New Preimage Attacks on Hash Modes of AES-256

Deukjo Hong, Dong-Chan Kim, and Daesung Kwon

August 30, 2012

Abstract

We study the slow diffusion of the AES key schedule for 256-bit keys and find weakness which can be used in the preimage attack on its Davis-Meyer mode. Our preimage attack works for 8 rounds of AES-256 with the computational complexity of $2^{124.9}$, while the best previous attack works for 7 rounds of AES-256. It is also extended to the preimage attack on some well-known double-block-length hash modes assuming the underlying block cipher is 8-round AES-256, whose computational complexity is $2^{252.9}$.

1 Introduction

Block ciphers and hash functions are widely used popular cryptographic primitives. Many hash functions are often designed based on block-cipher-like components. Some of them can be regarded as hash modes of block ciphers. PGV modes [22] are representative single-block-length (SBL) hash modes, where the length of the chaining and hash values is the same as the block length of the underlying block cipher. There are several double-block-length (DBL) hash modes, where the length of the chaining and hash value is twice as long as the block length of the underlying block cipher.

Sasaki presented the preimage attacks on PGV modes of 7-round AES [24]. In Sasaki's framework, the pseudo-preimage attack on Davis-Meyer mode is firstly researched and then extended to preimage attacks or second-preimage attacks on 12 secure PGV hash modes. His attack works for 7 rounds of AES, regardless of key size because it does not use any properties from the key schedule. So, it can be applied to 7 rounds of SQUARE [6] and CRYPTON [18], which are the block ciphers having almost same encryption structure as AES. [11] and [21] apply Sasaki's framework to reduced rounds of several block ciphers — 5 rounds of ARIA, 7 rounds of Camellia, and 4 rounds of Serpent, and a few rounds of block cipher structures — Feistel, generalized Feistel, Misty, and generalized Misty schemes, respectively. [20] discusses the preimage and second-preimage resistance of various double-block-length hash modes based on Sasaki's framework.

In this paper, we study the slow diffusion of the AES key schedule for 256-bit keys to find some weakness which can be used in the meet-in-the-middle preimage attack. Then, we shows a preimage attack on Davis-Meyer mode with 8-round AES-256. It requires the computational complexity of $2^{124.9}$. Our approach follows the previous preimage attacks on dedicated hash functions rather than block ciphers, because our attack uses the neutral bytes from the round keys produced by the key schedule instead of the internal state. For 256-bit keys, our result improves the previous Sasaki's work. We also point out this result can be extended to the preimage attacks on the well-known double-block-length hash modes assuming the underlying block cipher is 8-round AES-256, whose computational complexity is $2^{252.9}$.

This paper is organized as follows. In Section 2, we give brief descriptions about treated algorithms and schemes. In Section 3, we explain the previous work for the meet-in-the-middle preimage attack. Section 4 presents a method to construct neutral bytes for the AES key schedule of 256-bit keys with 8 rounds. Sections 5 describes the preimage attacks on Davis-Meyer modes of 8-round AES-256, and its complexities. In Section 6, we discuss the extension of our attack to the well-known DBL hash modes. In Section 7, we conclude this paper.

2 Specifications

2.1 AES

Advanced Encryption Standards (AES) [9] is a 128-bit block cipher supporting three different key sizes of 128, 192, and 256 bits. AES-256 has 14 rounds and a 256-bit key, which is two time larger than the internal state. Since each round needs a 128-bit round key, the key schedule consists of only 7 rounds. Let K be a 256-bit secret key and $K[i]$ be the i-th byte of K for $0 \leq i \leq 31$. The first two round keys RK_0 and RK_1 are defined as $RK_0 = K[0]\| \cdots \|K[15]$ and $RK_1 = K[16]\| \cdots \|K[31]$, respectively. The other round keys are generated as follows: for $l = 0, 1, ...,$

$$\begin{aligned}
RK_{2l+2}[i] &\leftarrow S(RK_{2l+1}[12 + (i+1 \bmod 4)]) \\
&\quad \oplus RK_{2l}[i] \oplus RC_l[i], \quad 0 \leq i \leq 3; \\
RK_{2l+2}[4i+j] &\leftarrow RK_{2l+2}[4(i-1)+j] \oplus RK_{2l}[4i+j], \\
&\quad 1 \leq i \leq 3, 0 \leq j \leq 3; \\
RK_{2l+3}[i] &\leftarrow S(EK_{2l+2}[12+i]) \oplus RK_{2l+1}[i], \\
&\quad 0 \leq i \leq 3; \\
RK_{2l+3}[4i+j] &\leftarrow RK_{2l+3}[4(i-1)+j] \oplus RK_{2l+1}[4i+j], \\
&\quad 1 \leq i \leq 3, 0 \leq j \leq 3,
\end{aligned}$$

where $S()$ and RC_l stand for the S-box and the round-dependent constant, respectively.

