Improved Meet-in-the-Middle Preimage Attacks against AES Hashing Modes

Zhenzhen Bao[1,2], Lin Ding[3], Jian Guo[1], Haoyang Wang[1], and Wenying Zhang[1,4]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
{zzbao,guojian}@ntu.edu.sg, wang1153@e.ntu.edu.sg
[2] Strategic Centre for Research in Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore
[3] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
dinglin@sjtu.edu.cn
[4] School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
zhangwenying@sdnu.edu.cn

Abstract. Hashing modes are ways to convert a block cipher into a hash function, and those with AES as the underlying block cipher are referred to as AES hashing modes. Sasaki in 2011 introduced the first preimage attack against AES hashing modes with the AES block cipher reduced to 7 rounds, by the method of meet-in-the-middle. In his attack, the key schedules are not taken into account, hence the same attack applies to all three versions of AES. In this paper, by introducing neutral bits from key, extra degrees of freedom are gained, which are utilized in two ways, *i.e.*, to reduce the time complexity and to extend the attack to more rounds. As an immediate result, the complexities of 7-round pseudo-preimage attacks are reduced from 2^{120} to 2^{112}, 2^{96}, and 2^{96} for AES-128, AES-192, and AES-256, respectively. By carefully choosing the neutral bits from key to cancel those from state, the attack is extended to 8 rounds for AES-192 and AES-256 with complexities 2^{120} and 2^{96}. Similar results are obtained for Kiasu-BC, a tweakable block cipher based on AES-128, and interestingly the additional input tweak helps reduce the attack complexities further. To the best of our knowledge, these are the first preimage attacks against 8-round AES hashing modes.

Keywords: AES, MITM, preimage, hashing mode, key schedule

1 Introduction

The Advanced Encryption Standard (AES). Designed by Daemen and Rijmen in 1998 [1], and later formally standardized by the U.S. National Institute of Standards and Technology (NIST) in 2001, AES becomes the most widely deployed block cipher nowadays in the world among both industry and government agencies, for its long-standing security against massive cryptanalysis and efficiency in both software and hardwares. There are three variants, and according to the key sizes and hence security level in bits from the set $\{128, 192, 256\}$, they are named as AES-128, AES-192, and AES-256, respectively.

The PGV Hashing Modes. Hashing modes are ways to convert block ciphers to compression functions, and then to hash functions under some domain extensions. Compared with designs from scratch, hashing modes enjoy both inherited security and performance efficiencies from the underlying block ciphers. Security proofs of hashing modes, which deduce the security such as collision resistance of a hash function to the security of the underlying block cipher, ensure that no attack against the hash function could be possible before an attack against the underlying block cipher is found. This removes the hassle of intensive cryptanalysis required by hash functions designed from scratch, when the hashing mode is instantiated by a secure block cipher such as AES. For environments like resource constrained hardware where a block cipher is already implemented, a hashing mode could be the most economic way to achieve a hash function for purposes like digest and signature, since in most of the cases implementing a hashing mode on top of an existing block cipher costs much lesser than that of a standalone hash function.

In [23], Preneel, Govaerts, and Vandewalle summarized that there are 12 secure ways to convert a block cipher to a compression function, and these constructions are referred as PGV modes nowadays after the name of the authors. Out of them, there are three modes named Davies-Meyer (DM), Matyas-Meyer-Oseas (MMO), Miyaguchi-Preneel (MP) commonly used in practice. For instance, all the hash functions in the MD-SHA hash family including MD4, MD5, SHA-1, and SHA-2, can be viewed as DM modes.

The MITM Preimage Attack. The preimage attack against a hash function is to find a message whose digest equals to the given value. The most naive way is to randomly select message, evaluate its digest and check against the given value. This bruteforce method costs 2^n hash evaluations for an n-bit hash function. A method running faster than 2^n is considered as an attack. Sasaki and Aoki introduced the Meet-in-the-Middle (MITM) preimage attack in 2008 [25], and the technique was extended and used to break the theoretical preimage security claims of MD4 [10], MD5 [26], Tiger [10, 27], HAVAL [12, 25] and round-reduced variants of many other hash functions such as SHA-0 and SHA-1 [5, 9, 18], SHA-2 [4], BLAKE [9], HAS-160 [13], RIPEMD and RIPEMD-160 [28], Stribog [2], and Whilwind [3]. It is interesting to see that the idea of MITM preimage attack also leads to progress of collision attack against reduced SHA-2 [21].

MITM Preimage Attack against AES Hashing Modes. AES hashing modes refer to the hashing modes instantiated by AES block cipher. In 2011, Sasaki [24] introduced the first attack against AES hashing modes with the underlying AES reduced to 7 rounds and the last round without the MixColumn operation. The complexity of the attack was sightly improved by Wu *et al.* [29]

Target	# Rounds	Time-1	Time-2	Memory	(d_1, d_2, m)	Source
AES-128	7	2^{120}	2^{125}	2^8	$(8, 8, 32)$	[24]
	7	$2^{120-\min(t,24)}$	2^{123}	$2^{8+\min(t,24)}$	$(8, 32, 32)$	[29]
	7	$2^{112-\min(t,8)}$	2^{117}	$2^{16+\min(t,8)}$	$(16, 32, 24)$	Section 4.3
AES-192	7	2^{120}	2^{125}	2^8	$(8, 8, 32)$	[24]
	7	2^{96}	2^{113}	2^{32}	$(32, 32, 32)$	Section 4.4
	8	$2^{120-\min(t,24)}$	2^{123}	$2^{8+\min(t,8)}$	$(8, 32, 32)$	Section 5.3
AES-256	7	2^{120}	2^{125}	2^8	$(8, 8, 32)$	[24]
	8	2^{96}	2^{113}	2^{32}	$(32, 32, 32)$	Section 5.2
Kiasu-BC	7	2^{104}	2^{117}	2^{24}	$(24, 32, 24)$	Section 4.5
	8	$2^{120-\min(t,24)}$	2^{123}	$2^{8+\min(t,8)}$	$(8, 32, 32)$	Section 5.4

Table 1: Summary of our improved pseudo-preimage attacks against the round-reduced compression function of AES hashing modes, compared with previous works. Here 2^t is the number of available targets for preimage attacks or the number of blocks of given message for second-preimage attacks, and Time-1 is the complexity for pseudo-preimage, second-preimages, or preimages which require no additional conversion from pseudo-preimage such as MMO and MP modes; Time-2 is the complexity of preimages which require a conversion and here a single target is given $(t = 0)$.

in 2012. To the best of our knowledge, there is no more public progress on this topic since then.

The general idea of MITM preimage attack is to split the cipher (or compression function) into two independent chunks, which can be computed independently from each other with respective to some *neutral bits*. Technically, the source of neutral bits of most previous works is the key bits of block cipher or the message bits of compression function. However, in [24], the neutral bits are chosen from the state while the key bits are not used and fixed to some random constants. The key bits are not used because finding neural bits in key is difficult due to the key schedule which diffuses all key bits quickly. It is then natural to ask whether it is possible to eventually find neutral bits from key to either improve the attack complexity or to extend the number of attacked rounds. In this paper, we achieve both.

Our Contributions. On one hand, additional neutral bits from key improves the attack in two direction, *i.e.*, improving the time complexity directly due to more neutral bits and extending the attack to more rounds since local collisions of neutral bits from the state and key are possible. On the other hand, to avoid dealing with the quick diffusion of the key schedule of AES, we choose neutral bits from key for one chunk v.s. for both chunks. This is possible thanks to the improvement of the attack in [29], where the unbalanced 8 and 32 bits neutral bits are found for the two chunks. The additional neutral bits from key makes it closer to the balanced 32 and 32 bits, which improves the time complexity of the final attack by a factor of at most $2^{32-8} = 2^{24}$.

Larger key sizes allow more degrees of freedom for the choices of neutral bits, and also AES with larger key size comes with a slower key diffusion. These factors lead us to a higher attacked rounds and lower time complexities for AES-192 and AES-256, compared with the previous attacks against AES-128 in [24, 29]. The details of attacks, including the number of attacked rounds and time/memory complexities, compared with the previous works, are summarized in Table 1.

Organization. The rest of the paper is organized as follows. Section 2 gives the preliminaries of AES and the PGV hashing modes, followed by a general description of the MITM preimage attack in Section 3. Results of 7 and 8 rounds are given in Section 4 and 5, respectively. Section 6 concludes the paper. Some details of the attacks are postponed to Appendix.

2 Preliminaries

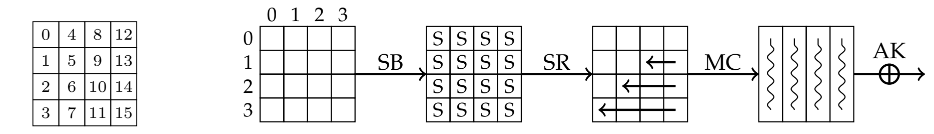
2.1 Description of AES

The Advanced Encryption Standard (AES) is an iterated block cipher which encrypts 128-bit plaintext with secret key of sizes 128, 192, and 256 bits. AES with 128-bit (192, 256) master keys is denoted by AES-128 (192, 256). AES-128, AES-192, and AES-256 share the same round function with different number of rounds: 10, 12, and 14, respectively. The rounds are numbered $0, \cdots, N_r - 1$, where $N_r \in \{10, 12, 14\}$ is the number of rounds. Its internal state can be represented as a 4×4 matrix whose elements are byte value (8 bits) in a finite field of $GF(2^8)$. The round function consists of four basic transformations in the following order:

- `SubBytes` (SB) is a nonlinear substitution that applies the same S-box to each byte of the internal state.
- `ShiftRows` (SR) is a cyclic rotation of i-th row by i bytes to the left, for $i = 0, 1, 2, 3$.
- `MixColumns` (MC) is a multiplication of each column with a Maximum Distance Separable (MDS) matrix over $GF(2^8)$.
- `AddRoundKey` (AK) is an exclusive-or with the round-dependent key.

