Collisions and other Non-Random Properties for Step-Reduced SHA-256*

Sebastiaan Indesteege[1,2,**], Florian Mendel[3], Bart Preneel[1,2], and Christian Rechberger[3]

[1] Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium.
sebastiaan.indesteege@esat.kuleuven.be
[2] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
[3] Institute for Applied Information Processing and Communications Inffeldgasse 16a, A–8010 Graz, Austria.

Abstract. We study the security of step-reduced but otherwise unmodified SHA-256. We show the first collision attacks on SHA-256 reduced to 23 and 24 steps with complexities 2^{18} and $2^{28.5}$, respectively. We give example colliding message pairs for 23-step and 24-step SHA-256. The best previous, recently obtained result was a collision attack for up to 22 steps. We extend our attacks to 23 and 24-step reduced SHA-512 with respective complexities of $2^{44.9}$ and $2^{53.0}$. Additionally, we show non-random behaviour of the SHA-256 compression function in the form of free-start near-collisions for up to 31 steps, which is 6 more steps than the recently obtained non-random behaviour in the form of a free-start near-collision. Even though this represents a step forwards in terms of cryptanalytic techniques, the results do not threaten the security of applications using SHA-256.

Keywords: SHA-256, SHA-512, hash functions, collisions, semi-free-start collisions, free-start collisions, free-start near-collisions.

1 Introduction

In the light of previous break-through results on hash functions such as MD5 and SHA-1, the security of their successors, SHA-256 and sisters, against all kinds of cryptanalytic attacks deserves special attention. This is even more important as many products and services that used to rely on SHA-1 are now migrating to SHA-256.

* This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, and in part by the Austrian Science Fund (FWF), project P19863. This work was done during a visit of the first author to the Graz University of Technology.

** F.W.O. Research Assistant, Fund for Scientific Research — Flanders (Belgium).

1.1 Previous work on members of the SHA-2 family

Below, we briefly discuss existing work. Results on older variants of the larger MD4 related hash function family, including SHA-1, suggest that the concept of local collisions might also be important for the SHA-2 family. The first published analysis on members of the SHA-2 family, by Gilbert and Handschuh [2], goes in this direction. They show that there exists a 9-step local collision with probability 2^{-66}. Later on, the result was improved by Hawkes *et al.* [3]. By considering modular differences, they increased the probability to 2^{-39}. Using XOR differences, local collisions with probability as high as 2^{-38} where used by Hölbl *et al.* [4]. Local collisions with lower probability but with other properties were studied by Sanadhya and Sarkar in [13].

Now we turn our attention to the analysis of simplified variants of SHA-256. In [17], Yoshida and Biryukov replace all modular additions by XOR. For this variant, a search for pseudo-collisions is described, which is faster than brute force search for up to 34 steps. Matusiewicz *et al.* [8] analysed a variant of SHA-256 where all Σ- and σ-functions are removed. The conclusion is that for this variant, collisions can be found much faster than by brute force search. The work shows that the approach used by Chabaud and Joux [1] in their analysis of SHA-0 is extensible to that particular variant of SHA-256. The message expansion as a building block on its own was studied by Matusiewicz *et al.* [8] and Pramstaller *et al.* [12].

Finally, we discuss previous work that focuses on step-reduced but otherwise unmodified SHA-256. The first study was done by Mendel *et al.* [9]. The results obtained are a practical 18-step collision and a differential characteristic for 19-step SHA-224 collision. Also, an example of a pseudo-near-collision for 22-step SHA-256 is given. Similar techniques have been studied by Matusiewicz *et al.* [8] and recently also by Sanadhya and Sarkar [15]. Using a different technique, Nikolić and Biryukov [10] obtained collisions for up to 21 steps and non-random behaviour in the form of semi-free-start near-collisions for up to 25 steps. Very recently, Sanadhya and Sarkar [16] extended this, and showed a collision example for 22 steps of SHA-256 in [14].

1.2 Our Contribution

We extend the work of Nikolić and Biryukov [10] to collisions for 23- and 24-step SHA-256 with respective time complexities of 2^{18} and $2^{28.5}$ reduced SHA-256 compression function evaluations. These 23- and 24-step attacks are also applied to SHA-512, with complexities of $2^{44.9}$ and

Table 1. Comparison of our results with the known results in the literature for each type. Effort is expressed in (equivalent) calls to the respective reduced compression functions.

function	steps	type	effort	source	example
SHA-256	18	collision	2^0	[9]	yes
SHA-256	20	collision	$2^{1.58}$	[10]	no
SHA-256	21	collision	2^{15}	[10]	yes
SHA-256	22	collision	2^9	[14]	yes
SHA-256	**23**	collision	2^{18}	this work	yes
SHA-256	**24**	collision	$2^{28.5}$	this work	yes
SHA-512	**23**	collision	$2^{43.9}$	this work	yes
SHA-512	**24**	collision	$2^{53.0}$	this work	no
SHA-256	23	semi-free-start collision	2^{17}	[10]	yes
SHA-256	**24**	semi-free-start collision	2^{17}	this work	no
SHA-224	**25**	free-start collision	2^{17}	this work	no
SHA-256	22	free-start near-collision	2^0	[9]	yes
SHA-256	25	semi-free-start near-collision	2^{34}	[10]	yes
SHA-256	**31**	free-start near-collision	2^{32}, Table 6	this work	no

$2^{53.0}$ for 23-step SHA-512 and 24-step SHA-512, respectively. Example collision pairs for 23-step SHA-256 and SHA-512, and for 24-step SHA-256 are given. The collision attacks presented in this work do not extend beyond 24 steps, but we investigate several weaker collision style attacks on a larger number of rounds. Our results are summarised in Table 1.

We use the terminology introduced by Lai and Massey [5] for different types of attacks on (iterated) hash functions. A collision attack aims to find two distinct messages that hash to the same result. In a semi-free-start collision attack, the attacker is additionally allowed to choose the initial chaining value, but the same value should be used for both messages. In a free-start collision attack, a (small) difference may appear in the initial chaining value. Near-collision attacks relax the requirement that the hash results should be equal and allow for small differences.

