

Trivial Collisions for Simplified and Reduced SHA-2^{*}

Sebastiaan Indestege^{**}

Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit
Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
`sebastiaan.indestege@esat.kuleuven.be`

Abstract. In this report we describe a trivial method to find collisions for strongly simplified variants of the SHA-2 family of hash functions. The simplifications include the linearization of the message expansion by replacing modular addition with XOR, and the reduction of the number of steps to 24 out of 64 (or 80). These simplifications allow to find collision pairs for any digest length with an expected time complexity of just 2^8 compression function evaluations. The same method can be applied to 30 steps of SHA-1, where the expected workload is about 2^5 compression function evaluations.

Key words: hash functions, collisions, SHA-2

1 Introduction

This report describes a collision finding attack on simplified variants of the SHA-2 family of hash functions.

The SHA-2 Family. The SHA-2 family consists of several iterated cryptographic hash functions with different digest sizes built on similar compression functions, e.g., SHA-256, SHA-384 and SHA-512. For a complete specification of the SHA-2 hash functions, we refer to [4]. Because of the similarity between the members of the SHA-2 family, we will focus on SHA-256 here. All the results in this paper can be equally applied to the other members of the SHA-2 family.

Brief Summary of Related Work. Gilbert and Handschuh [2] showed a 9 step local collision for SHA-256 with probability 2^{-66} . This was improved by Hawkes et al. [1] to 2^{-39} . In [6], a variant of SHA-256 where all modular additions are replaced by XOR was studied, resulting in pseudo-collisions for 34 steps of this variant. Mendel et al. [3] reported collision producing characteristics for step-reduced, but otherwise unmodified SHA-256, for up to 18 steps.

^{*} This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

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

2 Description of SHA-256

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, \dots, H in 64 identical steps, each using one expanded message word. The message expansion can be defined recursively as follows.

$$W_i = \begin{cases} M_i & \text{for } 0 \leq i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 16 \leq i < 64 \end{cases} \quad (1)$$

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

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

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

$$\begin{aligned} \Sigma_0(x) &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22) , \\ \Sigma_1(x) &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25) . \end{aligned} \quad (3)$$

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

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

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

2.1 A Simplified Variant of SHA-256

The simplified variant studied in this article differs from the real SHA-256 in two ways. First, all additions modulo 2^{32} in the message expansion are replaced by XORs, making it GF(2)-linear. Second, the number of steps is reduced to 24. Hence, the simplified message expansion becomes

$$W_i = \begin{cases} M_i & \text{for } 0 \leq i < 16 \\ \sigma_1(W_{i-2}) \oplus W_{i-7} \oplus \sigma_0(W_{i-15}) \oplus W_{i-16} & \text{for } 16 \leq i < 24 \end{cases} \quad (5)$$

We will refer to this variant as SHA-256-XOR-24. The same simplifications can be applied to the other members of the SHA-2 family.

3 Finding Collisions

In this section we describe how collisions for SHA-256-XOR-24 (and other simplified SHA-2 members like SHA-384-XOR-24 and SHA-512-XOR-24) can be found using a technique that is very similar to single-message modification. Single-message modification was first introduced by Wang [5] in collision attacks on MD5, SHA-0 and others.

3.1 Alternate Description of SHA-256

In SHA-0 and 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 two state variables are updated in every step.

To accomplish this, we derive from the state update equations (4) a series of equations which express the inputs of the i -th state update transformation, A_i, \dots, H_i , as a function of only A_i through A_{i-7} .

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

Substituting these into the state update transformation of SHA-256 (4) yields an alternative description requiring only a single state variable. This description can be written as

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

The function $F(\cdot)$ encapsulates (4) and (6) except for the addition of the expanded message word W_i .

Ignoring the message expansion, it is easy to see that control over eight consecutive expanded message words allows for any difference in the state variables to be eliminated. This is very similar to the idea of single message modification [5]. Note that this description of SHA-256 is only interesting for analysis purposes, not for implementation.

3.2 Inserting Odd Additive Differences.

Consider a pair of 32-bit words $\langle X, X' \rangle$ having an XOR difference of `0xffffffff`, i.e., there is a difference in every bit. What are the possible additive differences that can be achieved by such a pair?

