Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool

Yu Sasaki

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585 Japan
`sasaki.yu@lab.ntt.co.jp`

Abstract. We study the security of AES in the open-key setting by showing an analysis on hash function modes instantiating AES including Davies-Meyer, Matyas-Meyer-Oseas, and Miyaguchi-Preneel modes. In particular, we propose preimage attacks on these constructions, while most of previous work focused their attention on collision attacks or distinguishers using non-ideal differential properties. This research is based on the motivation that we should evaluate classical and important security notions for hash functions and avoid complicated attack models that seem to have little relevance in practice. We apply a recently developed meet-in-the-middle preimage approach. As a result, we obtain a preimage attack on 7 rounds of Davies-Meyer AES and a second preimage attack on 7 rounds of Matyas-Meyer-Oseas and Miyaguchi-Preneel AES. Considering that the previous best collision attack only can work up to 6 rounds, the number of attacked rounds reaches the best in terms of the classical security notions. In our attacks, the key is regarded as a known constant, and the attacks thus can work for any key length in common.

Keywords: AES, hash function, Davies-Meyer, Matyas-Meyer-Oseas, Miyaguchi-Preneel, PGV, preimage, meet-in-the-middle, Whirlpool

1 Introduction

Block ciphers are taking important roles in various aspects of our life. Currently, one of the most widely used block-ciphers all over the world is AES [12, 34].

Since 2009, great progress in the cryptanalysis on AES has been made. Related-key attacks against full-round AES-256 [6, 7], full-round AES-192 [6], 7-round AES-128 [9], and 10-round AES-256 with a practical complexity [5] have been proposed. Regarding AES-128, besides the above related-key boomerang attack [9] several non-marginal single-key attacks have been proposed; an impossible differential attack [25] and a single-key attack [15] based on a collision attack [16]. In any attack, the maximum number of attacked rounds is 7.

On the other hand, block ciphers are sometimes used as hash functions through mode-of-operations. For example, if one needs both a block-cipher and a hash function in a resource-restricted environment such as RFID Tag, only a block-cipher is implemented and a hash function is built using it. Besides, many Tag-based applications, such as authentication or anonymity/privacy, do not need the collision resistance [10]. Hence, building a 128-bit hash function with AES is a possible candidate. In fact, [10] proposed 80-bit and 64-bit hash functions using block-cipher PRESENT. Another concern is that many hash functions, even in the SHA-3 competition [35], are designed based on block-ciphers. Hence, block-ciphers' security in hashing modes is an interesting research object.

The known-key attack proposed by Knudsen and Rijmen [21] is the framework for this context. In this model, a secret key is randomly chosen and given to attackers. Then, attackers aim to efficiently detect a certain property of a random instance of the block cipher, where the same property cannot be observed for a random permutation with the same complexity. The attack can be extended to the chosen-key model. e.g. [7]. In the known-key model, the key size is irrelevant to the attack. In other words, all key sizes are simultaneously attacked. While, in the chosen-key model, the attack depends on the key-schedule algorithm. Hence, different strategies is necessary for different key sizes.

The first known-key attack was presented by Knudsen and Rijmen [21], which found a non-ideal property of 7-round AES. Then, Mendel *et al.* presented the known-key attack on 7-round AES [26] based on the rebound attack proposed by Mendel *et al.* [27]. Finally, Gilbert and Peyrin [17] and Lamberger *et al.* [22] independently applied Super-Sbox analysis to the rebound attack. Gilbert and Peyrin [17] showed that 8-round AES was not ideal in the known-key setting. Regarding the chosen-key attack, Biryukov *et al.* [7] presented a chosen-key distinguisher on full-round AES-256, which is converted to a q-pseudo-collision attack on AES-256 based compression functions. Biryukov and Nikolić also discovered a chosen-key distinguisher on 8-round AES-128 [8].

Although the above results led to significant progress for theoretical cryptanalysis in the secret-, known-, and chosen-key settings, one major drawback is the use of complicated attack models, which are sometimes too theoretic such as related-subkey attacks on block ciphers and distinguishers on block-cipher based compression functions. From this background, several researchers recently have attempted to analyze AES in a simple attack model. For example, Dunkelman *et al.* [15] and Wei *et al.* [36] avoided the related-key model and proposed attacks on 8 round AES-256 or AES-192 in the single-key model.

In this paper, we follow the similar principle as Dunkelman *et al.* [15] and Wei *et al.* [36]. That is to say, we analyze the security of hashing modes instantiating AES in terms of the classical security notions of hash functions, which are actually important for their applications. In particular, we study the preimage resistance of hash functions rather than compression functions.

For hash functions, three security notions are classically considered to be important; collision resistance, second-preimage resistance, and preimage resistance. Among these three, the collision resistance of reduced-round AES can be attacked by applying the techniques used in the rebound attack [27]. In fact, Lamberger *et al.* [23, Section 5.3] describe a collision attack on an AES-based hash function Whirlpool [30] reduced to 5.5 rounds, which is trivially converted

Table 1. Comparison of attacks. 0.5 round of collision and near-collision attacks on Whirlpool by [23] is omitted.

Attack	Rounds	Key-size	Mode	Comp. Func.	Hash	Ref.
				(Time, Mem.)	(Time, Mem.)	
Attacks on AES Hasing modes						
Collision	6	128/192/256	MMO,MP	$(2^{56}, 2^{32})$	$(2^{56}, 2^{32})$	[23]
2nd preimage	6	128/192/256	MMO,MP	$(2^{112}, 2^{16})$	$(2^{112}, 2^{16})$	Ours
2nd preimage	7	128/192/256	MMO,MP	$(2^{120}, 2^{8})$	$(2^{120}, 2^{8})$	Ours
Preimage	6	128/192/256	DM	$(2^{112}, 2^{16})$	$(2^{121}, 2^{16})$	Ours
Preimage	7	128/192/256	DM	$(2^{120}, 2^{8})$	$(2^{125}, 2^{8})$	Ours
Near collision	7	128/192/256	MMO,MP	$(2^{32}, 2^{32})$	$(2^{32}, 2^{32})$	[23]
Distinguisher	8	128/192/256	MMO,MP	$(2^{48}, 2^{32})$	-	[17]
q-multicollision	14	256	DM	$(q \cdot 2^{67}$, negl.)	-	[7]
Attacks on Whirlpool						
Collision	5	-	-	$(2^{120}, 2^{64})$	$(2^{120}, 2^{64})$	[23]
2ne Preimage	5	-	-	$(2^{504}, 2^{8})$	$(2^{504}, 2^{8})$	Ours
Near collision	7	-	-	$(2^{112}, 2^{64})$	$(2^{112}, 2^{64})$	[23]
Collision	7	-	-	$(2^{184}, 2^{8})$	-	[23]
Near collision	9	-	-	$(2^{176}, 2^{8})$	-	[23]
Distinguisher	10	-	-	$(2^{176}, 2^{8})$	-	[23]

to a collision attack on the Matyas-Meyer-Oseas mode [28, Algorithm 9.41] instantiating 6-round AES. As far as we know, there is no result that attacks second-preimage resistance or preimage resistance of such an AES usage. Note that the attack by [23] can generate near-collisions on some PGV compression functions with 7-round AES, which might be a valid security notion.

