Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE

Leibo Li[1,2], Keting Jia[2] and Xiaoyun Wang[1,2,3⋆]

[1] Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
lileibo@mail.sdu.edu.cn
[2] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
ktjia@mail.tsinghua.edu.cn
[3] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
xiaoyunwang@mail.tsinghua.edu.cn

Abstract. This paper studies key-recovery attacks on AES-192 and PRINCE under single-key model by methodology of meet-in-the-middle attack. A new technique named *key-dependent sieve* is proposed to further reduce the memory complexity of Demirci *et al.*'s attack at EUROCRYPT 2013, which helps us to achieve 9-round attack on AES-192 by using a 5-round distinguisher; the data, time and memory complexities are 2^{121} chosen plaintexts, 2^{185} encryptions and 2^{185} 128-bit memories, respectively. The new technique is also applied to attack block cipher PRINCE. Instead of 6-round results in the previous cryptanalysis, we first present attacks on 8-round (out of 12) PRINCEcore and PRINCE with about 2^{53} and 2^{60} encryptions, respectively. Furthermore, we construct an interesting 7-round distinguisher and extend the attack to 9-round PRINCE; the attack needs about 2^{57} chosen plaintexts, 2^{64} encryptions and $2^{57.3}$ 64-bit memories.
Keywords: AES-192, PRINCE, Block Cipher, Meet-in-the-Middle Attack, Differential Characteristic.

1 Introduction

The meet-in-the-middle (MITM) attack was first applied to block cipher by Diffie and Hellman [1]. The idea is as follows: A block cipher E_K can be seen as the cascade of two subciphers $E_K = E^2_{K_2} \circ E^1_{K_1}$; for a plaintext-ciphertext pair (P, C), the adversary guesses K_1, K_2 and checks whether $E^1_{K_1}(P) = {E^2_{K_2}}^{-1}(C)$. If so, then the adversary might have the right key; otherwise, the guessed key must be wrong. This attack will be slower than the exhaustive search if the key information included in the union set of K_1 and K_2 ($K_1 \cup K_2$) is more than that of K. However, the adversary can apply the time-memory trade-off technique to pre-compute the value of $E^1_{K_1}(P)$ under each K_1 and store them in a hash table; then he guesses K_2, calculates ${E^2_{K_2}}^{-1}(C)$, and looks for it in the precomputed table. Nevertheless, for a block cipher with a good key schedule, the number of rounds that could be broke is still limited.

Demirci *et al.* [2,3] went a step further when they applied the MITM attack to AES. They follow an idea similar to in [4] and treated the cipher E as $E_K = E^2_{K_2} \circ E^m \circ E^1_{K_1}$. In their strategy, at first, the adversary builds a distinguisher in E^m associated with the so-called $\delta-$set: A set including 2^8 states, where a byte traverses all values and the other bytes are constant. If one considers encryptions of a $\delta-$set $(X^0, \cdots, X^{255})$ for the input of E^m, then the output value $(E^m(X^0), \cdots, E^m(X^{255}))$ can be expressed as a function of some intermediate variables of X^i and partial subkeys involved in E^m. In this paper, we symbol the sequence $(E^m(X^0), \cdots, E^m(X^{255}))$ as $\mathcal{S}$, and all intermediate variables along with sufficient subkeys for the computation of $\mathcal{S}$ as $\mathcal{V}$. If the total number of bits of $\mathcal{S}$ is small enough, then the distinguisher will work, and the adversary pre-computes all possible values of $\mathcal{S}$ and stores them in a table $\mathcal{H}$. With the distinguisher, the adversary mounts an attack by guessing the value of K_1, choosing

⋆ Corresponding author.

suitable plaintexts (thus obtaining the corresponding ciphertexts) to construct a $\delta - set$ of E^m, then partially decrypting the ciphertexts (by guessing K_2) to get the corresponding sequence $\mathcal{S}$, and checking whether the sequence $\mathcal{S}$ lies in $\mathcal{H}$. By this means, some wrong (K_1, K_2) pairs will be discarded while the right one will be kept.

At ASIACRYPT 2010, combining MITM attacks with the differential enumeration technique, Dunkelman, Keller and Shamir [5] proposed a novel idea for MITM attacks on AES. They set up a correlation in $\mathcal{V}$ and a truncated differential characteristic $\mathcal{D}$ with average probability. That is to say, if a pair conforms to $\mathcal{D}$, where they assume one pair of message belongs to the $\delta-$set $(X^0, \cdots, X^{255})$, then the possible values of $\mathcal{V}$ will be restricted to a small subset of the value space. The attack also contains two phases. In the offline phase, the adversary precomputes all possible values of $\mathcal{S}$ in the case of that one message of the $\delta-$set that follows the truncated differential characteristic $\mathcal{D}$, and stores the unordered sequences in a table $\mathcal{H}$. In the online phase, the adversary guesses subkeys (K_1, K_2) and looks for a pair that follows the differential characteristic $\mathcal{D}$. Once such a pair is found, the adversary takes one a pair of message and constructs a $\delta-$set for the input of E^m, then computes the corresponding sequence $\mathcal{S}$ by the guessed (K_1, K_2). In the end, he detects whether $\mathcal{S}$ belongs to $\mathcal{H}$ or not. Apparently, the direct advantage of this attack is reducing the memory requirement, while the disadvantage is increasing the data complexity of looking for a pair which conforms to $\mathcal{D}$ in the online phase.

More recently, Derbez, Fouque and Jean presented a significant improvement to Dunkelman *et al.*'s attacks at EUROCRYPT 2013 [6]. Combined with the rebound-like view of the cipher, they showed that the number of possible values of $\mathcal{S}$ could be further reduced. In fact, it is determined by the number of intermediate difference states of the pair satisfying $\mathcal{D}$. Their work not only reduces the memory requirement of precomputation, but also extends the distinguisher to one more round for AES-256.

Our contribution. While great progress have been made in previous works, we know that the bottleneck of Dunkelman *et al.*'s attack to be extended to more rounds is still the memory complexity. So it is very meaningful for us to look for some approaches to further reduce the memory requirement in the precomputation phase of the attack.

In this paper, we introduce an interesting technique which makes us utilize the 5-round distinguisher to 9-round attack on AES-192. In [6], Derbez *et al.* pointed out that for each possible value of $\mathcal{V}$ satisfying the truncated differential characteristic $\mathcal{D}$, there exists a corresponding subkey of E^m. More precisely, for AES block cipher, there exists 64-bit subkey for 4-round distinguisher and 192-bit subkey for 5-round distinguisher. In their work, a 4-round distinguisher was applied to attack 7-round AES-128 and 8-round AES-192, and 5-round distinguisher was applied to attack 9-round AES-256. On the basis of that, we take into account the application of the 5-round distinguisher to AES-192. We find that 16-bit information of subkey will be deduced twice by two independent approaches in such case, which means the corresponding $\mathcal{V}$ should be incorrect if the two 16-bit subkeys are different. Thus, we can perform a filter from all possible $\mathcal{V}$ with probability 2^{-16}, subsequently, the number of $\mathcal{S}$ should also be reduced by a factor of $1 - 2^{-16}$. As a result, combined with time/memory trade-off and other sophisticated methods, we present a non-marginal 9-round attack on AES-192 with about 2^{121} chosen plaintexts, 2^{185} encryptions and 2^{185} 128-bit memories, this result is the most efficient one compared with the previous works. Since this technique utilizes the self-contradictory phenomenon of subkeys involved in distinguisher E^m to reduce the number of possible values of $\mathcal{S}$, we call it *key-dependent sieve*.

