared to hash functions like SHA-1 and MD5, finding good differential characteristics by hand is almost impossible. Therefore, we use an automatic tool to find complex nonlinear differential characteristics. The tool is also used for solving nonlinear equations involving conditions on state words and free message bits, i.e. to find confirming message pairs. The tool is based on the approach of Mendel et al. [7] to find nonlinear differential characteristics and conforming message pairs for SHA-2.

4.1 Generalized Conditions

The tool and search algorithm is based on the concept of generalized conditions introduced in [2]. Generalized conditions are inspired by signed-bit differences and take all 16 possible conditions on a pair of bits into account. Table 1 lists all these possible conditions and introduces the notation for the various cases.

Using these generalized conditions and propagating them in a bitsliced manner, we can construct complex differential characteristics in an efficient way. The basic idea of the search algorithm is to randomly pick a bit from a set of bit positions with predefined conditions, impose a more restricted condition and compute how this new condition propagates. This is repeated until an inconsistency is found or all unrestricted bits from the set are eliminated. Note that this

Table 1. Notation for possible generalized conditions on a pair of bits [2]

(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$	(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

general approach can be used for both, finding differential characteristics and conforming message pairs. There are three important aspects of the automated tool: using a good starting point, using an efficient condition propagation and using a sophisticated search strategy. We discuss each aspect in the following sections.

4.2 Defining a Starting Point

Similar to the attack on SHA-256 [7] we construct a local collision with differences in a few steps which results in a attack on a large number of steps. Since the message expansion of SM3 is linear finding a starting point is easier than for SHA-256.

To find a good starting point for SM3, a system of linear equations representing the message expansion is constructed. Afterwards, linear constraints are added and the system is solved. In that way several good starting points for up to 24 steps have been found. The starting points for 20 and 24 steps are given in the Appendix in Table 5 and Table 7.

Note that unlike in most hash function attacks so far, the non-linear part for 24 steps is placed at the end instead of the beginning. This has several reasons. First of all the differential characteristic for the message expansion is more sparse. Furthermore, after step 16 the Boolean function IF and MAJ instead of XOR are used. This again results in more sparse characteristic. In general the more sparse a characteristic the easier it is to find conforming message pairs.

4.3 Efficient Condition Propagation

The efficient propagation of new conditions is crucial for the performance of the algorithm, since it is the most often needed operation in the search algorithm. Due to the nature of the search algorithm where changes to the characteristic (using generalized conditions) are done on bit-level, we perform the propagation of conditions also on bit-level. At the beginning of the search every bit has at least one of the 16 generalized conditions (see Table 1). During the search we impose conditions on specific bits. These bits are inputs or outputs of functions. If a bit in the output is changed then all bits which are used to determine this

output bit are updated. We call such a set of bits a bit-slice. If the changed bit is an input of a function then all other bits of the corresponding bit-slice are updated. The following example illustrates this process.

Example 1 (Condition Propagation). Let $f : \mathbb{F}_{32}^3 \to \mathbb{F}_{32}$ be the Boolean IF function operating on 32-bit words and defined as follows:

$$f(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) = o.$$

Then the output bit o_i depends on the bits $\{x_i, y_i, z_i\}$ and $\{x_i, y_i, z_i, o_i\}$ forms a bit-slice. If the generalized condition ∇x_i changes then the conditions of the set $\{\nabla x_i, \nabla y_i, \nabla z_i, \nabla o_i\}$ are updated.

In our approach the update process is done exhaustively by computing all possible conditions of a bit-slice. This seems at first to be inefficient but we are using two techniques to significantly speed up the process. The first one splits the state update in smaller functions and the second one utilizes a cache. However, the update process for a modular addition is done in a slightly different way. The bit-slices of a modular addition contain also input carry and output carry. Hence, the bit-slices are connected through the carry bits. If the condition for a carry bit changes, then the connected bit-slice is updated as well. Furthermore, the whole update process is iterative and updates bits until conditions do not change any more.