Our contributions. In this paper, we propose preimage attacks on AES hashing modes including Davies-Meyer (DM) [28, Algorithm 9.42], Matyas-Meyer-Oseas (MMO), and Miyaguchi-Preneel (MP) [28, Algorithm 9.43] modes. As a result, we obtain a preimage attack on 7 rounds of DM-AES with a complexity of 2^{125} 7-round AES computations and the memory of 2^{8} AES state. We also obtain a second preimage attack on 7 rounds of MMO-AES and MP-AES with a complexity of 2^{120} 7-round AES computations and the memory of 2^{8} AES state. Our attack can also generate second preimages of 5-round Whirlpool with a complexity of 2^{504}. The attack results are summarized in Table 1.

We apply a meet-in-the-middle preimage approach developed by Aoki and Sasaki [3]. This approach has successfully been applied to many hash functions e.g. MD5 [32] and Tiger [18]. All of previously analyzed hash functions adopt the DM mode with a relatively weak message schedule, and the weak message schedule is in fact exploited by the attack. However, for AES, the situation is very different because AES has a heavy round function and key schedule. Moreover, it is unclear how to perform preimage attacks against MMO and MP modes.

In our attacks, we fix the value of key-input to a randomly chosen value and search for a plaintext-input that achieves the given hash target. This allows us to attack All PGV modes [29] in the same procedure. We then show that the splice-and-cut technique proposed by [3] and the omission of a MixColumns operation in the last round can be combined well and lead to a significant improvement for the preimage attack. Intuitively, this is because the round function without MixColumns is computed as a middle round. This breaks the AES design principle, where AES two rounds achieve the full diffusion, and leads to an attack improvement. Finally, we optimize several techniques of the meet-in-the-middle preimage attack for AES. Specifically, the initial-structure and matching through MixColumns contribute to increase the number of attacked rounds.

Paper outline. In Sect. 2, we describe AES. In Sect. 3, we introduce previous work. In Sect. 4, we explain a basic idea of our attack. In Sect. 5, we present a preimage attack on 7-round AES. In Sect. 6, we give observations on our attack and apply it to 5-round Whirlpool. Finally, we conclude this paper in Sect. 7.

2 Specifications

Advanced Encryption Standard (AES) [34, 12] is a 128-bit block cipher supporting three different key sizes; 128, 192, 256 bits. AES computes 10, 12, and 14 rounds for AES-128, -192, and -256, respectively.

By using the key schedule function, round keys are generated from the original secret key. We omit its description because our attacks regard round keys as given constant numbers and thus they are irrelevant to our attacks.

When the data is processed, the internal state is represented by a $4 * 4$ byte array. At the first, the original secret key is XORed to the plaintext, and then, a round function consisting of the following four operations is iteratively applied.

- SubBytes(SB): substitute each byte according to an S-box table.
- ShiftRows(SR): apply the j-byte left rotation to each byte at row j.
- MixColumns(MC): multiply each column by an MDS matrix. MDS guarantees that the sum of active bytes in the input and output of the MixColumns operation is at least 5 unless all bytes are non-active. The matrices for the encryption and decryption are shown below. Note that $X[j]$ is the input value and $Y[j]$ is the updated value. Numbers with $_x$ are hexadecimal numbers.

$$\begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix} = \begin{pmatrix} 2\ 3\ 1\ 1 \\ 1\ 2\ 3\ 1 \\ 1\ 1\ 2\ 3 \\ 3\ 1\ 1\ 2 \end{pmatrix} \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix}, \quad \begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix} = \begin{pmatrix} {}_xe\ {}_xb\ {}_xd\ {}_x9 \\ {}_x9\ {}_xe\ {}_xb\ {}_xd \\ {}_xd\ {}_x9\ {}_xe\ {}_xb \\ {}_xb\ {}_xd\ {}_x9\ {}_xe \end{pmatrix} \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix} \quad (1)$$

- AddRoundKey(AK): apply bit-wise exclusive-or with a round key.

Note that the MixColumns operation is not computed at the last round.

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

Fig. 1. Byte positions

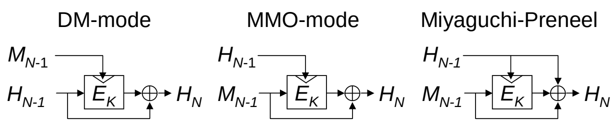

Fig. 2. Illustrations for DM, MMO, and MP modes

Byte positions in a state S are denoted by integer numbers $B, B \in \{0, 1, 2, \ldots, 15\}$, as shown in Fig. 1, where the byte $4j + i$ corresponds to the byte in the i-th row and j-th column of S, and is denoted by $S[4 \cdot j + i]$. We often denote several bytes of state S by $S[a, b, \ldots]$, e.g. 4 bytes in the right most column are denoted by $S[12, 13, 14, 15]$.

Hash functions based on block ciphers. To build a hash function, we need a domain extension for iteratively applying the compression function. The Merkle-Damgård domain extension is probably mostly used one in practice. It applies the padding to the input message M so that the last block includes the original message length, and splits the padded message to $M_0 \| M_1 \| \cdots \| M_{L-1}$, where the size of each M_N is the block length. An initial value H_0 is defined, and $H_N = \mathrm{CF}(H_{N-1}, M_{N-1})$ is iteratively applied for $N = 1, 2, \ldots, L$. Finally, H_L is output as a hash value of M. This paper assumes that the Merkle-Damgård domain extension is used as a domain extender.

The PGV modes [29] are mode-of-operations to build a compression function from a block cipher. In fact, many hash functions, e.g. MD5, SHA-2, and several SHA-3 candidates, use the PGV modes. Among PGV modes, the DM, MMO, and MP modes are used in practice. Let us denote a block cipher E with a key K by E_K. The construction of each mode is as follows, which is shown in Fig. 2.

$$\begin{aligned}
&\textbf{DM mode:} && \mathrm{CF}(H_{N-1}, M_{N-1}) = E_{M_{N-1}}(H_{N-1}) \oplus H_{N-1},\\
&\textbf{MMO mode:} && \mathrm{CF}(H_{N-1}, M_{N-1}) = E_{H_{N-1}}(M_{N-1}) \oplus M_{N-1},\\
&\textbf{MP mode:} && \mathrm{CF}(H_{N-1}, M_{N-1}) = E_{H_{N-1}}(M_{N-1}) \oplus M_{N-1} \oplus H_{N-1}.
\end{aligned}$$

3 Previous Work

3.1 Meet-in-the-Middle Preimage Attacks

To mount the preimage attack, we apply a meet-in-the-middle (MitM) preimage approach developed by Aoki and Sasaki [3], which is based on the pioneering work by Leurent [24][1]. In this approach, the compression function is divided into two sub-functions so that a portion of bits of the input message only affect one sub-function and another portion of bits of the input message only affect the other sub-function as shown in Fig. 3. This allows attackers to mount the meet-in-the-middle attack. Sub-functions are called *chunks* (stands for chunks of steps

[1] A basic idea of the MitM preimage finding technique can also be seen in [4] and [19].

