Improved Differential Cryptanalysis of CAST-128 and CAST-256

Shaomei Wang, Tingting Cui, and Meiqin Wang(✉)

Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
mqwang@sdu.edu.cn

Abstract. CAST-128 and CAST-256 are two symmetric algorithms designed by Adams in 1990s. Both of them adopt the CAST design procedure which makes them process a number of desirable cryptographic. CAST-128 is notably used as the default cipher in some versions of GNU Privacy Guard (GPG) and Pretty Good Privacy (PGP) systems. As an extension of CAST-128, CAST-256 was submitted as a candidate for the Advanced Encryption Standard (AES). Since they are widely used, there are many different attacks on them. Differential cryptanalysis is one of the most powerful tools. In this paper, we achieve improved differential cryptanalysis of both CAST-128 and CAST-256 based on the technique of accessing differential tables. Firstly, we propose a differential attack on 9-round CAST-128 with 2^{73} encryptions and 2^{58} chosen plaintexts. Although we cannot improve the number of attacked rounds, the time complexity is significantly reduced. Then we mount an improved differential attack on 10 quad-rounds of modified CAST-256 which increase one quad-round than previous attack. The time complexity of this attack is 2^{217} encryptions, and the data complexity is 2^{123} chosen plaintexts. As far as we know, these are the best known attacks on CAST-128 and CAST-256 under weak key assumption.

Keywords: Differential analysis · CAST-128 · CAST-256 · Weak key assumption

1 Introduction

In 1997, Adams proposed the CAST design procedure for constructing a family of DES-like Substitution-Permutation Network (SPN) cryptosystems in [1]. The ciphers, known as CAST family, appear to have good resistance to differential [7], linear [12] and related-key cryptanalysis [5]. CAST-128 [2] and CAST-256 [3] both adopt the CAST design procedure. CAST-128 is notably used as the default cipher in some versions of GNU Privacy Guard (GPG) and Pretty Good Privacy (PGP) systems. It has also been approved for Canadian government being used by the Communications Security Establishments. As an extension of CAST-128, CAST-256 was submitted as a candidate for the Advanced Encryption Standard (AES) [13] in June 1998.

K. Chen et al. (Eds.): Inscrypt 2016, LNCS 10143, pp. 18–32, 2017.
DOI: 10.1007/978-3-319-54705-3_2

Since CAST-128 and CAST-256 were proposed, they have been wildly attacked by differential attack, linear attack, boomerang attack [17] and multidimensional zero-correlation linear attack [8]. Adams *et al.* firstly investigated the resistance of CAST-256 to linear and differential cryptanalysis, but they did not give any concrete attack [4]. Then at FSE 1999, Wagner presented a boomerang attack on 16-round CAST-256 [17]. The first concrete linear cryptanalysis of 3-round CAST-128 and 12-round CAST-256 was presented in [14]. Then at SAC 2008, Wang *et al.* improved the results of [14] and mounted linear attacks on 6-round CAST-128 and 24-round CAST-256 [19]. Zhao *et al.* also mounted a linear cryptanalysis of 32 rounds of CAST-256, and they recovered partial key information of round 32 of CAST-256.

In [16], Seki *et al.* presented a differential attack on 9 quad-rounds of modified CAST-256 with a differential characteristic of 8 quad-rounds which remove the rotation keys of f_2 functions. Then in [18], Wang *et al.* presented a 6-round differential characteristic of CAST-128. With this characteristic, they recovered at less 104-bit subkey of 9-round CAST-128. These are the well known differential cryptanalysis of CAST-128 and CAST-256.

It is also worth mentioning that Bogdanov *et al.* mounted a multidimensional zero-correlation attack on 28-round CAST-256 at ASIACRYPT 2012 [8]. And in [9], Cui *et al.* proposed a statistical attack on 29-round CAST-256 by exploiting the statistical integral distinguisher. This is the best attack on CAST-256 in the single-key model without weak key (i.e. key which makes the cipher behave in some undesirable way [10]) assumption.

Our Contribution. In this paper, we use the known technique of the "Looking up Differential Tables" [11] and mount improved differential attacks on CAST-128 and CAST-256. We improve the previous attacks and achieve the best known attacks on both CAST-128 and CAST-256 under the weak key assumption.