In the second part of this paper, we apply this technique to attack the lightweight block cipher PRINCE recently proposed by Borghoff *et al.* at ASIACRYPT 2012. We first achieve an 8-round attack on PRINCEcore with a 6-round distinguisher, where about 2^{-16} of $\mathcal{S}$ in precomputation phase are kept after key-dependent sieve, and the attack needs about 2^{53} encryptions, 2^{53} chosen plaintexts and 2^{28} 64-bit memories. Then we extend the attack to 8-round PRINCE; the time, data and memory complexities are about 2^{60} encryptions, 2^{53} chosen plaintexts and 2^{30} 64-bit memories, respectively. Making further efforts, different from the previous distinguisher with byte-based or nibble-based sequence, we construct a 7-round distinguisher of PRINCEcore with bit-based sequences on account of the property of diffusion layer (with the branch number of 4). That leads to reducing $\mathcal{S}$ by 2^{16} times, because it is unnecessary

to consider all input nibbles of diffusion layer if we only need to compute one-bit of an output nibble. Combined with the key-dependent sieve (eliminating about $1 - 2^{-28}$ of $\mathcal{S}$), we launch a 9-round attack on PRINCE with about 2^{64} encryptions, 2^{57} chosen plaintexts and $2^{57.3}$ 64-bit memories.

Table 1 summarizes our results along with some major previous results of AES-192 and PRINCE under single-key model. The rest of this paper is organized as follows. Section 2 describes the improved attack on 9-round AES-192. In Section 3, we apply the new technique to block cipher PRINCE, which includes 8-round attacks on PRINCEcore and PRINCE, and 9-round attack on PRINCE. Finally, we conclude the paper in Section 4.

Table 1. Summary of the Attacks on AES-192 and PRINCE in the Single-key Model

Cipher	Rounds	Attack Type	Data	Time	Memory	Source
AES-192	8	MITM	2^{113}	2^{172}	2^{129}	[5]
	8	MITM	2^{113}	2^{172}	2^{82}	[6]
	8	MITM	2^{113}	2^{140}	2^{130}	[7]
	9	Bicliques	2^{80}	$2^{188.8}$	2^{8}	[8]
	9	MITM	2^{121}	2^{185}	2^{185}	Section 3
	Full	Bicliques	2^{80}	$2^{189.4}$	2^{8}	[8]
PRINCEcore	5	Integral	$5 \cdot 2^{4}$	2^{21}	2^{8}	[9]
	6	Differential	2^{48}	$2^{56.26}$	2^{48}	[10]
	6	Integral	$5 \cdot 2^{16}$	2^{30}	2^{16}	[9]
	8	MITM	2^{53}	2^{53}	2^{28}	Section 4.2
	12	Biclique	2^{40}	$2^{62.72}$	2^{8}	[10]
PRINCE	5	Integral	$5 \cdot 2^{4}$	2^{64}	2^{8}	[9]
	6	Integral	2^{16}	2^{64}	2^{16}	[9]
	8	MITM	2^{53}	2^{60}	2^{30}	Section 4.3
	9	MITM	2^{57}	2^{64}	$2^{57.3}$	Section 4.4
	12	MITM	2^{1}	$2^{125.47}$	Neglected	[9]

2 Primitive

3 The Meet-in-the-Middle Attack on 9-Round AES-192

Throughout the paper, the bit-wise exclusive OR (XOR) operation is denoted by $\oplus$, and bit string concatenation is denoted by $||$. A 128-bit (or 64-bit) state A is represented as a 4×4 matrix, we use the symbol $A[i]$ to express the bytes (or nibbles), where i ($i = 0, \cdots, 15$) represents the order of the byte (or nibble). We also define the column of A which includes four bytes (or nibbles) as $(A[4j] \cdots A[4j+3])$ for $j = 0, \cdots, 3$.

3.1 Brief Description of AES-192 and Previous Results

AES-192 [11] is an iterated block cipher which encrypts a 128-bit plaintext with a 192-bit key. It has 12 rounds, where each round is composed of four basic operations (see Fig. 1):

- SubBytes (SB) is a nonlinear byte-wise substitution that applies an 8 by 8 S-box to every byte.
- ShiftRows (SR) is a linear operation that rotates on the left of the $i-$th row by i bytes.
- MixColumns (MC) is a matrix multiplication over a finite field applied to each column.
- AddRoundKey (ARK) is an exclusive-or operation with the round subkey.

Before the first round an additional whitening ARK operation is performed, and in the last round the MC operation is omitted. In this section, X_i, Y_i, Z_i and W_i denote the state before SB, SR, MC, and ARX in round i, respectively.

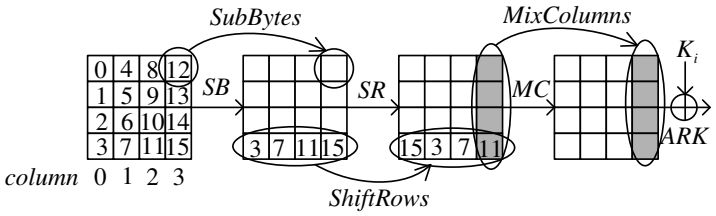

Fig. 1. The AES round function

Key schedule. The 192-bit master key is divided into 6 words of 32 bits each $(w_0, w_1, \cdots, w_5)$. Then the following algorithm is utilized to generate the 13 subkeys of 128-bit which consists of 52 words of 32-bit. Here, k_i represents the 128-bit subkey of round i, and $u_i = MC^{-1}(k_i)$.

- For $i = 6$ to $i = 51$ do the following:
  - If $i \equiv 0 \bmod 6$, then $w_i = w_{i-6} \oplus \text{SB}(\text{RotByte}(w_{i-1})) \oplus \text{Rcon}_{i/6}$,
  - Else $w_i = w_{i-6} \oplus w_{i-1}$,

where RotByte represents one byte rotation $(a_0, a_1, a_2, a_3) \rightarrow (a_1, a_2, a_3, a_0)$, and Rcon is an array of fixed constants. For more details about AES, we refer the readers to [11].

Previous attacks. For the reason of importance and popularity, there are many significant cryptanalysis results of AES-192 with various methods in previous years, such as [12,2,13,8,14,15,16]. For the single-key model, until 2010, the best result is an 8-round cryptanalysis with MITM attack given by Dunkelman *et al.* [5], which costs about 2^{172} encryptions. Then at Asiacrypt 2011, Bogdanov *et al.* [8] proposed 9-round and full-round attacks with bicliques method, which need $2^{188.8}$ and $2^{189.4}$ encryptions, respectively. Recently, Derbez *et al.* [7] improved the 8-round result of MITM attack to about 2^{140} encryptions.

3.2 The Meet-in-the-Middle Attack on 9-Round AES-192

Our attack is based on a 5-round distinguisher which is outlined by dotted line in Fig. 3 (Appendix A). The distinguisher was first proposed by Derbez *et al.* in [6], and was used to attack on 9-round AES-256. To be consistent with previous works, the $\delta-$set utilized in this section contains 2^8 values which traverses all values in byte $W_0[12]$ and are constants in the other bytes. Nevertheless, for the purpose of obtaining a better attack, we define it as an ordered sequence, and symbol it as $(W_0^0, \cdots, W_0^{255})$.

Proposition 1. *Considering the encryption of the first 2^5 values $(W_0^0, \cdots, W_0^{31})$ of the $\delta-$set through 5-round AES-192, in the case that a pair (W_0^i, W_0^j) $(0 \le i, j \le 255)$ conforms to the truncated differential characteristic outlined in Fig. 3, the corresponding 256-bit sequence $(Y_6^i[2] \oplus Y_6^0[2], \cdots, Y_6^i[2] \oplus Y_6^{31}[2])$ only takes about 2^{192} values.*

Proof. The proof is similar to those of the previous works in [5] and [6]. Firstly, the sequence $(Y_6^i \oplus Y_6^0, \cdots, Y_6^i \oplus Y_6^{31})$ is computed by the 42-byte variable

$$\mathcal{V} = (X_1^i[12], X_2^i[12, \cdots, 15], X_3^i[0, \cdots, 15], k_3[0, \cdots, 15], k_4[0, 5, 10, 15], k_5[2]),$$

where X^i represents the corresponding intermediate value of W_0^i.