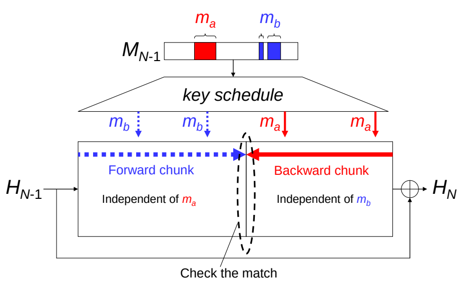

Fig. 3. Basic MitM attack on DM mode

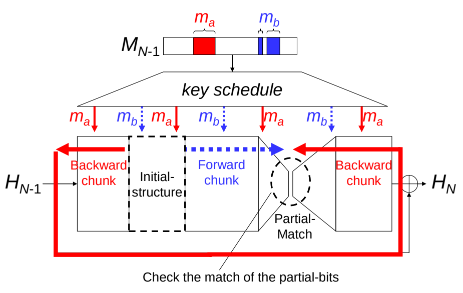

Fig. 4. Advanced techniques for MitM

or rounds) and bits affecting only one chunk are called *neutral*. In this paper, we call the chunk that computes the round function in the forward direction *forward chunk* and in the backward direction *backward chunk*.

In addition to the basic concept, several techniques have been proposed to extend the attack framework. 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, as a side-effect, generated items become pseudo-preimages rather than preimages. The *local-collision* technique [31], *initial-structure* technique [32] and *probabilistic initial-structure* technique [18] ignore the order of message words at the start point of the MitM attack. For example in Fig. 4, the order of neutral words m_a and m_b is reversed between the start points of the forward and backward chunks. These techniques enables MitM attacks even in such a situation. Finally, the *partial-matching/-fixing* techniques [3] and *indirect partial-matching* technique [1] match two chunks partially and efficiently. A framework with these techniques is illustrated in Fig. 4.

In n-bit narrow-pipe iterated hash functions, pseudo-preimage attacks with a complexity of 2^m, where $m < n-2$, can be converted to preimage attacks with a complexity of $2^{\frac{m+n}{2}+1}$ in generic [28, Fact9.99]. Several researchers showed that pseudo-preimage attacks satisfying certain special properties can be converted to preimage attacks more efficiently than the generic approach [11, 18, 24]. Because our attacks cannot satisfy such properties, we omit their details.

The MitM preimage approach has been applied to many hash functions such as HAVAL [31], MD5 [32], reduced SHA-0/-1 [2], reduced SHA-2 [1], and Tiger [18]. All of previously attacked hash functions adopt the DM mode and their weak key-schedules are exploited by the attack. This strategy cannot work for AES because, in the AES key-schedule, the impact of any change on the secret key or a subkey always propagate to all other subkeys. This indicates that neutral words such as described in Fig. 3 or Fig. 4 do not exist for AES. Moreover, if the message is input as a plaintext in the MMO and MP modes, no input value is available to separate the compression function into two parts.

3.2 Previous Analysis on AES

The security of AES in hash function modes was first evaluated by Knudsen and Rijmen [21]. They showed a non-ideal property of 7-round AES. Lamberger *et al.* showed a collision attack on 5.5-round Whirlpool based on the rebound attack [27], which is trivially converted to a collision attack on 6-round AES. As far as we know, no result is known on the second-preimage or preimage resistance. Note that current collision attacks can be applied only if the mode-of-operation is MMO or MP, where attackers fix H_{N-1} and search for (M_{N-1}, M'_{N-1}). If the DM mode is used instead, attackers fix M_{N-1} and search for (H_{N-1}, H'_{N-1}). Hence, only pseudo-collisions on the compression function can be generated.

As analysis methods against the AES block cipher, there exist attacks named collision attack [16] and Meet-in-the-Middle attack [13] (and their extension [15]). These attacks are not related to attacks on hash functions. These attacks based on an observation that a function from a certain input byte to a certain output byte after 4 rounds can be described by 25-byte parameters. Hence, this collision attack does not find paired messages producing an identical state, or this Meet-in-the-Middle attack does not separate the cipher into two independent sub-functions. The goal of these attacks is recovering the secret key of the AES block cipher. Their applicability to hashing modes is not understood well.

4 Basic Idea of Our Attack and Techniques for Extension

We first explain a basic idea of our attack by using 4-round AES as an example (Sect. 4.1). We fix the block-cipher's key to a constant. This approach is different from previous work in Sect. 3.1 which utilize the independence among subkeys. In this attack, for simplicity, we only apply the splice-and-cut technique. We then explain several techniques to extend the number of attacked rounds (Sect. 4.2).

4.1 Basic Attack for 4-Round AES

Goal of the attack. We fix the key-input when we perform the MitM attack, and the goal is to find the plaintext-input that provides the given target. This approach is irrelevant to the mode-of-operation used. That is, in the DM-mode, we fix a message M_{N-1} to some constant and try to find a chaining variable H_{N-1} that produces the given target H_N. Similarly, in the MMO- and MP-modes, we fix a chaining variable H_{N-1} and search for a message M_{N-1}. Generated pseudo-preimages are later converted to preimages with a technique in Sect. 3.1.

Chunk separation. We separate 4-round operations into two chunks as shown in Fig. 5. The start point of each chunk is state #9. We choose #9[0] as a neutral byte for the forward chunk and #9[12] for the backward chunk. We fix the other bytes, i.e. $\#9[1, 2, \ldots, 11, 13, 14, 15]$, to randomly chosen values. The backward chunk covers the computation from state #9 to #5 and the forward chunk covers from state #9 to #16 and #0 to #5. Results from two chunks will match at #5.