The structure of this paper is as follows. We give a short description of SHA-256 in Sect. 2. Section 3 gives an alternative description of the semi-free-start collision attack by Nikolić and Biryukov [10], which will make the subsequent description of the new attacks easier to understand. We then discuss our collision attacks on 23- and 24-step SHA-256 in Sect. 4. In Sect. 5, we apply our results to step-reduced SHA-512. Finally, Sect. 6 concludes.

Table 2. The notation used in this paper.

$X \ggg s$	X rotated over s bits to the right
$X \gg s$	X shifted over s bits to the right
$\overline{X}$	One's complement of X
$X \oplus Y$	Bitwise exclusive OR of X and Y
$X + Y$	Addition of X and Y modulo 2^{32}
$X - Y$	Subtraction of X and Y modulo 2^{32}
$A_i, \cdots, H_i$	State variables at step i, for the first message
$A'_i, \cdots, H'_i$	Idem, for the second message
W_i	i-th expanded message word of the first message
W'_i	Idem, for the second message
δX	Additive difference in X, *i.e.*, $X' - X$
$\delta\sigma_0(X)$	Additive difference in $\sigma_0(X)$, *i.e.*, $\sigma_0(X') - \sigma_0(X)$

2 Description of SHA-256

This section gives a short description of the SHA-256 hash function, using the notation from Table 2. For a detailed specification, we refer to [11].

The compression function of SHA-256 consists of a message expansion, which transforms a 512-bit message block into 64 expanded message words W_i of 32 bits each, and a state update transformation. The latter updates eight 32-bit state variables $A, \ldots, H$ in 64 identical steps, each using one expanded message word. The message expansion can be defined recursively as follows.

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \le i < 64 \end{cases} . \tag{1}$$

The functions $\sigma_0(X)$ and $\sigma_1(X)$ are given by

$$\begin{aligned} \sigma_0(X) &= (X \ggg 7) \oplus (X \ggg 18) \oplus (X \gg 3) \ , \\ \sigma_1(X) &= (X \ggg 17) \oplus (X \ggg 19) \oplus (X \gg 10) \ . \end{aligned} \tag{2}$$

The state update transformation updates two of the state variables in every step. It uses the bitwise Boolean functions f_{ch} and f_{maj} as well as the GF(2)-linear functions Σ_0 and Σ_1.

$$\begin{aligned} f_{\mathrm{ch}}(X,Y,Z) &= XY \oplus \overline{X}Z \ , \\ f_{\mathrm{maj}}(X,Y,Z) &= XY \oplus YZ \oplus XZ \ , \\ \Sigma_0(X) &= (X \ggg 2) \oplus (X \ggg 13) \oplus (X \ggg 22) \ , \\ \Sigma_1(X) &= (X \ggg 6) \oplus (X \ggg 11) \oplus (X \ggg 25) \ . \end{aligned} \tag{3}$$

The following equations describe the state update transformation, where K_i is a step constant.

$$
\begin{aligned}
T_1 &= H_i + \Sigma_1(E_i) + f_{\mathrm{ch}}(E_i, F_i, G_i) + K_i + W_i \ , \\
T_2 &= \Sigma_0(A_i) + f_{\mathrm{maj}}(A_i, B_i, C_i) \ , \\
A_{i+1} &= T_1 + T_2 \ , \quad B_{i+1} = A_i \ , \quad C_{i+1} = B_i \ , \quad D_{i+1} = C_i \ , \\
E_{i+1} &= D_i + T_1 \ , \quad F_{i+1} = E_i \ , \quad G_{i+1} = F_i \ , \quad H_{i+1} = G_i \ .
\end{aligned} \tag{4}
$$

After 64 steps, the initial state variables are fed forward using word-wise addition modulo 2^{32}.

3 Review of the Nikolić-Biryukov Semi-Free-Start Collision Attack

In this section, we review the 23-step semi-free-start collision attack by Nikolić and Biryukov [10]. The new results presented in this paper are extensions of this attack. The notations we use are given in Table 2.

The attack uses a nine step differential, which is presented in Table 3. All additive differences are fixed, as well as the actual values of some of the internal state variables. Fixing these values ensures that the differential is followed, as will be explained later. The constants α, β, γ and ϵ are determined by the attack. The first difference is inserted via the message word W_9. There are no differences in expanded message words other than those indicated in Table 3, *i.e.*, only W_9, W_{10}, W_{11}, W_{12}, W_{16} and W_{17} can have a difference.

The attack algorithm consists of two phases. The first phase finds suitable values for the constants α, β, γ and ϵ as well as two expanded message words, W_{16} and W_{17}. A detailed description of this phase of the attack will be given in Sect. 3.2, as it is more instructive to describe the second phase first.

3.1 The Second Phase of the Attack

The second phase of the attack finds, when given suitable values for α, β, γ, ϵ, W_{16} and W_{17}, a pair of messages and a set of initial values that lead to a semi-free-start collision for 23 steps of SHA-256. It works by carefully fixing the internal state at step 11 as indicated in Table 3, and then computing forward and backward. At each step, the expanded message word W_i is computed such that the differential from Table 3 is followed. During this, four extra conditions appear, involving only the constants

Table 3. A 9 step differential, using additive differences (left) and conditions on the value (right). Zero differences resp. unconstrained values are denoted by blanks.

step	δA	δB	δC	δD	δE	δF	δG	δH	δW	A	B	C	D	E	F	G	H
8										α				γ			
9									1	α	α			$\gamma+1$	γ		
10	1				1				-1	-1	α	α		-1	$\gamma+1$	γ	
11		1			-1	1			δ_1	α	-1	α	α	ϵ	-1	$\gamma+1$	γ
12			1			-1	1		δ_2	α	α	-1	α	β	ϵ	-1	$\gamma+1$
13				1			-1	1		α	α	α	-1	β	β	ϵ	-1
14					1			-1			α	α	α	-1	β	β	ϵ
15						1						α	α	0	-1	β	β
16							1		1				α	-2	0	-1	β
17								1	-1						-2	0	-1
18																-2	0

determined by the first phase of the attack.