- For CAST-128, We append three rounds to the end of the 6-round differential characteristic proposed by Wang *et al.* in [18] and recover all subkeys of the reduced 9-round CAST-128. In [18], Wang *et al.* recovered at less 104-bit subkey of 9-round CAST-128 with 2^{57} chosen plaintexts and $2^{101.8}$ encryptions. In our attack, we reduce the time complexity to 2^{73} encryptions while the data complexity is also 2^{58} chosen plaintexts. This is the best known attack on CAST-128 under the weak key assumption.
- For CAST-256, Seki *et al.* proposed a differential characteristic of 8 quad-rounds in [16], and they recovered 74-bit subkey of a 9 quad-rounds of modified CAST-256 under 2^{123} chosen plaintexts and 2^{95} encryptions. With the same differential characteristic, we recover 222 bits of subkeys including 148-bit subkey in the 10-th quad-round and 74-bit subkey in the 9-th quad-round with 2^{123} chosen plaintexts and 2^{217} encryptions which increases one more quad-round than previous result in [16]. As far as we known, it is also the best known attack on CAST-256 with regard to the attack rounds under weak key assumption.

In order to make a more accurate comparison, we summarise the related attacks on CAST-128 and CAST-256 in Table 1.

Table 1. Summary of attacks on CAST-128 and CAST-256

Target	Rounds	Attack type	Data(KP/CP)[a]	Time	Source	Rate of weak key
CAST-128	3	Linear	2^{37}KP	$2^{72.5}$	[14]	
	6	Linear	$2^{53.96}$KP	$2^{88.51}$	[19]	
	9	Differential	2^{58}CP	$2^{101.8}$	[18]	$2^{-23.8}$
	9	Differential	2^{58}CP	2^{73}	Sect. 3	$2^{-23.8}$
CAST-256	12	Linear	2^{101}KP	2^{101}	[14]	
	16	Boomerang	$2^{49.3}$CP	$2^{49.3}$	[17]	
	24	Linear	$2^{124.10}$KP	$2^{156.20}$	[19]	
	28	Multidimensional ZC[b]	$2^{98.8}$KP	$2^{246.9}$	[8]	
	29	Statistic integral	$2^{96.8}$CP	$2^{219.4}$	[9]	
	32	Linear	$2^{126.8}$KP	2^{250}	[20]	
	36	Differential	2^{123}CP	2^{95}	[16]	2^{-35}
	40	Differential	2^{123}CP	2^{217}	Sect. 4	2^{-35}

[a]KP: Known Plaintext; CP: Chosen Plaintext.
[b]Multi ZC: Multidimensional Zero-Linear.

Overview of This Paper. In Sect. 2, we give a brief description of CAST-128 and CAST-256. Then the improved differential attacks on CAST-128 and CAST-256 are proposed in Sects. 3 and 4 respectively. Finally, we conclude in Sect. 5.

2 Brief Description of CAST-128 and CAST-256

2.1 Brief Description of CAST-128

CAST-128 [2] is a DES-like Substitution-Permutation Network (SPN) cryptosystem. It is Feistel network encryption algorithm with 64-bit block size and key size of 40 to 128 bits (in 8-bit increments). When key size is no more than 80 bits, the algorithm use 12 rounds, and when key size is greater than 80 bits, the algorithm uses the full 16 rounds.

CAST-128 adopts three different round functions f_1, f_2, f_3. Different round functions share the same operations but different order of the these operations. f_1, f_2 and f_3 are described as following and shown in Fig. 1.

$$\begin{aligned} f_1: \quad & I = ((K_{m_i} + D) \lll K_{r_i}) \\ & f = ((S_1[I_a] \oplus S_2[I_b]) - S_3[I_c]) + S_4[I_d] \end{aligned}$$

$$\begin{aligned} f_2: \quad & I = ((K_{m_i} \oplus D) \lll K_{r_i}) \\ & f = ((S_1[I_a] - S_2[I_b]) + S_3[I_c]) \oplus S_4[I_d] \end{aligned}$$

$$\begin{aligned} f_3: \quad & I = ((K_{m_i} - D) \lll K_{r_i}) \\ & f = ((S_1[I_a] + S_2[I_b]) \oplus S_3[I_c]) - S_4[I_d] \end{aligned}$$