MDS guarantees that the sum of active bytes (a.k.a. non-zero bytes) in the input and output of the MixColumns operation is at least 5 unless all bytes are non-active (a.k.a. zeros). 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 in typewritter font, `e`, `b`, `d`, and `9`, are in hexadecimal.

$$\begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix}, \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{pmatrix} = \begin{pmatrix} \mathtt{e} & \mathtt{b} & \mathtt{d} & \mathtt{9} \\ \mathtt{9} & \mathtt{e} & \mathtt{b} & \mathtt{d} \\ \mathtt{d} & \mathtt{9} & \mathtt{e} & \mathtt{b} \\ \mathtt{b} & \mathtt{d} & \mathtt{9} & \mathtt{e} \end{pmatrix} \begin{pmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{pmatrix}$$



(a) Byte order **(b)** AES round function

Figure 1: AES byte order and AES round function

One round of AES is depicted in Figure 1 as follows:

At the very beginning of the encryption, an additional whitening key addition is performed, and the last round does not contain MixColumns.

The key schedule of AES transforms the master key into N_r+1 subkeys of 128 bits each. The master key is divided into N_k 32-bit words $(W[0], W[1], ..., W[N_k-1])$, then $W[i]$ for $i = N_k, \cdots, 4 \cdot N_r + 3$ is computed as

$$W[i] = \begin{cases} W[i-N_k] \oplus \texttt{SB}(\texttt{RotByte}(W[i-1])) \oplus Rcon[i/N_k], & i \equiv 0 \bmod N_k; \\ W[i-8] \oplus \texttt{SB}(W[i-1]), & N_k = 8 \text{ and } i \equiv 4 \bmod 8; \\ W[i-N_k] \oplus W[i-1], & \text{otherwise.} \end{cases}$$

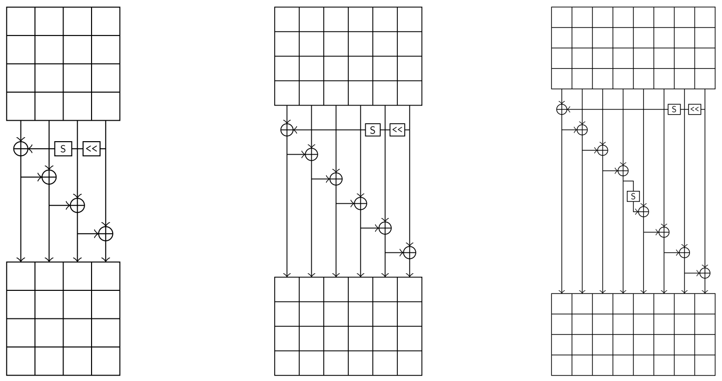


Figure 2: Round functions of key schedule of AES-128, AES-192, and AES-256 [15]

The i-th *round key* is the concatenation of 4 words $W[4i] \parallel W[4i+1] \parallel W[4i+2] \parallel W[4i+3]$. RotByte is a cyclic shift by one byte to the left, and *Rcon* is the round constant, for which we refer to [1] for details. The graphic representation of the key schedules is depicted in Figure 2.

2.2 Description of Kiasu-BC

Kiasu-BC is the underlying tweakable block cipher (TBC) used in the authenticated encryption scheme Kiasu proposed by Jean *et al.* [16] alongside their

TWEAKEY framework [17] at ASIACRYPT 2014. The TBC is almost identical to the AES-128 except for the additional input tweak, which renders it an attractive primitive for various modes of operation and applications requiring tweakable block ciphers. Therefore, studying how the additional tweak input affects the security strength compared to that of AES is highly valuable to gain trust for more adoptions.

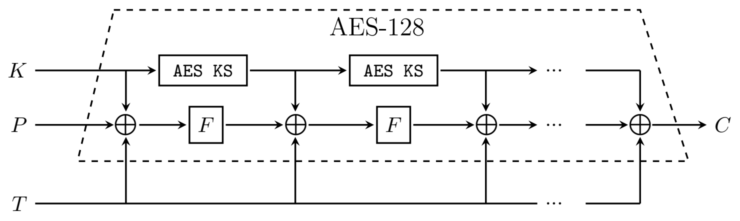


Figure 3: The Kiasu-BC tweakable block cipher based on AES-128

The TBC Kiasu-BC takes three inputs: a 64-bit tweak T, a 128-bit key K and a 128-bit plaintext P. It outputs a 128-bit ciphertext $C = E_K(T, P)$ as the encryption of P under the key K for the tweak value T. As depicted in Figure 3, Kiasu-BC is exactly the AES-128 cipher, but with a 64-bit tweak value XOR-ed to the first two rows of the internal state after each round key addition in the round function of the encryption, including after the pre-whitening key addition. There is no tweak schedule, *i.e.*, the same T is used every time in its original form. Kiasu-BC can actually be viewed as one of the simplest instances of the TWEAKEY framework based on AES.

3 The MITM Preimage Attack

The MITM Attack. In its early stages of development, the meet-in-the-middle approach proposed by Diffie and Hellman in [8] is mainly used as a generic time-memory trade-off technique to attack against encryption schemes with clear separations, *e.g.*, Double DES. That is because, it is straightforward to divided the whole computation into two or multiple independent computational chunks. Thus, the whole 'for' loop in a brute force attack can be separated into two or multiple independent 'for' loops, which have quite smaller and mutually balanced amount of computations, and which independently generate lists of candidates (of partial solutions). The independence between the smaller 'for' loops makes each element in one list be able to make a pair with any element in other lists to form a candidate solution. Such effect of taking cartesian product between two sets, enlarges the number of candidates of the correct computation dramatically. Then, according to the birthday paradox, say for two lists of 2^ℓ entries of n-bit values, to find a match with high probability, it is required $2^{(\ell+\ell)} \geq 2^n$, *i.e.*, $\ell \geq n/2$. Thus, the minimum time complexity of a simple MITM attack is $2^{n/2}$, together with $2^{n/2}$ memory.

3.1 Application to Pseudo-Preimage Attacks

Using the MITM approach in preimage finding on hash function can be seen in [7, 14, 20]. Aoki and Sasaki in [25] for the first time combined the MITM and local-collision approaches to devise preimage attacks on hash function HAVAL. Whereas, before that local-collision approach is mainly used in collision attacks on hash functions. Based on these primary works, the MITM-based preimage attack on hash functions developed in a series of papers and advanced further.

Techniques Developed for MITM Preimage Attacks. Several important techniques are invented along the development and the application of the attacks are as follows.

Splice-and-cut and neutral words. Aoki and Sasaki in [6] invented the splice-and-cut MITM attack and proposed the concept of "neutral word". In the splice-and-cut MITM attack as depicted in Figure 4, the first and last steps of the attack target can be spliced to be consecutive steps by feed-forward mechanisms in the compression function of hashing modes (*e.g.*, DM-mode) or by querying the decryption in encryption schemes. The chain of computational steps of the attack target is cut starting from an internal step (named starting point), such that the chain is divided into two chunks of steps. Conventionally, the chunk of steps, which need to be computed forward (resp. backward) to reach the matching point, is named forward chunk (resp. backward chunk). The starting point is chosen so that each chunk includes at least one message (or key) word that is independent from the other chunk, where such message (or key) words are called "neutral words".

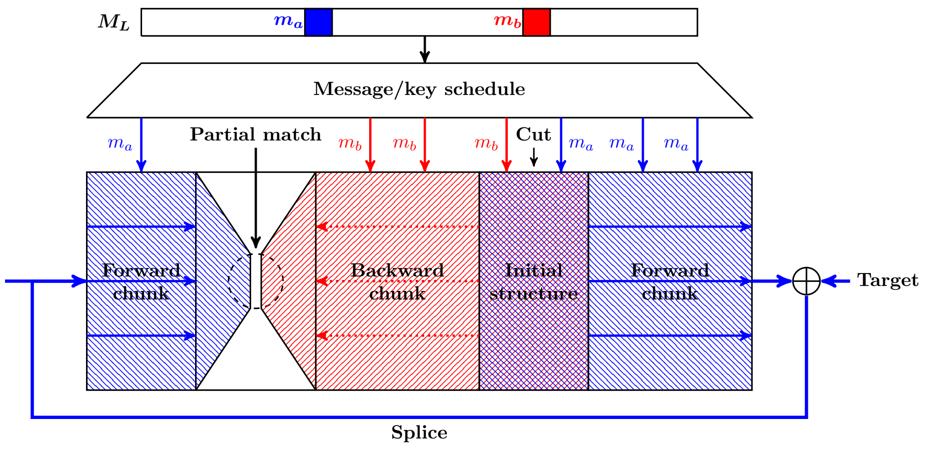
Initial-structure technique [26]: Initial-structure is a generalization of the local-collision technique that enables to skip several steps at the beginning of chunks.

> "Initial structure is a few consecutive steps including at least two neutral words named m^{2nd} and m^{1st}, where steps after the initial structure (2nd chunk) can be computed independently of m^{1st} and steps before the initial structure (1st chunk) can be computed independently of m^{2nd} [26]."

Partial matching [6, 26]: In primary MITM approach, the final phase of the attack involves matching between all-word in two states computed independently. Whereas, by executing only one-word (or several-words) matching instead of all-word matching, the required independent computations can be expanded by more steps, thus enables to attack more steps of the target.

Partial-fixing [6, 26]: fixing a part of the neutral words enables to partially compute more steps within a chunk even if in company with a neutral word for the other chunk.

MULTI-TARGETS [10]: when incorporating multi-target scenarios into the MITM framework, the available multiple targets can directly provide additional freedoms to one computation chunk without influencing the other.



Let the space for both neutral words m_a and m_b be 2^ℓ, the time complexity is $2^{n-\ell}$, and memory complexity is 2^ℓ.

Figure 4: The advanced MITM pseudo-preimage attack on DM-mode [24]

The attack framework. As depicted in Figure 4, the attack framework of the splice-and-cut MITM attack using initial structure and partial match is as follows. Before the execution of the attack procedure, the configuration of the attack should be set up, which involves:

1. Chunk separation: by splicing and cutting, decide where the computation be the starting point of the forward/backward computation, and at which state, be the matching point. The principle of the chunk separation is to find the best balance between freedom degrees and the size of the matching point that the freedom degrees are fully used. That requires one to decide:
2. The neutral bytes for each chunk – the selection on the neutral bytes will determine the freedom degrees.
3. The bytes for match – the derivation on the bytes for match also depends on the selection of neutral bytes and the computation rule of the attack target.