In the encryption procedure, the internal state is represented by a 4×4 byte array, where the index (i, j) of each entry is corresponded to the byte order $4i + j$ for $0 \leq i, j \leq 3$ (Fig. 1). The data is processed as follows. After the key schedule generates 128-bit round keys $RK_0, RK_1, ..., RK_{14}$, the plaintext is translated to State_{00} via XORing with the first round key RK_0. Then, a round function consisting of the following operations is iteratively applied:

- SubBytes (SB): substitute each byte according to an S-box table. The output of this operation is denoted by State_{i1}.
- ShiftRows (SR): apply the j-byte left rotation to each byte at row j. The output of this operation is denoted by State_{i2}.
- MixColumns (MC): multiply each column by an MDS matrix. MDS guarantees that the number of active bytes in the input and output of the MixColumns operation is at least 5 unless all bytes are non-active. The output of this operation is denoted by State_{i3}. The MDS matrix and its inverse matrix are defined as follows.

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}^{-1} = \begin{pmatrix} e_x & b_x & d_x & 9_x \\ 9_x & e_x & b_x & d_x \\ d_x & 9_x & e_x & b_x \\ b_x & d_x & 9_x & e_x \end{pmatrix}.$$

- AddRoundKey (AK): apply XOR with the round key RK_{i+1}. The output of this operation is denoted by $\text{State}_{(i+1)0}$.

Note that the MixColumns operation is not computed at the last round. We often denote several bytes of a state State_{ij} by $\text{State}_{ij}[a, b, ...]$ for the indices $a, b, ...$ or $\text{State}_{ij}[\mathcal{I}]$ for the index set $\mathcal{I} = \{a, b, ...\}$. For example, 4 bytes in the right most column of State_{00} are denoted by $\text{State}_{00}[12, 13, 14, 15]$. We apply the same notation to bytes of round keys.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Figure 1: Byte position

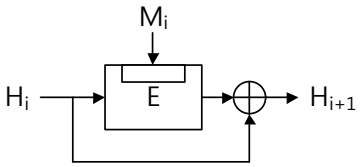

Figure 2: Davis-Meyer mode

2.2 Merkle-Damgård Domain Extension

One popular method to build a hash function with variable-length inputs is to use a domain extension for iteratively applying a compression function. The Merkle-Damgård scheme [19] is a mostly used domain extension in practice. It applies the padding to the input message M so that the last block includes the original message length, and parses the padded message to $M_0\|M_1\|...\|M_{L-1}$, where the size of each M_i is the block length. In this paper, we assume the most popular padding rule, used in SHA-1 and SHA-2 [8]; appending the bit strings '100...' and the 64-bit encode of the message length to the message. H_0 is fixed as an initial value, and $H_{i+1} = \mathrm{CF}(H_i, M_i)$ is iteratively applied for $i = 0, 1, ..., L-1$. Finally, H_L is output as a hash value of M.

Throughout this paper, we assume that we are given a hash function based on the Merkle-Damgård domain extension and the compression function CF. For a given hash value y, a preimage of the hash function is a message M leading to the hash value y. For a randomly chosen message M, a second-preimage of the hash function is a message M' leading to the same hash value for M. For a given output y, a pseudo-preimage is a pair of $(x, M), x \neq H_0$ such that $\mathrm{CF}(x, M) = y$.

2.3 Davis-Meyer Mode

We denote a block cipher E with a key K by E_K. Davis-Meyer mode [19, Algorithm 9.42] is a way to build a single-block-length compression function from a block cipher for the application to Merkle-Damgård domain extension. It defines the compression function $\mathrm{CF}(H_i, M_i)$ as follows (Fig. 2):

$$\mathrm{CF}(H_i, M_i) = E_{M_i}(H_i) \oplus H_i. \tag{1}$$

Davis-Meyer mode was proved to be collision-resistant and preimage-resistant in the ideal cipher model [5], as one of secure PGV hash modes [22].

2.4 Double-block-length Hash Modes

We apply our attack results on AES-256 to DBL hash modes whose underlying block cipher has n-bit key and $2n$-bit block. Let E be a block cipher with n-bit key and $2n$-bit block. For an n-bit plaintext P and a $2n$-bit key $K = K_0\|K_1$, we define a function G as follows:

$$G(P, K_0, K_1) = E_{K_0\|K_1}(P) \oplus P, \tag{2}$$

In [10], Hirose presented a DBL hash mode whose compression function $\mathrm{CF}(H_i, M_i) = H_{i+1}$ is defined as follows (Fig. 3):

$$H_{i+1,0} = G(H_{i,0}, H_{i,1}, M_i); \tag{3}$$
$$H_{i+1,1} = G(H_{i,0} \oplus Con, H_{i,1}, M_i), \tag{4}$$

where $H_i = H_{i,0}\|H_{i,1}$ and $H_{i+1} = H_{i+1,0}\|H_{i+1,1}$ are $2n$-bit chaining variables, M_i is an n-bit message block, and Con is an n-bit nonzero constant.

