A Collision Attack on a Double-Block-Length Compression Function Instantiated with Round-Reduced AES-256

Jiageng Chen[1], Shoichi Hirose[2], Hidenori Kuwakado[3], and Atsuko Miyaji[1]

[1] School of Information Science, Japan Advanced Institute of Science and Technology, Japan
[2] Graduate School of Engineering, University of Fukui, Japan
[3] Faculty of Informatics, Kansai University, Japan

Abstract. This paper presents the first non-trivial collision attack on the double-block-length compression function presented at FSE 2006 instantiated with round-reduced AES-256: $f_0(h_0\|h_1, M)\|f_1(h_0\|h_1, M)$ such that

$$f_0(h_0\|h_1, M) = E_{h_1\|M}(h_0) \oplus h_0 \ ,$$
$$f_1(h_0\|h_1, M) = E_{h_1\|M}(h_0 \oplus c) \oplus h_0 \oplus c \ ,$$

where $\|$ represents concatenation, E is AES-256 and c is a non-zero constant. The proposed attack is a free-start collision attack. It uses the rebound attack proposed by Mendel et al. It finds a collision with time complexity 2^8, 2^{64} and 2^{120} for the instantiation with 6-round, 8-round and 9-round AES-256, respectively. The space complexity is negligible. The attack is effective against the instantiation with 6-/8-round AES-256 if the 16-byte constant c has a single non-zero byte. It is effective against the instantiation with 9-round AES-256 if the constant c has four non-zero bytes at some specific positions.

Keywords: Double-block-length compression function · Free-start collision attack · Rebound attack · AES-256

1 Introduction

Background. Cryptographic hash functions are very important primitives and used in almost all cryptographic protocols. They are often called hash functions, and we follow this convention.

There are several design strategies of hash functions, and the most popular ones are block-cipher-based and permutation-based. The block-cipher-based approach is much more classical than the permutation-based approach. The permutation-based approach is fairly new, and the SHA-3 Keccak [2] is designed with the approach. Well-known hash functions such as MD5 [33], SHA-1 and SHA-2 [11] can be regarded as being designed with the block-cipher-based approach using dedicated block ciphers. Hash functions MDC-2 and MDC-4 [6] using DES predate them.

Hash functions using an existing block cipher seem useful for resource-constrained devices such as low-end microcontrollers and RFIDs. Even for high-end devices with AES-NI, hash functions using AES [8, 12] may be an option. How to construct secure hash functions using a block cipher has been an important research topic [4, 30]. When using existing block ciphers such as AES, one should adopt double-block-length construction [6, 14, 15, 20, 27] for sufficient level of collision-resistance.

Our Contribution. This paper presents a non-trivial collision attack on the double-block-length (DBL) compression function [15] instantiated with round-reduced AES-256. As far as the authors know, this is the first collision attack on the DBL compression function instantiated with AES-256. The DBL compression function is defined as $f_0(h_0\|h_1, M)\|f_1(h_0\|h_1, M)$ such that

$$f_0(h_0\|h_1, M) = E_{h_1\|M}(h_0) \oplus h_0 \ ,$$
$$f_1(h_0\|h_1, M) = E_{h_1\|M}(h_0 \oplus c) \oplus h_0 \oplus c \ ,$$

where $\|$ represents concatenation, E is AES-256 and c is a non-zero constant. The proposed collision attack assumes that the final round of round-reduced AES-256 does not have the `MixColumns` operation. The time complexity of the attack is 2^8, 2^{64}, and 2^{120} for the instantiation with AES-256 of 6 rounds, 8 rounds, and 9 rounds, respectively. The space complexity is negligible.

The proposed collision attack makes use of the following fact: If $(h_0\|h_1, M)$ and $((h_0 \oplus c)\|h_1, M)$ are a colliding pair for f_0, then they are also a colliding pair for f_1. The rebound attack [21] is used to find such a colliding pair for f_0. Thus, it largely depends on the value of c whether the proposed attack works well or not. The attack is effective against the instantiation with 6-/8-round AES-256 if the 16-byte constant c has a single non-zero byte. It is effective against the instantiation with 9-round AES-256 if the constant c has four non-zero bytes at some specific positions.

Related Work. The rebound attack was proposed by Mendel et al. [26], and was applied to the hash functions Whirlpool [31] and Grøstl [18], which have similar structure to AES. The rebound attack on Whirlpool was further improved by Lamberger et al. [21]. The rebound attack was also applied to a few other SHA-3 finalists [9, 17, 32].

There is some work on cryptanalyses of single-block-length hashing modes of AES. Biryukov, Khovratovich and Nikolić [3] presented a q-multicollision attack on the Davies-Meyer (DM) compression function instantiated with full-round AES-256. It is very powerful and its time complexity is $q \cdot 2^{67}$. The proposed attack does not seem to be able to use their attack since their attack needs some difference on the key input of AES. Mendel et al. [25] presented a collision attack on the DM compression function instantiated with 5-round AES-128 with time complexity 2^{56}. The collision attack on 5.5-round Whirlpool [21] can easily be extended to a collision attack on the DM, Matyas-Meyer-Oseas (MMO), Miyaguchi-Preneel (MP) compression functions instantiated with 6-round AES-128 or the DM compression function instantiated with 6-round AES-192/256. Its

time complexity is 2^{56}. Jean, Naya-Plasencia and Peyrin [16] presented a collision attack on the DM compression function instantiated with 6-round AES-128 with time complexity 2^{32}. Sasaki presented preimage and second-preimage attacks on DM, MMO and MP modes of 7-round AES [34].