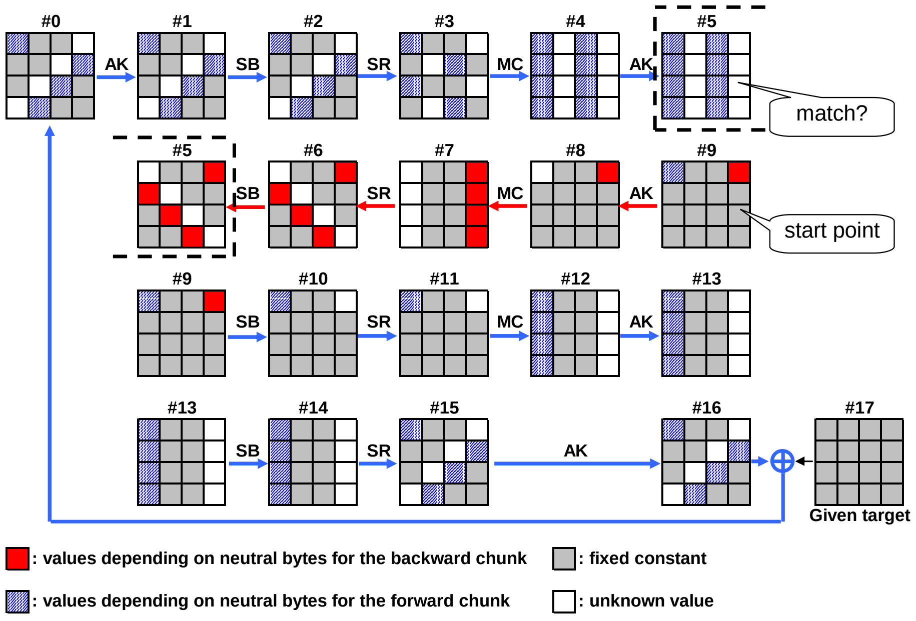

Fig. 5. Basic attack idea: chunk separation for 4-round AES

Forward computation. The forward computation starts from #9. Because #9[12] is a neutral byte for the backward chunk, we regard #9[12] to be unknown during the forward computation. Hence, one byte is unknown at #11 and the unknown byte is expanded to 4 bytes at #12. Similarly, by simply tracing the computation, we obtain 8-byte information at #5 (#5[0,1,2,3,8,9,10,11]).

An important observation is that the omission of the MixColumns operation in the last round extends the number of rounds that can be computed independently. The diffusion of AES is designed so that the full diffusion can be achieved after the 2-round operation. In fact, if MixColumns exists between #15 and #16, all bytes become unknown after this operation and it limits the attack efficiency strongly. However, the omission of MixColumns yields 12 known bytes at #16, and to make things worse, the positions of unknown bytes will overlap by the next ShiftRows operation, and thus attackers can compute MixColumns for another round. As a conclusion, we can summarize this property as follows;

AES 2-rounds achieve the full diffusion, however, if MixColumns *is omitted in the second round, 4 rounds are needed to achieve the full diffusion.*

This property is illustrated in Fig. 6. This situation does not seem to occur for the AES block cipher. However, in hash function modes, the splice-and-cut can exploit it by starting the forward chunk from the second last round.

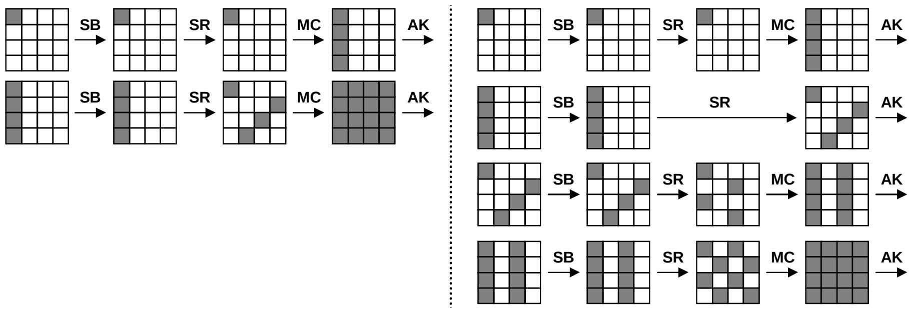

Fig. 6. Slow diffusion with the omission of MixColumns in the second round

Overall attack procedure. Because the backward computation is similar and straight-forward, we omit its explanation. Overall, if we fix 14 bytes at #9 as shown in Fig. 5, we can compute the forward chunk for 2^8 values of #9[0] and obtain 8-byte information at #5. The obtained results are stored in a table and sorted. Similarly, we can compute the backward chunk for 2^8 values of #9[12] and obtain 12-byte information at #5. Because 2 bytes (#5[1,11]) are overlapped between the results from both chunks, we can efficiently check the match of those results. If the match is not found, the attack is repeated by changing the values of fixed 14 bytes at #9 or the values for the AES key-input.

Each chunk can be built using 2^8 possible neutral values with 2^8 computations. The results are stored in a table of the size 2^8 AES state and then sorted. After that, 2^{16} pairs are tested in the 2-byte match and 1 pair will succeed. Hence, if we repeat the attack 2^{112} times, we will find a pair that also matches other 14 bytes. The final complexity for generating pseudo-preimages is $2^8 \cdot 2^{112} = 2^{120}$. This is converted to a preimage attack on the hash function with a complexity of $2^{\frac{120+128}{2}+1} = 2^{125}$ using a generic conversion in Sect. 3.1.

Note that the attack efficiency is not optimized because the purpose of this attack is to demonstrate the basic idea of our attack. Also note that, the impact of the change of #9[12] does not propagate to all bytes at the matching stage. This is because the backward chunk is too short. If the number of attacked rounds is extended as explained in Sect. 5, the impact will propagate to all bytes.

4.2 Techniques for Attacking More Rounds

We show that a technique similar to the initial-structure [32] can extend the number of rounds in each chunk by one round (in total 2 rounds), and by considering the MixColumns operation deeply, we can include one more round during the matching stage. These techniques are directly applied to our 7-round attack that will be explained in Sect. 5. Specifically, the differential path described in Fig. 7 and Fig. 8 are the copy of a part of differential path in Fig. 9.

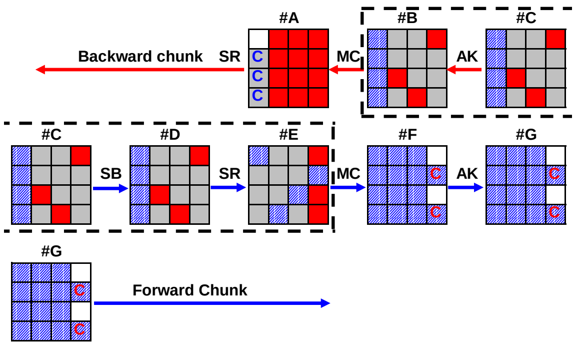

Fig. 7. Initial-structure technique for AES

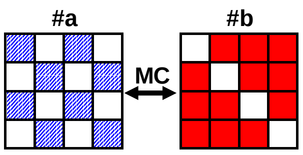

Fig. 8. Known-byte patterns for matching through MixColumns

Initial-structure. The idea is choosing several bytes as neutral bytes, and determining these values so that several output bytes of the MixColumns or InverseMixColumns operations can be constant values. This minimizes the number of unknown bytes after the first MixColumns operation in each chunk, and thus, the number of attacked rounds is extended by one round in each chunk. The construction of the initial-structure is shown in Fig. 7.