$$\begin{aligned}\sigma_1(W_{16}+1) - \sigma_1(W_{16}) - \Sigma_1(\epsilon - 1) + \Sigma_1(\epsilon) \\ - f_{\mathrm{ch}}(\epsilon - 1, 0, \gamma + 1) + f_{\mathrm{ch}}(\epsilon, -1, \gamma + 1) = 0 \ .\end{aligned} \tag{5}$$

$$\sigma_1(W_{17}-1) - \sigma_1(W_{17}) - f_{\mathrm{ch}}(\beta, \epsilon - 1, 0) + f_{\mathrm{ch}}(\beta, \epsilon, -1) = 0 \ . \tag{6}$$

$$\beta = \alpha - \Sigma_0(\alpha) \ . \tag{7}$$

$$f_{\mathrm{ch}}(\beta, \beta, \epsilon - 1) - f_{\mathrm{ch}}(\beta, \beta, \epsilon) = -1 \ . \tag{8}$$

The first phase guarantees that the constants are such that these conditions are satisfied. The second phase of the attack has a negligible complexity and is guaranteed to succeed. Since there is still a lot of freedom left, many 23-step semi-free-start collisions can be found, with only a negligible additional effort, by repeating this second phase several times. A detailed description of this phase, including the origins of (5)–(8), is given in Appendix A.

3.2 The First Phase of the Attack

The goal of the first phase of the attack is to determine suitable values for the constants α, β, γ and ϵ, as well as two expanded message words, W_{16} and W_{17}. Suitable values imply that the four conditions (5)–(8) are satisfied. Nikolić and Biryukov [10] do not give much detail on this procedure, hence we clarify it below.

1. Make a random choice for γ and ϵ and search for a value of W_{16} such that condition (5) is satisfied. This condition is of the form $\sigma_1(x+1) - \sigma_1(x) = \delta$. There exists a simple, generic method to solve equations of this form, which is described in Appendix B. We note however that for this particular case, a faster method exists. An exhaustive search over every possible value of x resulted in the observation that only 6 181 additive differences δ can ever be achieved. These can be stored in a lookup table, together with one or more solutions for each difference. Hence, solving an equation of this form can be done with a simple table lookup.

   If no solution exists, simply retry with different choices for γ and/or ϵ. If the right hand side difference δ is selected uniformly at random, the probability that the equation has a solution is $2^{-19.5}$, so we expect to have to repeat this step about $2^{19.5}$ times.
2. Make a random choice for α, and compute β using (7). Now check condition (8). As described in [10], this equation is satisfied if the bits of β are zero in the positions where the bits of $\epsilon - 1$ and ϵ differ. This occurs with a probability of approximately 1/3, so this condition is fairly easy to satisfy.
3. The last condition, (6), is of the same form as the first condition, so it can be solved in exactly the same way. The expected probability that a solution exists is again $2^{-19.5}$.

Note that, because not all conditions depend on all of the constants determined in this phase of the attack, the first condition can be treated independently of the last three. Thus, the first and last step of this phase of the attack are executed about $2^{19.5}$ times and the second step about 2^{21} times. One of these steps requires much less work than an evaluation of the compression function of (reduced) SHA-256 — a bit less than one step. Hence, the overall time complexity of the entire attack, when expressed in SHA-256 compression function evaluations, is below 2^{17}.

4 Our Collision Attacks on Step-Reduced SHA-256

In this section we describe a novel, practical collision attack on SHA-256, reduced to 23 steps. It has a time complexity of about 2^{18} evaluations of the reduced SHA-256 compression function. We also extend this to 24 steps of SHA-256, with an expected time complexity of $2^{28.5}$ compression function evaluations.

4.1 23-Step Collision

Our collision attack for SHA-256, reduced to 23 steps, consists of two parts. First, we construct a semi-free-start collision for 23 steps, based on the attack from Sect. 3. Then we transform this semi-free-start collision into a real collision.

Finding "Good" Constants. Finding a 23 step semi-free-start collision is done using the same attack as described in Sect. 3, with a slight change to the first phase. In Sect. 3.2, it was described how to find constants α, β, γ and ϵ such that there exist values for W_{16} and W_{17} ensuring that the conditions (5) and (6) are satisfied. There are still some degrees of freedom left in this process. Indeed, it is possible to determine the constants α, β, γ and ϵ such that there are *many* values for W_{16} and W_{17} satisfying (5) and (6).

We performed an exhaustive search for such good constants. Condition (5) depends only on ϵ and γ. An exhaustive search for this condition can be performed with approximately 2^{37} evaluations of (5), because for each value of ϵ, only some of the bits in γ can have an influence. We found several values for ϵ and γ for which more than 2^{29} choices for W_{16} ensure that (5) is satisfied, for instance

$$\gamma = \texttt{0000017c}_x \ , \quad \epsilon = \texttt{7f5f7200}_x \ . \tag{9}$$

Conditions (6) and (8) depend on ϵ and β, which in turn depends on α through (7). An interesting property is that condition (6) becomes independent of ϵ if we assume that condition (8) is satisfied. Indeed, since this assumption implies that the bits of β are zero where ϵ and $\epsilon - 1$ differ, (6) reduces to

$$\sigma_1\left(W'_{17} + 1\right) - \sigma_1\left(W'_{17}\right) = \overline{\beta} \ . \tag{10}$$

Because of this, an exhaustive search for good values of α and β is feasible. There are many of the optimal values for α and β which are consistent with (several of) the optimal values for ϵ, thus yielding a global optimum. For instance, with γ and ϵ as in (9), the following values for α and β are one of many optimal choices:

$$\alpha = \texttt{00b321e3}_x \ , \quad \beta = \texttt{fcffe000}_x \ . \tag{11}$$

There are 2^{16} possible choices for W_{17} which satisfy (6) with these constants. Thus, these values for α, β, γ and ϵ give us an additional freedom of 2^{45} in the choice of W_{16} and W_{17}. This phase can be considered a precomputation, or alternatively, one can reduce the effort spent in this phase by only searching a smaller part of the available search space, which likely leads to less optimal results. It may however be a worthwhile trade-off in practice.

Transforming into a Collision. Note that only 7 expanded message words, W_{11} until W_{17}, are actually fixed to a certain value when constructing a semi-free-start collision, ignoring the freedom left in W_{16} and W_{17} for now. The others are chosen arbitrarily or computed from the message expansion when necessary. Using this freedom, it is possible to construct many semi-free-start collisions with only a negligible additional effort. But it is also possible to use this freedom in a controlled manner to transform the semi-free-start collision into a real collision.