There is little work on cryptanalyses of instantiations of DBL hashing modes. Ferguson [10] presented a few generic attacks on H-PRESENT-128 [5]. Wei et al. [35] presented collision and preimage attacks on various hashing modes instantiated with the block cipher IDEA [19]. They concluded that IDEA should not be used for hashing. The hashing modes include the DBL modes such as Abreast-DM, Tandem-DM [20], the mode by Hirose [15], the mode by Peyrin et al. [29] and MJH [23]. Our proposed collision attack is unlikely to be applied to them except for the Hirose mode.

The collision resistance and the preimage resistance were provided proofs in the ideal cipher model for Abreast-DM [1, 13, 22], Tandem-DM [1, 24], the Hirose compression function [1, 15]. In particular, Abreast-DM and the Hirose compression function were shown to be optimally collision-resistant in the ideal cipher model.

Organization. A brief description of AES is given in Sect. 2. An overview of the proposed collision attack on the DBL compression function is described in Sect. 3. The collision attacks on the compression function instantiated with AES-256 of 6 rounds, 8 rounds and 9 rounds are detailed in Sect. 4, Sect. 5 and Sect. 6, respectively. A concluding remark is given in Sect. 7.

2 Preliminaries

2.1 AES

This section gives a description of the AES [8, 12] together with some properties of its components necessary for the discussions later.

AES is a block cipher with 128-bit block length and 128/192/256-bit key length. The transformations of AES are performed on a (4×4)-byte array called the state. Each byte is regarded as an element in $\mathrm{GF}(2^8)$. Multiplication is performed modulo $x^8 + x^4 + x^3 + x + 1$. The state is initially a plaintext.

The encryption of AES consists of four transformations: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. It starts with the `AddRoundKey` transformation followed by iteration of a round function. The round function applies `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey` transformations in this order to the state. The final round does not have the `MixColumns` transformation.

The `SubBytes` transformation is byte-wise application of the nonlinear S-box function. For the S-box S, an input x satisfying the equation $\mathsf{S}(x) \oplus \mathsf{S}(x \oplus \Delta I) = \Delta O$ is called an admissible input for the pair of an input difference ΔI and an output difference ΔO. We will say that an input difference and an output difference are compatible with each other if there exist admissible inputs for the pair.

Table 1 shows the numbers of the pairs of input and output differences which have the specified numbers of admissible inputs. The probability that there exist any admissible inputs for a pair of input and output differences $(\Delta I, \Delta O)$ chosen uniformly at random is about $1/2$. An admissible input of the `SubBytes` transformation is defined similarly.

Table 1. Correspondence between the number of input-/ouput-difference pairs and the number of their admissible inputs for the AES S-box

the number of admissible inputs	0	2	4	256
the number of difference pairs	33150	32130	255	1

The `ShiftRows` transformation is byte-wise cyclic transposition of each row. It shifts the i-th row by i-bytes cyclically to left for $0 \leq i \leq 3$.

The `MixColumns` transformation is linear transformation of each column. It can be represented with a matrix. For a 4-byte column b of a state, it is represented by Mb, where

$$M = \begin{pmatrix} \texttt{02 03 01 01} \\ \texttt{01 02 03 01} \\ \texttt{01 01 02 03} \\ \texttt{03 01 01 02} \end{pmatrix} \quad \text{and} \quad M^{-1} = \begin{pmatrix} \texttt{0e 0b 0d 09} \\ \texttt{09 0e 0b 0d} \\ \texttt{0d 09 0e 0b} \\ \texttt{0b 0d 09 0e} \end{pmatrix} \ .$$

The `AddRoundKey` transformation is bitwise XOR of a round key to a state. The round keys are generated by a key expansion algorithm. The round keys of AES-256 are generated in the following way. Let (4×4)-byte array K_r be the round key of the r-th round for $r \geq 0$, where K_0 is for the initial `AddRoundKey` transformation. The 256-bit key input is given to K_0 and K_1. Let $K_r[j]$ be the j-th column of K_r for $0 \leq j \leq 3$. For $r \geq 2$, if r is even, then

$$\begin{aligned} K_r[0] &= K_{r-2}[0] \oplus \mathsf{SW}(K_{r-1}[3]^{\uparrow}) \oplus C_r \ , \\ K_r[j] &= K_{r-2}[j] \oplus K_r[j-1] \quad \text{for } 1 \leq j \leq 3 \ , \end{aligned}$$

where SW represents byte-wise application of the AES S-box, $K_{r-1}[3]^{\uparrow}$ represents cyclic 1-byte shift of $K_{r-1}[3]$ to the top, and C_r is a specified constant. If r is odd, then

$$\begin{aligned} K_r[0] &= K_{r-2}[0] \oplus \mathsf{SW}(K_{r-1}[3]) \ , \\ K_r[j] &= K_{r-2}[j] \oplus K_r[j-1] \quad \text{for } 1 \leq j \leq 3 \ . \end{aligned}$$

For simplicity, the `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey` transformations are denoted by SB, SR, MC and AK, respectively.

The state in the r-th round is denoted by S_r. S_r^{SB}, S_r^{SR}, S_r^{MC} and S_r^{AK} represent the state S_r just after SB, SR, MC and AK transformations, respectively. S_{-1} represents a plaintext input.

For $0 \leq i \leq 3$ and $0 \leq j \leq 3$, $S_r[i][j]$ represents the byte of S_r in the i-th row and the j-th column. $S_r[j]$ represents the j-th column of S_r.