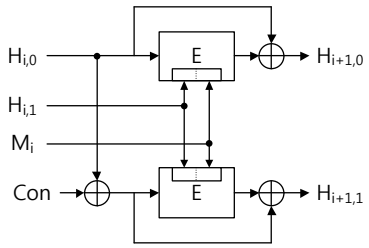

Figure 3: Hirose's DBL mode

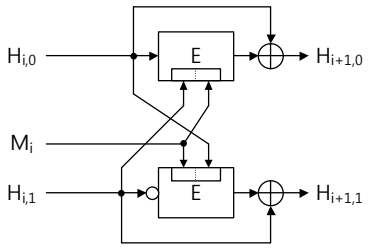

Figure 4: Abreast-DM mode

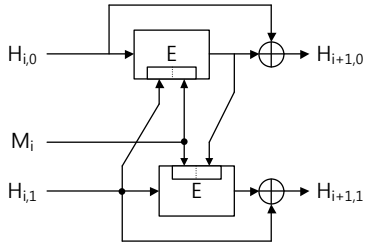

Figure 5: Tandem-DM mode

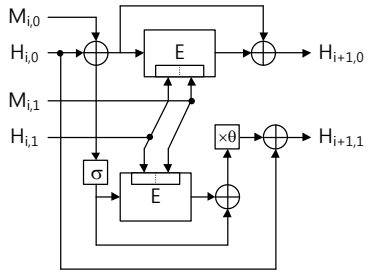

Figure 6: MJH mode

In [13], Lai and Massey proposed two DBL hash modes, Abreast-DM and Tandem-DM. In Abreast-DM, the compression function $\mathrm{CF}(H_i, M_i) = H_{i+1}$ is defined as follows (Fig. 4):

$$H_{i+1,0} = G(H_{i,0}, H_{i,1}, M_i); \quad (5)$$
$$H_{i+1,1} = G(\overline{H_{i,1}}, M_i, H_{i,0}) \oplus 1^n, \quad (6)$$

where $H_i = H_{i,0} \| H_{i,1}$ and $H_{i+1} = H_{i+1,0} \| H_{i+1,1}$ are $2n$-bit chaining variables, M_i is an n-bit message block, 1^n is all-1-bit string 11...1 with n-bit length, and $\overline{X}$ is the complement of any n-bit value X, i.e., $\overline{X} = X \oplus 1^n$. In Tandem-DM, the compression function $\mathrm{CF}(H_i, M_i) = H_{i+1}$ is defined as follows (Fig. 5):

$$H_{i+1,0} = G(H_{i,0}, H_{i,1}, M_i); \quad (7)$$
$$H_{i+1,1} = G(H_{i,1}, M_i, H_{i,0} \oplus H_{i+1,0}). \quad (8)$$

In [15], Lee and Stam proposed a DBL hash mode, MJH. It can use any block cipher with $n \leq k$, where n is block size and k is key size, but we fix $k = 2n$ in order to apply our attack on AES-256. With this setting, its compression function $\mathrm{CF}(H_i, M_i) = H_{i+1}$ is defined as follows (Fig. 6):

$$H_{i+1,0} = G(H_{i,0} \oplus M_{i,0}, H_{i,1}, M_{i,1}); \quad (9)$$
$$H_{i+1,1} = G(\sigma(H_{i,0} \oplus M_{i,0}), H_{i,1}, M_{i,1}) \cdot \theta \oplus H_{i,0}, \quad (10)$$

where $H_i = H_{i,0} \| H_{i,1}$ and $H_{i+1} = H_{i+1,0} \| H_{i+1,1}$ are $2n$-bit chaining variables, $M_i = M_{i+1,0} \| M_{i+1,1}$ is a $2n$-bit message block, σ is an involution map, and $\cdot\theta$ is a multiplication with a constant $\theta \in \mathbb{F}_{2^n} \setminus \{0, 1\}$.

The above DBL hash modes were proved to be collision-resistant in the ideal cipher model [10, 14, 15, 16]. Hirose, Abreast-DM, and Tandem-DM hash modes were also proved to be preimage-resistant in the ideal cipher model [4].

2.5 Meet-in-the-middle Preimage Attack

The meet-in-the-middle (MITM) preimage attack framework developed by Aoki and Sasaki [2, 3] is based on Leurent's work [17]. The basic approach of this framework is the separation of the compression function. The compression function is divided into *backward and forward chunks* such that a piece of the

input message affects only the backward chunk and another piece affects only the forward chunk. We call such pieces of input message *neutral words (or bytes or bits)*. This separation allows us to apply the meet-in-the-middle technique.

Additionally, several techniques have been developed to extend this basic approach. The *splice-and-cut* technique [3] regards that the first and last steps are consecutive, and thus any step can be the start point or matching point of the MITM attack. However, the results become pseudo-preimages rather than preimages. The *partial-matching/-fixing* techniques [3] and *indirect partial-matching* technique [1] allow to check the match at several steps instead of a point, and so make the matching check of two chunks efficient. Finally, the *initial-structure* technique [23] allow to start the meet-in-the-middle attack with some skipped steps.

In n-bit narrow-pipe iterated hash functions based on Merkle-Damgård domain extender, pseudo-preimage attacks with a complexity of 2^x, where $x < n - 2$, can be converted to preimage attacks with a complexity of $2^{\frac{n+x}{2}+1}$ in generic [19, Fact 9.99]. If the pseudo-preimage attack is applied to the last compression function and the padding rule can be satisfied with the message length of 2 blocks, the conversion is briefly described as follows:

1. Perform the pseudo-preimage attack for the given hash value and $2^{\frac{n-x}{2}}$ distinct settings of random choices, and keep all the attack results (H_1, M_1) in a table.