To this end, we first introduce an alternative description of SHA-256. In older variants of the same design strategy, like MD5 or SHA-1, only a single state variable is updated in every step. This naturally leads to a description where only the first state variable is considered. Something similar can be done with the SHA-2 hash functions, even though in the standard description, two state variables are updated in every step.

From the state update equations (4), we derive a series of equations expressing the inputs of the i-th state update transformation, $A_i, \ldots, H_i$, as a function of only A_i through A_{i-7}.

$$\begin{aligned} A_i &= A_i \ , \ \ B_i = A_{i-1} \ , \ \ C_i = A_{i-2} \ , \ \ D_i = A_{i-3} \ , \\ E_i &= A_{i-4} + A_i \quad - \Sigma_0(A_{i-1}) - f_{\text{maj}}(A_{i-1}, A_{i-2}, A_{i-3}) \ , \\ F_i &= A_{i-5} + A_{i-1} - \Sigma_0(A_{i-2}) - f_{\text{maj}}(A_{i-2}, A_{i-3}, A_{i-4}) \ , \\ G_i &= A_{i-6} + A_{i-2} - \Sigma_0(A_{i-3}) - f_{\text{maj}}(A_{i-3}, A_{i-4}, A_{i-5}) \ , \\ H_i &= A_{i-7} + A_{i-3} - \Sigma_0(A_{i-4}) - f_{\text{maj}}(A_{i-4}, A_{i-5}, A_{i-6}) \ . \end{aligned} \tag{12}$$

Substituting these into (4) yields an alternative description requiring only a single state variable. This description can be written concisely as

$$A_{i+1} = F\left(A_i, A_{i-1}, A_{i-2}, A_{i-3}, A_{i-4}, A_{i-5}, A_{i-6}\right) + A_{i-7} + W_i \ . \tag{13}$$

The function $F(\cdot)$ encapsulates (4) and (12), except for the addition of the expanded message word W_i and the state variable A_{i-7}. From (12), it is clear that one can easily transform an internal state in the standard description, $\langle A_i, \cdots, H_i \rangle$, to the corresponding internal state in the alternative description, $\langle A_i, \cdots, A_{i-7} \rangle$, and vice versa. Analogous to what is done for MD5 and SHA-1, the initial values can be redefined as $A_{-7}, \cdots, A_0$.

This alternative description of SHA-256 can be used to transform a 23 step semi-free-start collision for SHA-256 into a real collision. Since control over one expanded message word W_i gives full control over one state variable A_{i+1}, control over eight consecutive expanded message words gives full control over the entire internal state.

1. Start from a 23-step semi-free-start collision pair. Set $\langle A_0, \cdots, A_{-7} \rangle$ to the SHA-256 initial values, in the alternative description. Make arbitrary choices for W_0, W_1 and W_2, and recompute the first three steps.
2. The eight message words W_3 until W_{10} are now modified such that A_4 until A_{11} remain unchanged. This implies that the internal state at step 11, $\langle A_{11}, \cdots, H_{11} \rangle$ does not change, and thus we connect to the rest of the semi-free-start collision. More specifically, for every step i, $3 \leq i \leq 10$, the new value of the i-th message word is computed as

$$W_i = A_{i+1} - F(A_i, A_{i-1}, A_{i-2}, A_{i-3}, A_{i-4}, A_{i-5}, A_{i-6}) - A_{i-7} \ . \quad (14)$$

   In the message words W_9 and W_{10} there is an additive difference of 1 and -1, respectively. This does not pose a problem since the construction of the semi-free-start condition guarantees that these will have the intended effect, regardless of the values of W_9 and W_{10}, see Appendix A.
3. Now we need to verify again if conditions (5) and (6) are still satisfied, since they depend on W_{16} and W_{17}, which may have changed. If the conditions are not satisfied, simply restart and make different choices for W_0, W_1 and/or W_2.
   Recall however that we have spent extra effort in the first phase of the attack to choose the constants α, β, γ and ϵ such that there are *many* values for W_{16} and W_{17} that satisfy the conditions. For the constants given in (9) and (11), there are 2^{45} allowed values for these two expanded message words. This translates into a probability of 2^{-19} that the conditions (5) and (6) are indeed still satisfied. We hence expect to have to repeat this procedure about 2^{19} times. Every trial requires an effort equivalent to about 10 steps of SHA-256.
4. After a successful modification of the first message words, the expanded message words W_{18} until W_{22} need to be recomputed, and also the corresponding steps need to be redone. The construction of the semi-free-start collision still guarantees that no differences will be introduced.

If we consider the first phase to be a precomputation, the overall attack complexity is about 2^{18} evaluations of the compression function of SHA-256 reduced to 23 steps. An example collision pair for 23-step reduced SHA-256 is given in Table 4.

Table 4. Example Colliding Message Pair for 23-Step Reduced SHA-256.

M	29f1ebfb 4468041a 1e6565b6 4cc17e75 4ea4f993 33a77104 864a828d 1dcec3d2
	d33d7b02 bcd4a2d7 3b10201d 39953548 8e127f2b 0304fc01 e7118577 43b12ca7
M'	29f1ebfb 4468041a 1e6565b6 4cc17e75 4ea4f993 33a77104 864a828d 1dcec3d2
	d33d7b02 bcd4a2d8 3b10201c 3995d548 91129f2a 0304fc01 e7118577 43b12ca7
H	c77405ea 8bfe2016 ff0531b6 a89b81f6 e98cf052 491a6c62 fd009a40 3969dc83

4.2 24-Step Collision

The same approach can be extended to 24 steps of SHA-256, using the 24-step semi-free-start collision attack given in detail in Sect. 4.3. Simply put, the 23-step attack is simply shifted down by a single step, and no difference is introduced into W_0 by the message expansion in the backward direction.

When turning the semi-free-start collision into a collision, however, the value of the expanded message word W_{16} (which was the non-expanded message word W_{15} in the 23-step attack) should not change. In a straightforward extension of the 23-step collision attack to 24 steps, this extra condition would only be satisfied with a probability of 2^{-32}. Using the available freedom in a better way, this can be improved substantially.