From two's complement arithmetic, we know that the additive inverse of a word X can be found as $-X = \overline{X} + 1$, where \overline{X} is the one's complement of X . The additive difference of the pair $\langle X, X' \rangle$ is

$$\delta X = X' - X = \overline{X} - X = \overline{X} + (\overline{X} + 1) = 2\overline{X} + 1 . \quad (8)$$

Hence, any *odd* additive difference can be generated by an appropriate choice of the word X . There are exactly two possible choices for X for each odd additive difference as the most significant bit of X does not influence the difference δX .

3.3 The Message Difference

For SHA-256-XOR-24, there exists a message difference that will produce a difference of `0xffffffff` in the eight expanded message words W_8 through W_{15} and a zero difference in W_{16} through W_{23} . Indeed, since the message expansion of this simplified SHA-256 variant is GF(2)-linear one can simply use (5) to determine the differences in the first eight expanded message words. Table 1 shows the expanded message difference.

Table 1. The SHA-256-XOR-24 Expanded Message Difference.

$\Delta W_0 = 0x56c4b38b$	$\Delta W_8 = 0xffffffff$	$\Delta W_{16} = 0x00000000$
$\Delta W_1 = 0x1e01bbe2$	$\Delta W_9 = 0xffffffff$	$\Delta W_{17} = 0x00000000$
$\Delta W_2 = 0x71721c34$	$\Delta W_{10} = 0xffffffff$	$\Delta W_{18} = 0x00000000$
$\Delta W_3 = 0x037ab391$	$\Delta W_{11} = 0xffffffff$	$\Delta W_{19} = 0x00000000$
$\Delta W_4 = 0x0c28f460$	$\Delta W_{12} = 0xffffffff$	$\Delta W_{20} = 0x00000000$
$\Delta W_5 = 0xefbc47ff$	$\Delta W_{13} = 0xffffffff$	$\Delta W_{21} = 0x00000000$
$\Delta W_6 = 0xfdc03800$	$\Delta W_{14} = 0xffffffff$	$\Delta W_{22} = 0x00000000$
$\Delta W_7 = 0x1fffffff$	$\Delta W_{15} = 0xffffffff$	$\Delta W_{23} = 0x00000000$

3.4 The Collision Search

Putting the pieces together yields a simple collision finding algorithm for SHA256-XOR-24. The algorithm proceeds as follows:

1. Choose the message words W_0 through W_7 arbitrarily. Imposing the message difference from Tbl. 1 allows to compute W'_0 through W'_7 .
2. For each i , $8 \leq i < 16$,
 - (a) Determine the additive difference that needs to be introduced in step i to ensure a zero difference in A_{i+1} , i.e.,

$$\begin{aligned} \delta W_i &= W'_i - W_i \\ &= F(A_i, A_{i-1}, A_{i-2}, A_{i-3}, A_{i-4}, A_{i-5}, A_{i-6}, A_{i-7}) \\ &\quad - F(A'_i, A'_{i-1}, A'_{i-2}, A'_{i-3}, A'_{i-4}, A'_{i-5}, A'_{i-6}, A'_{i-7}) . \end{aligned} \quad (9)$$

- (b) With probability 1/2 the difference δW_i is odd. In this case, there are two suitable choices for W_i . In case δW_i is even, we start over with a different choice of W_0 through W_7 .
- 3. After 16 steps, a zero difference is reached in the internal state. Because the message difference is zero in the next eight rounds, this zero difference is maintained with certainty for an additional eight rounds.

Under reasonable independence assumptions, the overall success probability is 2^{-8} , hence we expect to find a collision pair after about 2^8 attempts. An early abort strategy and not fully backtracking slightly improves this. An example collision pair is given in Tbl. 2.

Note that a very similar approach can be applied to 30 steps of SHA-1.

4 Conclusion

We described a trivial method to find collisions for strongly simplified variants of the SHA-2 family of hash functions. Linearizing the message expansion by replacing modular addition with XOR and reducing the number of steps to 24 allows to find collision pairs with an expected effort of just 2^8 compression function evaluations, for any digest length.