3 Collision Attack on DBL Compression Function Instantiated with Round-Reduced AES-256

This section gives an overview of the proposed free-start collision attack on a DBL compression function [15] instantiated with round-reduced AES-256. The target DBL compression function

$$v_0 \| v_1 = F(h_0 \| h_1, M) = f_0(h_0 \| h_1, M) \| f_1(h_0 \| h_1, M)$$

is defined by

$$f_0(h_0 \| h_1, M) = E_{h_1 \| M}(h_0) \oplus h_0 \ ,$$
$$f_1(h_0 \| h_1, M) = E_{h_1 \| M}(h_0 \oplus c) \oplus h_0 \oplus c \ ,$$

where $\|$ represents concatenation, E is a block cipher and c is a non-zero constant. F is depicted in Fig. 1.

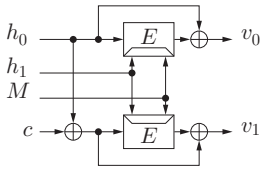

Fig. 1. The target DBL compression function. c is a non-zero constant and E is a block cipher.

The proposed attack uses the following simple fact:

Fact 1 *Suppose that* $(h_0 \| h_1, M)$ *and* $((h_0 \oplus \Delta h_0) \| h_1, M)$ *cause a collision for* f_0*, that is,* $f_0(h_0 \| h_1, M) = f_0((h_0 \oplus \Delta h_0) \| h_1, M)$ *and that* $\Delta h_0 = c$*. Then,* $(h_0 \| h_1, M)$ *and* $((h_0 \oplus \Delta h_0) \| h_1, M)$ *also cause a collision for* f_1*.*

The algorithm of the collision attack on F is given below:

1. Find a colliding pair of inputs $(h_0 \| h_1, M)$ and $((h_0 \oplus \Delta h_0) \| h_1, M)$ for f_0.
2. Output $(h_0 \| h_1, M)$ and $((h_0 \oplus \Delta h_0) \| h_1, M)$ if $\Delta h_0 = c$. Otherwise, return to Step 1.

The first step returns a colliding pair of inputs for f_0 such that the non-zero bytes of Δh_0 are located at the same positions as the non-zero bytes of the constant c. Thus, it largely depends on the value of c if the proposed attack is effective or not. The attack is effective against F instantiated with 6-/8-round AES-256 if the 16-byte constant c has a single non-zero byte. It is effective against F instantiated with 9-round AES-256 if the constant c has four non-zero bytes at some specific positions. Sections 4, 5 and 6 present how the collision attack is applied to F instantiated with AES-256 of 6, 8 and 9 rounds, respectively.

4 Collision Attack on F with 6-Round AES-256

This section presents a collision attack on f_0 instantiated with 6-round AES-256. It returns a pair of colliding inputs for any given $\Delta h_0 (= c)$ whose bytes are zero except for the first byte. The time complexity is 2^8, and the space complexity is negligible.[1] Thus, the total time complexity of the collision attack on the compression function F instantiated with 6-round AES-256 is also 2^8.

The collision attack on f_0 is based on the rebound attack on the 5.5-round Whirlpool hash function by Lamberger et al. [21]. Different from the attack by Lamberger et al., it is a free-start collision attack. Its goal is to find a pair of inputs, $(h_0 \| h_1, M)$ and $(h_0' \| h_1, M)$, which follow the differential path given in Fig. 2. Colored bytes in Fig. 2 are non-zero differences.

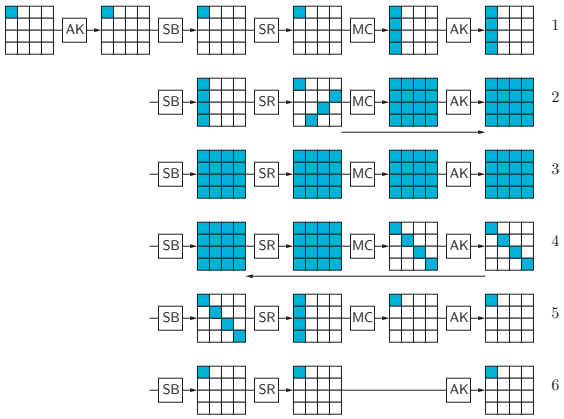

Fig. 2. The differential path used by the collision attack on the compression function f_0 instantiated with 6-round AES-256. Colored bytes are non-zero differences.

A detailed description of the attack is given below. It proceeds in two phases. It first fixes the values of differences on the differential path so that the pair of input and output differences of each SB transformation are compatible, that is, they have admissible inputs (Steps 1 to 5). Then, it selects admissible inputs of the SB transformations and connects them using the round keys (Steps 6 to 15).

Input: $\Delta h_0 = \Delta S_{-1}$.

Procedure:

1. Select the difference ΔS_1^{SB} compatible with $\Delta S_0^{\mathsf{AK}} = \Delta S_{-1}$. Then, compute $\Delta S_1^{\mathsf{AK}} = \Delta S_1^{\mathsf{MC}} = \mathsf{MC}(\mathsf{SR}(\Delta S_1^{\mathsf{SB}}))$.
2. Select ΔS_2^{SB} compatible with ΔS_1^{AK}. Then, compute ΔS_2^{AK}.
3. Select $\Delta S_5^{\mathsf{AK}} = \Delta S_5^{\mathsf{MC}}$ compatible with $\Delta S_6^{\mathsf{SB}} = \Delta S_6^{\mathsf{AK}} = \Delta S_{-1}$. Then, compute $\Delta S_5^{\mathsf{SB}} = \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(\Delta S_5^{\mathsf{MC}}))$.

[1] The complexity to compute admissible inputs of the AES S-box is omitted.