1. Start from a 24-step semi-free-start collision pair. Set $\langle A_0, \cdots, A_{-7}\rangle$ to the SHA-256 initial values. Make an arbitrary choice for W_0 and recompute the first step. Now, it follows from (4) that $(A_2 - W_1)$ is a constant:
$$c_1 = A_2 - W_1 \ . \tag{15}$$
2. The new value of W_9 is determined from (14), *i.e.*, it depends on A_2 through A_{10}. The state variables A_5 through A_{10} have already been fixed in the semi-free-start collision. If we additionally fix A_4 and A_3 to arbitrary values, it is possible to compute the sum of W_9 and A_2,
$$c_2 = W_9 + A_2 = A_{10} - F\left(A_9, \cdots, A_3\right) \ . \tag{16}$$
3. Combining (1) and (15)–(16), results in
$$W_{16} - \sigma_1(W_{14}) - c_2 + c_1 - W_0 = \sigma_0(W_1) - W_1 \ . \tag{17}$$
It is easy to find a suitable value for W_1 that ensures that W_{16} has the proper value, if it exists. It suffices to guess the 15 least significant bits of W_1 to compute all 32 bits of W_1, satisfying the above condition with probability 2^{-14}. A conservative estimate is that each trial requires an effort equivalent to one step update of SHA-256.

Table 5. Example Colliding Message Pair for 24-Step Reduced SHA-256.

M	0187e08e 865cedaf 5b69e21a e0f7485e 50b98993 217e4650 51e3cf65 c2997c68
	2c267e16 82ffa4e9 37b5af09 5b28721d 1be35597 7ff22aa1 e807a758 c1519aaa
M'	0187e08e 865cedaf 5b69e21a e0f7485e 50b98993 217e4650 51e3cf65 c2997c68
	2c267e16 82ffa4e9 37b5af0a 5b28721c 1be3f597 82f24aa0 e807a758 c1519aaa
H	1584074c 8b810a94 01ea31b1 81bffd02 d29c817d e4e04b51 b9f5ac4f 6b34d1f8

4. Now all the internal state variables have been fixed. The corresponding message words can be found from (14) and the message expansion. Just as in the 23-step collision attack, however, there are still some conditions left. As explained in Sect. 4.1, these are satisfied with a probability of 2^{-19}.

Hence, the overall expected time complexity is equivalent to about $2^{19} \cdot (2^{14} + 10)$ SHA-256 step computations, or about $2^{28.5}$ evaluations of the SHA-256 compression function reduced to 24 steps. An example collision pair for 24-step reduced SHA-256 is given in Table 5. An extension of this attack method beyond 24 steps fails, because then a difference in the first or in the last message word becomes unavoidable. In [14], another differential than the one shown in Table 3 is used to find 22-step collisions for SHA-256. We tried to use this differential in our extended attacks, but even for 23 steps, using this differential fails.

4.3 Further Extensions

This section discusses further extensions using weaker attack models. The starting point is the 23-step semi-free-start collision attack of Nikolić and Biryukov [10], which was described in Sect. 3.

Semi-Free-Start Collisions for 24 Steps of SHA-256. We keep the entire attack algorithm from Sect. 3 unchanged, but shift everything down by a single step. Because of this, one more message word, W_0, needs to be computed from the message expansion in the reverse direction. From (1), it follows that the additive difference in this word is

$$\delta W_0 = \delta W_{16} - \delta\sigma_1 (W_{14}) - \delta W_9 - \delta\sigma_0 (W_1) \quad . \tag{18}$$

None of these expanded message words has a difference, so also $\delta W_0 = 0$. This yields 24-step semi-free-start collisions of SHA-256 with the same complexity of 2^{17} compression function evaluations.

Free-Start Collisions for 25 Steps of SHA-224. SHA-224 differs from SHA-256 in two ways. First, it has different initial values, and second, the output is truncated to the leftmost 224 bits. We can thus extend the 24-step semi-free-start collision of SHA-256 to a 25-step free-start collision of SHA-224 by simply shifting the same attack down one more step. Now a difference will inevitably appear in W_0, which propagates to the initial value H_0. The other initial values, A_0 through G_0 still have a zero difference. Because the word H is truncated away in SHA-224, this results in free-start collisions for 25 steps of SHA-224, with the same complexity. Note that this attack would not apply if a different method of truncation would have been chosen in the design of SHA-224.

Free-Start Near-Collisions of SHA-256. Extending the attack to more steps is possible, provided that some differences are allowed both in the initial value and in the hash result, *i.e.*, when considering free-start near-collisions. The starting point is again the 23-step semi-free-start collision attack from Sect. 3. It is extended by adding a number of extra forward and backward steps.

As explained above, no difference is introduced in the first backward step. Note that, in general, the diffusion of differences is slower in the backward direction than in the forward direction. A difference introduced in an expanded message word W_i affects both A_{i+1} and E_{i+1} in the forward direction, as opposed to only H_i when going in the backward direction. Thus, in the forward direction, all state words can be affected by a single difference in an expanded message word after only four rounds. In the backward direction, this takes eight rounds.

We have done several experiments, each equivalent to an effort of 2^{32} reduced SHA-256 compression function evaluations. The results of our experiments are summarised in Table 6. The first three columns give the total number of steps, the number of extra forward and extra backward steps, respectively. The fourth column gives $k_{\min}$, the smallest Hamming distance found. The last eight columns contain the 2-logarithm of the number of solutions with a Hamming distance k of at most 8, 16, ..., 64 bits.