Having decided the above configurations, the attack procedure goes as follows (Figure 4 illustrates the MITM pseudo-preimage attack integrating with these advanced techniques on Davies-Meyer mode): Denote the neutral words for the forward chunk and backward chunk by $N^{\mathtt{f}}$ and $N^{\mathtt{b}}$, respectively:

1. Fix all other words except for the neutral words $N^{\mathtt{f}}$ and $N^{\mathtt{b}}$ in the initial structure to arbitrary values.

2. For all possible values of $N^{\mathtt{f}}$, forward compute from the starting point to the matching point at the terminal state of forward chunk to get a list $Lfor$ of candidate values indexed by the value of $N^{\mathtt{f}}$.
3. For all possible values of $N^{\mathtt{b}}$, backward compute from the starting point to the matching point at the terminal state of backward chunk to get a list *Lback* of candidate values indexed by the value of $N^{\mathtt{b}}$.
4. Sorting the two lists $Lfor$ and *Lback* using hash tables, check whether there is a match/partial-match between them.
5. In case of partial-matching used in the above step, for the surviving pairs, check for full match.
6. Repeat the whole procedure to find full state matches by changing values of fixed words.

The Complexity Analysis. Denote the size of the internal state by n, the freedom degrees in the forward and backward directions by d_1 and d_2 respectively, and the number of bits for match by m.

1. forward computing to get a list $Lfor$ of size 2^{d_1} requires 2^{d_1} computations of the forward chunk.
2. backward computing to get a list *Lback* of size 2^{d_2} requires 2^{d_2} computations of the backward chunk.
3. matching between $Lfor$ and *Lback* requires $2^{\max(d_1,d_2)}$ memory access which is usually ignored, compared with the $2^{\max(d_1,d_2)}$ computations of the compression function of the target in above steps.
4. $2^{d_1+d_2-m}$ pairs are left after partial matching, hence the same complexity is required for full-match checking.
5. finding a full match requires $2^{m-(d_1+d_2)} \times 2^{n-m} = 2^{n-(d_1+d_2)}$ repetitions.

When d_1, d_2 are different, *i.e.*, unbalanced, we use $\max(2^{d_1}, 2^{d_2})$ to denote the sum of complexities for computing the two chunks. Note, when $d_1 = d_2$, the computation complexity is 2^{d_1} full target (= forward chunk + backward chunk). Hence $\max(2^{d_1}, 2^{d_2})$ is used for all cases. The above complexity analysis gives

$$2^{n-(d_1+d_2)} \cdot \left(2^{\max(d_1,d_2)} + 2^{d_1+d_2-m}\right) = 2^{n-\min(d_1,d_2)} + 2^{n-m} \simeq 2^{n-\min(d_1,d_2,m)}. \quad (1)$$

From this formula, it can be seen that, the critical point for the attack being optimized is to reach a balance between the freedom degrees for forward chunk and backward chunk. Because at last, it will only depend on the minimum between the two freedom degrees. Besides, the memory complexity can be $2^{\min(d_1,d_2)}$ by only storing the list computed by the direction with less freedom and make a match at once a candidate for the other direction being available. Accordingly, the larger the $\min(d_1, d_2)$ the lesser the time complexity and at the same time the larger the memory complexity. However, generally, in such MITM preimage attack on hash function, the $\min(d_1, d_2)$ is less than $n/2$, thus, compared with memory complexity, time complexity is the bottleneck. Therefore, generally, the larger the $\min(d_1, d_2)$ the better the attack. The second term 2^{n-m} is usually minor when $m > \min(d_1, d_2)$, *i.e.*, the number of matching bits is more than that of neutral bits.

3.2 Conversion from Pseudo-Preimages to Preimages

Note that, the above MITM attack only gives pseudo-preimage. Below we consider the methods to convert pseudo-preimages to preimages.

Converting pseudo-preimages to a preimage [22, Fact9.99]: for n-bit narrow-pipe iterated hash function, using the unbalanced meet-in-the-middle approach, a pseudo-preimage attack with computational complexity of 2^{ℓ} where $\ell < n-2$ can be converted into a preimage attack with computational complexity of $2^{\frac{n+\ell}{2}+1}$, by finding $2^{\frac{n-l}{2}}$ pseudo-preimages and $2^{\frac{n+\ell}{2}+1}$ links, among which one will lead to one of the psuedo-preimages and form a preimage. Figure 5 depicts how to convert pseudo-preimage attacks to preimage attacks using a higher layer of MITM procedure, and how to use an unbalanced-tree and an expandable message to convert a multi-target pseudo-preimage to preimage attack [19]. The time complexity for the latter is

$$((n-\ell)\cdot\ln(2)+1)\cdot 2^{\ell} = (\min(d_1,d_2)\cdot\ln(2)+1)\cdot 2^{n-\min(d_1,d_2)}, \qquad (2)$$

where a single pseudo-preimage attack costs $2^{\ell-t}$ ($\ell = n-\min(d_1,d_2)$ in our notation) computations when 2^t targets are given. This is possible when target can be used as the additional source of neutral words. Without loss of generality, let us assume targets can be used as part of forward chunk, hence the size of $Lfor$ list is increased from 2^{d_1} to 2^{d_1+t}, then Eq. (1) gives complexities

$$\text{Time: } 2^{n-\min(d_1+t,d_2,m)};\ \text{Memory: } 2^{\min(d_1+t,d_2,m)} \qquad (3)$$

In other words, the complexity of psuedo-preimage attack reduces linearly with respective to the amount of targets available, as long as $d_1+t<d_2$ and $m > \min(d_1+t,d_2)$. When these conditions could not be met, the complexity of pseudo-preimage attack remains as

$$2^{n-\max(d_1,d_2)} + 2^{n-m} \simeq 2^{n-\min(\max(d_1,d_2),m)} \qquad (4)$$

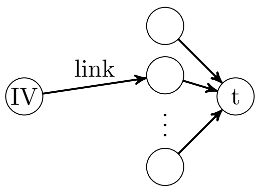
for $d_1+t \geq d_2$ (*i.e.*, $t \geq d_2-d_1$), for which the time complexity of preimage conversion will follow the original unbalanced MITM

$$2^{n+1-\min(\max(d_1,d_2),m)/2}. \qquad (5)$$

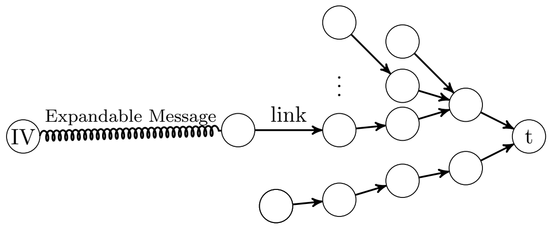
3.3 Converting Block cipher to Compression Function

In [23], Preneel *et al.* summarized 17 mode-of-operations to build a compression function for hash from a block cipher. Twelve of them are shown to be secure.

Denote the compression function composed of a hash function by CF and denote the block cipher E with a key K by E_K. Then, the twelve secure PGV constructions of CF can be expressed by formulas in Table 2, where H_i denote the state value (chaining value), M_i denote the message block, and X_i denote the XOR of the state H_{i-1} and the message block M_i.



(a) Traditional Conversion



(b) Multi-Target Pseudo-Preimage

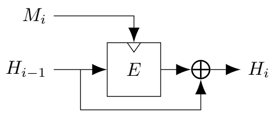
Figure 5: Converting Pseudo-Preimages to Preimages: circle denotes state, arrow denotes message block [11]

Table 2: Twelve secure PGV constructions [23, 24].

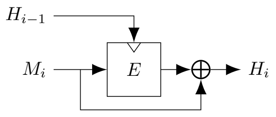
	No.	Computation	No.	Computation	No.	Computation	No.	Computation
Class 1	1	$E_{H_{i-1}}(M_i) \oplus M_i$	2	$E_{H_{i-1}}(X_i) \oplus X_i$	3	$E_{H_{i-1}}(M_i) \oplus X_i$	4	$E_{H_{i-1}}(X_i) \oplus M_i$
Class 2	5	$E_{M_i}(H_{i-1}) \oplus H_{i-1}$	6	$E_{M_i}(X_i) \oplus X_i$	7	$E_{M_i}(H_{i-1}) \oplus X_i$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

X_i represents $H_{i-1} \oplus M_i$.

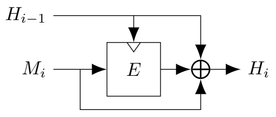
These 12 PGV constructions of CF can be classified according to the material fed in through the key into three classes: Class 1 – chaining values are fed in through the key (row 1 in Table 2); Class 2 – messages are fed in through the key (row 2 in Table 2); Class 3 – XOR sum of message and the chaining values are fed in through the key (row 3 in Table 2). Among those PGV schemes, three of them are used in practice, which are named Davies-Meyer (DM)-mode, Matyas-Meyer-Oseas (MMO)-mode, and Miyaguchi-Preneel (MP)-mode (see Figure 6)



(a) DM-mode
$H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1}$



(b) MMO-mode
$H_i = E_{H_{i-1}}(M_i) \oplus M_i$



(c) MP-mode
$H_i = E_{H_{i-1}}(M_i) \oplus M_i \oplus H_{i-1}$

Figure 6: Illustrations for DM, MMO, and MP modes [15, 24]

In the original attack by Sasaki [24], the key is preset to random constants, which can be used by attacker, and the input state is determined by the attack, hence such psuedo-preimage can be converted into preimages for modes in Class 1 such as MMO and MP with the same complexity as pseudo-preimage. Other modes in Class 2 and 3 requires conversion, and hence result in higher complexities for preimages as discussed in the subsection above.

Converting TBC to Compression Function. The tweakable block cipher, denoted here as $\widetilde{\mathsf{E}}(K_t, T_t, P)$, has an additional input tweak T, compared with block ciphers $\mathsf{E}(K, P)$. We consider here converting a TBC to a block cipher, then to a compression function through the modes above. The TBC-to-BC conversion can be divided into 3 types:

- Type-I: $\mathsf{E}(K, P) = \widetilde{\mathsf{E}}(K_t = K, T_t = C, P)$, where C is a constant.
- Type-II: $\mathsf{E}(K, P) = \widetilde{\mathsf{E}}(K_t = C, T_t = K, P)$, where C is a constant.
- Type-III: $\mathsf{E}(K, P) = \widetilde{\mathsf{E}}(K_t = K_1, T_t = K_2, P)$, where $K = K_1||K_2$, *i.e.*, both key and tweak of $\widetilde{\mathsf{E}}$ are used as the key of the block cipher.