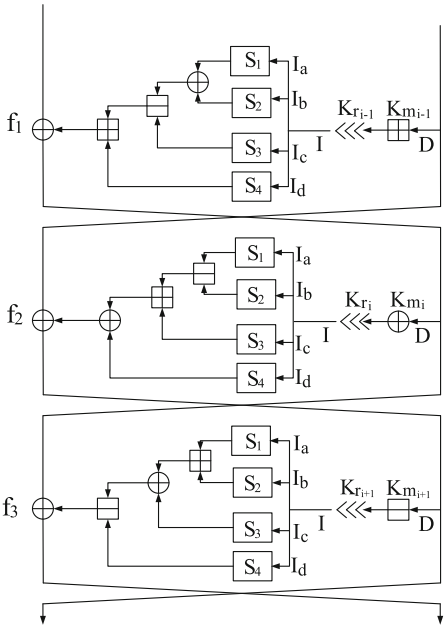

Fig. 1. Encryption procedure of CAST-128

where D is the data input to the round function. K_{r_i} is 5-bit "rotation key" of round i, and K_{m_i} is 32-bit "masking key" of round i. $I = (I_a, I_b, I_c, I_d)$ is the state after rotation key operation. S_1, S_2, S_3 and S_4 are four 8×8 bits S-boxes. $\lll, \oplus, +$ and $-$ denote circular left-shift operation, bitwise XOR, addition modulo 2^{32} and subtraction modulo 2^{32} respectively.

Note that the round number starts from 1 in this paper, so for CAST-128, rounds 1, 4, 7, 10, 13 and 16 use f_1 as round function, round 2, 5, 8, 11 and 14 use f_2 as round function, and rounds 3, 6, 9, 12, and 15 use f_3 as round function. The consecutive 3-round encryption procedure is described in Fig. 1.

Since in our attacks, we do not care for the key schedule. We assume that all subkeys are independent. So we will not describe it in more details. For more details about the key schedule, please refer to [2].

2.2 Brief Description of CAST-256

CAST-256 [3] was published in June 1998 adopting CAST design procedure. It is the extension of CAST-128 and was submitted as a candidate for the Advanced Encryption Standard (AES). Its block size and key size are 128 bits and 256 bits respectively. The whole construction adopts a generalized Feistel network including four 32-bit branches. Between two adjacent branches there are three different functions f_1, f_2 and f_3 which are same as the ones used in CAST-128.

Specially, CAST-256 consists of two types of round function: forward quad-round and reverse quad-round, which are described as following:

$$
\begin{aligned}
forward\ quad-round: \quad & C = C \oplus f_1(D, K^i_{r_0}, K^i_{m_0}) \\
& B = B \oplus f_2(C, K^i_{r_1}, K^i_{m_1}) \\
& A = A \oplus f_3(B, K^i_{r_2}, K^i_{m_2}) \\
& D = D \oplus f_1(A, K^i_{r_3}, K^i_{m_3})
\end{aligned}
$$

$$
\begin{aligned}
reverse\ quad-round: \quad & D = D \oplus f_1(A, K^i_{r_3}, K^i_{m_3}) \\
& A = A \oplus f_3(B, K^i_{r_2}, K^i_{m_2}) \\
& B = B \oplus f_2(C, K^i_{r_1}, K^i_{m_1}) \\
& C = C \oplus f_1(D, K^i_{r_0}, K^i_{m_0})
\end{aligned}
$$

where $K^i_{r_j}, K^i_{m_j} (j = 0, 1, 2, 3)$ are "rotation key" and "masking key" of the i-th quad-round respectively. A, B, C, D together denotes a 128-bit block, and each one of them is a 32-bit word.

The overall encryption procedure is composed of 12 quad-rounds including 6 forward quad-rounds then 6 reverse quad-rounds, which is described in Fig. 2.

Similar to CAST-128, we do not care about the key schedule and assume that all subkeys are independent. So we ignore it here. For more details, please refer to [3].