For comparison, also the expected values for a generic birthday attack with an equal effort of 2^{32} is given. For a generic (free-start) near-collision attack on an ideal n-bit hash function, using the birthday paradox with an effort of 2^w compression function evaluations, the lowest expected Hamming distance is the lowest k for which

$$2^{2w} \cdot \sum_{i=0}^{k} 2^{-n} \binom{n}{i} \geq 1 \ . \qquad (19)$$

Table 6. Experimental results of the free-start near-collision attack on SHA-256. For each number of steps, only the combination of forward/backward steps that gave the best results is shown. For comparison, the expected numbers of solutions for a generic birthday attack with an equal effort are also given.

steps	fwd.	bwd.	$k_{\min}$	2-logarithm of the number of solutions with k							
				≤ 8	≤ 16	≤ 24	≤ 32	≤ 40	≤ 48	≤ 56	≤ 64
25	1	1	2	31.95	32.00	32.00	32.00	32.00	32.00	32.00	32.00
26	1	2	8	24.17	31.55	31.99	32.00	32.00	32.00	32.00	32.00
27	1	3	11	$-\infty$	15.41	26.20	30.65	31.89	32.00	32.00	32.00
28	1	4	18	$-\infty$	$-\infty$	8.77	20.41	27.24	30.63	31.80	31.99
29	1	5	32	$-\infty$	$-\infty$	$-\infty$	1.58	14.31	22.86	28.19	30.93
30	1	6	43	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	10.73	19.58	25.68
31	2	6	53	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	6.34	15.50
Birthday Attack			57	−143.41	−108.84	−80.49	−56.36	−35.51	−17.37	−1.57	12.14

For instance, with $w = 32$ and for SHA-256 (*i.e.*, $n = 256$), this gives $k = 57$ bits. Our attack performs significantly better for up to 30 steps of SHA-256. For 31 steps, we still found 208 free-start near-collisions with a Hamming distance of at most 57 bits, whereas a birthday attack is only expected to find one with the same effort.

5 Collision Attacks on Step-Reduced SHA-512

SHA-512 is a 512-bit hash function from the SHA-2 family. Its structure is very similar to SHA-256. The sizes of all words are increased to 64 bits and the number of rounds is increased to 80. It uses a different initial chaining value, and different step constants. Finally, the GF(2)-linear functions are redefined. Refer to [11] for details on SHA-512. In this section, we extend the collision attacks on SHA-256 that were described in Sect. 4.1 and 4.2 to SHA-512. The first phase of the attacks needs to be adapted, since an exhaustive search as in Sect 3.2 is no longer feasible.

Finding "Good" Constants for SHA-512. Recall from Sect. 3.2 that the goal of the first phase of the attack is to find values for the constants α, β, γ, ϵ such that the conditions (5)–(8) are satisfied for many values of the expanded message words W_{16}, W_{17}. Since an exhaustive search for good constants is infeasible, we suggest the following approach.

1. First, make a list L of additive differences δ for which the equation

$$\sigma_1 (x + 1) - \sigma_1 (x) = \delta \tag{20}$$

has many solutions x. This can be accomplished by picking several values for x at random and computing the corresponding δ's. This procedure is likely to quickly find the "good" values for δ, since the more x's correspond to a δ, the more likely we are to find it. Using Appendix B, the number of solutions x for a given δ can be counted efficiently.

2. Since all conditions (5)–(8) will need to be satisfied, we can use (10) instead of (6). Hence, $\overline{\beta}$ should preferably be one of the "good" δ's from the list L. Knowing the value of β, we need to invert (7) to find α. This can, for instance, be done by guessing the 36 most significant bits of α and determining the other bits using (7). A guess succeeds with a probability of about 2^{-36}. Note that (7) cannot necessarily be inverted for all β's.
3. Now we make an arbitrary choice for ϵ which satisfies (8). Denote by l_β the length of the run of least significant "0"-bits in β. Then, (8) is satisfied if and only if the least significant "1"-bit of ϵ lies within the l_β least significant bits. Unfortunately, for SHA-512, this condition eliminates the best values for β.
4. If we choose a "good" value for $\sigma_1(W_{16}+1) - \sigma_1(W_{16})$ from the list L, and since ϵ has already been chosen, (5) can be rewritten as

$$C - f_{\mathrm{ch}}(\epsilon - 1, 0, \gamma + 1) + f_{\mathrm{ch}}(\epsilon, -1, \gamma + 1) = 0 \ , \tag{21}$$

where C is a known constant. The bits in which ϵ and $(\epsilon - 1)$ differ can be corrected by a proper choice of γ. Hence it is advantageous to choose ϵ with a long run of least significant "0"-bits. This again constrains β, as explained above. If no choice for γ can satisfy (21), retry with a different choice for ϵ and/or β.

Unlike the exhaustive search in Sect. 3.2, this procedure does not guarantee to find the optimal solution. However, experiments show that we can quickly find many good solutions. We found many values for the constants α, β, γ and ϵ for which the conditions (5) and (8) are satisfied for $2^{49.1}$ and 2^{34} values for W_{16} and W_{17}, respectively. Example values are

$$\begin{aligned} \alpha &= \texttt{3891fd20b54a8eb9}_x \ , & \beta &= \texttt{0001200000000000}_x \ , \\ \gamma &= \texttt{00000fff7f7fff46}_x \ , & \epsilon &= \texttt{0000100000000000}_x \ . \end{aligned} \tag{22}$$

23-step Collision. The second phase of the 23-step attack from Sect. 4.1 can directly be applied to SHA-512. With the constants from (22), a single attempt to turn a 23-step semi-free-start collision into a 23-step collision will succeed with an expected probability of $2^{-44.9}$ and costs about half of

Table 7. Example Colliding Message Pair for 23-Step Reduced SHA-512.

M	0000000017daf2ec	000000004b7adc8e	000000000d01f49d	54cce0ac731eb4c9
	5caf52c6f3e941cd	0224e6b804216305	95bbdc5df5b491c8	9f7f1453e39ee6c0
	3e345efecc818058	93dfcee7a268ce69	90561054da994c54	7262751c31b5bdd0
	54b1d56610b9e802	7f201dfcfce968c0	2b90cc3824ee5f13	05cfd16a7b4c4ab1
M'	0000000017daf2ec	000000004b7adc8e	000000000d01f49d	54cce0ac731eb4c9
	5caf52c6f3e941cd	0224e6b804216305	95bbdc5df5b491c8	9f7f1453e39ee6c0
	3e345efecc818058	93dfcee7a268ce6a	90561054da994c53	7266551c31b5bd18
	54b0b56610b9e801	7f201dfcfce968c0	2b90cc3824ee5f13	05cfd16a7b4c4ab1
H	dd44d89f178803f5	136802b223c880ba	bbb80917dda6a3e7	be1f118889bd5415
	98adc37a0f32d151	83d35099922ee2c6	670ac37463f224da	e0835506fb66503d

a reduced SHA-512 compression function evaluation. Hence, this results in a collision attack on 23-step SHA-512 with an expected time complexity of $2^{43.9}$ reduced compression function evaluations. An example collision pair for 23-step reduced SHA-512 is given in Table 7.