Secondly, the 42-byte value is determined by the 26-byte variable of difference

$$(\Delta Y_1[12], \Delta Y_2[12, \cdots, 15], \Delta Y_3[0, \cdots 15], \Delta X_5[0, 5, 10, 15], \Delta X_6[2]),$$

where ΔY and ΔX denote corresponding difference values $(Y^i \oplus Y^j)$ and $(X^i \oplus X^j)$ for the pair (W_0^i, W_0^j). Moreover, by the aid of 208-bit difference, we can get a 192-bit subkey

$$(u_2[3, 6, 9, 12], k_3[0, \cdots, 15], k_4[0, 5, 10, 15]).$$

According to key schedule of AES-192, we can linearly deduce the forth column of k_0 and k_1, and the first two columns of k_2 with the knowledge of k_3. Meanwhile, $k_5[2]$ can also be obtained by $k_3[10]$ and $k_4[15]$. Here, we notice that $u_2[3, 6]$ could be computed by two independent methods, one is deduced immediately by the truncated differential characteristic, the other is got by k_3. Therefore, the corresponding 208-bit difference must be an incorrect state if the two values of $u_2[3, 6]$ are not equal. Otherwise, it is a proper state. The probability that a proper state occurs is about 2^{-16}, so there exist about 2^{192} possible values of $\mathcal{V}$, actually.

Finally, with the knowledge of $k_0[12]$, $k_1[12, \cdots 15]$ and $k_5[2]$, we obtain the ordered sequence $(Y_6^i[2] \oplus Y_6^0[2], \cdots, Y_6^i[2] \oplus Y_6^{255}[2])$ of the $\delta-$set. □

Note that, for the reason of reducing the memory complexity, we only need to compute the first 32-byte value of the $\delta-$set since it is sufficient to distinguish a proper sequences with the probability of $2^{192}/2^{256} = 2^{-64}$. Furthermore, the time complexity can be reduced by 2^3 times in the online phase of the attack.

Attack procedure. The attack is composed of two phases: precomputation phase and online phase. In the precomputation phase, we get all possible 256-bit sequences described as Proposition 1 by using of differential match technique.

For each 128-bit k_3, do the following steps.

1. Compute the subkey $u_2[3, 6]$, $k_1[12, 13, 14, 15]$, $k_0[12]$ by the key schedule.
2. Traverse $(\Delta W_5[2], W_5[0, 1, 2, 3])$ to compute $(\Delta X_5[0, 5, 10, 15], X_5[0, 5, 10, 15])$, and store $(X_5[0, 5, 10, 15])$ in a table T_0 indexed by $\Delta X_5[0, 5, 10, 15]$. There are about 2^8 values of $X_5[0, 5, 10, 15]$ for each index.
3. For all 32-bit difference $\Delta Y_2[12, \cdots, 15]$ and $\Delta X_5[0, 5, 10, 15]$, we apply the super-sbox technique [**?**] to connect the differences $\Delta Y_2[12, \cdots, 15]$ and $\Delta X_5[0, 5, 10, 15]$, and deduce the intermediate values (X_3, W_4). Then $Y_2[14, 15]$ is obtained by X_3 and $u_2[3, 6]$. Store these values with the index of the 48-bit value $(\Delta Y_2[12, \cdots, 15], Y_2[14, 15])$ in a table T_1. There are about 2^{16} values of $(\Delta X_5[0, 5, 10, 15], X_3, W_4[0, 5, 10, 15])$ corresponding to the index $(\Delta Y_2[12, \cdots, 15], Y_2[14, 15])$.
4. For each $(\Delta W_0[12], W_0[12], X_1[1, 6, 11])$, execute the following substeps.
   (a) Compute $X_1[12], X_2[12, 13, 14, 15]$, $\Delta Y_2[12, 13, 14, 15]$ and $Y_2[12, 13, 14, 15]$ by partial encryption.
   (b) Then look up table T_1 to get about 2^{16} values $(\Delta X_5[0, 5, 10, 15], X_3, W_4[0, 5, 10, 15])$ by the values of $(\Delta Y_2[12, 13, 14, 15], Y_2[14, 15])$. And $u_2[9, 12]$ are obtained by X_3 and $Y_2[12, 13]$.
   (c) For every $(\Delta X_5[0, 5, 10, 15], X_3, W_4[0, 5, 10, 15])$, we get 2^8 values of $X_5[0, 5, 10, 15]$ by accessing the table T_0. Then $k_4[0, 5, 10, 15] = X_5[0, 5, 10, 15] \oplus W_4[0, 5, 10, 15]$, and $k_5[2] = S(k_4[15]) \oplus k_3[10] \oplus Rcon$. Here we obtain the intermediate values $\mathcal{V}$.
   (d) For each $\mathcal{V}$, construct the $\delta-$set, compute the corresponding sequence $(Y_6[2] \oplus Y_6^0[2], \cdots, Y_6[2] \oplus Y_6^{31}[2])$, and store them in table $\mathcal{H}$.

In the online phase, we deduce all possible subkeys for the plaintext-ciphertext pairs satisfying the truncated differential characteristic, and identify the first 32 bytes value of a $\delta-$set. Finally, detect whether it belongs to the precomputation table. The attack procedure is described as follows.

1. Ask for encrypting 2^{81} structures of 2^{32} plaintexts, such that $P[1, 6, 11, 12]$ takes all 32-bit values and other bytes are constants. There are 2^{144} pairs totally.

2. For each pair, do the following substeps.
   (a) Guess the difference value $\Delta Y_7[8,9,10,11]$, and compute the subkey u_8. Then deduce $u_7[2,5]$.
   (b) Compute the difference $\Delta X_7[9,10]$, delete the wrong guesses which do not lead to $\Delta Z_6[8,9,11] = 0$, and there are about 2^{24} guesses remaining after this step.
   (c) For each remaining guess, deduce subkey $u_7[8,15]$.
   (d) Guess the difference $\Delta W_0[12]$, and compute the subkey $k_{-1}[1,6,11,12]$.
3. For each deduced subkey, select a pair of message and get the value $W_0[12]$. Then change the value of $W_0[12]$ to be $(0,\cdots,32)$ and compute plaintexts $(P^0,\cdots,P^{31})$. Query their corresponding ciphertexts, and get the corresponding sequence $(Y_6[2]\oplus Y_6^0[2],\cdots,Y_6[2]\oplus Y_6^{31}[2])$ by partial decryption, where $Y_6[2]$ represents the intermediate value of the selected message under the deduced subkeys. Note that we don't guess $u_6[10]$ in such case.
4. Delete the wrong subkeys by verifying whether the sequence lies in table $\mathcal{H}$. There are about $2^{176}\times 2^{-64}$ subkeys remaining in the end. Then exhaustively search for $u_7[9,\cdots 14]$ to find the real key, which needs about 2^{160} encryptions.

Complexity analysis. The time complexity of the precomputation phase is about 2^{192} simple operations, which also needs about 2^{193} 128-bit memories. The time complexity of the online phase is dominated by Step 3, which is equivalent to about $2^{144}\times 2^{32}\times 2^5\times 2^{-4}=2^{177}$ 9-round encryptions. With time-memory tradeoff, the adversary only needs to precompute a fraction 2^{-8} of possible sequences, so the memory complexity reduces to 2^{185} 128-bit memories, the data and time complexities increase to 2^{121} chosen plaintexts and 2^{185} encryptions, respectively.

4 The Improved Attack on Block Cipher PRINCE

In this section, we introduce the application of the new technique to the lightweight block cipher PRINCE. We firstly present attacks on 8-round PRINCEcore and PRINCE utilizing a 6-round distinguisher. After that, we propose an interesting 7-round distinguisher and 9-round attack on PRINCE.

4.1 The Block Cipher PRINCE

PRINCE [17] is a 64-bit block cipher with a 128-bit key. The key is split into two parts, $k=k_0||k_1$, and generates the subkey $k_0'=(k_0\ggg 1)\oplus(k_0\gg 63)$, where k_0 and k_1 are 64 bits. The subkeys k_0 and k_0' are used as input and output whitening keys respectively, while k_1 is used as the internal key for the core of the block cipher which is named PRINCEcore (see Fig. 2).