4. Select ΔS_4^{AK} compatible with ΔS_5^{SB}. Then, compute ΔS_4^{SB}.
5. Compute ΔS_3^{SB} and ΔS_3^{AK} compatible with ΔS_2^{AK} and ΔS_4^{SB}, respectively, such that $\Delta S_3^{\mathsf{AK}} = \mathsf{MC}(\mathsf{SR}(\Delta S_3^{\mathsf{SB}}))$. They can be computed column by column, and the expected time complexity is $4 \times 2^4 = 2^6$.
6. Select an admissible input S_2^{AK} for the pair of ΔS_2^{AK} and ΔS_3^{SB}.
7. Select an admissible input S_3^{AK} for the pair of ΔS_3^{AK} and ΔS_4^{SB}.
8. Compute the round key $K_3 = S_3^{\mathsf{MC}} \oplus S_3^{\mathsf{AK}} = \mathsf{MC}(\mathsf{SR}(\mathsf{SB}(S_2^{\mathsf{AK}}))) \oplus S_3^{\mathsf{AK}}$.
9. Compute S_4^{MC} from S_3^{AK}.
10. Select the diagonal elements of an admissible input S_4^{AK} for the pair of ΔS_4^{AK} and ΔS_5^{SB}. Then, compute the diagonal elements of K_4: $K_4[i][i] = S_4^{\mathsf{MC}}[i][i] \oplus S_4^{\mathsf{AK}}[i][i]$ for $0 \le i \le 3$.
11. The diagonal elements of S_5^{SB} are fixed by those of S_4^{AK}, and they further fix $S_5^{\mathsf{MC}}[0]$.
12. Select an admissible input $S_5^{\mathsf{AK}}[0][0]$ for the pair of ΔS_5^{AK} and ΔS_6^{SB}. Then, compute $K_5[0][0] = S_5^{\mathsf{MC}}[0][0] \oplus S_5^{\mathsf{AK}}[0][0]$.
13. Select an admissible input $S_1^{\mathsf{AK}}[0]$ for the pair of ΔS_1^{AK} and ΔS_2^{SB}. $S_1^{\mathsf{AK}}[0]$ fixes $S_2^{\mathsf{SB}}[0]$. The following condition on the round key K_2 is obtained:

$$\begin{aligned} S_2^{\mathsf{SB}}[0] &= \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(S_2^{\mathsf{AK}} \oplus K_2))[0] \\ &= \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(S_2^{\mathsf{AK}}))[0] \oplus \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(K_2))[0] \ . \end{aligned}$$

14. Select an admissible input $S_0^{\mathsf{AK}}[0][0]$ for the pair of ΔS_0^{AK} and ΔS_1^{SB}. $S_0^{\mathsf{AK}}[0][0]$ fixes $S_1^{\mathsf{SB}}[0][0]$. The following condition on the round key $K_1[0]$ is obtained:

$$\begin{aligned} S_1^{\mathsf{SB}}[0][0] &= (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})(S_1^{\mathsf{AK}}[0] \oplus K_1[0]) \\ &= (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})S_1^{\mathsf{AK}}[0] \oplus (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})K_1[0] \ . \end{aligned}$$

15. Compute the round keys satisfying all the conditions obtained so far. The following bytes of the round keys are already fixed: K_3, $K_4[0][0]$, $K_4[1][1]$, $K_4[2][2]$, $K_4[3][3]$ and $K_5[0][0]$. The conditions on the other bytes of round keys can be expressed by equations on K_2. The expected time complexity to compute K_2 is 2^8. Details are given in Sect. 4.1.
16. Compute the input S_{-1} from S_2^{AK} and the round keys. Output $h_0 = S_{-1}$, $h_1 = K_0$ and $M = K_1$.

The time complexity of the collision attack is about 2^8. The space complexity is negligible. An example of collision is given in Appendix A.

The same kind of differential path as the one in Fig. 2 is able to be constructed when given an input difference Δh_0 with a single non-zero byte at any byte position. Due to the asymmetry of the key expansion algorithm, however, a little more analyses are required to confirm whether the same kind of very efficient attack really works or not.

4.1 Conditions on the Round Key K_2

The following five conditions are led from the key expansion algorithm:

$$K_2[0][0] = K_4[0][0] \oplus \mathsf{S}(K_3[1][3]) \oplus RC_2 \tag{1}$$
$$K_2[1][0] \oplus K_2[1][1] = K_4[1][1] \oplus \mathsf{S}(K_3[2][3]) \tag{2}$$
$$K_2[2][0] \oplus K_2[2][1] \oplus K_2[2][2] = K_4[2][2] \oplus \mathsf{S}(K_3[3][3]) \tag{3}$$
$$K_2[3][0] \oplus K_2[3][1] \oplus K_2[3][2] \oplus K_2[3][3] = K_4[3][3] \oplus \mathsf{S}(K_3[0][3]) \tag{4}$$
$$K_2[0][1] \oplus K_2[0][2] \oplus K_2[0][3] = K_4[0][0] \oplus K_4[0][3] \ , \tag{5}$$

where RC_2 is a constant in the key expansion algorithm. $K_4[0][3] = \mathsf{S}^{-1}(K_3[0][0] \oplus K_5[0][0])$. Notice that all the bytes of the round keys on the right side of the equations above are fixed.