24-Step Collision. Also the second phase of the 24-step attack from Sect. 4.2 can be applied to SHA-512. One slight modification is required when determining a suitable value for W_0, due to the redefinition of the σ_0-function in SHA-512. Guessing the 8 least significant bits of W_0 allows to compute all of W_0, satisfying (17) with probability 2^{-8}. This results in a collision attack on 24-step SHA-512 with an expected time complexity of $2^{53.0}$ reduced compression function evaluations.

Further Extensions. The attacks on SHA-512 can also be extended, much like the extensions described for the SHA-256 attacks in Sect. 4.3. Adding more rounds trivially leads to several (semi-) free-start (near-) collision attacks. One noteworthy case is a free-start collision attack on 26 steps of SHA-384. It is analogous to the 25-step free-start collision attack on SHA-224 from Sect. 4.3, but as two words are truncated away in the case of SHA-384, the attack extends to 26 steps.

6 Conclusion

Our results push the limit for cryptanalysis of step reduced but otherwise unmodified SHA-256; we found practical collisions for up to 24 steps. For almost half of the steps (31 out of 64) non-random properties of the compression function are detectable in practice. The results also apply to SHA-512, albeit with higher time complexities.

Acknowledgements

The authors would like to thank Vincent Rijmen for his advice. We acknowledge the use of the VIC computer cluster of K.U.Leuven, which was used to obtain most of the experimental results presented in this paper. Finally, thanks to Ralph Wernsdorf for bringing [14] to our attention.

References

1. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
2. Henri Gilbert and Helena Handschuh. Security Analysis of SHA-256 and Sisters. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *LNCS*, pages 175–193. Springer, 2003.
3. Philip Hawkes, Michael Paddon, and Gregory G. Rose. On corrective patterns for the SHA-2 family. Cryptology ePrint Archive, Report 2004/207, August 2004. `http://eprint.iacr.org/`.
4. Marko Hölbl, Christian Rechberger, and Tatjana Welzer. Searching for messages conforming to arbitrary sets of conditions in SHA-256. In *Proceedings of WEWORC 2007*, volume 4945 of *LNCS*. Springer, 2008. To appear.
5. Xuejia Lai and James L. Massey. Hash function based on block ciphers. In *EUROCRYPT*, pages 55–70, 1992.
6. Helger Lipmaa and Shiho Moriai. Efficient algorithms for computing differential properties of addition. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *LNCS*, pages 336–350. Springer, 2001.
7. Helger Lipmaa, Johan Wallén, and Philippe Dumas. On the additive differential probability of exclusive-or. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 317–331. Springer, 2004.
8. Krystian Matusiewicz, Josef Pieprzyk, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of simplified variants of SHA-256. In *Proceedings of WEWoRC 2005*, LNI P-74, pages 123–134, 2005.
9. Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of Step-Reduced SHA-256. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *LNCS*, pages 126–143. Springer, 2006.
10. Ivica Nikolić and Alex Biryukov. Collisions for Step-Reduced SHA-256. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, March 26-28, 2008*, LNCS. Springer, 2008. To appear.
11. National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.
12. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Preliminary Analysis of the SHA-256 Message Expansion. NIST - First Cryptographic Hash Workshop, October 31-November 1, 2005.
13. Somitra Kumar Sanadhya and Palash Sarkar. New Local Collisions for the SHA-2 Hash Family. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 193–205. Springer, 2007.

14. Somitra Kumar Sanadhya and Palash Sarkar. 22-step collisions for SHA-2. arXiv e-print archive, arXiv:0803.1220v1, March 2008. `http://de.arxiv.org/abs/0803.1220`.
15. Somitra Kumar Sanadhya and Palash Sarkar. Attacking Reduced Round SHA-256. In *Proceedings of ACNS 2008*, LNCS. Springer, 2008. To appear.
16. Somitra Kumar Sanadhya and Palash Sarkar. Non-Linear Reduced Round Attacks Against SHA-2 Hash Family. In *Proceedings of ACISP 2008*, LNCS. Springer, 2008. To appear.
17. Hirotaka Yoshida and Alex Biryukov. Analysis of a SHA-256 Variant. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *LNCS*, pages 245–260. Springer, 2005.

A Detailed Description of the Second Phase of the Nikolić-Biryukov Attack

This appendix gives a detailed description of the second phase of the Nikolić-Biryukov attack [10]. When given suitable values for α, β, γ, ϵ, W_{16} and W_{17} by the first phase, as described in Sect. 3.2, it constructs a pair of messages and a set of initial values that lead to a semi-free-start collision for 23 steps of SHA-256.

1. Start at step 11 by fixing the state variables in this step, $A_{11}, \cdots, H_{11}$ as indicated in Table 3. The constants α, β, γ and ϵ are given by the first phase of the attack.
2. Calculate W_{11} such that $A_{12} = \alpha$ and W'_{11} such that $A'_{12} = \alpha$. Now $E_{12} = \beta$ only depends on α, and we find condition (7) from Sect. 3.1.

$$E_{12} = \alpha - \Sigma_0(\alpha) = \beta \ . \tag{23}$$

3. In a similar way, calculate W_{12} such that $E_{13} = \beta$ and W'_{12} such that $E'_{13} = \beta$. This also guarantees that $A_{13} = A'_{13}$ because the majority function absorbs the difference in C_{12}.
4. Calculate W_{13} such that $E_{14} = -1$ and set $W'_{13} = W_{13}$. Now, see Table 3, δE_{14} should be equal to 1. This yields the condition

$$\delta E_{14} = f_{\text{ch}}(\beta, \beta, \epsilon - 1) - f_{\text{ch}}(\beta, \beta, \epsilon) + 2 = 1 \ . \tag{24}$$

   It was given before as (8), and is satisfied by the first phase of the attack. Note that this also ensures that $\delta A_{14} = 0$.
5. Calculate W_{14} such that $E_{15} = 0$ and set $W'_{14} = W_{14}$. Since the values of E_{14} and E'_{14} were chosen in the previous step to be fixed points of the function Σ_1, $\delta\Sigma_1(E_{14}) = \delta E_{14} = 1$ cancels with $\delta H_{14} = -1$. Also, f_{ch} absorbs the difference in E_{14}, so no new differences are introduced.