From cryptanalyst's point of view, Type-III gives additional input to the attacker, hence it is likely more rounds could be attacked for Type-III, compared with the other two types.

4 Improving the Complexities of 7-Round Attacks

4.1 The Original 7-Round Attack by Sasaki

Following the framework of splice-and-cut MITM preimage attack depicted in Figure 4, Sasaki [24] invented a MITM preimage attack on AES hashing mode. Different with previous MITM preimage attacks on hash functions such as MD5 and Tiger, neutral words in the attack on AES hashing mode in [24] are not chosen from message, instead, they are from the internal state values of the compression function. This choice is mainly due to the fast diffusion of the AES key schedule, *i.e.*, the key schedule quickly diffuses any choice of key byte (as the neutral word) to all other key bytes in just a few rounds. As a result, in [24] the key-input/message-input are fixed to arbitrary constant values. Although the number of round of AES is much smaller than that of many dedicated hash functions such as MD5 and SHA-1, the round function is relatively heavier and the computation has faster diffusion, which prevents the MITM attacks from penetrating many rounds. Thus, more techniques were invented for attacking more rounds (yet the results are still not for the full rounds, and currently leaves comfortable margins).

There are two main techniques used for attacking more rounds in [24]. In a nutshell, the first is carefully crafted initial structure, and the second is matching through indirect but efficiently computable `MixColumns` relations.

Initial Structure (IS). Concretely, when constructing initial structure, by restricting the values of neutral bytes to a special set, the initial structure can be extended by one more round. These special set of values for the neutral bytes are chosen in such a way that their influences on particular output bytes of the `MixColumns` (or `InverseMixColumns`) are known constants, *i.e.*, have no influence. This reduces the number of unknown bytes after (resp. before) the first `MixColumns` (resp. `InverseMixColumns`) in each chunk, which is the starting point of each chunk. For example, when 4 bytes $B^{\mathsf{b}}[0, 1, 2, 3]$, *i.e.*, the bytes in the first column of the state $B^{\mathtt{b}}$, are chosen as the neutral bytes for the forward chunk, there are 2^{32} possible values. Among them, we can choose a small subset of size 2^8 in the following way: first fix an arbitrary constant $C^{\mathtt{neut}}$ of 3 bytes, then impose the restriction of $B^{\mathtt{f}} = \mathrm{MC}^{-1}(B^{\mathtt{b}}[0,1,2,3])$ and $B^{\mathtt{f}}[1,2,3] = C^{\mathtt{neut}}$. Since $C^{\mathtt{neut}}$ is of 24 bits and $B^{\mathtt{b}}[0,1,2,3]$ is of 32 bits , there will be 2^8 solutions for $B^{\mathtt{b}}[0,1,2,3]$, which can be obtained by enumerating all possible values of $B^{\mathtt{b}}[0]$ and for each computing the unique value of the other 3 bytes according to the restriction imposed through $C^{\mathtt{neut}}$.

Matching Through MixColumns (MTM). In the second technique, when matching at the meeting point, properties of the `MixColumns` are used again: instead of directly matching values of the same bytes in a state, matching via deterministic relations between bytes in different states (specifically, the states immediately before and after the `MixColumns`). This can extend the attack by one more round. Concretely, the AES `MixColumns` has the following feature: knowledge of any 4 out of the 8 input/output bytes to the MC will determine all other bytes, and knowledge of any additional byte(s) (*i.e.*, more than 4 bytes) can be used as a $8 \cdot (x-4)$-bit filter when a total of x bytes are known for $4 < x \leq 8$. With an example below, we demonstrate how this can be done in a way of meet in the middle.

To efficiently check whether the paired values in the two lists match through the `MixColumns` operation, one can test the consistence between the bytes column-wise, individually for each column in sequence. Suppose the two states before and after the `MixColumns` are $B^{\mathtt{f}}$ and $B^{\mathtt{b}}$, of which some bytes have been obtained independently via the forward and backward chunks, and the candidate values have been stored in two lists, *e.g.*, $B^{\mathtt{f}}[0,2]$ and $B^{\mathtt{b}}[1,2,3]$ in the first column. These bytes are related through MC as follows:

$$B^{\mathtt{f}}[0] = (\mathtt{e},\mathtt{b},\mathtt{d},\mathtt{9}) \cdot (B^{\mathtt{b}}[0], B^{\mathtt{b}}[1], B^{\mathtt{b}}[2], B^{\mathtt{b}}[3])^T,$$
$$B^{\mathtt{f}}[2] = (\mathtt{d},\mathtt{9},\mathtt{e},\mathtt{b}) \cdot (B^{\mathtt{b}}[0], B^{\mathtt{b}}[1], B^{\mathtt{b}}[2], B^{\mathtt{b}}[3])^T.$$

Let us denote by $g'(B^{\mathtt{b}}[1,2,3]) = (\mathtt{b},\mathtt{d},\mathtt{9}) \cdot (B^{\mathtt{b}}[1], B^{\mathtt{b}}[2], B^{\mathtt{b}}[3])^T$ a linear mapping from 3 bytes to 1 byte, and by $g''(B^{\mathtt{b}}[1,2,3]) = (\mathtt{9},\mathtt{e},\mathtt{b}) \cdot (B^{\mathtt{b}}[1], B^{\mathtt{b}}[2], B^{\mathtt{b}}[3])^T$ another linear mapping. Then

$$B^{\mathtt{f}}[0] = \mathtt{e} \cdot B^{\mathtt{b}}[0] \oplus g'(B^{\mathtt{b}}[1,2,3]) \quad \text{and} \quad B^{\mathtt{f}}[2] = \mathtt{d} \cdot B^{\mathtt{b}}[0] \oplus g''(B^{\mathtt{b}}[1,2,3]).$$

Cancelling out $B^{\mathtt{b}}[0]$ implies

$$\mathtt{d} \cdot B^{\mathtt{f}}[0] \oplus \mathtt{e} \cdot B^{\mathtt{f}}[2] = \mathtt{d} \cdot g'(B^{\mathtt{b}}[1,2,3]) \oplus \mathtt{e} \cdot g''(B^{\mathtt{b}}[1,2,3]).$$

For the sake of simplicity, denote by $M^{\mathtt{f}} = f(B^{\mathtt{f}}[0,2]) = \mathtt{d} \cdot B^{\mathtt{f}}[0] \oplus \mathtt{e} \cdot B^{\mathtt{f}}[2]$ a linear mapping from 2 bytes to 1 byte, and by $M^{\mathtt{b}} = g(B^{\mathtt{b}}[1,2,3]) = \mathtt{d} \cdot g'(B^{\mathtt{b}}[1,2,3]) \oplus \mathtt{e} \cdot g''(B^{\mathtt{b}}[1,2,3])$ another linear mapping from 3 bytes to 1 byte. Then find matches with $M^{\mathtt{f}} = M^{\mathtt{b}}$, which can be computed independently and stored in hash tables, then matched in the way of MITM. To be more general,

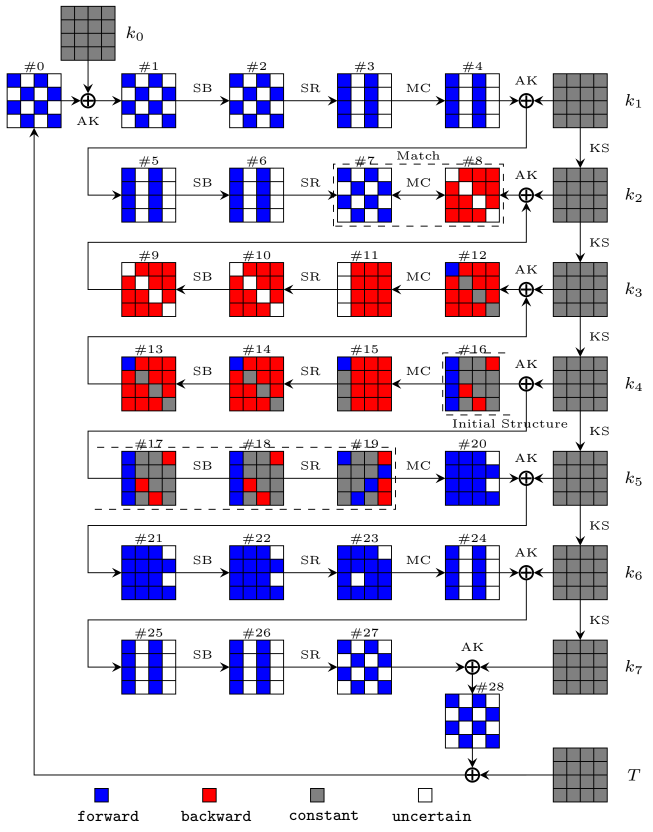


Figure 7: The 7-round MITM preimage attack on AES in [24]

Property 1. When x out of the 8 input and output bytes of the `MixColumns` are known, there is a filter of $(x-4)$ bytes $(= 8 \cdot (x-4)$ bits), and such filtering can be done in the way of meet in the middle.

This property is used in the matching of all attacks to be given in the rest of the paper, and for the sake of simplicity, the explicit expressions of $M^{\mathtt{f}}$ and $M^{\mathtt{b}}$, *i.e.*, the derived matching functions f and g from the input $B^{\mathtt{f}}$ and $B^{\mathtt{b}}$, are omitted. The total bits of filtering m is then the sum of 4 independent columns.

Sasaki's Attack. As depicted in the Figure 7, details of the pseudo-preimage attack in [24] are as follows.