4.4 Increasing the Propagation Performance

In the state update transformation of SM3, only two state variables are updated in each step, namely A_i and E_i. Therefore, we can redefine the state update such that only these two variables are involved. In this case, we get the following mapping between the original and new state variables:

A_i	B_i	C_i	D_i	E_i	F_i	G_i	H_i
A_i	A_{i-1}	A_{i-2}	A_{i-3}	E_i	E_{i-1}	E_{i-2}	E_{i-3}

Hence, only two state variables need to be stored. Furthermore, the complexity of propagating generalized conditions increases exponentially with the number of input bits and additions. Similar as in SHA-2 the number of input bits in the update of A_i,E_i and W_i is high. To reduce the computational complexity of the propagation, we further split the update of W_i, E_i and A_i into sub-steps. The decision where to split need to be done carefully. If too many sub-steps are introduced we are losing too much information resulting in a worse propagation and late detection of contradictions. If too few sub-steps are introduced the performance of the propagation is too slow. Therefore, we found the following separation of the SM3 state update which leads to a good performance/propagation ratio:

$$
\begin{aligned}
S_i &= W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15,\\
P_i &= S_i \oplus S_i \lll 15 \oplus S_i \lll 23,\\
W_i &= P_i \oplus W_{i-13} \lll 7 \oplus W_{i-6},\\
W'_i &= W_i \oplus W_{i+4},\\
L_i &= A_{i-1} \lll 12 + E_{i-1} + K_i \lll 7,\\
F_i &= A_{i-1} \oplus A_{i-2} \oplus A_{i-3} \lll 9,\\
A_i &= F_i + W'_i + A_{i-4} \lll 9 + (L_i \lll 7 \oplus A_{i-1} \lll 12),\\
G_i &= E_{i-1} \oplus E_{i-2} \oplus E_{i-3} \lll 19,\\
R_i &= E_{i-4} \lll 19 + L_i \lll 7 + W_i + G_i,\\
E_i &= R_i \oplus R_i \lll 9 \oplus R_i \lll 17.
\end{aligned}
$$

By carefully analyzing the state update and message expansion we have split up the computations such that one step does not have more than 5 inputs. Using this representation of SM3 we can use a cache during the propagation efficiently. Furthermore, for those steps with only three inputs we are able to compute all possibilities beforehand, changing the propagation of this steps to a simple table lookup.

4.5 Search Strategy

To reduce the complexity of the system and eventually find a solution, random additional conditions are introduced. In other words, some variables are guessed. Even the most efficient method to propagate information may not result in a solution if we make poor guesses. We need a guessing strategy, which can efficiently use the new information generated by the propagation of information introduced by previous guesses. The goal of a good guessing strategy is to discard invalid solutions and to find a valid solution as soon as possible. The guessing strategy depends in first place on the shape of the equations and the storeable information propagated. Furthermore, external knowledge of the structure of the attacked cryptographic system can help to improve the guessing strategy.

Our guessing strategy is similar to the one used by Mendel et al. in the attack on SHA-256 reduced to 32 steps [7]. However, there are some small but important modifications. In our approach for SM3 we further refine the search strategy for SM3 by considering specific output words for guessing. As in the attack of [7], our search strategy consists of several stages and each stage can basically be divided into three parts: decision, deduction and backtracking. Note that the same separation is done in many other fields, like SAT solvers [3]. In the decision part, we decide which bit is chosen and which constraints are imposed at its position. In the deduction part we compute the propagation of the new information and check for consistence. In the case of an inconsistency we need to backtrack and undo previous decisions, which is the third and last part.

Let U be a set of generalized conditions. Repeat the following until U is empty:

Decision

1. Pick according to some heuristic (or randomly) a bit in U.
2. Impose new constraints on this bit according to Table 2.

Deduction

3. Propagate the Information to the other variables and equations as described in Section 4.3.
4. If an inconsistency is detected start backtracking, else continue with step 1.

Backtracking

5. Try the second choice for the decision bit.
6. If this still results in an inconsistency mark this bit as critical.
7. Jump back until the critical bit can be resolved.
8. Continue with step 1.