The neutral bytes for the forward chunk are 4 bytes #B[0,1,2,3]. We choose 2^8 values of these bytes and use them to compute the forward chunk independently of the backward chunk. These values are determined as follows;

1. Randomly choose constant values for 3 bytes at #A (#A[1,2,3]).
2. For all possible 2^8 values of #B[0], we calculate the values of #B[1,2,3] so that the chosen 3 bytes at #A (#A[1,2,3]) can be achieved through the InverseMixColumns operation. Note that, according to Eq. (1), #A[1,2,3] is computed by Eq.(2). Because there are 3 free variables to control 3 bytes, this is possible by solving a system of equations.

$$\begin{pmatrix} \#A[1] \\ \#A[2] \\ \#A[3] \end{pmatrix} = \begin{pmatrix} {}_x9 & {}_xe & {}_xb & {}_xd \\ {}_xd & {}_x9 & {}_xe & {}_xb \\ {}_xb & {}_xd & {}_x9 & {}_xe \end{pmatrix} \begin{pmatrix} \#B[0] \\ \#B[1] \\ \#B[2] \\ \#B[3] \end{pmatrix} \qquad (2)$$

As a result, for any 2^8 neutral values of #B[0,1,2,3], #A[1,2,3] become constant, and thus, the backward computation from #A can start with 15 known bytes and only 1 unknown byte.

The neutral bytes for the backward chunk are 3 bytes #E[12,14,15]. We choose 2^8 values of these bytes as follows;

1. Randomly choose constant values for 2 bytes, which will be the impact from #E[12,14,15] to the chosen 2 bytes at #F (#F[13,15]). In details, by consid-

ering the MixColumns operation in Eq. (1), #F[13,15] are written as follows:

$$\#F[13] = (1 \cdot \#E[12]) \oplus (2 \cdot \#E[13]) \oplus (3 \cdot \#E[14]) \oplus (1 \cdot \#E[15]), \quad (3)$$
$$\#F[15] = (3 \cdot \#E[12]) \oplus (1 \cdot \#E[13]) \oplus (1 \cdot \#E[14]) \oplus (2 \cdot \#E[15]). \quad (4)$$

The impacts on #F[13] and #F[15] from #E[12,14,15] mean the following values respectively.

$$(1 \cdot \#E[12]) \oplus (3 \cdot \#E[14]) \oplus (1 \cdot \#E[15]), \quad (5)$$
$$(3 \cdot \#E[12]) \oplus (1 \cdot \#E[14]) \oplus (2 \cdot \#E[15]). \quad (6)$$

2. For all possible 2^8 values of #E[12], we calculate the values of #E[14,15] by solving a system of equations so that the impact on the chosen 2 bytes at #F (#F[13,15]) can be achieved through the MixColumns operation. Because there are 2 free variables to control 2 bytes, this is always possible.

As a result, for any 2^8 neutral values of #E[12,14,15], the impact from these values to #F[13,15] becomes the determined constant. Note that #F[13,15] are also influenced by #E[13], and thus, final values of #F[13,15] are exclusive-or of the determined constant and values depending on #E[13]. Finally, the forward computation from #F can start with 14 known bytes and only 2 unknown bytes.

Match through MixColumns. Assume that many values of the partially known states of the form #a and #b in Fig. 8 are stored in tables. The goal of this match is efficiently finding paired values (#a, #b) that match through the MixColumns operation. Because MixColumns is applied column by column, the match is also tested column by column. We explain the match for the first column as an example. The other columns can be tested in the same procedure.

Let us consider the InverseMixColumns operation from #b to #a. From Eq. (1), #a[0] and #a[2] are expressed as follows;

$$\#a[0] = (\ _xe \cdot \#b[0]\ ) \oplus (\ _xb \cdot \#b[1]\ ) \oplus (\ _xd \cdot \#b[2]\ ) \oplus (\ _x9 \cdot \#b[3]\ ) \quad (7)$$
$$\#a[2] = (\ _xd \cdot \#b[0]\ ) \oplus (\ _x9 \cdot \#b[1]\ ) \oplus (\ _xe \cdot \#b[2]\ ) \oplus (\ _xb \cdot \#b[3]\ ) \quad (8)$$

Considering that #b[1,2,3] are known values, the equations can be transformed by using some constant numbers C_0 and C_1 as follows;

$$\#a[0] \oplus C_0 = \ _xe \cdot \#b[0], \qquad \#a[2] \oplus C_1 = \ _xd \cdot \#b[0]. \quad (9)$$

Whether or not these equations are satisfied can be checked efficiently by using the idea based on the indirect partial-matching [1]. Namely, $_xd \cdot (\#a[0] \oplus C_0) = {}_xe \cdot (\#a[2] \oplus C_1)$ is obtained from Eq. (9), and then obtain the following equation:

$$_xd \cdot \#a[0] \ \oplus \ _xe \cdot \#a[2] = \ _xd \cdot C_0 \ \oplus \ _xe \cdot C_1. \quad (10)$$

Let us denote $_xd \cdot \#a[0] \ \oplus \ _xe \cdot \#a[2]$ and $_xd \cdot C_0 \ \oplus \ _xe \cdot C_1$ by C_{for} and C_{back}, respectively. By computing C_{for} and C_{back} in the computation for each chunk, we can perform the match by just comparing these values.

Note that AES has 4 columns in a state. Because the number of candidates for the match can be reduced by a factor of 2^{-8} per column, for 4 columns, the number of candidates is reduced by a factor of 2^{-32}.