The following conditions are mentioned in the step 13 of the algorithm in Sect. 4:

$$(\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})K_2[0] = S_2^{\mathsf{SR}}[0][0] \oplus (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})S_2^{\mathsf{AK}}[0] \tag{6}$$
$$(\mathtt{0b}, \mathtt{0d}, \mathtt{09}, \mathtt{0e})K_2[1] = S_2^{\mathsf{SR}}[3][1] \oplus (\mathtt{0b}, \mathtt{0d}, \mathtt{09}, \mathtt{0e})S_2^{\mathsf{AK}}[1] \tag{7}$$
$$(\mathtt{0d}, \mathtt{09}, \mathtt{0e}, \mathtt{0b})K_2[2] = S_2^{\mathsf{SR}}[2][2] \oplus (\mathtt{0d}, \mathtt{09}, \mathtt{0e}, \mathtt{0b})S_2^{\mathsf{AK}}[2] \tag{8}$$
$$(\mathtt{09}, \mathtt{0e}, \mathtt{0b}, \mathtt{0d})K_2[3] = S_2^{\mathsf{SR}}[1][3] \oplus (\mathtt{09}, \mathtt{0e}, \mathtt{0b}, \mathtt{0d})S_2^{\mathsf{AK}}[3] \ . \tag{9}$$

The following last condition is nonlinear. It is also led from the key expansion algorithm:

$$\mathsf{SW}(K_2[3]) = K_1[0] \oplus K_3[0] \ ,$$

where $K_1[0]$ satisfies

$$(\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})K_1[0] = S_1^{\mathsf{SB}}[0][0] \oplus (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})S_1^{\mathsf{AK}}[0] \ . \tag{10}$$

SW represents transformation of each byte with the AES S-box.

We compute K_2 satisfying the conditions above by first computing $K_2[3]$ satisfying the last nonlinear condition. $K_2[3]$ is computed as follows:

1. Choose $K_2[3]$ satisfying Eq. (9) uniformly at random, and compute $K_1[0] = \mathsf{SW}(K_2[3]) \oplus K_3[0]$.
2. Check if $K_1[0]$ satisfies Eq. (10).

In the second step of the procedure, the probability that Eq. (10) holds is 2^{-8}. It is easy to compute the remaining twelve bytes of K_2 satisfying the linear equations from (1) to (9).

5 Collision Attack on F with 8-Round AES-256

This section presents a free-start collision attack on f_0 instantiated with 8-round AES-256. It returns a pair of colliding inputs with difference Δh_0 whose bytes are zero except for one byte at any specified position. The time complexity is

2^{56}, and the space complexity is negligible. The probability that $\Delta h_0 = c$ is 2^{-8} if c has a single non-zero byte at the same position as the non-zero byte of Δh_0. Thus, the total time complexity of the collision attack on F instantiated with 8-round AES-256 is 2^{64}.

The collision attack on f_0 is based on the rebound attack on the 7.5-round Whirlpool compression function by Lamberger et al. [21]. The goal of the attack is to find a pair of inputs, $(h_0 \| h_1, M)$ and $(h_0' \| h_1, M)$, which follow the differential path given in Fig. 3.

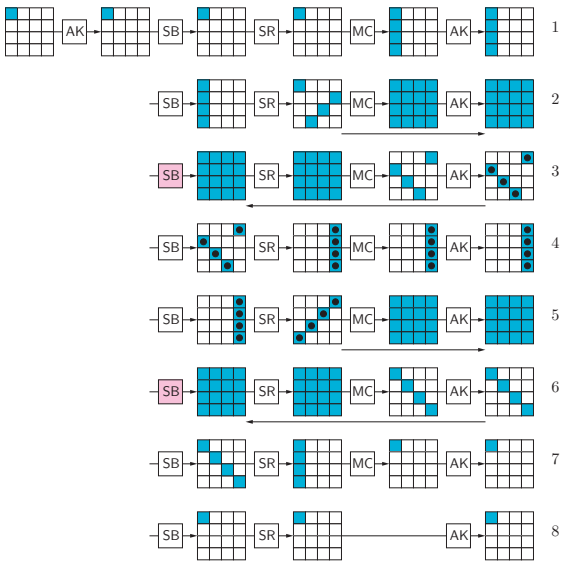

Fig. 3. The differential path used by the collision attack on the compression function f_0 instantiated with 8-round AES-256. Colored bytes are non-zero differences.

The proposed attack uses two inbound phases: The first one is in the second and the third rounds, and the second one is in the fifth and the sixth rounds. The algorithm of the attack is described below. It first selects the values of differences of the two inbounds (Steps 1 and 2) and those between the two inbounds (Step 3). Then, for each pair of an admissible input of SB in the third round and that of SB in the sixth round, it connects them with the round keys (Steps 4a to 4c), and extends the state transformation to the outbounds to check if a colliding pair of inputs are obtained (Steps 4d to 4f).

1. This step looks for a pair of compatible input/output differences of `SubBytes` of the third round in the following way:
   (a) Select ΔS_2^{SR}, $\Delta S_3^{\mathsf{AK}} = \Delta S_3^{\mathsf{MC}}$ uniformly at random, and compute
   $$\Delta S_2^{\mathsf{AK}} = \Delta S_2^{\mathsf{MC}} = \mathsf{MC}(\Delta S_2^{\mathsf{SR}}) \ ,$$
   $$\Delta S_3^{\mathsf{SB}} = \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(\Delta S_3^{\mathsf{MC}})) \ .$$

(b) If there are no admissible inputs for the pair of ΔS_2^{AK} and ΔS_3^{SB}, then return to Step 1a.

The expected number of repetitions of this step is 2^{16}. The number of admissible inputs obtained for S_2^{AK} with this step is 2^{16}. Actually, this step can be made more efficient since the trials can be done column by column. However, this speed-up does not change the time complexity of the overall algorithm.