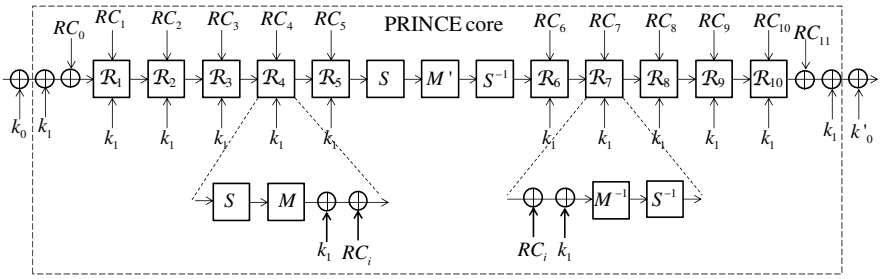

Fig. 2. Description of PRINCE block cipher

The PRINCEcore is a Substitution-Permutation Network composed of 12 rounds. Each round function R_i includes a key addition, a Sbox-layer, a linear layer, and a constant addition layer.

– The S-box layer applies a 4-bit S-box S to every nibble.

– In the linear layer, the 64-bit state is multiplied with a 64×64 matrix M (or M'), respectively. More precisely, for 64-bit state $(x_0||x_1||x_2||x_3)$, the linear layer M' is defined as

$$M'(x_0||x_1||x_2||x_3) \ = \ M_0(x_0)||M_1(x_1)||M_1(x_2)||M_0(x_3),$$

where M_0 and M_1 are two different 16×16 matrixes, of which each output bit is only determined by three bits of the input value. The linear diffusion layer M is composed of M' and an AES-like ShiftRows operation: $M \ = \ SR \circ M'$.
– Key addition is defined by the bitwise addition of the 64-bit subkey k_1.
– The constant addition layer is a XORed operation of the 64-bit state with a 64-bit constant RC_r ($r = 1 \cdots 12$). Note that the difference $RCr \oplus RC_{11-i}$ equals to a constant value $\alpha = 0xc0ac29b7c97c50dd$.

The encryption of PRINCEcore has three parts: forward rounds, middle rounds and backward rounds. The round functions is defined as follows.

– The forward rounds are defined as

$$\mathcal{R}_i(x) = M(S(x \oplus k_1 \oplus RC_i)) \qquad for \ i = 0 \cdots 4.$$

– The middle two rounds are defined as

$$\mathcal{R}(x) = S^{-1}(M'(S(x \oplus k_1 \oplus RC_5))) \oplus k_1 \oplus RC_6.$$

– The backward rounds are the inversion of forward rounds, which are defined as

$$\mathcal{R}_i(x) = S^{-1}(M^{-1}(x)) \oplus k_1 \oplus RC_i \quad for \ i = 7 \cdots 11.$$

The designers claim that given 2^n encryption/decryption pairs, the adversary can not recover the key with a complexity significantly lower than 2^{127-n} under the single-key setting. For more details of PRINCE, please refer to [17].

Previous results. As far as we know, the best results for PRINCEcore under single-key model are 6 rounds with 2^{30} encryptions [9] and full rounds with $2^{62.72}$ encryptions [10]. For PRINCE, the best results are 6 rounds with 2^{64} encryptions and full rounds with $2^{125.47}$ encryptions [9,18].

4.2 The Meet-in-the-Middle Attack on 8-Round PRINCEcore

In order to simplify the attack, we denote the equivalent key $M^{-1}(k_1)$ as K_1, seen Fig. 4 (Appendix B). Firstly, we present two useful propositions in the following.

Proposition 2. *Consider the function $y = g(x) = S^{-1}(M_i(S(x)))$ $(i = 0, 1)$, where S represents the S-box layer, M_i is defined in Section 4.1, x and y are 16-bit variables. Given two nonzero differences Δx and Δy, the equation*

$$g(x) \oplus g(x \oplus \Delta x) = \Delta y,$$

has a solution on average.

Proof. The function $y = g(x)$ is actually a super-sbox $\{0,1\}^{16} \rightarrow \{0,1\}^{16}$. According to the general property of S-box, there exists a pair on average that satisfies the known nonzero input and output differences. □

Thus, the function $(y_0||y_1||y_2||y_3) = S^{-1}(M'(S(x_0||x_1||x_2||x_3)))$ is sliced as four independent $\{0,1\}^{16} \rightarrow \{0,1\}^{16}$ super-sboxes. For equation

$$(\Delta y_0||\Delta y_1||\Delta y_2||\Delta y_3) = S^{-1}(M'(S(\Delta x_0||\Delta x_1||\Delta x_2||\Delta x_3))),$$

if all 16-bit differences Δx_i and Δy_i $(i = 0 \cdots 3)$ are nonzero values, we averagely get a pair to satisfy a given difference. Note that this operation can be performed by constructing two small difference distribution tables, the time complexity is about 2^{32} simple computations.

In this section, we define the δ-set including 2^4 values $(X^0, \cdots X^{15})$, which traverses all values of a nibble $X_1[0]$ (with an ordered sequence) and keeps the values in the other nibbles to be constants. The six round distinguisher is outlined with dotted line in Fig. 4, where the black boxes represent the nonzero differences for truncated differential characteristic, the gray boxes represent uncertain differences (zero or nonzero), and the white boxes represent the zero difference.

Proposition 3. *If there exists a pair $(X_1^i, \widetilde{X}_1)$ which satisfies the truncated difference characteristic as described in Fig. 4, then after 6-round encryption of a δ-set $(X_1^0, \cdots X_1^{15})$ which traverses the nibble $X[0]$, the corresponding sequence of ciphertexts $(X_6^0[0], \cdots, X_6^{15}[0])$ takes about 2^{28} values out of 2^{64} theoretical values.*

Proof. We consider the computation of $X_6^l[0]$ for $0 \leq l \leq 15$ and $l \neq i$. Actually, $X_6^l[0]$ is determined by a 96-bit intermediate variable

$$\mathcal{V} = (X_2^i[0, 7, 10, 13], X_3^i[0, \cdots, 15], K_4[0, 7, 10, 13]),$$

where X_2^i and X_3^i denote the corresponding intermediate variables of X_1^i. Using the known value of $(X_1^i[0], X_1^l[0])$, we obtain the value of $(Y_2^i[0] \oplus Y_2^l[0])$, and then get $(X_2^i[0, 7, 10, 13] \oplus X_2^l[0, 7, 10, 13])$. After that, based on the known value of $(X_2^i[0, 7, 10, 13])$, we get the difference $(X_3^i[0 \cdots 15] \oplus X_3^l[0 \cdots 15])$. Then according to the value of $(X_3^i[0, \cdots, 15])$, we obtain $(Y_4^l[0, 7, 10, 13])$. In the end, we get the value $X_6^l[0]$ through the known $(K_4[0, 7, 10, 13])$.

Then we focus on the possible 96-bit intermediate variables $\mathcal{V}$, if there exists a pair $(X_1^i, \widetilde{X}_1)$ which satisfies the truncated differential characteristic $\mathcal{D}$ as described in Fig. 4. We conclude that 96-bit $\mathcal{V}$ is determined by a 44-bit difference variable

$$(\Delta Y_2[0, 1], \Delta Y_3[0, 7, 10, 13], \Delta Y_4[0, 7, 10, 13], \Delta Y_5[0]),$$

where ΔY_j represents the intermediate difference $(Y_j^i \oplus \widetilde{Y}_j)$ for the pair $(X_1^i, \widetilde{X}_1)$. According to differential match technique, if all gray boxes are active, we deduce the value $(X_2^i[0, 7, 10, 13])$ with the difference $(\Delta Y_2[0, 1], \Delta Y_3[0, 7, 10, 13])$. Similarly, the value $(X_5^i[0, 7, 10, 13])$ is obtained by the difference $(\Delta Y_4[0, 7, 10, 13], \Delta Y_5[0])$, and the value (X_3^i, X_4^i) is deduced by the difference $(\Delta Y_3[0, 7, 10, 13]$, $\Delta Y_4[0, 7, 10, 13])$. In the end, $(K_4[0, 7, 10, 13])$ is computed subsequently. However, different from AES, the branch number of diffusion layer is 4 for PRINCE, so it may be possible that one or some gray boxes are inactive in some $\mathcal{D}$, then the differential match does not work for such $S-$boxes, and we have to search all possible input and output values of these $S-$boxes. But it is clear that the probability of this occurrence is not so high, we presume that it does not influence the total number of possible values of $\mathcal{V}$. Therefore, there are only about 2^{44} possible values for 96-bit $\mathcal{V}$ of X_1^i if a pair $(X_1^i, \widetilde{X}_1)$ conforms to the expected $\mathcal{D}$.