Attack 1: The original pseudo-preimage attack on 7-round AES hashing mode [24]

Attack configuration

1. Initial structure:
   - Neutral bytes for forward: 2^8 possible values of $\#16[0,1,2,3]$, s.t., $\#15[1,2,3]$ equals predefined constant $C_0^{\texttt{neut}}$, i.e., $\mathrm{MC}^{-1}(\#16[0,1,2,3])[1,2,3] = C_0^{\texttt{neut}}$.
   - Neutral bytes for backward: 2^8 possible values of $\#19[12,14,15]$, s.t., their impacts on $\#20[13,15]$ equal predefined constant $C_1^{\texttt{neut}}$, i.e., $\mathrm{MC}(\#19[12,13,14,15])[13,15] = C_1^{\texttt{neut}}$.
2. Chunk separation:
   - Forward chunk: the computation following $\#20 - \#28 - \#0 - \#7$
   - Backward chunk: the computation following $\#15 - \#8$
3. Bytes for match:
   - $B^{\texttt{f}} = \#7[0,2;5,7;8,10;13,15]$
   - $B^{\texttt{b}} = \#8[1,2,3;4,6,7;8,9,11;12,13,14]$

Attack procedure

1. Fix constants:
   - key-input
   - $C^{\texttt{init}}$: 9 bytes values for $\#16[4,5,7,8,9,10,13,14,15]$
   - $C_0^{\texttt{neut}}$: 3 bytes values for $\#15[1,2,3]$
   - $C_1^{\texttt{neut}}$: 2 bytes values for impacts on $\#20[13,15]$.
2. Forward computation: for 2^8 values of the neutral bytes for forward – $\#16[0,1,2,3]$ s.t. $\#15[1,2,3] = C_0^{\texttt{neut}}$
   - compute in forward from $\#20 - \#28 - \#0 - \#7$
   - compute $M^{\texttt{f}}$ from $B^{\texttt{f}}$, store the results in the hash table $L^{\texttt{f}}$
3. Backward computation: for 2^8 values of the neutral bytes for backward – $\#19[12,14,15]$ s.t. $\#20[13,15] = C_1^{\texttt{neut}}$
   - compute in backward from $\#15 - \#8$
   - compute $M^{\texttt{b}}$ from $B^{\texttt{b}}$, store the results in the hash table $L^{\texttt{b}}$
4. Matching: match between $L^{\texttt{f}}$ and $L^{\texttt{b}}$, for each such match of partial state, all bytes at the initial structure are fixed, and hence the full states of #7 and #8 can be tested for full-state matching. Output (H_{N-1}, M_N) if full match is found, otherwise, go back to step 1 with some other values for the fixed constants at initial structure and repeat.

Attack complexity. The formula in Eq. (1) can be directly applied for the complexity analysis of Attack 1. In the setting of Attack 1, freedom degrees for both of the forward and the backward chunks are 8 bits, *i.e.*, $d_1 = d_2 = 8$, the number of bits for match come from 4 bytes, *i.e.*, $m = 32$ (for each column, there are $x = 2 + 3 = 5$ know bytes, resulting in a filter of $x - 4 = 1$ byte according to Property 1). Plugging $(d_1 = 8, d_2 = 8, m = 32)$ into Eq. (1) obtains the time complexity $2^{128-8} = 2^{120}$. The memory complexity is 2^8 which comes from the cost for storing $L^{\texttt{f}}$ and $L^{\texttt{b}}$.

Attack features. Attack 1 has the following features:

– Advantage: it is general because it is independent with key schedule algorithms and fixes keys as constant. Thus, it is applicable for AES-128/192/256 simultaneously and applicable for all PGV hash modes in [23].
– Disadvantage: the freedom degrees of neutral bytes for backward chunk are not fully exploited; freedom degrees from the key/message are not utilized. Thus, the computational complexity is not optimal.

4.2 The improved 7-Round Attack by Wu *et al.*

By exploiting the additional degrees of freedom of neutral bytes from the internal states for backward chunk and assuming multiple targets, the attack in [24] was improved to be a more efficient multi-target pseudo-preimage attack in [29].

In the multi-target preimage attack, it is assumed there are 2^t available target T's. As depicted in Figure 13, details of the improved multi-target pseudo-preimage attack in [29] are as follows:

Attack 2: The improved multi-target pseudo-preimage attack on 7-round AES hashing mode [29]

Attack configuration

1. Initial structure:
   – Neutral bytes for forward: same with Attack 1
   – Neutral bytes for backward: $(2^8)^4 = 2^{32}$ possible values of $\#19[1,2,3], \#19[4,5,6], \#19[8,9,11], \#19[12,14,15]$, s.t., their impacts on $\#20[0,2], \#20[5,7], \#20[8,10], \#20[13,15]$ equal to predefined constants $C_1^{\mathtt{neut}} = C_{1,0}^{\mathtt{neut}} \| C_{1,1}^{\mathtt{neut}} \| C_{1,2}^{\mathtt{neut}} \| C_{1,3}^{\mathtt{neut}}$.
2. Chunk separation: same with Attack 1
3. Bytes for match: same with Attack 1

Attack procedure: Given 2^t values of target T

1. Fix constants:
   – key-input: same with Attack 1
   – $C_0^{\mathtt{neut}}$: same with Attack 1
   – $C_1^{\mathtt{neut}}$: 8 bytes values for impacts on $\#20[0,2,5,7,8,10,13,15]$.
2. Forward computation: for 2^8 values of the neutral bytes for forward – $\#16[0,1,2,3]$ s.t. $\#15[1,2,3] = C_0^{\mathtt{neut}}$
   – compute in forward from $\#20 - \#28$, for each of the 2^t given values of the target T
     • xor state $\#28$ with the value of the target T, and compute $\#0 - \#7$
     • compute $M^{\mathtt{f}}$ from $B^{\mathtt{f}}$, store results in $L^{\mathtt{f}}$

3. Backward computation: for each of the 2^{32} values of the neutral bytes for backward – #19[1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15] s.t. their impact on bytes #20[0, 2, 5, 7, 8, 10, 13, 15] $= C_1^{\mathtt{neut}}$
   – compute in backward from #15 – #8
   – compute M^{b} from B^{b}, store results in L^{b}
4. Matching: same with Attack 1.

Attack complexity. Attack 2 fits the multi-target setting discussed in Sect. 3.2, thus the complexity can be analyzed using Eq. (3) directly. In the setting of this attack, $n = 128$, $d_1 = 8$, and $m = 32$ which are all same with that in Attack 1. Whereas, $d_2 = 32$ is much larger than that in Attack 1. Additionally, we assume there are 2^t targets. Thus, plugging these values to Eq. (3) obtains the time complexity $2^{128-\min(8+t,32)} = 2^{120-\min(t,24)}$ and the memory complexity is $2^{8+\min(t,24)}$.

It is easy to see from here that the improvement only applies when the pseudo-preimages are converted into preimages, where multiple targets ($k > 0$) are possible. When only one single target ($t = 0$) is available, the attack complexities of [29] remain the same as in [24].

Attack features. Attack 2 has the following features:

– Advantage: it fully exploits the freedom that lies in neutral bytes for backward chunk, which increases the candidates for matching from 2^8 to 2^{32}.
– Disadvantage: it has to assume that there are more available target values to obtain the balanced freedom for forward chunk; freedom from the key/message are not utilized.

4.3 Introducing Neutral Bytes in Key

Sasaki in [24] has already discussed the possibility to improve the attack described in Sect. 4.1 in chosen-key setting, *i.e.*, introduce neutral bytes from key values. However, the conclusion is negative in consideration of the difficulties in adopting splice-and-cut technique in the key schedule function.

However, in this attack, neutral bytes from key are introduced and used as part of the forward chunk only, *i.e.*, none is used for backward chunk. This enables us to avoid separating the key bytes in every round key generated by the key schedule function, and at the meantime to exploit freedom from the key/message to balance the computations of forward and backward.

Explicitly, we introduce neutral bytes in key state in addition to the original neutral bytes in the encryption state. Those neutral bytes are all for forward chunk, while the effect of neutral bytes in the encryption state and those in the key state are different: neutral bytes in the encryption state affect the candidate values at the matching point through the AES round functions in each direction, and neutral bytes in the key state affect through all subsequent round keys (via key schedule). Thus, although they both affect on the same bytes at the matching point, they both provide possible candidate values for the matching bytes. Moreover, because the independent constructions between the AES round function and the AES key schedule, even the values chosen for the neutral bytes in encryption state and key state are within the same algebraic structure, *e.g.*, the same linear subspace, their effect on the matching values can be seen as independent.

Next, we present an attack exploiting freedom in the key input, which achieves better complexities when there are less available targets compared with the attack in [29].

Suppose there are 2^t available target T, details of our improved multi-target pseudo-preimage attack are as follows (which is illustrated in Figure 8):

Attack 3: Our improved multi-target pseudo-preimage attack on 7-round AES hashing mode [new]

Attack configuration

1. Initial structure:
   - Neutral bytes for forward (in state): same with Attack 1 and 2.
   - Neutral bytes for forward (in key): unlike in Attack 1 and 2, compute 2^8 possible values of $k_4[0,1,2,3]$,
     s.t., their impacts on $\#15[1,2,3]$ (when computing backward) equals predefined constant $C_2^{\mathtt{neut}}$.
     Other bytes in k_4 equals predefined constant $C^{\mathtt{key}}$.
     According to the key schedule algorithm of AES-128, $(k_4[0,1,2,3] \oplus k_3[4,5,6,7])$ equals to $k_4[4,5,6,7]$ which is a constant (*i.e.*, part of $C^{\mathtt{key}}$) and $(k_4[0,1,2,3] \oplus k_3[0,1,2,3])$ equals to a constant. Thus, $(k_3[0,1,2,3] \oplus k_3[4,5,6,7])$ equals to a constant. Thus, their impacts on $\#11[1,2,3]$ and $\#11[5,6,7]$ are also constants when computing backward. Denote the constant influence on $\#11[5,6,7]$ by $C_2^{\mathtt{neut}''}$ which can be deduced from $C_2^{\mathtt{neut}}$ and $C^{\mathtt{key}}$ and will be used during backward computations.
     As a result, for the forward chunk, the 2^8 additional degrees of freedom from neutral bytes $k_4[0,1,2,3]$ are obtained.
   - Neutral bytes for backward: same with Attack 2.
2. Chunk separation: same with Attack 1 and 2.
3. Bytes for match: unlike in Attack 1 and 2, there are equivalently 8×3 bits for match: Denote the equivalent sub-key of k_2 by uk_2, which can be computed via $\mathrm{MC}^{-1}(k_2)$:
   - $B^{\mathrm{f}} = (\#7 \oplus uk_2)[0,2;8,10;13,15]$, note the second column is not ulitized.
   - $B^{\mathrm{b}} = \#8[1,2,3;8,9,10;12,13,14]$, note the second column is not ulitized either.