2. Compute $H_1' = \mathrm{CF}(H_0, M_0)$ for the initial value H_0 and $2^{\frac{n+x}{2}}$ distinct random choices for the first message block M_0, and check whether each computation result (H_1', M_0) has a match $H_1' = H_1$ for any entry (H_1, M_1) in the table. If a match is found, then the corresponding $M_0 \| M_1$ is a preimage for the given hash value.

It is easy to see the whole complexity is estimated as $2^{\frac{n-x}{2}} \cdot 2^x + 2^{\frac{n+x}{2}} = 2^{\frac{n+x}{2}+1}$. This conversion method requires a memory for $2^{\frac{n-x}{2}}$ values of (H_1, M_1), and can be modified to a memoryless method with the computational complexity $2^{\frac{n+x}{2}+2}$ [19, Chapter 9]. However, we do not need to consider the memoryless method since $2^{\frac{n-x}{2}}$ is usually very small.

In [24], Sasaki presents how to apply MITM preimage attack to block ciphers. He fixes a key input of the block cipher as a constant, and uses some pieces of an intermediate variable as neutral words to perform a kind of meet-in-the-middle attack. Sasaki's attack finds a pseudo-preimage of Davis-Meyer mode for 7-round AES, regardless of key size. It leads to a preimage attack on Davis-Meyer mode with 7-round AES. He also showed that if key size is fixed as 128 bits, then it can be used to estimate preimage or second-preimage attack complexities for other PGV modes. This framework can be applied to 7 rounds of SQUARE and CRYPTON, which are the block ciphers having almost same encryption structure as AES. [11] and [21] exploited Sasaki's framework to reduced rounds of several block ciphers — 5 rounds of ARIA, 7 rounds of Camellia, and 4 rounds of Serpent, and a few rounds of block cipher structures — Feistel, generalized Feistel, Misty, and generalized Misty schemes, respectively. [20] discusses the preimage and second-preimage resistance of various double-block-length hash modes using the Sasaki's pseudo-preimage attack on underlying block ciphers.

Since our approach in this paper exploits the slow diffusion of the key schedule for 256-bit keys, our attack is similar to previous attack on dedicated hash functions rather than block ciphers.

2.6 Expandable Message

If we can intentionally control the length information contained in the last block of the padded message to a small number (e.g., 2) in the pseudo-preimage attack, the conversion of the pseudo-preimage attack to the preimage attack follows the scenario mentioned in the previous Section 2.5. However, if the length of messages is not controlled, we cannot use the conversion method directly. In this case, expandable messages [7, 12] are helpful. $(a, b, H_{\mathrm{start}}, H_{\mathrm{end}})$-expandable message is a set of messages which have the

same starting chaining variable H_{start} and the same ending chaining variable H_{end} in the Merkle-Damgård hash function with the lengths between $a < b$ blocks.

For the compression function CF, we call (H, M), a fixed-point, such that $H = \text{CF}(H, M)$. Assume the length of each chaining variable is ℓ bits. If it takes the computational complexity of 2^x to find a fixed-point with $x < \ell$, then we can construct an expandable message with H_{start} as follows:

1. Find $2^{\frac{\ell-x}{2}}$ fixed points (H, M).
2. For $2^{\frac{\ell+x}{2}}$ message blocks M^*, compute chaining variables $V = \text{CF}(H_{\text{start}}, M^*)$.
3. If there are H and V such that $H = V$, output $\{M^*, M\}$ as a $(2, \infty, H_{\text{start}}, H)$-expandable message for the corresponding M^* and M.

We can use this expandable message to construct any message of length greater than 1 block as the form of $M^*\|M\|...\|M$ such the chaining variables start with H_{start} and end with H. Note that it is very easy to find a fixed-point for the Davis-Meyer mode by letting the output of the inner permutation be zero and choosing a message block.

3 Neutral Byte Arrangement

We arrange the neutral bytes as Fig. 7. We choose $RK_3[0, 1, 2]$ as the neutral bytes for the forward chunk. These bytes have effects on $RK_1[0, 1, 2, 4, 5, 6]$, $RK_5[all \setminus \{3, 7, 11, 15\}]$, $RK_6[all \setminus \{2, 6, 10, 14\}]$, $RK_7[all]$, and $RK_8[all \setminus \{2, 6, 10, 14\}]$, where *all* means the set of all byte positions, i.e., $\{0, 1, ..., 15\}$. They are colored in blue in Fig. 7. Furthermore, we minimize the effects of $RK_3[0, 1, 2]$ toward backward direction at the cost of the degree of freedom. We choose two 1-byte constants δ_0 and δ_1, and use the solutions to the following equations for the neutral bytes:

$$e_x \cdot RK_3[0] \oplus b_x \cdot RK_3[1] \oplus d_x \cdot RK_3[2] = \delta_0; \quad (11)$$

$$9_x \cdot RK_3[0] \oplus e_x \cdot RK_3[1] \oplus b_x \cdot RK_3[2] = \delta_1. \quad (12)$$

In backward computation, State_{30} goes to State_{23} by XORing with RK_3, and State_{23} is translated into State_{22} by the inverse of MixColumn. Since MixColumn is linear, the values of $RK_3[0, 1, 2]$ satisfying (11) and (12) are related with only two bytes $\text{State}_{22}[2, 3]$.