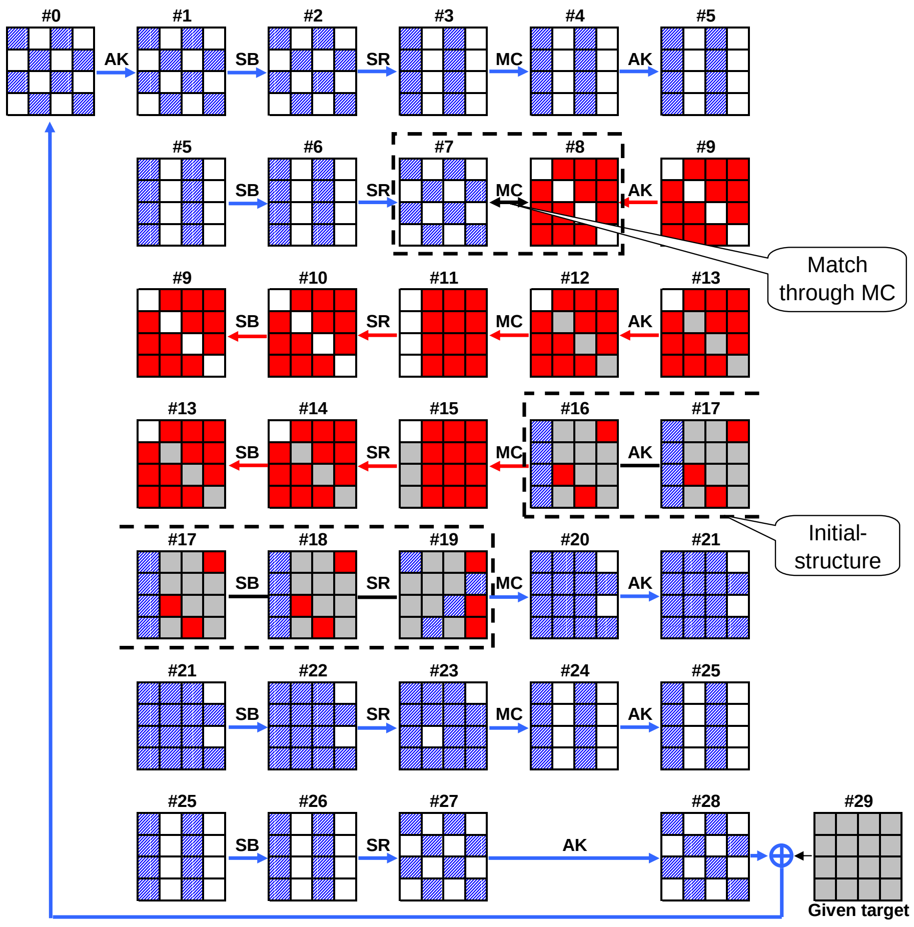

Fig. 9. Chunk separation for 7-round AES

5 Preimage Attack against 7-Round AES

By considering the techniques explained in Sect. 4.2, we can attack up to 7 rounds of AES. The chunk separation for this attack is depicted in Fig. 9.

In this attack, states #16 to #19 are chosen as the initial-structure and we apply the match between states #7 and #8. The neutral bytes for the forward computation are 4 bytes at #16, namely, #16[0,1,2,3]. The neutral bytes for the backward computation are 3 bytes at #19, namely, #19[12,14,15].

To make the initial-structure work, we choose neutral bytes for the forward chunk so that 3 bytes #15[1,2,3] can be pre-determined constant values. Similarly, neutral bytes for the backward chunk are computed so that impacts on 2 bytes #20[13,15] can be pre-determined constant values. The matching procedure is exactly the same as the one explained in Section 4.2. The detailed attack procedure is explained below. Note that the procedure below is a preimage attack on the compression function. To convert this attack to the one for a hash function, we need additional effort depending on the mode-of-operation used.

1. Choose a value for the key-input, and compute all sub-keys. How to choose the value depends on the mode-of-operation. In this procedure, we assume that the key-input can be any value. We remove this assumption later.
2. Randomly choose constant values for 9 bytes in state #16 (#16[4,5,7,8,9,10, 13,14,15]), for 3 bytes in state #15 (#15[1,2,3]), and for impacts on 2 bytes in state #20 (#20[13,15]) from the neutral bytes of the backward chunk.
3. For all 2^8 values of #16[0], do as follows.
   (a) Calculate #16[1,2,3] so that 3 bytes #15[1,2,3] can be pre-determined constant values.
   (b) Compute the forward chunk from #20 to #28, and then, from #0 to #7. Also compute C_{for} in Eq. (10) for each column.
   (c) Store the results in a table T_{for}.
4. Sort the table T_{for} after obtaining 2^8 values.
5. For all 2^8 values of #19[12], do as follows.
   (a) Calculate #19[14,15] so that the impacts from these values to 2 bytes #20[13,15] can be pre-determined constant values.
   (b) Compute the backward chunk from #15 to #8.
   (c) From 12 known bytes of #8, compute C_{back} in Eq. (10) for each column.
   (d) Check if there exists an entry in T_{for} that matches the computed C_{back}.
   (e) If exits, compute all bytes of #7 and #8 with matched values and check if all 128-bit values of #7 and #8 match.
   (f) If all 128-bits match, output the corresponding (H_{N-1}, M_{N-1}).
6. If the attack does not succeed with 2^8 values of #19[12], go back to Step 2 (or Step 1 if necessary) and repeat the attack with different constant values.

Complexity evaluation. In this attack, the sum of the complexity for computing 2^8 results of #7 and #8 is roughly 2^8 7-round AES computations. At Step 3c, we need a memory to store $2^8 \cdot 4$-byte information for C_{for}. At Step 5d we search for the table of the size 2^8. Hence, for all 2^8 values of #19[12], the cost is about $2^8 \cdot \log 2^8$ memory access, which is enough small compared to 2^8 7-round AES for computing #7 and #8. The success probability of the match is 2^{-8} for each column, and thus 2^{-32} for 4 columns. Hence, after generating 2^8 results for #7 and 2^8 results for #8, $2^8 \cdot 2^8 \cdot 2^{-32} = 2^{-16}$ candidate will remain, where a remaining candidate satisfies 4-byte linear relations in a state. Therefore, if we iterate the above procedure 2^{112} times, we obtain $2^{112} \cdot 2^{-16} = 2^{96}$ candidates satisfying the match and one of them will satisfy the other 12-byte linear relations in a state, in other words, a preimage on the compression function is found. Note that, at Step 6, the algorithm can go back to Step 2 up to 2^{112} times for a fixed key-input. To repeat the attack more, we need to change the key-input at Step 1. The final complexity of the attack is $2^8 \cdot 2^{112} = 2^{120}$ AES 7-round computations and we need a memory for storing $2^8 \cdot 4$-byte information.

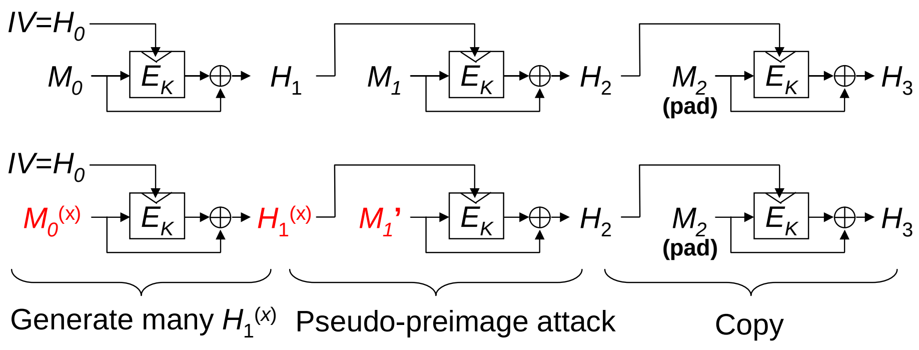

Fig. 10. (Top) Given first-preimage (Bottom) Second-preimage construction

Conversion to hash function scenario. How to convert this attack to the hash function scenario depends on the mode-of-operation used. For the DM mode, this attack finds (H_{N-1}, M_{N-1}), where the value of M_{N-1} is chosen at Step 1 and the value of H_{N-1} is determined randomly during the attack. In this scenario, we can choose the message so that the padding string can be satisfied. Instead, H_{N-1} cannot be fixed to IV. Hence, we choose M_{N-1} at Step 1 so that the padding string for 2-block messages is satisfied, and convert pseudo-preimages into preimages with the conversion in Sect. 3.1. Finally, the attack generates preimages of 2-block long with a complexity of $2^{1+(120+128)/2} = 2^{125}$.