Finally, we prove that there are only 2^{28} possible values of $\mathcal{V}$ by key-dependent sieve. For each 44-bit difference, we can get the corresponding subkey $(K_4[0, 7, 10, 13])$. Moverover, we can also obtain the subkey $(K_3[0, 7, 10, 13])$ at same time. In the line of key schedule of PRINCEcore, they should be the same value. The probability of the subkeys to be equal to each other is about 2^{-16}, then the number of $\mathcal{V}$ is only about 2^{28}.

In a word, the sequence $(X_6^0[0], \cdots, X_6^{15}[0])$ only takes about 2^{28} possible values if there exists a pair $(X_1^i, \widetilde{X}_1)$ satisfying the truncated differential characteristic. □

Remark 1. As described in Proposition 3, $Y_5^i[1, 2, 3]$ $(i = 0, \cdots, 15)$ are also obtained in the computation of the sequence. Thus, the 192-bit sequence

$$(W_6^i[j] \oplus W_6^0[j], \cdots, W_6^i[j] \oplus W_6^{15}[j]) \quad for\ j = 1, 2, 3,$$

are deduced in the computation of the sequence $(X_6^0[0], \cdots, X_6^{15}[0])$. There are 2^{28} 192-bit sequences, too.

Attack procedure. We mount an attack on 8-round PRINCEcore by extending one round forward and one round backward of the 6-round distinguisher (see Fig. 4). The attack procedure is similar to that of the AES-192, we first need to precompute all possible sequences as follows.

1. As described in Proposition 2, construct two tables T_1 and T_2, which list all input and output difference values $(\Delta x, \Delta y)$ and their corresponding pairs (x, y, x', y') for functions $y = S^{-1} \circ M_0 \circ S(x)$ and $y = S^{-1} \circ M_1 \circ S(x)$, respectively.
2. For all $(\Delta Y_2[0,1], Y_2[0,1,2,3])$, deduce the value $W_2[0,7,10,13]$ and $\Delta Y_3[0,7,10,13]$, and then store the difference $(\Delta Y_2[0,1], Y_2[0], W_2[0,7,10,13])$ indexed by the value $\Delta Y_3[0,7,10,13]$ in table T_3. There are about 2^8 values for each index.
3. Similarly, for all $(\Delta Y_5[0], Y_5[0,1,2,3])$, deduce the values $(W_5[0,7,10,13], \Delta Y_4[0,7,10,13])$, and then store these values $(\Delta Y_5[0], Y_5[0], W_5[0,7,10,13])$ with the index of $\Delta W_5[0,7,10,13]$ in table T_4.
4. For each possible difference $(\Delta Y_3[0,7,10,13], \Delta Y_4[0,7,10,13])$, perform the following substeps:
   (a) Compute the differences ΔX_3 and ΔX_4, and then deduce the values $Y_3[0,7,10,13]$ and $Y_4[0,7,10,13]$ by querying the tables T_1 and T_2 in the first step.
   (b) Query the table T_3 with values $\Delta Y_3[0,7,10,13]$ to get $(\Delta Y_2[0,1], Y_2[0], W_2[0,7,10,13])$. And access the table T_4 with values $\Delta Y_4[0,7,10,13]$ to get $(\Delta Y_5[0], Y_5[0], W_5[0,7,10,13])$. If the equation $(W_2[0,7,10,13] \oplus Y_3[0,7,10,13]) = (W_5[0,7,10,13] \oplus Y_4[0,7,10,13])$ holds, $K_4[0,7,10,13] = W_5[0,7,10,13] \oplus Y_4[0,7,10,13]$. Then we compute $X_2[0,7,10,13]$ and $X_1[0]$. Here, we obtain $\mathcal{V}$.
   (c) On the basis of $\mathcal{V}$, we construct a $\delta-$set, compute the corresponding sequence $(X_6^0[0], \cdots, X_6^{15}[0])$, and store it in a table $\mathcal{H}$.

Then, in the online phase, we retrieve the master key as follows.

1. Ask for the encryptions of 2^{21} structures of 2^{32} plaintexts, such that nibbles $P[0 \cdots 7]$ traverse all possible values and the other nibbles are constants. For each structure, perform the following substeps:
   (a) Store the ciphertexts in a hash table indexed by nibbles $C[4 \cdots 15]$, the pairs which lie in the same line form right pairs. This step performs a 48-bit filter, the expected number of the remaining pairs is about 2^{15}.
   (b) Guess 16-bit value $k_1[0,1,2,3]$, compute $(\Delta Z_1[0,1,2,3], \Delta Z_6[0,1,2,3])$, delete pairs which do not satisfy $\Delta Z_1[1,2,3] = 0$ and $\Delta Z_6[1,2,3] = 0$. We expect that there are about 2^{-9} pairs remaining for each structure. In total, there are about 2^{12} pairs left for each 16-bit key guess after this step.
2. For every above 16-bit subkey guess, guess $k_1[4,5,6,7]$ and get the value $\Delta Z_1[4,5,6,7]$, then keep the pair which satisfies $\Delta Z_1[4,6,7] = 0$. Thus, for each 32-bit subkey guess, we obtain a pair which conforms to the expected differential characteristic on average.
3. For the obtained pair under each 32-bit key guess, select one message and compute the value $X_1[0]$, then deduce the corresponding plaintexts for $\delta-$set $(X_1^0, \cdots, X_1^{15})$. Afterwards, encrypt the plaintexts and get the corresponding ciphertexts. Partially decrypt the ciphertexts and get the sequence $(X_6^0[0], \cdots, X_6^{15}[0])$, detect whether the sequence lies in the table $\mathcal{H}$, if not, go back to step 2 for the next subkey guess. Otherwise, exhaustively search for 32-bit key $k_1[8, \cdots, 15]$ and recover the master key.

Complexity analysis. The time complexity of the precomputation phase is about 2^{40} simple computations. In the online phase, it is obvious that the time complexity is dominated by step 1, which needs about 2^{53} encryptions to get the plain-ciphertexts. The data complexity of the attack is about 2^{53} chosen plaintexts. The memory complexity is about 2^{28} 64-bit memory which is used to store all possible values of $(X_6^0[0], \cdots, X_6^{15}[0])$.

4.3 Extend the Attack to 8-Round PRINCE

For the 8-round attack on PRINCE, the adversary has to retrieve the equivalent key $\hat{k}_1[0,\cdots 7]$ and $\tilde{k}_1[0,1,2,3]$, where $\hat{k}_1 = k_0 \oplus k_1$ and $\tilde{k}_1 = k_0' \oplus k_1$. The attack procedure is demonstrated as follows:

1. Similar to the attack on PRINCEcore, look for a proper pair for every key guess, select one message and construct a $\delta-$set, then retrieve 64-bit key $(\hat{k}_1[0,\cdots 7], \tilde{k}_1[0,1,2,3], k_1[0,1,2,3])$, where $k_1[0,1,2,3]$ is decided by $\hat{k}_1[0,1,2,3]$, $\tilde{k}_1[0,1,2,3]$ and 1-bit key guess of k_1. This step needs about 2^{60} computations.
2. For the selected pair and 64-bit key obtained in the above step, guess the equivalent key $\tilde{k}_1[4,5,6,7]$, and compute
$$(W_6[1] \oplus W_6^0[1], \cdots, W_6[1] \oplus W_6^{15}[1]),$$
then find the correct guess by remark 1, where $W_6[1]$ represents the intermediate value of the message.
3. Similar to step 2, retrieve the key $\tilde{k}_1[8,9,10,11]$ and $\tilde{k}_1[12,13,14,15]$, respectively.
4. Search for the remaining 32-bit key $\hat{k}_1[8,\cdots 15]$ and find the master key.