Note that in each stage different bits are chosen (guessed). In total we have two stages which can be summarized as follows.

Stage 1: In the first stage we search for a consistent differential characteristic in the state words. Therefore, we add all unconstrained bits of A_i andE_i that are `?` or `x` to the set U. Furthermore, we add bits of L_i as well to U. Experience has shown that guessing the output of modular additions first provides a significant speed up. Due to the additional freedom added by the carry bits, the propagation of conditions is slow towards the output of modular additions if these bits are not included in U.

Stage 2: In the second stage we search for conforming inputs. Therefore, we pick decision bits with many two-bit conditions, since this ensures that bits which influence a lot of other bits are guessed first. Furthermore, many other bits propagate by defining the value of a single bit. Hence, this way inconsistent characteristics are discarded earlier and valid solutions are found faster. The concept of two-bit conditions was introduced in [7] .

Note that we dynamically switch between the two stages. Additionally, we restart the search from scratch after a certain amount of inconsistencies to terminate branches which appear to be stuck because of exploring a search space far from a solution.

Table 2. Decision rules of our guessing strategy with $r \in \{0, 1\}$ a random value

Decision bit	r	Choice 1	Choice 2
`?`	1,0	`-`	`x`
`x`	0	`u`	`n`
	1	`n`	`u`
`-`	0	`0`	`1`
	1	`1`	`0`

5 Results for Reduced SM3

To find collisions for reduced SM3 we apply the techniques described in Section 4. We first construct a differential characteristic with low Hamming weight in the message expansion which functions as starting point for the automatic search algorithm. Next the search algorithm is applied. Running on a cluster with 72 nodes, the algorithm finds a differential characteristic in less than 1 hour. Afterwards, we continue the search for a conforming message pair which can be found in several seconds.

5.1 Collision Attack

Using the starting point given in Table 5 and our automatic search algorithm described in Section 4, we are able to construct collisions for up to 20 steps of SM3. The differential characteristic is given Table 6. In Table 3 we present colliding message pairs. Note that we have used an additional first message block to generate several different initial values for the second message block. These degrees of freedom were needed for the attack to work, otherwise we could not find a confirming message pair.

Table 3. Collision for 20 steps of SM3

h_0	7380166f 4914b2b9 172442d7 da8a0600 a96f30bc 163138aa e38dee4d b0fb0e4e
m_0	03c98f41 a8bda164 709a299c d76610eb 26b351ac 53547024 8efdff59 7e818400 4188b7f1 954faf0e a32f9984 6e5d8975 dc3b528a 973480e4 f6be9d9b cf07f13e
h_1	5e801aac 4b8c7a46 c8f34646 3b2420c1 97e775ae e3a6c399 83a05d40 6a257995
m_1	559654cd 8d4f9e94 8ca64e4a b85d989c 8c185880 b51caad1 03eca739 be66a265 ca21ab71 9c341028 2c043967 d4617038 bf6744ca d8772f12 a58e12c0 35f4f9f2
m_1^*	559654cd 8d1f9e84 8ca64e5a b85d989c 8c185880 950cbad1 03eca739 be66a265 ca71ab61 9c341028 2c043967 d4617038 bf6744ca d8772f12 a58e12c0 35f4f9f2
Δm_1	00000000 00500010 00000010 00000000 00000000 20101000 00000000 00000000 00500010 00000000 00000000 00000000 00000000 00000000 00000000 00000000
h_2	b2033829 677c16d2 a6de9db9 fd898668 a9119d20 476364d6 a0838adc 08d3833d

5.2 Free-Start Collision

Using the starting point given in Table 7 and our automatic search algorithm described in Section 4, we are able to construct free-start collisions for up to 24 steps of SM3. The differential characteristic is given in Table 8. In Table 4 we present a colliding message pair and IV pair resulting in a collision after 24 steps. Again this attack has practical complexity. The main difference in this attack to previous attack is, that we had to place the non-linear part at the end to get sparse characteristics.