6. Calculate W_{15} such that $E_{16} = -2$ and set $W'_{15} = W_{15}$. The difference in F_{15} is absorbed by f_{ch}.
7. The value for W_{16} is computed in phase one of the attack. The difference $\delta W_{16} = 1$ is cancelled by the output of f_{ch}. Indeed, since the binary representation of $E_{16} = -2$ is $111 \cdots 10_b$, the f_{ch} function passes only the difference in the least significant bit.
8. Also the value for W_{17} is computed in phase one of the attack. The difference $\delta W_{17} = -1$ cancels with $\delta H_{17} = 1$, thereby eliminating the final difference in the state variables. Thus, a collision is reached.
9. Now, go back to step 11 and proceed in the backward direction. Make an arbitrary choice for W_{10}. The differential from Table 3 is followed because of the careful choice of the state variables in step 11.
10. Make an arbitrary choice for W_9, and proceed one step backward. The difference $\delta W_9 = 1$ cancels with δA_{10} and with δE_{10} such that there is a zero difference in the state variables A_9 through H_9. Now randomly choose W_8 down to W_2 and calculate backward. Because no new differences appear in these expanded message words, there is also a zero difference in the state variables A_2 through H_2.
11. It is not possible to freely choose W_0 or W_1 as 16 expanded message words have already been chosen, *i.e.*, W_2 until W_{17}. Hence, these are computed using the message expansion in the backward direction. Although some of the message words used to compute W_0 and W_1 have differences, these differences always cancel out.
12. Continuing forward from step 18 again, note that the collision is preserved as long as no new differences are introduced via the expanded message words. From the message expansion, it follows that

$$\begin{aligned}\delta W_{18} = \sigma_1\left(W_{16}+1\right) - \sigma_1\left(W_{16}\right) - \Sigma_1\left(\epsilon - 1\right) + \Sigma_1\left(\epsilon\right) \\ - f_{\mathrm{ch}}\left(\epsilon - 1, 0, \gamma + 1\right) + f_{\mathrm{ch}}\left(\epsilon, -1, \gamma + 1\right) = 0 \ . \end{aligned} \tag{25}$$

    This is condition (5), which is satisfied by the first phase of the attack.
13. Similarly, in step 19, we require that $\delta W_{19} = 0$, which results in

$$\sigma_1\left(W_{17} - 1\right) - \sigma_1\left(W_{17}\right) - f_{\mathrm{ch}}\left(\beta, \epsilon - 1, 0\right) + f_{\mathrm{ch}}\left(\beta, \epsilon, -1\right) = 0 \ . \tag{26}$$

    This condition was given in (6), and is also satisfied by the first phase of the attack.
14. In steps 20–22, the message expansion guarantees that no new differences are introduced. In step 23, however, a difference of 1 is impossible to avoid, hence the attack stops after 23 steps.

B Solving $\mathcal{L}(x+\delta) = \mathcal{L}(x) + \delta'$

This appendix describes a generic method to solve equations of the form $\mathcal{L}(x+\delta) = \mathcal{L}(x) + \delta'$ where δ and δ' are given n-bit additive differences, and $\mathcal{L}$ is an n-bit to n-bit GF(2)-linear transformation. This is similar to the problems studied by Lipmaa and Moriai [6] and Lipmaa *et al.* [7].

Consider the modular addition $x+\delta$ and let $\Delta = (x+\delta) \oplus x$. This addition is described by the following equations, where x_i is the i-th bit of x and the c_i's are the carry bits:

$$\begin{aligned} (x+\delta)_i &= x_i \oplus \delta_i \oplus c_i \\ c_{i+1} &= f_{\mathrm{maj}}(x_i, \delta_i, c_i) \\ c_0 &= 0 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} c_i &= \delta_i \oplus \Delta_i \\ c_{i+1} &= f_{\mathrm{maj}}(x_i, \delta_i, \delta_i \oplus \Delta_i) \\ c_0 &= 0 \end{aligned} \ . \tag{27}$$

Hence, once we fix both the additive difference δ and the XOR difference Δ, all the carries c_i are fixed. Some of the x_i's are also fixed: when $\Delta_i = 1$ and $i < n-1$, it must hold that $x_i = c_{i+1} = \delta_{i+1} \oplus \Delta_{i+1}$. The other x_i's can be chosen arbitrarily. Thus, the allowed values for x lie in an affine space. Note that not all additive differences are consistent with all XOR differences, *i.e.*, the following conditions must be satisfied

$$\begin{cases} c_0 = \delta_0 \oplus \Delta_0 = 0 \\ \delta_i = \delta_{i+1} \oplus \Delta_{i+1} \quad \text{when} \ \ \Delta_i = 0 \ \text{ and } \ i < n-1 \end{cases} . \tag{28}$$

Solving an equation of the form $\mathcal{L}(x+\delta) = \mathcal{L}(x) + \delta'$ can be done as follows. Let $\Delta' = (\mathcal{L}(x) + \delta') \oplus \mathcal{L}(x)$, *i.e.*, the XOR-difference associated with the modular addition $\mathcal{L}(x) + \delta'$. Since $\mathcal{L}(x+\delta) = \mathcal{L}(x) + \delta'$ and $\mathcal{L}$ is GF(2)-linear, it follows that $\Delta' = \mathcal{L}(\Delta)$. We can thus simply enumerate all the XOR-differences Δ consistent with the given additive difference δ, compute $\Delta' = \mathcal{L}(\Delta)$ and check if this is consistent with the other additive difference δ'. If it is, both additions restrict x to a (different) affine space. The intersection of these spaces, which can be computed by solving a system of linear equations over GF(2), gives the solutions x for the chosen XOR-difference Δ. Note that this intersection may be empty. If no solutions are found for any value of the XOR-difference Δ, the equation $\mathcal{L}(x+\delta) = \mathcal{L}(x) + \delta'$ has no solutions. Note that the number of solutions of the equation can be counted efficiently using this method, as the number of solutions of a linear system over GF(2) is straightforward to compute.

The time complexity of this method is proportional to the minimum of the number of XOR differences consistent with the given additive differences δ or δ'. This follows from the fact that one can easily modify the method to choose Δ' instead of Δ.