We choose $RK_6[2]$ as the neutral byte for the backward chunk. It has effects on $RK_0[2, 6, 10, 14]$, $RK_2[2, 6, 10]$, $RK_4[2, 6]$, and $RK_8[2, 6, 10, 14]$. They are colored in red in Fig. 7, except $RK_8[2, 6, 10, 14]$, which are colored white. Note that $RK_8[2, 6, 10, 14]$ cannot be considered as neutral bytes because they are affected from both backward and forward neutral bytes. The other bytes in Fig. 7 are colored in gray. Consequently, in Fig. 7, gray bytes are fixed as constants, and red and blue bytes do not interfere with each other in the computations.

4 Preimage Attack on Davis-Meyer mode of 8-round AES-256

Fig. 8 depicts the overview of the pseudo-preimage attack on Davis-Meyer mode of 8-round AES-256. It starts at State_{40}, whose value is randomly chosen. The forward chunk consists of the forward computations from State_{43} to State_{80} and from State_{00} to State_{02}. The backward chunk consists of the backward computations from State_{40} to State_{03}. We have a partial-matching check between State_{02} and State_{03}. Firstly, we explain how to check the match between two chunks. Then, we describe whole procedure of the pseudo-preimage attack.

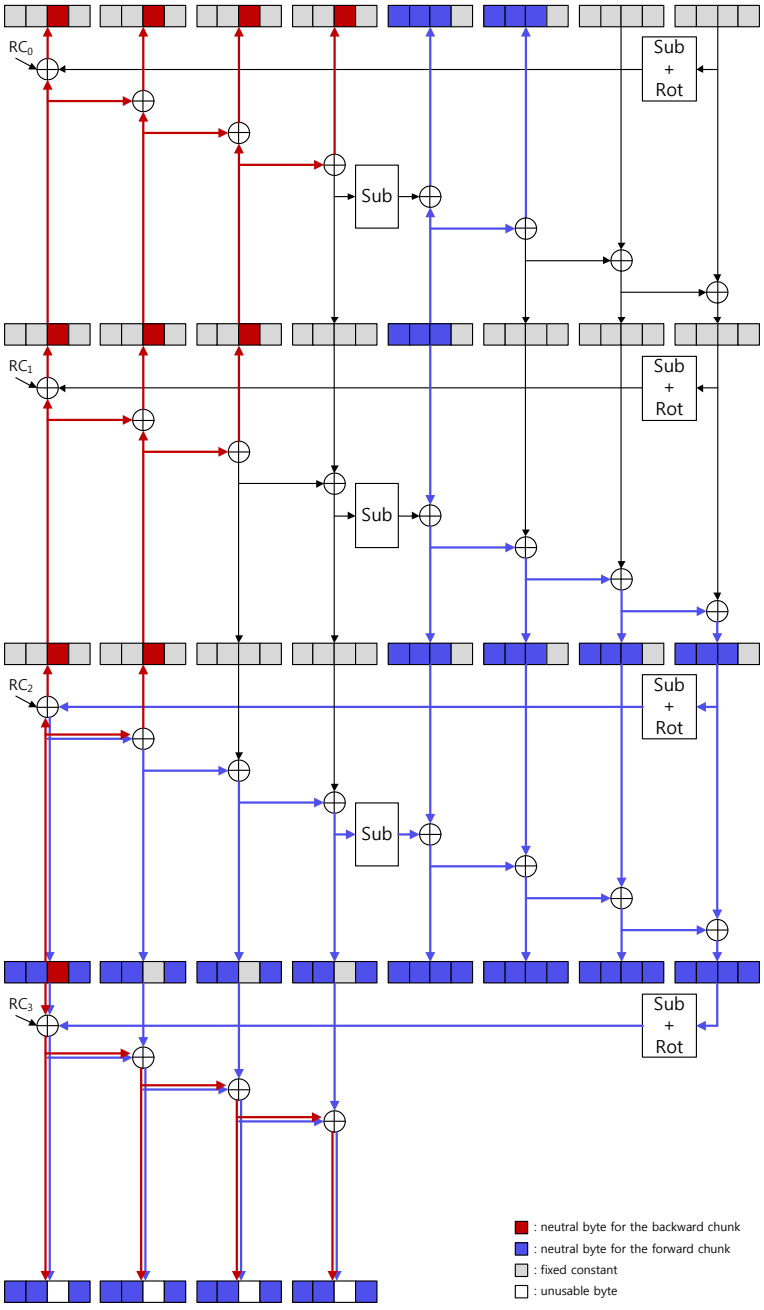

Figure 7: Neutral bytes of AES-256

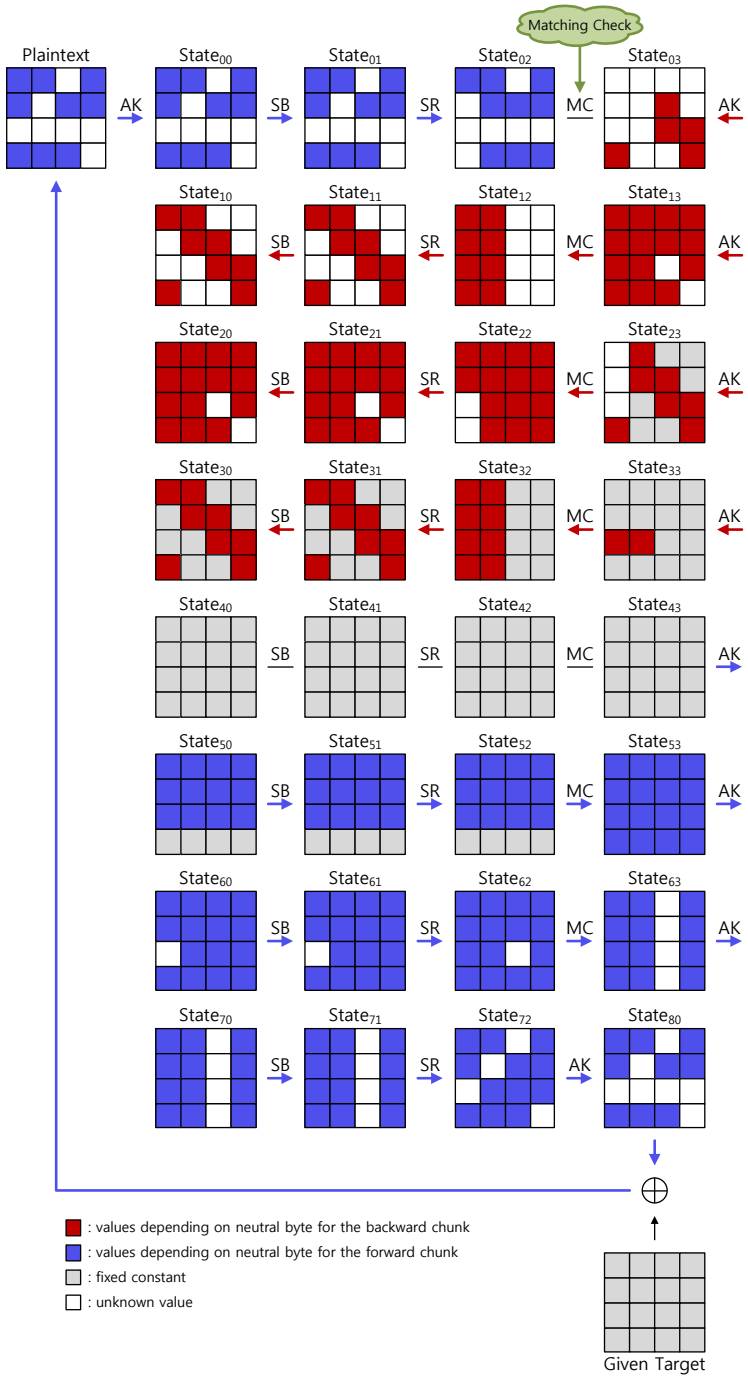

Figure 8: Pseudo-preimage attack on Davis-Meyer compression function of 8-round AES-256

4.1 Matching-Check Equation

We use an equation to check partially the match of the computation results in backward and forward chunks. In our attack, we obtain 9 bytes $\text{State}_{02}[0, 4, 5, 7, 9, 11, 12, 13, 15]$ as the result of the forward chunk computation and 5 bytes $\text{State}_{03}[3, 9, 10, 14, 15]$ as the result of the backward chunk computation. From the definition of the MixColumns layer, we have the following equations for the fourth columns of State_{02} and State_{03}.

$$Y[2] = X[0] \oplus X[1] \oplus 2_x \cdot X[2] \oplus 3_x \cdot X[3]; \quad (13)$$

$$Y[3] = 3_x \cdot X[0] \oplus X[1] \oplus X[2] \oplus 2_x \cdot X[3], \quad (14)$$