In the total, the data complexity is about 2^{53}, the time complexity is about 2^{60}, the memory complexity is about 2^{30} 64-bit memory which is used to store 256-bit possible sequences.

4.4 The Meet-in-the-Middle Attack on 9-Round PRINCE

Our attack is based on a fact that each bit of the output value of M_i $(i = 0, 1)$ operation only depends on three bits of the input value. We first introduce a 7-round distinguisher which is composed of two forward rounds, two middle rounds and three backward rounds (without the last $S-$box layer). It is outlined with dotted line in Fig. 5 (Appendix C), where a represents a special value satisfying $M_1(a||0||0||0) = (b_1||b_2||b_3||0)$.

Proposition 4. *Suppose a pair $(X_1^i, \widetilde{X}_1)$ conforms to the differential characteristic described in Fig. 5, then after 7-round encryption of a $\delta-$set $(X_1^0, \cdots, X_1^{15})$, there are only about 2^{56} values for 152-bit sequence*
$$X_1^i[4,7], K_4[0,1,3,4,6,7,9,10,11,12,13,14], W_7^1[j]^{(1)}, \cdots, W_7^{15}[j]^{(1)},$$
for j=0,1,4,7,13,14, where $W_7[j]^{(1)}$ represents a bit value which is determined by three nibbles $Z_6[4,5,6]$.

Proof. According to the property of M' function, there must exist a bit of $Y_6[4]$ which doesn't depend on $Z_6[7]$, we symbol such bit as $Y_6[4]^{(1)}$. Then, for a $\delta-$set $(X_1^0, \cdots, X_1^{15})$, we consider the computation of $(Y_6^0[4]^{(1)} \cdots Y_6^{15}[4]^{(1)})$. In fact, this sequence is determined by a 128-bit intermediate variable

$$\mathcal{V} = (X_2^i[1,4,11,14], X_3^i[1,\cdots,15], K_4[0,1,3,4,6,7,9,10,11,12,13,14]).$$

Nevertheless, if there exists a pair $(X_1^i, \widetilde{X}_1)$ following the expected truncated differential characteristic, the 128-bit variable is determined by the 84-bit difference variable

$$(\Delta Y_2[4,7], \Delta Y_3[1,4,11,14], \Delta Y_4[0,1,3,4,6,7,9,10,11,12,13,14], \Delta Y_5[1,4,14]),$$

where ΔY_j means $(Y_j^i \oplus \widetilde{Y}_j)$.

However, the 76-bit subkeys are deduced from the difference variable

$$(K_3[1,4,11,14], K_4[0,1,3], K_4[4,6,7], K_4[9,10,11], K_4[12,13,14], K_5[1,4,14]).$$

By key schedule, we know $K_3[1,4,14] = K_4[1,4,14] = K_5[1,4,14]$ and $K_3[11] = K_4[11]$. Thus, there are only about $2^{84} \times 2^{-28} = 2^{56}$ values of $\mathcal{V}$.

Finally, with the knowledge of 48-bit equivalent key, it is sufficient to compute intermediate values

$$(W_6^j[0,1,3,4,6,7,12,13,14], Z_6^j[0,1,3,4,5,6,12,14,15]),$$

for $0 \leq j \leq 15$. Therefore, we obtain a bit-based sequence

$$(Y_6^1[j]^{(1)} \cdots, Y_6^{15}[j]^{(1)}),$$

for $j = 0 \cdots 7, 12, \cdots 15$. More precisely, for $j = 0 \cdots 3$, $Y_6^1[j]^{(1)}$ represents a bit value which doesn't depend on $Z_6[2]$. For $j = 4 \cdots 7$ and $j = 12 \cdots 15$, $Y_6^1[j]^{(1)}$ represent the values do not rely on $Z_6[7]$ and $Z_6[13]$, respectively. In our attack, we only take advantage of the case $j = 0, 1, 4, 7, 13, 14$ and compute the corresponding values of W_7. □

Note that, as a general approach, if we do not restrict the value of $\Delta Y_6[4]$ in such $\mathcal{D}$, and $\mathcal{S}$ of distinguisher is based on the nibble value of W_7, then the number of $\mathcal{S}$ is determined by 27 nibbles' difference

$$(\Delta Y_2[4, 7], \Delta Y_3[1, 4, 11, 14], \Delta Y_4, \Delta Y_5[1, 4, 11, 14], \Delta Y_6[4]).$$

Combined with key-dependent sieve (eliminating $1 - 2^{-32}$ of $\mathcal{S}$), there are 2^{76} possible values of $\mathcal{S}$, totally. It is infeasible since the complexity is too high to overstep the security claimed by designers. Therefore, we construct a bit-based sequence instead of a nibble-based sequence.

Attack procedure. Our attack is mounted by extending one round forward and one round backward for 7-round-like distinguisher (see Fig. 5). The precomputation is given as follows.

For each $\Delta Y_4[0, 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14]$, we do the following steps.

1. For all possible differences $\Delta Y_3[1, 4, 11, 14]$, deduce the values (Y_3, Y_4) by using of differential match technique, and store $(\Delta Y_3[1, 4, 11, 14], Y_3[1, 4, 11, 14], X_3, Y_4)$ indexed by 12-bit information $Y_4[1, 4, 14]$ in table T_1. There are about 2^4 values for every index.
2. For all the possible differences $\Delta Y_5[1, 4, 14]$ and the fixed value $\Delta Y_6[4]$.
   (a) Deduce the values $(W_5[0, 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14], Y_5[1, 4, 14], W_6[1, 4, 14], Z_6[4, 5, 6])$ by connecting the differences $\Delta Y_4[0, 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14]$, $\Delta Y_5[1, 4, 14]$ and $\Delta Y_6[4]$.
   (b) Then compute $K_5[1, 4, 14] = Y_5[1, 4, 14] \oplus W_6[1, 4, 14]$. Then we know $Y_4[1, 4, 14] = K_5[1, 4, 14] \oplus W_5[1, 4, 14]$, and access table T_1 to get $(\Delta Y_3[1, 4, 11, 14], Y_3[1, 4, 11, 14], X_3, Y_4)$. Thus, $K_4[0, 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14]$ is deduced from Y_4 and W_5.
   (c) Compute $W_2[1, 4, 11, 14] = Y_3[1, 4, 11, 14] \oplus K_4[1, 4, 11, 14]$, and deduce the values $X_2[1, 4, 11, 14]$ and $\Delta Y_2[4, 5, 6, 7]$ by partial decryption. If $\Delta Y_2[5] = \Delta Y_2[6] = 0$ holds , we obtain a value of $\mathcal{V}$, i.e., $(X_2[1, 4, 11, 14], X_3[1, \cdots, 15], K_4[0, 1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14])$.
   (d) By the value of $\mathcal{V}$, we construct the $\delta-$set and its corresponding 152-bit sequence as listed in Proposition 4, and store it in table $\mathcal{H}$.

Compared with the 8-round attacks, we need more complicated steps to identify the right key in the online phase as follows.