Attack procedure.

1. Fix constants:
   - $C^{\mathtt{key}}$: a value for 12-byte $k_4[4,5,6,7,8,9,10,11,12,13,14,15]$;
   - $C_0^{\mathtt{neut}}$ and $C_1^{\mathtt{neut}}$: same with Attack 2;
   - $C_2^{\mathtt{neut}}$: a value for 3-byte impact from neutral bytes in k_4 on $\#15[1,2,3]$; from $C_2^{\mathtt{neut}}$ and key schedule, compute the constant impact $C_2^{\mathtt{neut}\prime\prime}$ from $k_3[4,5,6,7]$ on $\#11[5,6,7]$.
2. Forward computation: for 2^8 values of neutral bytes for forward – $\#16[0,1,2,3]$ s.t. their impacts on $\#15[1,2,3] = C_0^{\mathtt{neut}}$ and for 2^8 values of neutral bytes in key $k_4[0,1,2,3]$ s.t. their impacts on $\#15[1,2,3] = C_2^{\mathtt{neut}}$
   - compute all required sub-key bytes from $k_4[0,1,2,3]$ and $C^{\mathtt{key}}$.
   - compute in forward from $\#20 - \#28$ (blue cells in Figure 8), for each of the 2^8 given values of the target T
     - xor state $\#28$ with T, and compute $\#0 - \#7$ (blue cells in Figure 8)
     - compute $M^{\mathtt{f}}$ from $B^{\mathtt{f}}$, store results in $L^{\mathtt{f}}$
3. Backward computation: for each of the 2^{32} values of the neutral bytes for backward – $\#19[1,2,3,4,5,6,8,9,11,12,14,15]$ s.t. their impact on bytes $\#20[0,2,5,7,8,10,13,15] = C_1^{\mathtt{neut}}$:
   - compute in backward from $\#15 - \#8$ (red cells in Figure 8)
   - compute $M^{\mathtt{b}}$ from $B^{\mathtt{b}}$, store the results in $L^{\mathtt{b}}$
4. Matching: same with Attack 1 and 2.

Attack complexity. Compared with Attack 2, this attack introduces one more neutral byte from the key, at the cost of losing one byte for matching. Thus, when using Eq. (3) to analysis the complexity, one has $d_1 = 8 + 8 = 16$, $d_2 = 32$, and $m = 24$. Thus, supposing there are 2^t targets, the time complexity of this attack is $2^{128-\min(16+t,24)} = 2^{112-min(t,8)}$ and the memory complexity is $2^{16+\min(t,8)}$.

Attack features. Attack 3 has the following features:

- Advantage: freedom degrees of neutral bytes for backward chunk are fully exploited; freedom degrees therefrom the key/message are also utilized. Thus, the computational complexity is better when less target are available.
- Disadvantage: the attack is not quite general because it dependents on key schedule algorithms. Thus, it cannot be applied without modification for AES-128/192/256.

4.4 Application to 7-Round AES-192 Hashing Mode

Compared with AES-128, the key schedule of AES-192 has relatively slow diffusion. Consequently, it is possible to select more neutral bytes from the key to provide more freedom for the forward chunk without influence on the backward. As a result, a similar pseudo-preimage attack on 7-round AES-192 hashing mode has

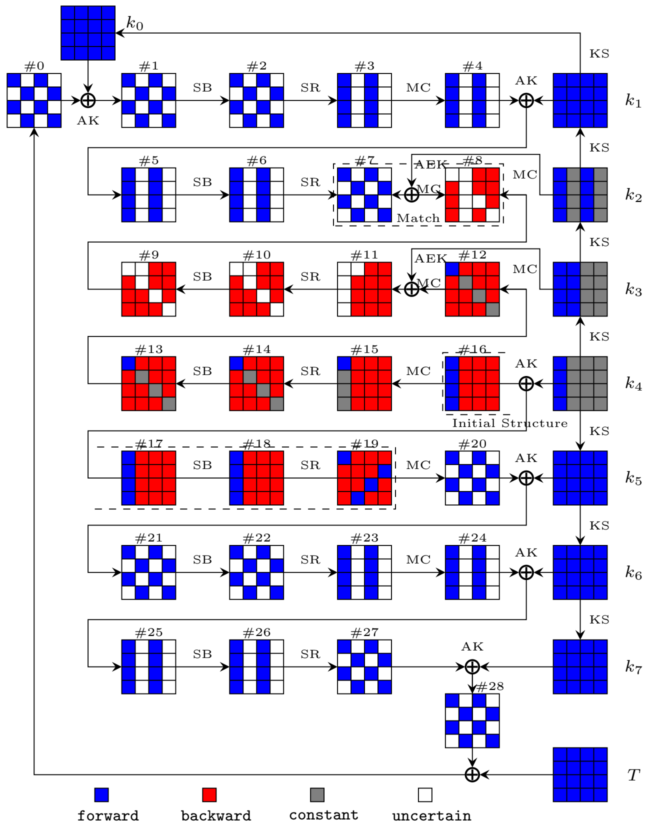


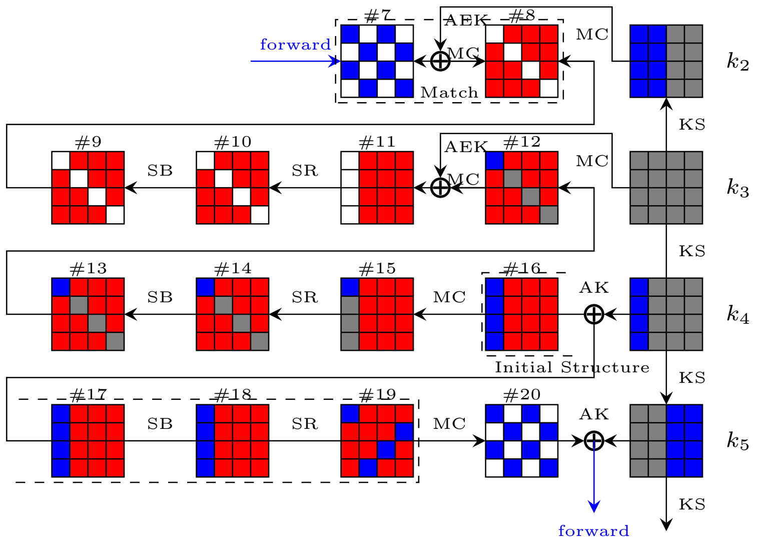
Figure 8: Introducing free bytes in key to improve the 7-round preimage attack on AES-128 hashing mode

lower computational complexity compared with that of 7-round AES-128 when there are less than 2^{16} targets.

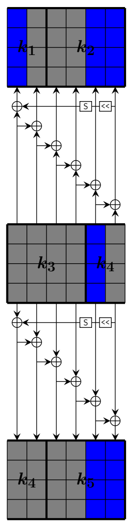
The main difference compared with the attack on AES-128 hashing mode lies in the choice of neutral bytes from key state k_4. In Attack 3, we restricted that neutral bytes $k_4[0,1,2,3]$ has constant influence on $\#15[1,2,3]$ such that $k_3[4,5,6,7]$ has constant influence on $\#11[5,6,7]$ if computing backward. For AES-192, according to the key schedule, the full key state k_3 can be fixed without dependence on neutral bytes in k_4 (as depicted in Figure 9b). Thus, the propagation of the influence brought by the neutral bytes in k_4 will not influence the backward computation which is relatively short.

Note that there are already 2^8 candidate values for neutral bytes $\#16[0,1,2,3]$ for forward chunk. There are 2^{32} candidate values for neutral bytes $k_4[0,1,2,3]$, out of which, we need only choose 2^{24} values considering that there are 2^{32} candidate values for neutral bytes for backward chunk. In this way, the number of candidate values in both $L^{\mathtt{f}}$ and $L^{\mathtt{b}}$ can be 2^{32} without assuming the existence of multiple targets.

The attack follows the same framework with previous attacks. The main different part is depicted in Figure 9. Follows the same complexity analysis of previous attack and applying Eq. (1), we conclude that, for this attack on 7-round AES-192 hashing mode, the total computational complexity is $2^{128-32} = 2^{96}$ computations of 7-round AES. The memory complexity is 2^{32}.



(a) The attack configuration



(b) Neutral bytes in k_4

Figure 9: Introducing free bytes in key to improve the 7-round preimage attack on AES-192 hashing mode

4.5 Application to 7-Round Kiasu-BC Hashing Mode

In the scenario where a teakable block cipher used in the PGV hashing mode and the tweaks can accept chosen inputs, freedom from this additional input might be exploited in similar attacks to the above ones.

Take Kiasu-BC for example whose encryption algorithm and key schedule are exactly that of AES-128. The Attack 2 can be applied and improved by introducing two additional neutral bytes from the tweak. Specifically, we choose $\#tk[0,1]$ as the neutral bytes, which can take 2^{16} values and which will not influence the backward computation (recall Figure 8). In this way, together with the 2^8 freedom from neutral byte $\#16[0,1,2,3]$, we can finally obtain $2^{8+16=24}$ candidates in $M^{\mathtt{f}}$ without any cost (which is unlike in Attack 3). Corresponding to the complexity analysis equation Eq. (1) and compared with that of Attack 2, the parameter b_1 increases from 8 to 24 and leaves all other parameters unchanged. Note the number of matching bits is already 32, thus, as done in Attack 3, *i.e.*, introducing one more neutral byte from the key at the cost of one byte from the matching bits, there will be no more gain.

Consequently, for 7-round Kiasu-BC hashing mode, a modified attack basing on Attack 2 and by introducing two neutral bytes from tweak will have a computational complexity $2^{128-\min(8+16,32)} = 2^{104}$ (recall Eq. 1). The memory complexity is 2^{24}. Note both key and tweak values are used here, so Type-III TBC-to-BC conversion is assumed here.

5 Extension to 8-Round Attacks

In this section, we extend the 7-round attacks of the previous section to 8-round attacks by introducing freedom from the key schedule. We first explain the technique used in our extension, then provide its application to AES-256, AES-192 and Kiasu-BC.