Table 4. Free-Start-Collision for 24 steps of SM3

h_0	898991b0 8de47668 6e54847c 9167ff5e 3c7d51fe e2101301 6c53d522 7b3809df
h_0^*	898991b0 8da47668 ee54847c 1127ff5e 3c7d51fe e2100301 ec53d522 fb3819df
Δh_0	00000000 00400000 80000000 80400000 00000000 00001000 80000000 80001000
m	07595c54 e01e0245 facd449a 07ca096d 510445e8 4e1d0dff 97f2a3c0 79f02f14 2ebfac50 48cdde2d e88f68e1 2b5032d1 3aa9a79f 656d1380 693417c1 ce82a62a
m^*	07595c54 e01e0245 7acd449a 07ca096d 510445e8 4e1d0dff 97f2a3c0 79f02f14 2ebfac50 48cdde2d e88f68e1 2b5032d1 3aa9a79f 656d1380 693417c1 ce82a62a
Δm	00000000 00000000 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
h_1	a83d45dc 9eda869d 48baa718 78ccd026 25696682 d0be9f4a e70babf3 f4852e70

6 Conclusions

Since the collision attacks of Wang et al. [13, 14] on MD5 and SHA-1 many cryptographers are convinced that these widely deployed hash functions can no longer be considered secure. As a consequence, NIST proposed the transition from SHA-1 to the SHA-2 family and many companies and organization are migrating to SHA-2. Furthermore, researchers are evaluating alternative hash functions in the SHA-3 initiative. In this work, we analyze the Chinese hash function standard SM3. SM3 was designed by Wang et al [1] and is published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems. The amount of cryptanalytic results on SM3 is low compared to other hash function standards.

The design of SM3 is very similar to the MD4 family, in particular to SHA-256. Since new collision attacks on SHA-256 and similar hash functions have been shown recently, a revaluation of the security of similar hash functions such as SM3 seems to be necessary. The attacks are based on a the concept of generalized conditions and an automatic search algorithm. Recently, Mendel et al. improved and extended the technique such that it can be applied on more complex ARX based hash function. In this paper we develop the methods by Mendel et al. for SHA-256 [7] further and apply them on SM3. We show how the technique can be effectively applied to SM3. Furthermore, we present a collision for 20 steps and a free-start collision for 24 steps of SM3. These are the first collision attacks on step-reduced SM3 and both attacks have practical complexity.

Acknowledgments. Part of this work was done while Florian Mendel was with KU Leuven. The work has been supported in part by the Austrian Science Fund (FWF), project P21936-N23 and by the European Commission under contract ICT-2007-216646 (ECRYPT II).

Table 5. Starting point for a collision for 20 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	----------	----------	----------	----------
-3	----------	----------	----------	----------
-2	----------	----------	----------	----------
-1	----------	----------	----------	----------
0	----------	----------	----------	----------
1	??????????	??????????	-----x----	--x-x-x---
2	??????????	??????????	----x-----	---x-x----
3	??????????	??????????	----------	----------
4	??????????	??????????	----------	---x-x----
5	----------	----------	--x---x---	--x-x-x---
6	----------	----------	----------	----------
7	----------	----------	----------	----------
8	----------	----------	-x-x------	---x-x-x--
9	----------	----------	----------	----------
10	----------	----------	----------	----------
11	----------	----------	----------	----------
12	----------	----------	----------	----------
13	----------	----------	----------	----------
14	----------	----------	----------	----------
15	----------	----------	----------	----------
16	----------	----------	----------	----------
17	----------	----------	----------	----------
18	----------	----------	----------	----------
19	----------	----------	----------	----------