1. Ask for encrypting 2^{25} structures of 2^{32} plaintexts, such that nibbles $P[0 \cdots 7]$ traverse all possible values and the other nibbles are constants. For each structure, store the ciphertexts with a hash table indexed by nibbles $C[0, \cdots, 3, 8, \cdots, 15]$, the pair which lie in the same line may be a right pair. There are about 2^{40} right pairs remaining.
2. Look for pairs which satisfy the truncated differential characteristic.
   (a) Guess 16-bit equivalent key $\hat{k}[4, 5, 6, 7]$, compute the value $\Delta Z_1[4, 5, 6, 7]$, and delete the pairs which do not satisfy $\Delta Z_1[5, 6, 7] = 0$, where $\hat{k} = k_0 \oplus k_1$. The expected number of left pairs is about 2^{28} after this step.
   (b) Guess 16-bit equivalent key $\tilde{k}[4, 5, 6, 7]$, compute the value $\Delta Z_7[4, 5, 6, 7]$ and only keep the pairs satisfying $\Delta Z_7[5, 6, 7] = 0$, where $\tilde{k} = k_0' \oplus k_1$. There are about 2^{16} remaining pairs.
   (c) By the 32-bit equivalent key

$$\begin{cases} \hat{k}[4, 5, 6, 7] = k_1\{17, \cdots, 32\} \oplus k_0\{17, \cdots, 32\} \\ \tilde{k}[4, 5, 6, 7] = k_1\{17, \cdots, 32\} \oplus k_0\{18, \cdots, 33\}, \end{cases}$$

we deduce the master key $(k_1\{17,\cdots,32\}, k_0\{18,\cdots,33\})$ by guessing 1-bit value $k_0\{17\}$, where $k\{j\}$ symbol the $j-$th bit of k. Compute the difference $\Delta W_7[4]$ and discard the pairs which do not satisfy $\Delta W_7[4] = a$. Thus, there are about 2^{12} pairs left.

(d) Guess $\hat{k}[0,1,2,3]$, compute the value $\Delta Z_1[0,1,2,3]$, keep the pairs which satisfy $\Delta Z_1[0,1,2] = 0$. There is a pair left for each 49-bit key guess on average.

3. For the left pairs under the 49-bit key guess, select one message and guess subkey $(k_1[1,14], k_0\{64\})$, then do the following substeps.

(a) Compute the value $(X_1[4,7])$, construct a $\delta-$set, and get its ciphertext. Then, compute 48-bit value

$$W_7^1[j]^{(1)}, \cdots, W_7^{15}[j]^{(1)} \quad for\ j = 1, 4, 14,$$

check the hash table $\mathcal{H}$, and find the sequences which have the same 56-bit value

$$(X_1[4,7], W_7^1[j]^{(1)}, \cdots, W_7^{15}[j]^{(1)}).$$

There are about a sequence obtained for each 58-bit key guess.

(b) For each 58-bit key guess along with its sequence, deduce the real key k_1 by the information of 48-bit K_4 (in sequence) and 24-bit k_1 (guessed). It is obvious that there exists 8-bit information redundance, so there are about 2^{50} 58-bit key guesses remaining after this step.

(c) For every the key guess along with its sequence, deduce $k_0\{1\cdots16\}$, compute $\tilde{k}[0,1,2,3]$, and get 48-bit value

$$W_7^1[j]^{(1)}, \cdots, W_7^{15}[j]^{(1)} \quad for\ j = 0, 7, 13.$$

Check whether it equals to the value of sequence. The expected number of proper 58-bit key guess is about $2^{50} \times 2^{-48}$.

4. For the remained 58-bit key guesses along with the computation of k_1, search for 30-bit $k_0\{34\cdots63\}$ to obtain the master key.

Complexity analysis. The time complexity of the precomputation phase is dominated by substeps 2.(b) and 2.(c), which are about 2^{64} computations. The memory requirement is about 2^{56} 152-bit, which is equivalent to $2^{57.3}$ 64-bit. In the online phase, step 1 needs about 2^{57} computations. The time complexity of step 2 is dominated by substep 2.(d), which needs about 2^{61} computations. The time complexity of step 3 is dominated by substep 3.(a), which is about $2^{58} \times 2^4$ computations. Overall, the attack needs about 2^{57} chosen plaintexts and $2^{57.3}$ 64-bit memory bytes, the time complexity is lower then 2^{64} 9-round encryptions.

5 Conclusion

In this article, we propose a new technique named key-dependent sieve to further reduce the memory complexity of MITM attack. In particular, we apply this technique to attack on 9-round AES-192, 8-round PRINCEcore and PRINCE. Furthermore, we propose a bit-based distinguisher and achieve 9-round attack on PRINCE. To the best of our knowledge, these are the most efficient results in single-key model for AES-192 and PRINCE. The new technique takes an interesting view of MITM attack, we believe that this view, in a sense, will give rise to more attentions about application of MITM cryptanalysis.

References

1. Diffie, W., Hellman, M.E.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer **10** (1977) 74–84
2. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In Nyberg, K., ed.: FSE 2008. Volume 5086 of Lecture Notes in Computer Science., Springer (2008) 116–126

3. Demirci, H., Taskin, I., Çoban, M., Baysal, A.: Improved meet-in-the-middle attacks on aes. In Roy, B.K., Sendrier, N., eds.: INDOCRYPT 2009. Volume 5922 of Lecture Notes in Computer Science., Springer (2009) 144–156
4. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference. (2000) 230–241
5. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Abe, M., ed.: Advances in Cryptology - ASIACRYPT 2010. Volume 6477 of Lecture Notes in Computer Science., Springer (2010) 158–176
6. Derbez, P., Fouque, P.A., Jean, J.: Improved key recovery attacks on reduced-round aes in the single-key setting. In: EUROCRYPT 2013 (to appear). (2013)
7. Derbez, P., Fouque, P.A., Jean, J.: Exahusting demirici-selçuk meet-in-the-middle attacks against reduced-round aes. In: FSE 2013 (to appear). (2013)
8. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In Lee, D.H., Wang, X., eds.: Advances in Cryptology - ASIACRYPT 2011. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 344–371
9. Jean, J., Nikolic, I., Peyrin, T., Wang, L., Wu, S.: Security analysis of prince. In: FSE 2013 (to appear). (2013)
10. Farzaneh, A., Eik, L., Stefan, L.: On the security of the core of prince against biclique and differential cryptanalysis. In: IACR Cryptology ePrint Archive 2012/712. (2012)
11. National Institute of Standards and Technology: ADVANCED ENCRYPTION STANDARD. In: In FIPS PUB 197, Federal Information Processing Standards Publication. (2001)
12. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In Schneier, B., ed.: Fast Software Encryption 2008. Volume 1978 of Lecture Notes in Computer Science., Springer (2000) 213–230
13. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New Impossible Differential Attacks on AES. In Chowdhury, D.R., Rijmen, V., Das, A., eds.: Progress in Cryptology - INDOCRYPT 2008. Volume 5365 of Lecture Notes in Computer Science., Springer (2008) 279–293
14. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full aes-192 and aes-256. In Matsui, M., ed.: ASIACRYPT 2009. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 1–18
15. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round aes-128. In Gong, G., Gupta, K.C., eds.: INDOCRYPT 2010. Springer (2010) 282–291
16. Wei, Y., Lu, J., Hu, Y.: Meet-in-the-middle attack on 8 rounds of the aes block cipher under 192 key bits. In Bao, F., Weng, J., eds.: ISPEC 2011. Volume 6672 of Lecture Notes in Computer Science., Springer (2011) 222–232
17. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: Prince - a low-latency block cipher for pervasive computing applications - extended abstract. In Wang, X., Sako, K., eds.: ASIACRYPT 2012. Volume 7658 of Lecture Notes in Computer Science., Springer (2012) 208–225
18. Soleimany, H., Blondeau, C., Yu, X., Wu, W., Nyberg, K., Zhang, H., Zhang, L., Wang, Y.: Reflection cryptanalysis of prince-like ciphers. In: FSE 2013 (to appear). (2013)