For the MMO or MP modes, the value of H_{N-1} is chosen at Step 1 and the value of M_{N-1} is determined randomly during the attack. Therefore, we can always start from the IV, but cannot satisfy the padding string. Because of the padding problem, this attack cannot generate preimages. Hence, we aim to generate second preimages. Assume that the given first preimage is 3-block long. The attack is depicted in Fig. 10. Our attack copies the last message block in which the padding string is included. For the first block, we choose several different message values $M_0^{(x)}$ to find several different chaining variables $H_1^{(x)}$. Finally, for the second block, we choose one of generated $H_1^{(x)}$ and search for a message M_1' that satisfies $\mathrm{CF}(M_1', H_1^{(x)}) = H_2$ using the pseudo-preimage attack. Hence, second preimages are generated with a complexity of 2^{120}. Note that, a fixed H_2 cannot be mapped from a fixed $H_1^{(x)}$ for any M_1' with a probability of $1 - e^{-1}$. In such a case, we choose another $H_1^{(x)}$ and repeat the attack. After several trials of $H_1^{(x)}$, we will find a valid M_1' with a probability almost 1.

6 Discussion

Other PGV modes. In PGV, 12 schemes in Table 2 are secure. Our attacks can be applied to all 12 schemes. In our attack, the key is chosen and fixed. Therefore, as long as the key is equivalent to H_i, the same attack as the MMO-

Table 2. Twelve secure PGV constructions. X_i represents $H_i \oplus M_i$.

No.	Computation	No.	Computation	No.	Computation	No.	Computation
1	$E_{H_i}(M_i) \oplus M_i$	2	$E_{H_i}(X_i) \oplus X_i$	3	$E_{H_i}(M_i) \oplus X_i$	4	$E_{H_i}(X_i) \oplus M_i$
5	$E_{M_i}(H_i) \oplus H_i$	6	$E_{M_i}(X_i) \oplus X_i$	7	$E_{M_i}(H_i) \oplus X_i$	8	$E_{M_i}(X_i) \oplus H_i$
9	$E_{X_i}(M_i) \oplus M_i$	10	$E_{X_i}(H_i) \oplus H_i$	11	$E_{X_i}(M_i) \oplus H_i$	12	$E_{X_i}(H_i) \oplus M_i$

mode can be performed. Hence, on No.1 to No.4, a second-preimage attack with 2^{120} computations is possible. Similarly, on No.5 to No.8, a preimage attack with 2^{125} computations is possible. On No.9 to No.12, the key X_i is chosen but either H_i or M_i cannot be chosen. Hence, only a second-preimage attack with 2^{125} computations is possible. Considering the long message attack [20], our attack has an advantage only if the length of the given message is 3- to 7-blocks.

As a further generalization, applications to the generalized PGV construction proposed by [33] seems interesting. We leave this work as an open problem.

Complexity on AES 6-rounds. To demonstrate the change of the complexity with a different number of rounds, we attacked 6-rounds. The idea is omitting the match through the MC and apply the direct match instead. This enables attackers to reduce the number of known bytes in the backward chunk, and thus to keep 2^{16} neutral values for each chunk. The final results are listed in Table 1.

Known-key attack on 7-round AES. Our attack can be regarded as a new approach of the known-key attack on AES, which finds fixed points on 7-round AES. The success probability of the attack is $1 - e^{-1}$.

The attacker is given a randomly chosen key k. Then, she carries out the pseudo-preimage attack on 7-round AES in Sect. 5 with setting the given target hash value to 0. With a complexity of 2^{120}, a plaintext p s.t. $p = E_k(p)$ will be found, while finding such p will cost 2^{128} for a 128-bit random permutation.

Note that Gilbert and Peyrin proposed a known-key distinguisher on 8-round AES [17]. They find some non-ideal differential property, while our attack finds a fixed point which has been discussed for a long time. The application of our known-key distinguisher to the hash function scenario is meaningful, which finds a preimage of 0 in several PGV constructions. However, [17] attacks 8-round with a feasible complexity, while ours attacks 7-round with an infeasible complexity.

Difficulties in chosen-key setting. In our attack, the key-input is fixed to a constant. It might be possible to extend the attack by actively choosing the key-input. However, this is not trivial. Firstly, the splice-and-cut technique cannot be used for the key-schedule function, namely, most part of the first round key cannot be obtained from the last round key without computing the inversion. Hence, the MitM attack would be difficult. Secondly, the previous related-key attacks on AES focused their attention on differential properties. It is unclear how to use the weak property of the key-schedule function to build preimages.

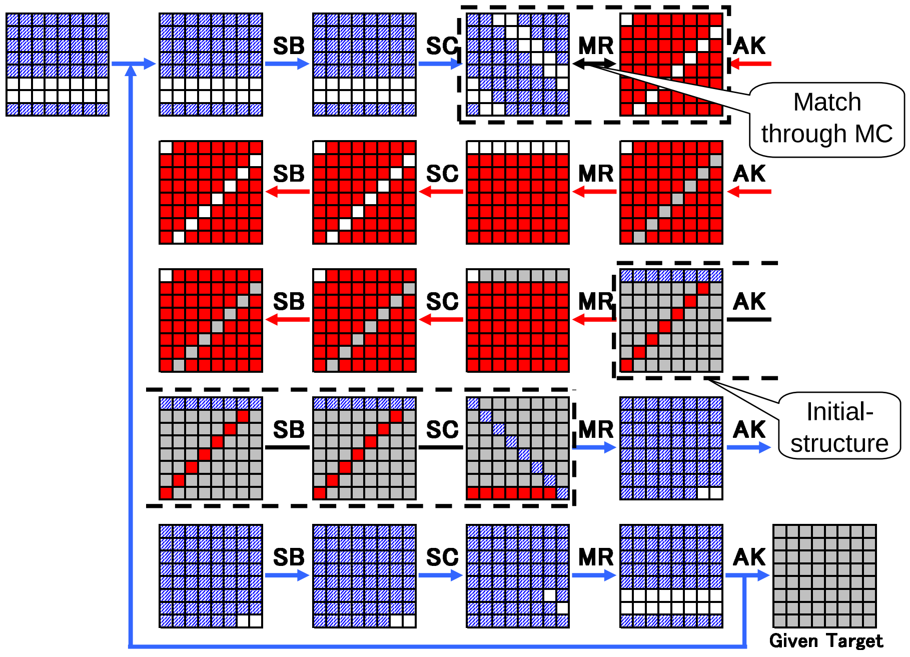

Fig. 11. Chunk separation for second preimage attack on 5-round Whirlpool. Whirlpool computes ShiftColumns(SC) and MixRows(MR) instead of SR and MC.