5.1 Techniques for Attacking 8 Rounds

We add one more round to the backward chunk based on the attack in Figure 13, and the new chunk separation is shown in Figure 10. Notice that $\#17[0]$ is a neutral byte for the forward computation and can have 2^8 values, and it is denoted by x. We regard the whole key schedule as a part of forward chunk and set the value of $k_4[0]$ to be $x \oplus c$, where c is a constant value, and the left 15 bytes of k_4 equals to predefined constant values. In this way, the forward chunk is computed with the knowledge on neutral bytes from both key and internal state. While for the backward computation, the value of $\#16[0]$ is fixed to be c, thus states $\#13$ to $\#15$ can be computed deterministically. As for k_3, the neutral byte $k_4[0]$ can only have impact on one column of k_3 to ensure a valid matching between state $\#7$ and $\#8$. And the impact on k_5 should not overlap with the neutral bytes for the backward chunk in state $\#20$. The value of k_2 does not have impact on the backward computation since the `MixColumns` is linear and we can instead find the match between $\#7 \oplus MC^{-1}(k_2)$ and $\#8$ through the `MixColumns`.

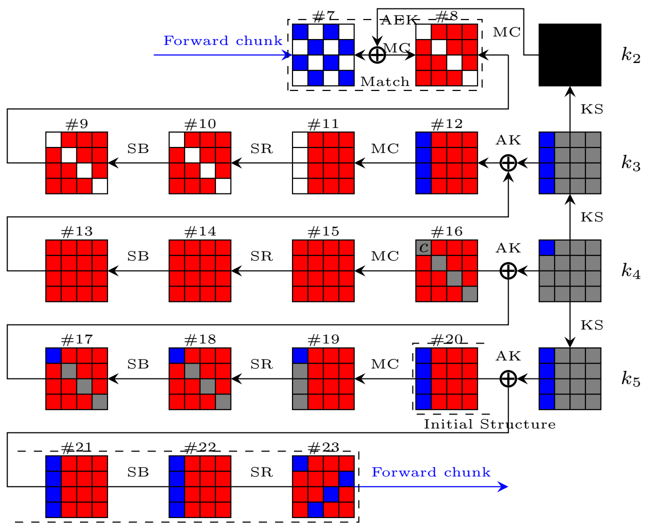
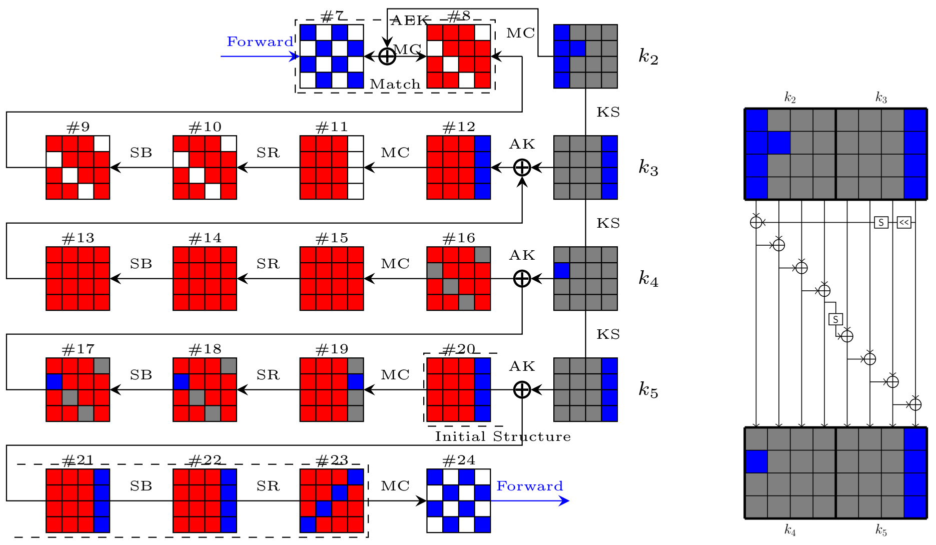


Figure 10: Basic idea of the extended 8-round attack



(a) The attack configuration **(b)** Key states

Figure 11: Introducing free bytes in key to launch a 8-round preimage attack on AES-256 hashing mode

5.2 The 8-Round Attack against AES-256

Due to the slower diffusion in the key schedule, we can apply the technique of Section 5.1 to the attack on 8-round AES-256 hashing mode. The key path is initialized from the 256-bit state k_4 and k_5 in the way that the freedom in k_4 could neutralize the randomness in backward chunk, and additional 2^{32} freedom degrees could be introduced in k_5, which is shown in Figure 11.

Attack 4: The pseudo-preimage attack on 8-round AES-256 hashing mode

Attack configuration

1. Initial structure:
   – Neutral bytes for forward (in state): 2^8 possible values of #20[12,13,14, 15], s.t., #19[12,14,15] equals predefined constant C_0^{neut},
   – Neutral bytes for forward (in key): 2^{32} possible values of $k_5[12, 13, 14, 15]$, s.t., their impacts on $\#19[12, 14, 15]$ equals predefined constant $C_1^{\texttt{neut}}$.
     The value of $k_4[1]$ is chosen in the way so that its XOR difference with #17[1] equals to a predefined constant $C^{\texttt{byte}}$, i.e., $k_4[1] = \#17[1] \oplus C^{\texttt{byte}}$.
     As a result, for the backward chunk, the states #13 to #16 can be computed deterministically.
     Other bytes in k_4 and k_5 equal to predefined constant $C^{\texttt{key}}$.
   – Neutral bytes for backward: $(2^8)^4 = 2^{32}$ possible values of $\#23[0, 1, 2], \#23[4, 5, 7], \#23[8, 10, 11], \#23[13, 14, 15]$,
     s.t., their impacts on $\#24[0, 2], \#24[5, 7], \#24[8, 10], \#24[13, 15]$ equal predefined constant $C_2^{\texttt{neut}} = C_{2,0}^{\texttt{neut}} \| C_{2,1}^{\texttt{neut}} \| C_{2,2}^{\texttt{neut}} \| C_{2,3}^{\texttt{neut}}$.
2. Chunk separation:
   – Forward chunk: the computation following #24 - #32 - #0 - #7
   – Backward chunk: the computation following #19 - #8
3. Bytes for match: Denote the equivalent sub-key of k_2 by uk_2, which can be computed via $\mathrm{MC}^{-1}(k_2)$:
   – $B^{\mathrm{f}} = (\#7 \oplus uk_2)[0, 2; 5, 7; 8, 10; 13, 15]$
   – $B^{\mathrm{b}} = \#8[0, 2, 3; 4, 5, 7; 8, 9, 10; 13, 14, 15]$

Attack procedure

1. Fix constants:
   – $C_0^{\texttt{neut}}$: a value for 3-byte impact from neutral bytes in #20 on #19[12, 14, 15];
   – $C_1^{\texttt{neut}}$: a value for 3-byte impact from $k_5[12, 13, 14, 15]$ on #19[12, 14, 15];
   – $C^{\texttt{byte}}$: a value for byte #16[1];
   – $C^{\texttt{key}}$: a 27-byte constant value in k_4 and k_5;
   – $C_2^{\texttt{neut}}$: a value for 8-byte impact from neutral bytes in #23 on #24[0, 2, 5, 7, 8, 10, 13, 15].

2. Forward computation: for 2^8 values of #20[12, 13, 14, 15] s.t. their impact on #19[12, 14, 15] equals to $C_0^{\texttt{neut}}$, and for 2^{24} out of 2^{32} values of $k_5[12, 13, 14, 15]$ s.t. their impact on #19[12, 14, 15] equals to $C_1^{\texttt{neut}}$:
   - Compute the value of #17[1].
   - Determine the value of $k_4[1]$: $k_4[1] = \#17[1] \oplus C^{\texttt{byte}}$.
   - Compute all sub-keys from k_4 and k_5 through the key schedule.
   - Compute the forward chunk from #24 to #32, and from #0 to #7.
   - Compute $M^{\texttt{f}}$ from $B^{\texttt{f}}$, and store the results in a table $L^{\texttt{f}}$.
3. Backward computation. For 2^{32} values of the neutral bytes #23[0, 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 15] s.t. their impact on bytes #24[0, 2, 5, 7, 8, 10, 13, 15] equals to $C_2^{\texttt{neut}}$.
   - Compute the backward chunk from #19 to #8. The involved subkey bytes are constant in the backward computation.
   - Compute $M^{\texttt{b}}$ from $B^{\texttt{b}}$, and store the results in a table $L^{\texttt{b}}$.
4. Matching: follow the same idea of the Attack 3.

Attack complexity The freedom degrees for both of the forward and backward chunk are 32, *i.e.*, $d_1 = d_2 = 32$, and the number of bits for match comes from 32 bits ($m = 32$). Thus according to Eq. (1), the time complexity is 2^{96} and the memory complexity is 2^{32}

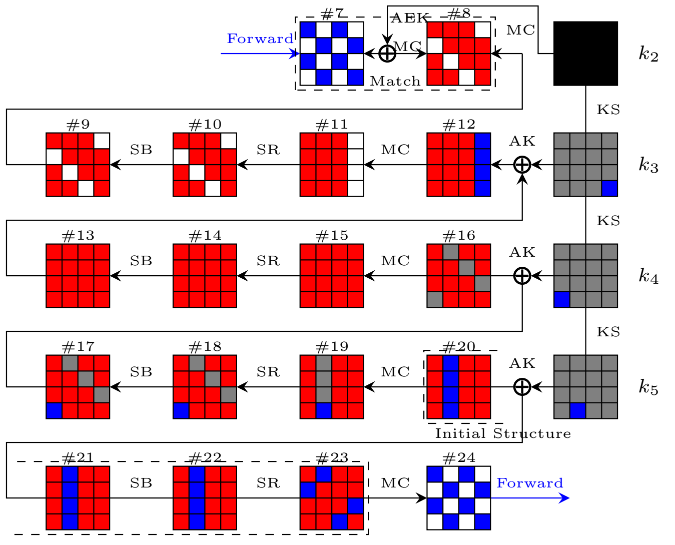
5.3 The 8-Round Attack against AES-192

The extended part of attack is illustrated in Figure 12a. The data path in the internal state is similar to the attack on AES-256, only the neutral bytes for the forward in state #20 is changed to the second column. As for the key state in Figure 12b, the value of $k_4[3]$ equals to the XOR difference between #17[3] and #16[3], and $k_3[15] = k_4[3]$. The left bytes of the 192-bit subkey state are set to be predefined values, and the next subkey can be computed accordingly.