2. This step looks for a pair of compatible input/output differences of `SubBytes` of the sixth round in the same way as the step 1. 2^{16} admissible inputs are obtained for S_5^{AK} with this step.
3. Select ΔS_4^{SB} compatible with ΔS_3^{AK} uniformly at random until $\Delta S_4^{\mathsf{AK}} = \Delta S_4^{\mathsf{MC}} = \mathsf{MC}(\mathsf{SR}(\Delta S_4^{\mathsf{SB}}))$ is compatible with ΔS_5^{SB}. The expected number of repetitions of this step is 2^4.
4. Perform the following procedure:
   (a) Select a new pair among the 2^{32} pairs of S_2^{AK} and S_5^{AK}. If there exists no new pair, then return to Step 1.
   (b) Compute $S_3^{\mathsf{SB}} = \mathsf{SB}(S_2^{\mathsf{AK}})$. Then, run the algorithm for connecting two inbound phases, which is given in Sect. 5.1, and obtain the round keys K_3, K_4 and K_5.
   (c) Compute the round keys K_0, K_1, K_2, K_6 and K_7.
   (d) Compute the corresponding input S_{-1} to AES and the difference ΔS_{-1} from S_2^{AK} and ΔS_2^{AK}. If any byte of ΔS_{-1} other than $\Delta S_{-1}[0][0]$ is non-zero, then return to Step 4a.
   (e) Compute the corresponding output S_8^{AK} from AES and the difference ΔS_8^{AK} from S_5^{AK} and ΔS_5^{AK}. If any byte of ΔS_8^{AK} other than $\Delta S_8^{\mathsf{AK}}[0][0]$ is non-zero, then return to Step 4a.
   (f) If $\Delta S_{-1} = \Delta S_8^{\mathsf{AK}}$, then proceed to Step 5. Otherwise, return to Step 4a.
5. Output the pair of inputs (K, S_{-1}) and $(K, S_{-1} \oplus \Delta S_{-1})$, which are mapped to the same hash value by f_0 instantiated with 8-round AES-256, where $K = K_0 \| K_1$.

For Step 4d in the algorithm above, the probability that only $\Delta S_{-1}[0][0]$ is non-zero (the transition from ΔS_1^{MC} to ΔS_1^{SR} is successful) is 2^{-24}. Similarly, for Step 4e, the probability that only $\Delta S_8^{\mathsf{AK}}[0][0]$ is non-zero is 2^{-24}. For Step 4f, the probability that $\Delta S_{-1} = \Delta S_8^{\mathsf{AK}}$ is 2^{-8}. Thus, the estimated time complexity of the algorithm above is $2^{24\times 2+8} = 2^{56}$.

The collision attack returns a colliding pair of inputs whose non-zero difference is located at the top-left corner. Owing to the symmetry of AES, the collision attack can easily be extended so that it returns a colliding pair of inputs whose non-zero difference is located at any specified byte position.

5.1 Algorithm to Connect Two Inbound Phases

An algorithm to connect two inbound phases is described in this section. It gives a pair of sequences of state values between SB in the third round and SB in the sixth round whose differences follow the differential path in Fig. 3. The initial

and final state values of the sequences are given to the algorithm as input as well as the values of the differences. The algorithm outputs the round keys (K_3, K_4 and K_5) which connect these values. The algorithm pays specific attention to the bytes of states with black circles in Fig. 3. They are given priority simply because they are bytes with non-zero differences.

Input: S_3^{SB}, S_5^{AK}, and ΔS_3^{SB}, ΔS_4^{SB}, ΔS_5^{AK}.

Output: Round keys K_3, K_4 and K_5.

Procedure:

1. Compute ΔS_3^{AK}, ΔS_4^{AK} and ΔS_5^{SB}:

$$\begin{aligned}
\Delta S_3^{\mathsf{AK}} &= \Delta S_3^{\mathsf{MC}} = \mathsf{MC}(\mathsf{SR}(\Delta S_3^{\mathsf{SB}})) \\
\Delta S_4^{\mathsf{AK}} &= \Delta S_4^{\mathsf{MC}} = \mathsf{MC}(\mathsf{SR}(\Delta S_4^{\mathsf{SB}})) \\
\Delta S_5^{\mathsf{SB}} &= \mathsf{SR}^{-1}(\mathsf{MC}^{-1}(\Delta S_5^{\mathsf{MC}})) \text{ , where } \Delta S_5^{\mathsf{MC}} = \Delta S_5^{\mathsf{AK}} \ .
\end{aligned}$$

2. Select admissible inputs of the S-boxes with non-zero differences of SB in the fourth round: $S_3^{\mathsf{AK}}[0][3]$, $S_3^{\mathsf{AK}}[1][0]$, $S_3^{\mathsf{AK}}[2][1]$ and $S_3^{\mathsf{AK}}[3][2]$.
3. Compute $K_3[0][3]$, $K_3[1][0]$, $K_3[2][1]$ and $K_3[3][2]$ from the corresponding bytes of $S_3^{\mathsf{MC}} = \mathsf{MC}(\mathsf{SR}(S_3^{\mathsf{SB}}))$ and S_3^{AK}.
4. Select admissible inputs of the S-boxes with non-zero differences of SB in the fifth round: $S_4^{\mathsf{AK}}[3]$. Then, compute $S_5^{\mathsf{SB}}[3]$.
5. $K_4[3] = S_4^{\mathsf{MC}}[3] \oplus S_4^{\mathsf{AK}}[3]$, where $S_4^{\mathsf{MC}}[3]$ can be computed from the corresponding bytes of S_3^{AK}.
6. Compute the round key K_5 satisfying the conditions obtained so far. They can be expressed by 8 linear equations on the bytes of K_5. The equations are given in Sect. 5.2.
7. Compute the remaining bytes of K_3 from K_5 and $K_4[3]$.
8. Compute S_4^{MC} and S_4^{AK} from S_3^{SB} with K_3 and from S_5^{AK} with K_5, respectively. Then, compute $K_4[j] = S_4^{\mathsf{MC}}[j] \oplus S_4^{\mathsf{AK}}[j]$ for $0 \leq j \leq 2$.