where $X[0, 1, 2, 3]$ denote $\text{State}_{02}[12, 13, 14, 15]$ and $Y[2, 3]$ denote $\text{State}_{03}[14, 15]$. We can cancel $X[2](= \text{State}_{02}[14])$ by multiplying (14) by 2_x and XORing the result with (13). Consequently, we get a matching-check equation as follows.

$$Y[2] \oplus 2_x \cdot Y[3] = 7_x \cdot X[0] \oplus 3_x \cdot X[1] \oplus 7_x \cdot X[3]. \quad (15)$$

4.2 Pseudo-Preimage Attack Procedure

Let M be the 256-bit padded message block of the last compression function. The procedure finding a pseudo-preimage for the last block is as follows.

1. **Initialization**: Initialize the message block M in terms of the round keys RK_i as follows.
   (a) $RK_1[7, ..15]$ are set such that they satisfy the padding rule of the message length of 2 blocks.
   (b) $RK_0[all \setminus \{2, 6, 10, 14\}]$, $RK_1[3]$, $RK_2[14]$, $RK_3[4, 5, 6]$, $RK_4[10]$, and $RK_6[6]$ are fixed to randomly chosen 1-byte constants.
   (c) Choose the internal state State_{40} randomly and then compute State_{43}.
   (d) The constants δ_0 and δ_1 in (11) and (12) are randomly chosen.
2. **Backward computation**: For all possible 2^8 values of $RK_6[2]$,
   (a) Compute the backward chunk from State_{40} to State_{03}. Note that we have unknown values for only two bytes at State_{22} due to (11) and (12).
   (b) Compute the left hand side of (15) with $\text{State}_{03}[14, 15]$.
   (c) Store the result in a table T_{back}.
3. **Forward computation**: For all possible 2^8 values of $RK_3[0, 1, 2]$ satisfying (11) and (12),
   (a) Compute the forward chunk from State_{43} to State_{02} according to Fig. 8.
   (b) Compute the right hand side of (15) with $\text{State}_{02}[12, 13, 15]$.
   (c) Check whether there exists an entry in T_{back} that matches the result of the step 3-b.
   (d) If a match is found in the step 3-c, compute all bytes of State_{02} and State_{03} for the matched pair, and check whether the other bytes match.
   (e) If all bytes match, output the corresponding chaining value and message block (H, M).
4. **Repetition**: If the above steps do not succeed, go back to the step 1 and repeat the attack with different setting of random choices.