A Attack on 9-Round AES-192

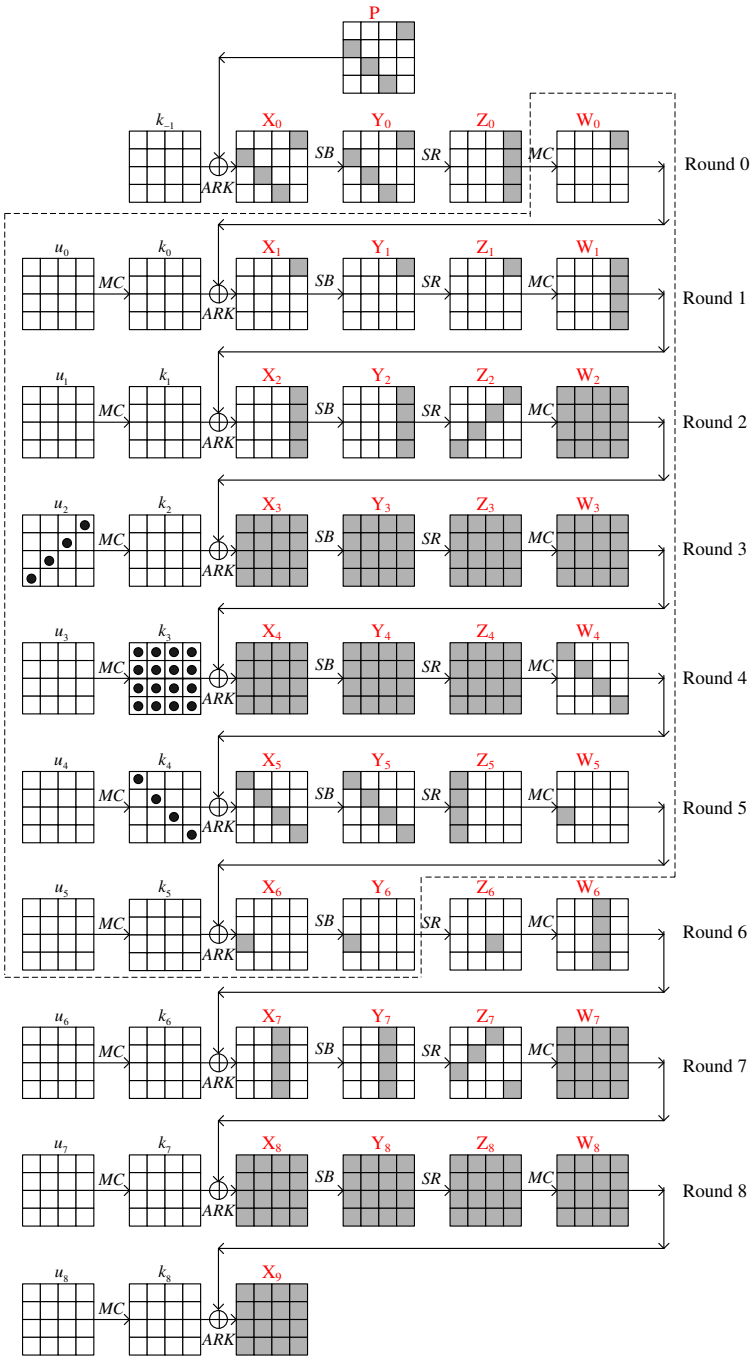

Fig. 3. The Truncated Differential Characteristic of the Attack on 9-Round AES-192

B Attack on 8-Round PRINCEcore

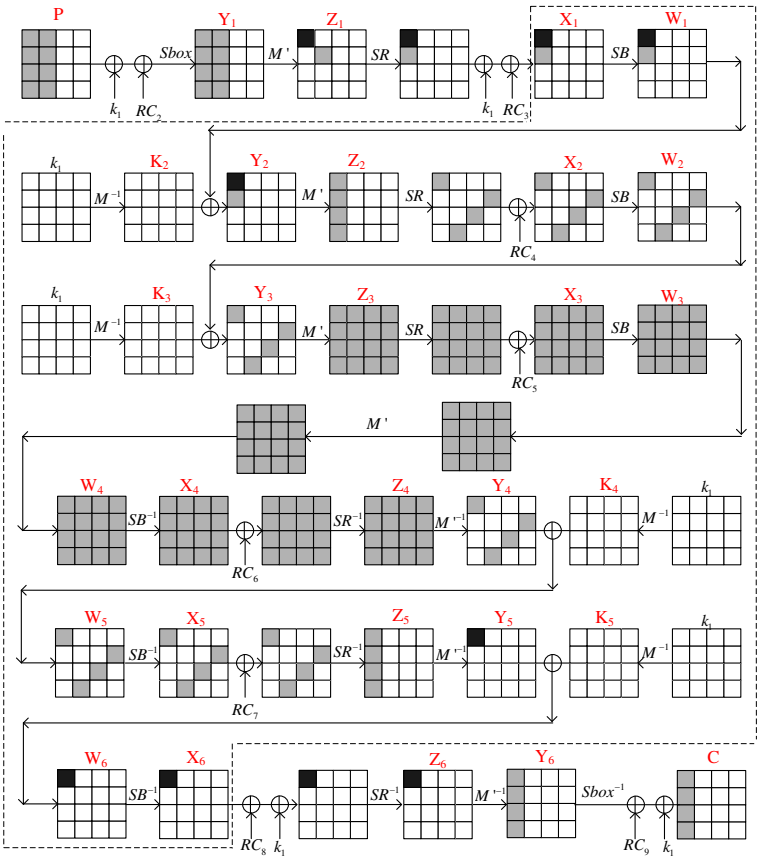

Fig. 4. The Truncated Differential Characteristic of the Attack on 8-Round PRINCEcore

C Attack on 9-Round PRINCE

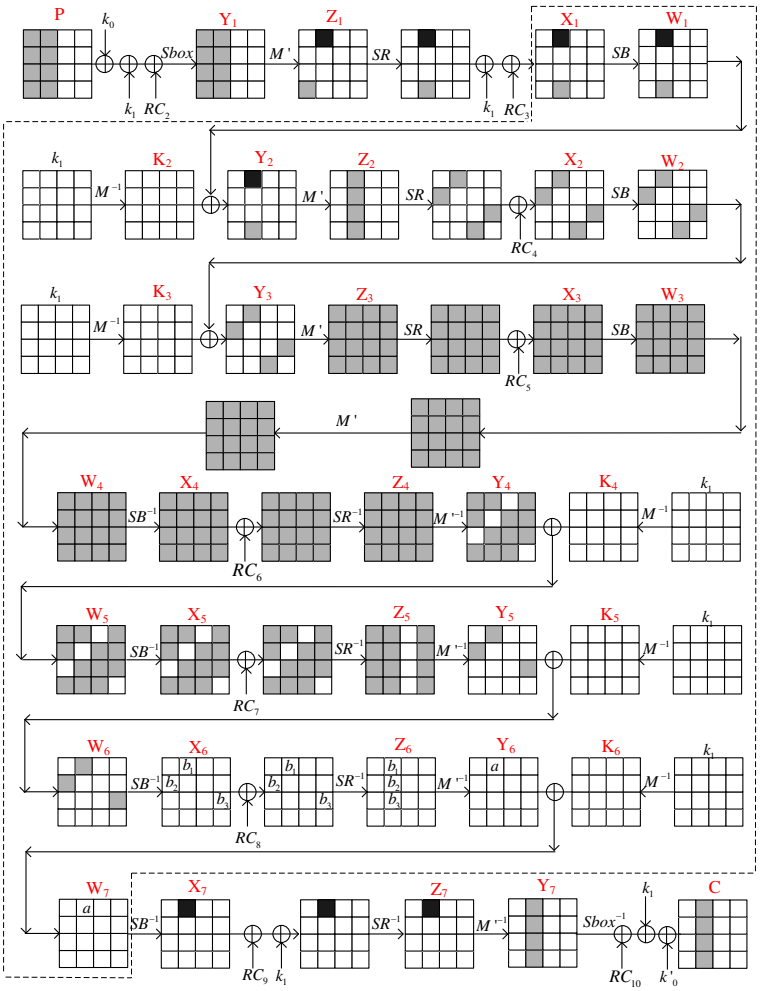

Fig. 5. The Truncated Differential Characteristic of the Attack on 9-Round PRINCE