5.2 Conditions on the Round Key K_5

The following four conditions are led from the key expansion algorithm:

$$\begin{aligned}
K_5[1][0] &= K_3[1][0] \oplus \mathsf{S}(K_4[1][3]) \\
K_5[2][0] \oplus K_5[2][1] &= K_3[2][1] \\
K_5[3][1] \oplus K_5[3][2] &= K_3[3][2] \\
K_5[0][2] \oplus K_5[0][3] &= K_3[0][3] \ .
\end{aligned}$$

Notice that all the bytes of K_3 and K_4 on the right side are already fixed by the algorithm.

The other condition comes from the fixed bytes of $S_5^{\mathsf{SB}}[3]$:

$$\mathsf{SR}(S_5^{\mathsf{SB}}[3]) = \mathsf{MC}^{-1}(S_5^{\mathsf{AK}})[3] \oplus \mathsf{MC}^{-1}(K_5)[3] \ .$$

Notice that S_5^{AK} is given to the algorithm as input. They can be expanded to the following four equations:

$$
\begin{aligned}
(\mathtt{0b}, \mathtt{0d}, \mathtt{09}, \mathtt{0e})K_5[0] &= S_5^{\mathsf{SR}}[3][0] \oplus (\mathtt{0b}, \mathtt{0d}, \mathtt{09}, \mathtt{0e})S_5^{\mathsf{AK}}[0] \\
(\mathtt{0d}, \mathtt{09}, \mathtt{0e}, \mathtt{0b})K_5[1] &= S_5^{\mathsf{SR}}[2][1] \oplus (\mathtt{0d}, \mathtt{09}, \mathtt{0e}, \mathtt{0b})S_5^{\mathsf{AK}}[1] \\
(\mathtt{09}, \mathtt{0e}, \mathtt{0b}, \mathtt{0d})K_5[2] &= S_5^{\mathsf{SR}}[1][2] \oplus (\mathtt{09}, \mathtt{0e}, \mathtt{0b}, \mathtt{0d})S_5^{\mathsf{AK}}[2] \\
(\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})K_5[3] &= S_5^{\mathsf{SR}}[0][3] \oplus (\mathtt{0e}, \mathtt{0b}, \mathtt{0d}, \mathtt{09})S_5^{\mathsf{AK}}[3] \ .
\end{aligned}
$$

6 Collision Attack on F with 9-Round AES-256

The collision attack on f_0 instantiated with 8-round AES-256 can be extended to the collision attack on f_0 instantiated with 9-round AES-256 for different choice of the constant c. The differential path used by this attack is presented in Fig. 4. $\Delta h_0 = \Delta S_{-1}$ has four non-zero bytes on its diagonal. The differential path from the third round to the sixth round is equal to the differential path from the second round to the fifth round in Fig. 3. Thus, the algorithm to connect two inbound phases shown in Sect. 5.1 can also be used here.

In the outbound phase of the attack,

- the success probability of the transition from ΔS_2^{MC} to ΔS_2^{SR} is 2^{-24},
- the success probability of the transition from ΔS_8^{SR} to ΔS_8^{MC} is 2^{-32}, and
- the probability that $\Delta S_{-1} = \Delta S_9^{\mathsf{AK}}$ is 2^{-32}.

Thus, the estimated time complexity of the attack is $2^{24+32+32} = 2^{88}$. Though it is beyond the complexity of the birthday attack for f_0, it is effective for our purpose.

Due to the symmetry of AES, the attack also works with the same kind of the differential paths with ΔS_{-1} such that the four non-zero bytes of ΔS_{-1} are

- $\Delta S_{-1}[0][1]$, $\Delta S_{-1}[1][2]$, $\Delta S_{-1}[2][3]$, $\Delta S_{-1}[3][0]$,
- $\Delta S_{-1}[0][2]$, $\Delta S_{-1}[1][3]$, $\Delta S_{-1}[2][0]$, $\Delta S_{-1}[3][1]$, or
- $\Delta S_{-1}[0][3]$, $\Delta S_{-1}[1][0]$, $\Delta S_{-1}[2][1]$, $\Delta S_{-1}[3][2]$.

The total time complexity of the collision attack on F is $2^{88+32} = 2^{120}$ since the probability that $\Delta h_0 = c$ is 2^{-32} for constant c which has four non-zero bytes at the same positions as the non-zero bytes of Δh_0.

7 Conclusion

This paper has presented a free-start collision attack on the DBL compression function [15] instantiated with round-reduced AES-256. A drawback of the attack is that it is effective against restricted constants. It is interesting if the restriction is reduced. It is also interesting to apply the attack to instantiations with other block ciphers.

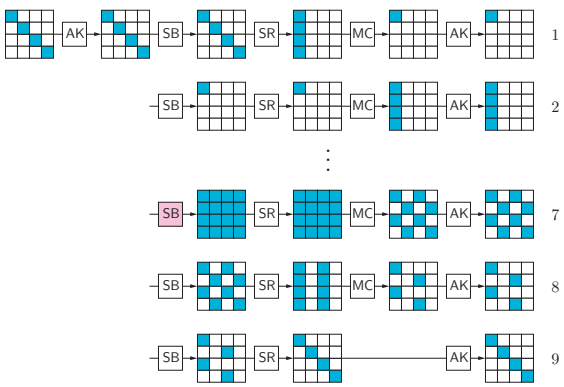

Fig. 4. The differential path used by the collision attack on the compression function f_0 instantiated with 9-round AES-256. Colored bytes are non-zero differences. The differential path from the third round to the sixth round is omitted since it is equal to the differential path from the second round to the fifth round in Fig. 3.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by JSPS KAKENHI Grant Numbers 21240001 and 25330150.