4.3 Complexity

In the backward computation phase of the attack procedure, we estimate the computational complexities of the steps for each trial as follows.

- Computation from State_{40} to State_{30}: 1 round,
- Computation from State_{30} to State_{20}: $\frac{14}{16}$ round,
- Computation from State_{20} to State_{10}: $\frac{8}{16}$ round.

So, the complexity of the backward computation for one iteration of the attack procedure is estimated as $2^8 \cdot \frac{19}{8}$ rounds.

Similarly, in the forward computation phase of the attack procedure, we estimate the computational complexities of the steps for each trial as follows.

- Computation from State_{43} to State_{53}: 1 round,
- Computation from State_{53} to State_{63}: $\frac{12}{16}$ round,
- Computation from State_{63} to State_{80}: $\frac{9}{16}$ round,
- Computation from State_{80} to State_{02}: $\frac{9}{16}$ round.

So, the complexity of the forward computation for one iteration of the attack procedure is estimated as $2^8 \cdot \frac{23}{8}$ rounds.

The 8-bit partial matching in the step 3-c occurs with the probability 2^{-8}. So, we expect 2^8 computations of step 3-d, whose complexity is estimated as $2^8 \cdot \frac{7}{4}$ rounds. In total, the computational complexity for one iteration of the attack procedure is estimated as

$$2^8 \cdot \left(\frac{19}{8} + \frac{23}{8} + \frac{7}{4} \right) = 2^8 \cdot 7$$

rounds, which is translated to $2^{7.8}$ encryptions of 8-round AES-256. For one iteration of the attack procedure, a full-byte match occurs with the probability $2^{16}/2^{-128} = 2^{-112}$. So, we expect the number of the repetition is close to 2^{112} until a match is found. It requires the computational complexity of $2^{119.8}$ and the memory for storing 2^8 bytes.

If we assume the most popular padding rule used in SHA family [8]: appending the bit strings '100...' and the 64-bit encode of the message length to the message, we can control the length information in the last block such that the length of the padded message is 2 blocks. Then, as explained in Section 3, we can convert this pseudo-preimage attack to a preimage attack on Davis-Meyer mode with 8-round AES-256 with the computational complexity of $2^{(128+119.8)/2+1} = 2^{124.9}$. The hash function may use a different padding method from what we assumed. In this case, we can use an expandable message with fixed-points mentioned in Section 2.6.

5 Application to Hash Modes

In this section, we discuss the securities of the DBL hash modes mentioned in Section 2.4, assuming 8-round AES-256 is the underlying block cipher. For Hirose's DBL hash mode, we apply our pseudo-preimage attack to (3), and obtain the attack result, $(H_{1,0}, H_{1,1}, M_1)$. Then, (4) is expected to hold for $(H_{1,0}, H_{1,1}, M_1)$ with the probability 2^{-128}. So, we can iterate the above steps 2^{128} times to get a pseudo-preimage. It requires the computational complexity of $2^{128+119.8} = 2^{247.8}$. With similar ways, we can find pseudo-preimages for Abreast-DM, Tandem-DM, and MJH mode with the same complexity.

From the discussion in Section 2.6, we know the fixed-points make expandable messages and expandable messages allow to overcome the padding issue. Fortunately, it takes 2^{n-1} compression operations to find a fixed-point for each of the target DBL modes, where n is the block length of the underlying block cipher. Each of them contains two Davis-Meyer modes. So, we compute a fixed-point for one Davis-Meyer mode, and then expect that it makes a fixed-point for the other one. The probability of this event is 2^{-n}. Therefore, we can expect a fixed-point for the compression function by repeating this computation 2^n times, and make an expandable message with the complexity of $2^{\frac{3n+1}{2}}$. Note that for MJH mode, we do not need to fix the output of the block cipher to zero. In the case of 8-round AES-256, this complexity is much less than that of the pseudo-preimage attack on the DBL compression function.

From the above arguments, we reach the preimage attacks for the Hirose, Abreast-DM, Tandem-DM, and MJH based on 8-round AES-256 with the computational complexity of $2^{(256+247.8)/2+1} = 2^{252.9}$.

6 Conclusion

We have presented the preimage attack on Davis-Meyer mode with 8-round AES-256, and extended it to the DBL hash modes such as Hirose, Abreast-DM, Tandem-DM, and MJH. Our result improves the previous preimage attack for AES-256, and shows that some weak property of the key schedule can be used for the preimage attack on hash modes of block ciphers like the message schedules of the dedicated hash functions.