Table 6. Differential characteristic for a collision for 20 steps of SM3

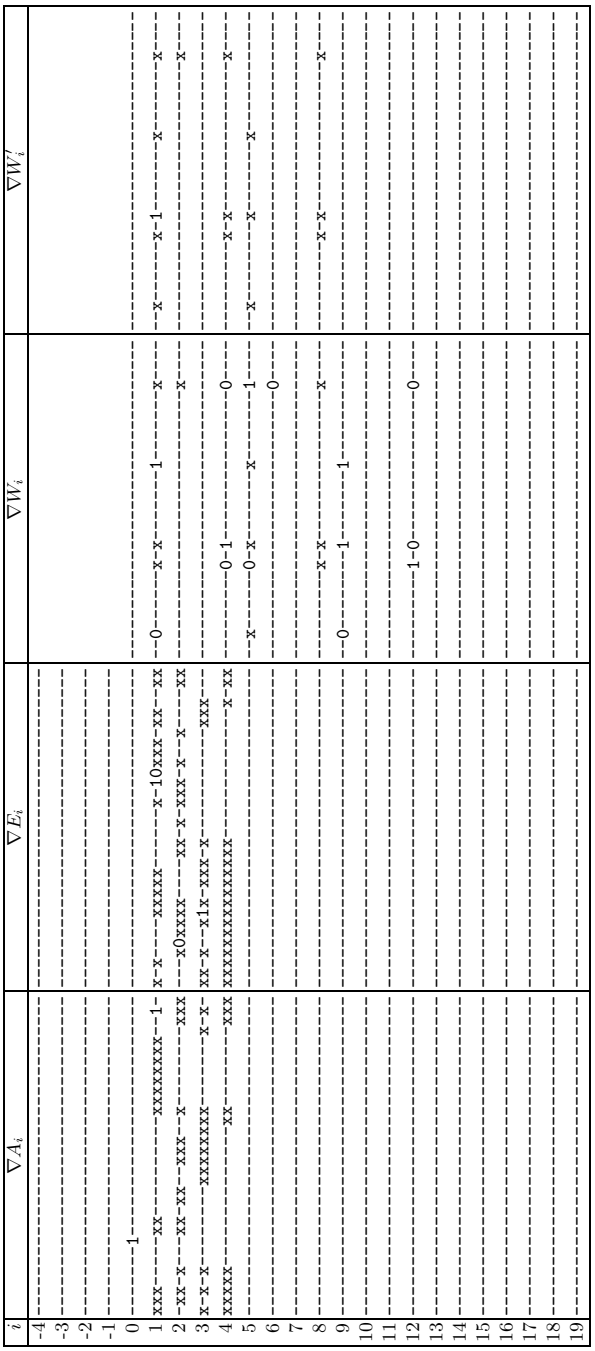
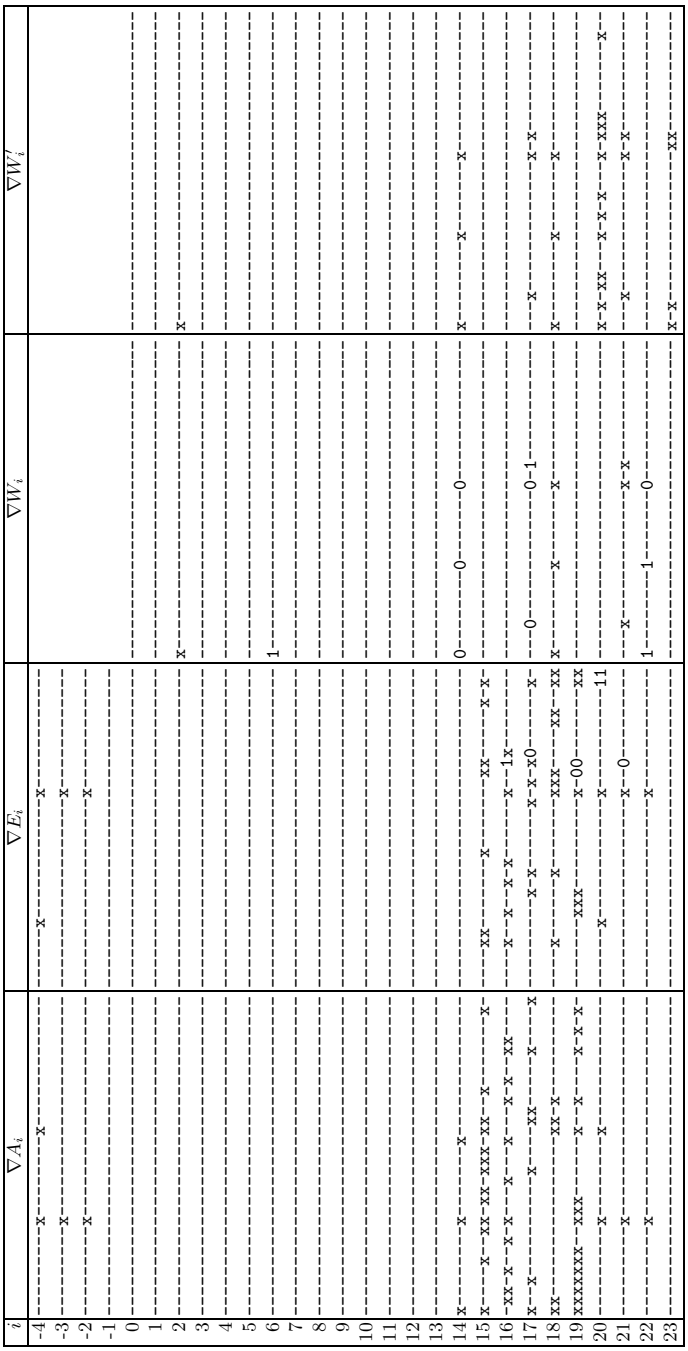