References

1. F. Armknecht, E. Fleischmann, M. Krause, J. Lee, M. Stam, and J. P. Steinberger. The preimage security of double-block-length compression functions. In D. H. Lee and X. Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 233–251. Springer, 2011.
2. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The KECCAK sponge function family, 2008. `http://keccak.noekeon.org`.
3. A. Biryukov, D. Khovratovich, and I. Nikolić. Distinguisher and related-key attack on the full AES-256. In S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009. An extended version is "Cryptology ePrint Archive: Report 2009/241" at `http://eprint.iacr.org/`.
4. J. Black, P. Rogaway, T. Shrimpton, and M. Stam. An analysis of the blockcipher-based hash functions from PGV. *Journal of Cryptology*, 23(4):519–545, 2010.
5. A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, and Y. Seurin. Hash functions and RFID tags: Mind the gap. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299. Springer, 2008.
6. B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas Jr., C. H. W. Meyer, J. Oseas, S. Pilpel, and M. Schilling. Data authentication using modification detection codes based on a public one-way encryption function, mar 1990. U. S. Patent # 4,908,861.

7. A. Canteaut, editor. *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*. Springer, 2012.
8. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002.
9. A. Duc, J. Guo, T. Peyrin, and L. Wei. Unaligned rebound attack: Application to Keccak. In Canteaut [7], pages 402–421.
10. N. Ferguson. Observations on H-PRESENT-128. Crypto 2011 Rump Session, 2011. `http://www.iacr.org/cryptodb/archive/2011/CRYPTO/video/rump/`.
11. FIPS PUB 180-4. Secure hash standard (SHS), Mar. 2012.
12. FIPS PUB 197. Advanced encryption standard (AES), 2001.
13. E. Fleischmann, M. Gorski, and S. Lucks. Security of cyclic double block length hash functions. In Parker [28], pages 153–175.
14. S. Hirose. Provably secure double-block-length hash functions in a black-box model. In C. Park and S. Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2004.
15. S. Hirose. Some plausible constructions of double-block-length hash functions. In M. J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.
16. J. Jean, M. Naya-Plasencia, and T. Peyrin. Multiple limited-birthday distinguishers and applications. In T. Lange, K. Lauter, and P. Lisonek, editors, *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 533–550. Springer, 2013.
17. D. Khovratovich, I. Nikolić, and C. Rechberger. Rotational rebound attacks on reduced Skein. In M. Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2010.
18. L. R. Knudsen, P. Gauravaram, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. Grøstl – a SHA-3 candidate. `http://www.groestl.info`, 2008.
19. X. Lai and J. L. Massey. A proposal for a new block encryption standard. In I. Damgård, editor, *EUROCRYPT*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 1990.
20. X. Lai and J. L. Massey. Hash function based on block ciphers. In R. A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.
21. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer. The rebound attack and subspace distinguishers: Application to Whirlpool. Cryptology ePrint Archive, Report 2010/198, 2010. `http://eprint.iacr.org/`.
22. J. Lee and D. Kwon. The security of Abreast-DM in the ideal cipher model. *IEICE Transactions*, 94-A(1):104–109, 2011.
23. J. Lee and M. Stam. MJH: A faster alternative to MDC-2. In A. Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236. Springer, 2011.
24. J. Lee, M. Stam, and J. P. Steinberger. The collision security of Tandem-DM in the ideal cipher model. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 561–577. Springer, 2011.
25. F. Mendel, T. Peyrin, C. Rechberger, and M. Schläffer. Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2009.

26. F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In O. Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.
27. O. Özen and M. Stam. Another glance at double-length hashing. In Parker [28], pages 176–201.
28. M. G. Parker, editor. *Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009, Cirencester, UK, December 15-17, 2009. Proceedings*, volume 5921 of *Lecture Notes in Computer Science*. Springer, 2009.
29. T. Peyrin, H. Gilbert, F. Muller, and M. J. B. Robshaw. Combining compression functions and block cipher-based hash functions. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 315–331. Springer, 2006.
30. B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.
31. V. Rijmen and P. S. L. M. Barreto. The Whirlpool hash function. `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`, 2000.
32. V. Rijmen, D. Toz, and K. Varici. Rebound attack on reduced-round versions of JH. In S. Hong and T. Iwata, editors, *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 286–303. Springer, 2010.
33. R. Rivest. The MD5 message-digest algorithm. Request for Comments 1321 (RFC 1321), The Internet Engineering Task Force, 1992.
34. Y. Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. *IEICE Transactions on Fundamentals*, E96-A(1):121–130, 2013.
35. L. Wei, T. Peyrin, P. Sokołowski, S. Ling, J. Pieprzyk, and H. Wang. On the (in)security of IDEA in various hashing modes. In Canteaut [7], pages 163–179. The full version is "Cryptology ePrint Archive: Report 2012/264" at `http://eprint.iacr.org/`.

A Example of Collision for f_0 Instantiated with 6-round AES-256

Table 2 gives an example of collision for f_0 instantiated with 6-round AES-256.

Table 2. An example of collision for f_0 instantiated with 6-round AES-256

Δh_0	`ff000000 00000000 00000000 00000000`
h_0	`5950c89a 7243695a b5561aa0 78899ca7`
h_1	`e9141904 6ab77163 f77410dc 429d3463`
M	`f5e6ee51 ac004900 1d47b1e7 8394e656`
output	`6ecccd37 579174c9 457e605d f2cdeecb`