References

[1] K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki and L. Wang, "Preimages for Step-Reduced SHA-2," In M. Matsui (Ed.), *ASIACRYPT 2009*, LNCS 5912, pp. 578–597, Springer-Verlag, 2009.

[2] K. Aoki and Y. Sasaki, "Meet-in-the-Middle Preimage Attacks against Reduced SHA-0 and SHA-1," In S. Halevi (Ed.), *CRYPTO 2009*, LNCS 5677, pp. 70–89, Springer-Verlag, 2009.

[3] K. Aoki and Y. Sasaki, "Preimage Attacks on One-Block MD4, 63-Step MD5 and More," In R. M. Avanzi, L. Keliher, and F. Sica (Eds.), *SAC 2008*, LNCS 5381, pp. 103–119, Springer-Verlag, 2009.

[4] F. Armknecht, E. Fleischmann, M. Krause, J. Lee, M. Stam, J. Steinberger, "The Preimage Security of Double-Block-Length Compression Functions," In D. H. Lee and X. Wang (Eds.), *ASIACRYPT 2011*, LNCS 7073, pp. 233-251, Springer-Verlag, 2011.

[5] J. Black, P. Rogaway, and T. Shrimpton, "Black-Box Analysis of the Block-Cipher-Based Hash-Function Construction from PGV," In M. Yung (Ed.), *CRYPTO 2002*, LNCS 2442, pp. 320–335, Springer-Verlag, 2002.

[6] J. Daemen, L. R. Knudsen, and V. Rijmen, "The Block Cipher Square," In E.Biham (Ed.), *FSE'97*, LNCS 1267, Springer-Verlga, pp. 149–165, 1997.

[7] R. D. Dean, *Formal Aspects of Mobile Code Security*, Ph. D Dissertation, Princeton University, January 1999.

[8] Secure Hash Standard (SHS), Federal Information Processing Standards Publication 180-2, August 1, 2002. Amended February 25, 2004.

[9] Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, November 26, 2001.

[10] S. Hirose, "Some Plausible Constructions of Double-Block-Length Hash Functions," In M. J. B. Robshaw (Ed.), *FSE 2006*, LNCS 4047, pp. 231–246, Springer-Verlag, 2006.

[11] D. Hong, B. Koo, and D.-C. Kim, "Preimage and Second-Preimage Attacks on PGV Hashing Modes of Round-Reduced ARIA, Camellia, and Serpent," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, Vol. 95-A, No. 1, pp. 372–380, 2012.

[12] J. Kelsey and B. Schneier, "Second Preimages in n-Bit Hash Functions for Much Less Than 2n Work," In R. Cramer (Ed.), *EUROCRYPT 2005*, LNCS 3494, pp. 474–490, Springer-Verlag, 2005.

[13] X. Lai and J. L. Massey, "Hash function based on block ciphers," In R. A. Rueppel (Ed.), *EUROCRYPT'92*, LNCS 658, pp. 55–70, Springer-Verlag, 1993.

[14] J. Lee and D. Kwon, "The Security of Abreast-DM in the Ideal Cipher Model," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, Vol. 94-A, No. 1, pp. 104–109, 2011.

[15] J. Lee and M. Stam, "MJH: A Faster Alternative to MDC-2," In A. Kiayias (Ed.), In A. Kiayias (Ed.), *CT-RSA 2011*, LNCS 6558, pp. 213–236, Springer-Verlag, 2011.

[16] J. Lee, M. Stam, and J. P. Steinberger, "The Collision Security of Tandem-DM in the Ideal Cipher Model," In P. Rogaway (Ed.), *CRYPTO 2011*, LNCS 6841, pp. 561–577, Springer-Verlag, 2011.

[17] G. Leurent, "MD4 is not one-way," In K. Nyberg (Ed.), *FSE 2008*, LNCS 5086, pp. 412–428, Springer-Verlag, 2008.

[18] C. H. Lim, "A Revised Version of Crypton — Crypton V1.0," In L. R. Knudsen (Ed.), *FSE'99*, LNCS 1636, Springer-Verlag, pp. 31–45, 1999.

[19] A. J. Menezes, P. C. Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[20] D. Moon, D. Hong, B. Koo and S. Hong, "Security Evaluation of Double-Block-Length Hash Modes with Preimage Attacks on PGV Schemes," In the 2011 FTRA international Symposium on Advances in Cryptography, Security and Applications for Future Computing.

[21] D. Moon, D. Hong, D. Kwon and S. Hong, "Meet-in-the-Middle Preimage Attacks on Hash Modes of Generalized Feistel and Misty Schemes with SP Round Function," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, Vol. 95-A, No. 8, 2012.

[22] B. Preneel, R. Govaerts and J. Vandewalle, "Hash Functions Based on Block Ciphers: A Synthetic Approach," In D. R. Stinson (Ed.), CRYPTO 1993, LNCS 773, pp. 363–378, Springer-Verlag, 1994.

[23] Y. Sasaki and K. Aoki, "Finding preimages in full MD5 faster than exhaustive search," In A. Joux (Ed.), *EUROCRYPT 2009*, LNCS 5479, pp. 134–152, Springer-Verlag, 2009.

[24] Y. Sasaki, "Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool," In A. Joux (Ed.), *FSE 2011*, LNCS 6733, pp. 378–396, Springer-Verlag, 2011.