References

1. P. Hawkes, M. Paddon and G. Rose, “*On Corrective Patterns for the SHA-2 Family*,” Cryptology ePrint Archive, Report 2004/207, 2004, <http://eprint.iacr.org/2004/207>.
2. H. Gilbert and H. Handschuh, “*Security Analysis of SHA-256 and Sisters*,” Selected Areas in Cryptography 2003, Lecture Notes in Computer Science, vol. 3006, pp. 175–193, Springer-Verlag, 2004.
3. F. Mendel, N. Pramstaller, C. Rechberger and V. Rijmen, “*Analysis of Step-Reduced SHA-256*,” Proceedings of Fast Software Encryption 2006, Lecture Notes in Computer Science vol. 4047, pp. 126–143, Springer-Verlag, 2006.
4. National Institute of Standards and Technology (NIST), “FIPS-180-2: Secure Hash Standard,” August 2002. Available online at <http://www.itl.nist.gov/fipspubs/>
5. X. Wang, H. Yu, “How to break MD5 and other hash functions,” Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science vol. 3494, pp. 19–35, Springer-Verlag, 2005.
6. H. Yoshida and A. Biryukov, “*Analysis of a SHA-256 Variant*,” Selected Areas in Cryptography 2005, Lecture Notes in Computer Science, vol. 3897, pp. 245–260, Springer-Verlag, 2006.

Table 2. An example collision pair for SHA-256-XOR-24.