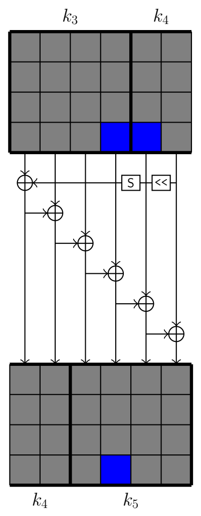
Note that the forward chunk has 2^8 freedom degrees that comes from #20[4,5,6,7], while the backward chunk has 2^{32} freedom degrees that comes from #23[0,2,3,5,6,7,8,9,10,12,13,15], thus we need multi-target scenario to balance the number of candidates in $L^{\texttt{f}}$ and $L^{\texttt{b}}$. The attack follows the same framework with Attack 2, and it requires $2^{120-min(t,24)}$ computations of 8-round AES and $2^{8+min(t,24)}$ memory.

5.4 The 8-Round Attack against Kiasu-BC

Since Kiasu-BC adds the same tweak to the first two rows of internal state for each round, we can introduce neutral bytes from tweak instead of key values, and the attack configuration in Figure 10 can be directly adopted with the only modification in the key and the tweak: the key states are set to be constant values, and the tweak is initialized in such a way that $tk[0] = \#17[0] \oplus C$ and the left 7 bytes are constant values. This attack follows the same framework with previous attacks, the time complexity is $2^{120-min(t,24)}$ and the memory complexity is



(a) The attack configuration



(b) Key states

Figure 12: Introducing free bytes in key to launch a 8-round preimage attack on AES-192 hashing mode

$2^{8+min(t,24)}$. Note the tweak values are used here, but not the key values, so both Type-II and Type-III TBC-to-BC conversions fit the attack setting here.

6 Conclusion

Under the general framework of meet-in-the-middle preimage attack against AES hashing modes introduced by Sasaki in 2011 and improved by Wu *et al.* in 2012, we made two observations: the key bits are not used, and the neutral bits in the two chunks are not balanced in Wu *et al.*'s improvement. In this paper, we introduced neutral bits from key, and to avoid dealing with the fast diffusion of the AES key schedule, neutral bits from key are introduced for one chunk only. By carefully choosing the key neutral bits, we found it was indeed possible while keeping the computation of the other chunk unaffected. Then the additional degrees of freedom are used in two ways, *i.e.*, to reduce time complexities and to extend the attack to more rounds. As a result, we improved the MITM preimage attack complexities for 7-round AES hashing modes under all 3 versions of AES, and extended the attack to 8 rounds under AES-192 and AES-256. The same was applied to Kiasu-BC.

Acknowledgements. We thank Lei Wang for helpful discussions. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative, Nanyang Technological University under research grant M4082123, and Singapore's Ministry of Education under grant M4012049.

References

1. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce (Nov 2001)
2. AlTawy, R., Youssef, A.M.: Preimage attacks on reduced-round Stribog. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 14: 7th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 8469, pp. 109–125. Springer, Heidelberg, Germany, Marrakesh, Morocco (May 28–30, 2014)
3. AlTawy, R., Youssef, A.M.: Second Preimage Analysis of Whirlwind. In: Lin, D., Yung, M., Zhou, J. (eds.) Information Security and Cryptology - 10th International Conference, Inscrypt 2014, Beijing, China, December 13-15, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8957, pp. 311–328. Springer (2014), https://doi.org/10.1007/978-3-319-16745-9_17
4. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for step-reduced SHA-2. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 578–597. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009)
5. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. Lecture Notes in Computer Science, vol. 5677, pp. 70–89. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2009)
6. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 5381, pp. 103–119. Springer, Heidelberg, Germany, Sackville, New Brunswick, Canada (Aug 14–15, 2009)
7. Aumasson, J.P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 5381, pp. 120–135. Springer, Heidelberg, Germany, Sackville, New Brunswick, Canada (Aug 14–15, 2009)
8. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. IEEE Computer 10(6), 74–84 (1977), https://doi.org/10.1109/C-M.1977.217750
9. Espitau, T., Fouque, P.A., Karpman, P.: Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 683–701. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
10. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 56–75. Springer, Heidelberg, Germany, Singapore (Dec 5–9, 2010)
11. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced meet-in-the-middle preimage attacks: First results on full tiger, and improved results on MD4 and SHA-2.

Cryptology ePrint Archive, Report 2010/016 (2010), http://eprint.iacr.org/2010/016
12. Guo, J., Su, C., Yap, W.: An improved preimage attack against HAVAL-3. Inf. Process. Lett. 115(2), 386–393 (2015), https://doi.org/10.1016/j.ipl.2014.10.016
13. Hong, D., Koo, B., Sasaki, Y.: Improved preimage attack for 68-step HAS-160. In: Lee, D., Hong, S. (eds.) ICISC 09: 12th International Conference on Information Security and Cryptology. Lecture Notes in Computer Science, vol. 5984, pp. 332–348. Springer, Heidelberg, Germany, Seoul, Korea (Dec 2–4, 2010)
14. Indesteege, S., Preneel, B.: Preimages for reduced-round tiger. In: Lucks, S., Sadeghi, A., Wolf, C. (eds.) Research in Cryptology, Second Western European Workshop, WEWoRC 2007, Bochum, Germany, July 4-6, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4945, pp. 90–99. Springer (2007), https://doi.org/10.1007/978-3-540-88353-1_8
15. Jean, J.: TikZ for Cryptographers. https://www.iacr.org/authors/tikz/ (2016)
16. Jean, J., Nikolić, I., Peyrin, T.: KIASU v1. Additional first-round candidates of CAESAR compeition, https://competitions.cr.yp.to/caesar-submissions.html (2014)
17. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)
18. Knellwolf, S., Khovratovich, D.: New preimage attacks against reduced SHA-1. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 367–383. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)
19. Leurent, G.: Message freedom in MD4 and MD5 collisions: Application to APOP. In: Biryukov, A. (ed.) Fast Software Encryption – FSE 2007. Lecture Notes in Computer Science, vol. 4593, pp. 309–328. Springer, Heidelberg, Germany, Luxembourg, Luxembourg (Mar 26–28, 2007)
20. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) Fast Software Encryption – FSE 2008. Lecture Notes in Computer Science, vol. 5086, pp. 412–428. Springer, Heidelberg, Germany, Lausanne, Switzerland (Feb 10–13, 2008)
21. Li, J., Isobe, T., Shibutani, K.: Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to SHA-2. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 264–286. Springer, Heidelberg, Germany, Washington, DC, USA (Mar 19–21, 2012)
22. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. The CRC Press series on discrete mathematics and its applications, CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA (1997)
23. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) Advances in Cryptology – CRYPTO'93. Lecture Notes in Computer Science, vol. 773, pp. 368–378. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 22–26, 1994)
24. Sasaki, Y.: Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. Lecture Notes in Computer Science, vol. 6733, pp. 378–396. Springer, Heidelberg, Germany, Lyngby, Denmark (Feb 13–16, 2011)
25. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) Advances in Cryptology – ASIACRYPT 2008. Lecture Notes in Computer Sci-

ence, vol. 5350, pp. 253–271. Springer, Heidelberg, Germany, Melbourne, Australia (Dec 7–11, 2008)

26. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) Advances in Cryptology – EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479, pp. 134–152. Springer, Heidelberg, Germany, Cologne, Germany (Apr 26–30, 2009)
27. Wang, L., Sasaki, Y.: Finding preimages of Tiger up to 23 steps. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption – FSE 2010. Lecture Notes in Computer Science, vol. 6147, pp. 116–133. Springer, Heidelberg, Germany, Seoul, Korea (Feb 7–10, 2010)
28. Wang, L., Sasaki, Y., Komatsubara, W., Ohta, K., Sakiyama, K.: (Second) preimage attacks on step-reduced RIPEMD/RIPEMD-128 with a new local-collision approach. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. Lecture Notes in Computer Science, vol. 6558, pp. 197–212. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14–18, 2011)
29. Wu, S., Feng, D., Wu, W., Guo, J., Dong, L., Zou, J.: (Pseudo) preimage attack on round-reduced Grøstl hash function and others. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 127–145. Springer, Heidelberg, Germany, Washington, DC, USA (Mar 19–21, 2012)

A Appendix 1

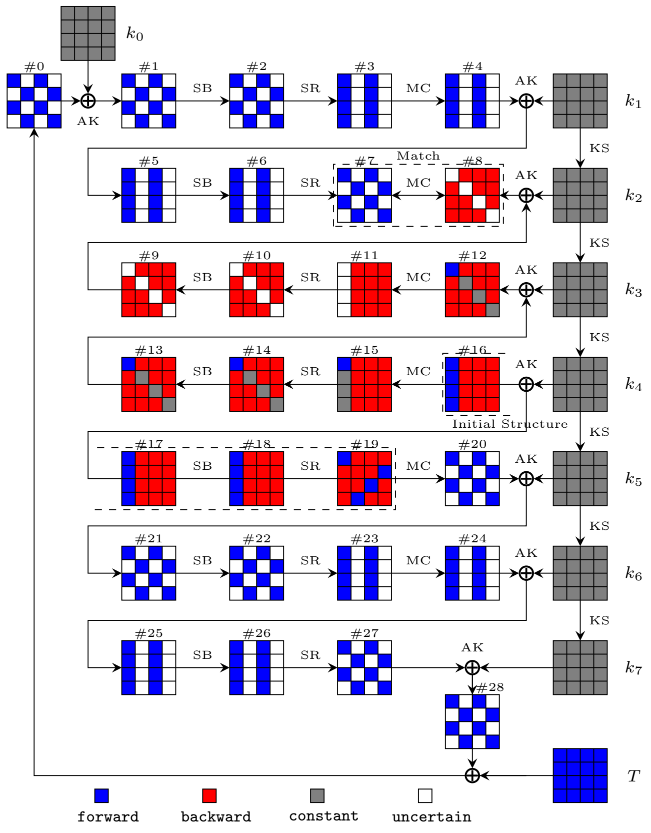


Figure 13: The multi-targets 7-round attack on AES in [29]