Table 7. Starting point for a free-start collision for 24 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	----------x----------x----------x----------	----------x----------x----------		
-3	----------x----------x----------	----------x----------x----------		
-2	----------x----------x----------	----------x----------x----------		
-1	----------	----------		
0	----------	----------	----------	----------
1	----------	----------	----------	----------
2	----------	----------x----------	----------	----------x----------
3	----------	----------	----------	----------
4	----------	----------	----------	----------
5	----------	----------	----------	----------
6	----------	----------	----------	----------
7	----------	----------	----------	----------
8	----------	----------	----------	----------
9	----------	----------	----------	----------
10	----------	----------	----------	----------
11	----------	----------	----------	----------
12	----------	----------	----------	----------
13	----------	----------	----------	----------
14	x---------x---------x----------	----------	----------	x---------x----------
15	??????????????????????????????	??????????????????????????????	----------	----------
16	??????????????????????????????	??????????????????????????????	----------	----------
17	??????????????????????????????	??????????????????????????????	----------x----------	----------x----------
18	??????????????????????????????	??????????????????????????????	x---------x---------x----------	x---------x----------
19	??????????????????????????????	??????????????????????????????	----------	----------
20	----------x----------x----------	----------x----------x----------	x-x-xx-x-x-x-xxx-x-x-xxx-x----------	----------
21	----------x----------	----------x----------	----------x----------	----------x----------
22	----------x----------	----------x----------	----------	----------
23	----------	----------	x-x----------	----------

Table 8. Differential characteristic for a free-start collision for 24 steps of SM3



References

1. Specification of SM3 cryptographic hash function (in Chinese), http://www.oscca.gov.cn/UpFile/20101222141857786.pdf
2. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
3. Gu, J., Purdom, P.W., Franco, J., Wah, B.W.: Algorithms for the Satisfiability (SAT) Problem: A Survey. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 19–152. American Mathematical Society (1996)
4. Kircanski, A., Shen, Y., Wang, G., Youssef, A.: Boomerang and Slide-Rotational Analysis of the SM3 Hash Function. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. LNCS. Springer (to appear, 2012)
5. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential Attacks on Reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 23–38. Springer, Heidelberg (2012)
6. Mendel, F., Nad, T., Schläffer, M.: Cryptanalysis of Round-Reduced HAS-160. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 33–47. Springer, Heidelberg (2012)
7. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
8. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012)
9. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
10. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
11. Stinson, D.R.: Some Observations on the Theory of Cryptographic Hash Functions. Des. Codes Cryptography 38(2), 259–277 (2006)
12. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
13. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
14. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
15. Zou, J., Wu, W., Wu, S., Su, B., Dong, L.: Preimage Attacks on Step-Reduced SM3 Hash Function. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 375–390. Springer, Heidelberg (2012)