	A_i	B_i	C_i	D_i	E_i	F_i	G_i	H_i	W_i
0	6a09e667	bb67ae85	3c6ef372	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0cd19	6667938f
1	62701bdc	6a09e667	bb67ae85	3c6ef372	ff2f7631	510e527f	9b05688c	1f83d9ab	7bd6934a
2	648a60cf	62701bdc	6a09e667	bb67ae85	3087407e	ff2f7631	510e527f	9b05688c	07dc201c
3	d738ee5a	648a60cf	62701bdc	6a09e667	39bdb81e	3087407e	ff2f7631	510e527f	2b2fc2a7
4	8d535940	d738ee5a	648a60cf	62701bdc	6c82ebbc	39bdb81e	3087407e	ff2f7631	2bf4b111
5	cd9aff09	8d535940	d738ee5a	648a60cf	c6baf39c	6c82ebbc	39bdb81e	3087407e	7292120a
6	867e3675	cd9aff09	8d535940	d738ee5a	3d18a3d9	c6baf39c	6c82ebbc	39bdb81e	0b748afe
7	67d3c637	867e3675	cd9aff09	8d535940	86c7ccdb	3d18a3d9	c6baf39c	6c82ebbc	72b15c2c
8	d4490d64	67d3c637	867e3675	cd9aff09	f26a5de3	86c7ccdb	3d18a3d9	c6baf39c	1da9165d
9	92b431e6	d4490d64	67d3c637	867e3675	1ff2b83a	f26a5de3	86c7ccdb	3d18a3d9	3a5cf5ca
10	04ef8437	92b431e6	d4490d64	67d3c637	b93eb1b4	1ff2b83a	f26a5de3	86c7ccdb	0202394b
11	7719af6b	04ef8437	92b431e6	d4490d64	eb6d55da	b93eb1b4	1ff2b83a	f26a5de3	10c5f64b
12	5333e55b	7719af6b	04ef8437	92b431e6	509ece8d	eb6d55da	b93eb1b4	1ff2b83a	3d868320
13	edb1efd1	5333e55b	7719af6b	04ef8437	f7a76eb7	509ece8d	eb6d55da	b93eb1b4	5a030974
14	011440c7	edb1efd1	5333e55b	7719af6b	cc7583d6	f7a76eb7	509ece8d	eb6d55da	61aa831e
15	b17ccc81	011440c7	edb1efd1	5333e55b	4fe69182	cc7583d6	f7a76eb7	509ece8d	06e69332
16	7c5b7561	b17ccc81	011440c7	edb1efd1	32f9cff8	4fe69182	cc7583d6	f7a76eb7	729011bf
17	4c06af0f	7c5b7561	b17ccc81	011440c7	6e939fad	32f9cff8	4fe69182	cc7583d6	d2d96dbe
18	eb01f745	4c06af0f	7c5b7561	b17ccc81	bdef5445	6e939fad	32f9cff8	4fe69182	a6774853
19	d1d36374	eb01f745	4c06af0f	7c5b7561	501a063e	bdef5445	6e939fad	32f9cff8	86987156
20	7b176155	d1d36374	eb01f745	4c06af0f	cc0f6a70	501a063e	bdef5445	6e939fad	a2c87883
21	40ca974d	7b176155	d1d36374	eb01f745	87d28e05	cc0f6a70	501a063e	bdef5445	fa5f5e00
22	288eedac	40ca974d	7b176155	d1d36374	01b7eb18	87d28e05	cc0f6a70	501a063e	3f535b7e
23	582f3054	288eedac	40ca974d	7b176155	647f5ebd	01b7eb18	87d28e05	cc0f6a70	b8c68fad
24	0e108a8a	582f3054	288eedac	40ca974d	1830da87	647f5ebd	01b7eb18	87d28e05	
H	781a70f1	1396ded9	64fde11e	e61a8c87	693f2d06	ff84c749	213bc4c3	e3b35b1e	
	A'_i	B'_i	C'_i	D'_i	E'_i	F'_i	G'_i	H'_i	W'_i
0	6a09e667	bb67ae85	3c6ef372	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0cd19	30a32004
1	2caba851	6a09e667	bb67ae85	3c6ef372	c96b02a6	510e527f	9b05688c	1f83d9ab	65d728a8
2	954cfd60	2caba851	6a09e667	bb67ae85	008d7692	c96b02a6	510e527f	9b05688c	76ae3c28
3	1f2514b1	954cfd60	2caba851	6a09e667	b0e0bf6b	008d7692	c96b02a6	510e527f	28557136
4	73d035fb	1f2514b1	954cfd60	2caba851	4a98e779	b0e0bf6b	008d7692	c96b02a6	27dc4571
5	bcd102a2	73d035fb	1f2514b1	954cfd60	9e3ff6d2	4a98e779	b0e0bf6b	008d7692	9d2e55f5
6	f05260e1	bcd102a2	73d035fb	1f2514b1	47a31cbb	9e3ff6d2	4a98e779	b0e0bf6b	f6b4b2fe
7	1771d0de	f05260e1	bcd102a2	73d035fb	132e2742	47a31cbb	9e3ff6d2	4a98e779	6d4ea3d3
8	e21633c9	1771d0de	f05260e1	bcd102a2	dd2570fe	132e2742	47a31cbb	9e3ff6d2	e256e9a2
9	92b431e6	e21633c9	1771d0de	f05260e1	9f2d07f4	dd2570fe	132e2742	47a31cbb	c5a30a35
10	04ef8437	92b431e6	e21633c9	1771d0de	67afafb8	9f2d07f4	dd2570fe	132e2742	fdfdc6b4
11	7719af6b	04ef8437	92b431e6	e21633c9	ad423400	67afafb8	9f2d07f4	dd2570fe	ef3a09b4
12	5333e55b	7719af6b	04ef8437	92b431e6	5e6bf4f2	ad423400	67afafb8	9f2d07f4	c2797cdf
13	edb1efd1	5333e55b	7719af6b	04ef8437	f7a76eb7	5e6bf4f2	ad423400	67afafb8	a5fcf68b
14	011440c7	edb1efd1	5333e55b	7719af6b	cc7583d6	f7a76eb7	5e6bf4f2	ad423400	9e557ce1
15	b17ccc81	011440c7	edb1efd1	5333e55b	4fe69182	cc7583d6	f7a76eb7	5e6bf4f2	f9196ccd
16	7c5b7561	b17ccc81	011440c7	edb1efd1	32f9cff8	4fe69182	cc7583d6	f7a76eb7	729011bf
17	4c06af0f	7c5b7561	b17ccc81	011440c7	6e939fad	32f9cff8	4fe69182	cc7583d6	d2d96dbe
18	eb01f745	4c06af0f	7c5b7561	b17ccc81	bdef5445	6e939fad	32f9cff8	4fe69182	a6774853
19	d1d36374	eb01f745	4c06af0f	7c5b7561	501a063e	bdef5445	6e939fad	32f9cff8	86987156
20	7b176155	d1d36374	eb01f745	4c06af0f	cc0f6a70	501a063e	bdef5445	6e939fad	a2c87883
21	40ca974d	7b176155	d1d36374	eb01f745	87d28e05	cc0f6a70	501a063e	bdef5445	fa5f5e00
22	288eedac	40ca974d	7b176155	d1d36374	01b7eb18	87d28e05	cc0f6a70	501a063e	3f535b7e
23	582f3054	288eedac	40ca974d	7b176155	647f5ebd	01b7eb18	87d28e05	cc0f6a70	b8c68fad
24	0e108a8a	582f3054	288eedac	40ca974d	1830da87	647f5ebd	01b7eb18	87d28e05	
H'	781a70f1	1396ded9	64fde11e	e61a8c87	693f2d06	ff84c749	213bc4c3	e3b35b1e	