Application for Whirlpool. Our attack exploits the omission of MixColumns in the last round. Here, we discuss a variant of AES, which computes MixColumns in the last round[2]. To check the number of attacked rounds, we again use Fig. 9. Due to MixColumns in the last round, we need to remove the 7th round from our target. Note that known byte positions at states #25 and #5 are identical. This indicates that we also need to remove the first round, and state #25 should be connected to state #6. As a result, we can attack only 5 rounds.

Whirlpool [30] is a hash function which is deeply based on AES with a larger state size. Because Whirlpool computes MixColumns during the last round, the above analysis can be directly applied as a preimage attack on reduced Whirlpool. Whirlpool uses $8 * 8$-byte (512-bit) state and consists of 10 naturally-expanded AES rounds. It produces the compression function output with the MP mode. In Fig. 11, we show the chunk separation for attacking 5-round Whirlpool.

The attack strategy is the same as 7-round attack on AES. Hence we omit the details. Pseudo-preimages can be generated faster than the brute force attack by a factor of 2^8, which is 2^{504}. Because Whirlpool uses the MP mode, this can be a second-preimage attack with a complexity of 2^{504} and a memory of 2^8.

[2] One may note another study on the effects of the omission of MixColumns [14].

7 Concluding Remarks

In this paper, we studied the security of AES hashing modes in terms of the classical and important security notions. We proposed a preimage attack on the PGV modes instantiating AES by applying the meet-in-the-middle approach. As a result, we obtained a preimage attack on 7 rounds of DM-AES and second-preimage attack on 7 rounds of MMO-AES and MP-AES. This attack can also generate second preimages of Whirlpool reduced to 5 rounds.

Note that our results do not give impact on other AES-based hash functions e.g. several SHA-3 candidates, in particular, those with the wide-pipe structure.

Acknowledgements

I would like to thank the anonymous reviewers of FSE 2011 for many fruitful comments, especially for the research motivation and applications of the attack.

References

1. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 578–597. Springer-Verlag, 2009.
2. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 70–89. Springer-Verlag, 2009.
3. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *SAC 2008*, volume 5381 of *LNCS*, pages 103–119. Springer-Verlag, 2009.
4. Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage attacks on 3-pass HAVAL and step-reduced MD5. In *SAC 2008*, volume 5381 of *LNCS*, pages 120–135. Springer-Verlag, 2009.
5. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 299–319. Springer-Verlag, 2010.
6. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer-Verlag, 2009.
7. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer-Verlag, 2009.
8. Alex Biryukov and Ivica Nikolić. A new security analysis of AES-128. Rump session of CRYPTO 2009, 2009. `http://rump2009.cr.yp.to/`.
9. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 322–344. Springer-Verlag, 2010.

10. Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, and Yannick Seurin. Hash functions and RFID tags: Mind the gap. In *CHES 2008*, volume 5154 of *LNCS*, pages 283–299. Springer-Verlag, 2008.
11. Christophe De Cannière and Christian Rechberger. Preimages for reduced SHA-0 and SHA-1. In *CRYPTO 2008*, volume 5157 of *LNCS*, pages 179–202. Springer-Verlag, 2008.
12. Joan Daemen and Vincent Rijmen. *The design of Rijndeal: AES – the Advanced Encryption Standard (AES).* Springer-Verlag, 2002.
13. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In *FSE 2008*, volume 5086 of *LNCS*, pages 116–126. Springer-Verlag, 2008.
14. Orr Dunkelman and Nathan Keller. The effects of the omission of last round's MixColumns on AES. Cryptology ePrint Archive, Report 2010/041, 2010. `http://eprint.iacr.org/2010/041`.
15. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 158–176. Springer-Verlag, 2010.
16. Henri Gilbert and Marine Minier. A collision attack on 7 rounds Rijndael. In *Third AES Candidate Conference (AES3)*, pages 230–241. Springer-Verlag, 2000.
17. Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE 2010*, volume 6147 of *LNCS*, pages 365–383. Springer-Verlag, 2010.
18. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 56–75. Springer-Verlag, 2010.
19. Sebastiaan Indesteege and Bart Preneel. Preimages for reduced-round Tiger. In *WEWoRC 2007*, volume 4945 of *LNCS*, pages 90–99. Springer-Verlag, 2007.
20. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 474–490. Springer-Verlag, 2005.
21. Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 315–324. Springer-Verlag, 2007.
22. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 126–143. Springer-Verlag, 2009.
23. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. The rebound attack and subspace distinguishers: Application to Whirlpool. Cryptology ePrint Archive, Report 2010/198, 2010. `http://eprint.iacr.org/2010/198`.
24. Gaëtan Leurent. MD4 is not one-way. In *FSE 2008*, volume 5086 of *LNCS*, pages 412–428. Springer-Verlag, 2008.
25. Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New impossible differential attacks on AES. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 279–293. Springer-Verlag, 2008.
26. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In *SAC 2009*, volume 5867 of *LNCS*, pages 16–35. Springer-Verlag, 2009.

27. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009*, volume 5665 of *LNCS*, pages 260–276. Springer-Verlag, 2009.
28. Alfred John Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
29. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO 1993*, volume 773 of *LNCS*, pages 363–378. Springer-Verlag, 1994.
30. Vincent Rijmen and Paulo S. L. M. Barreto. The WHIRLPOOL hashing function. Submitted to NISSIE, September 2000.
31. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 253–271. Springer-Verlag, 2008.
32. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 134–152. Springer-Verlag, 2009.
33. Martijn Stam. Blockcipher-based hashing revisited. In *FSE 2009*, volume 5665 of *LNCS*, pages 67–83. Springer-Verlag, 2009.
34. U.S. Department of Commerce, National Institute of Standards and Technology. *Specification for the ADVANCED ENCRYPTION STANDARD (AES) (Federal Information Processing Standards Publication 197)*, 2001. `http://csrc.nist.gov/encryption/aes/index.html\#fips`.
35. U.S. Department of Commerce, National Institute of Standards and Technology. *Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices*, 2007. `http://csrc.nist.gov/groups/ST/hash/documents/FR\_Notice\_Nov07.pdf`.
36. Yongzhuang Wei, Jiqiang Lu, and Yupu Hu. Meet-in-the-middle attack on 8 rounds of AES block cipher under 192 key bits. Cryptology ePrint Archive, Report 2010/537, 2010. `http://eprint.iacr.org/2010/537`, appeared in the accepted papers list of ISPEC 2011.