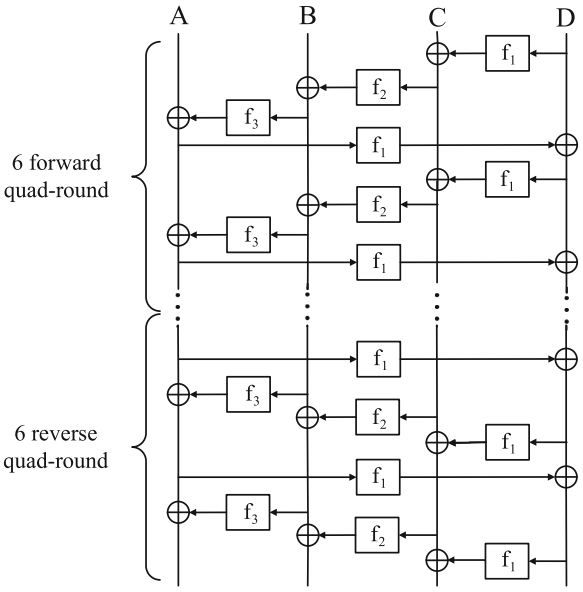

Fig. 2. Encryption procedure of CAST-256

3 Improved Differential Attack on 9-Round CAST-128

In this section, we will propose an improved differential attack on 9-round CAST-128. Firstly in Sect. 3.1, we recall a 6-round differential characteristic proposed by Wang *et al.* in [18]. Based on this characteristic, we present the detailed attack in Sect. 3.2. In this attack, we append three rounds to the end of the differential characteristic and recover all subkeys of 9-round CAST-128, which needs 2^{58} chosen plaintexts and 2^{73} encryptions under about $2^{106.2}$ weak keys.

3.1 6-Round Differential Characteristic of CAST-128

In this subsection, we recall the 6-round differential characteristic of CAST-128 presented by Wang *et al.* in [18]. Its probability is 2^{-53}, and it can be satisfied under $2^{-23.8}$ of the total key space, i.e., $2^{104.2}$ weak keys for CAST-128, which was found by studying the properties of round function f_1 and f_3. The form of such 6-round differential characteristic is $(f7e00000 \ggg K_{r_2}||00000000) \rightarrow (f7e00000 \ggg K_{r_2}||00000000)$, which is listed in the Fig. 3.

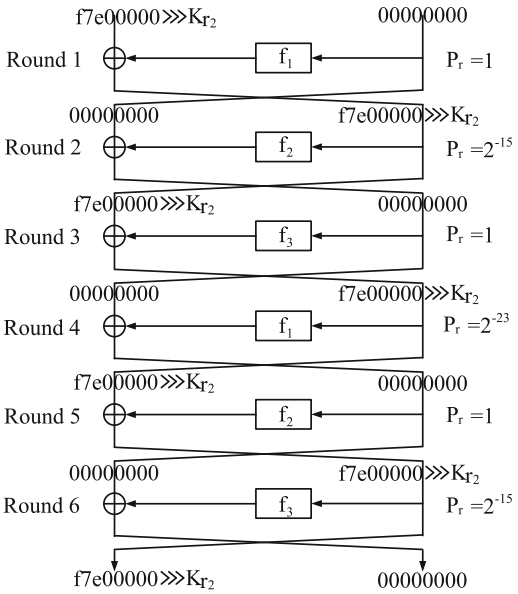

Fig. 3. 6-Round differential characteristic of CAST-128

Note that in the Fig. 3, K_{r_2} is the "rotation key" of the round 2. For each of 2^5 values of K_{r_2}, the differential characteristic is feasible.

3.2 Key Recovery Attack on CAST-128

In order to make the attack faster, we perform a precomputation: we compute the Differential Distribution Tables (called as DDT) for part of f_2 and f_3 functions, which means that we build a table according to all combinations of input

and output differences, then store all possible input-output pairs into the corresponding cells of the table that have the same input and output differences.

For f_3 function, we exclude the modular subtraction operation with K_{m_i} and compute the DDTs for remained part according to the value of K_{r_i}. Since there are 2^5 values of K_{r_i}, we need to compute 32 tables. Without loss of generality, we denote them as $DDT_j^3(j = 0, 1, \ldots, 31)$. However, it is worth noting that the input differences are subtractive differences (modulo 2^{32}) and the output differences are xor differences in these DDTs.

For example, if let $K_{r_i} = j$, and suppose a, b are two 32-bit inputs, then the input difference is $\Delta In = (a - b)$ (modulo 2^{32}), and the output difference is $\Delta Out = f_3'(a) \oplus f_3'(b)$, where f_3' denotes the remaining part of f_3 by excluding the modular subtraction operation with K_{m_i}. Then we need to store $(a, f'(a))$ into the cell satisfying the $\Delta In \rightarrow \Delta Out$ of DDT_j^3.

For f_2 function, we build a DDT for the part of f_2 without mixing subkeys operations (renamed as f_2'), which means we only consider the part from the state before S-boxes layer to the end of f_2 function. Without loss of generality, we denote this DDT as DDT^2, and its input and output differences are both xor differences. Until now, we build 33 DDTs.

As the sizes of input and output of f_3' and f_2' functions are both 32 bits, there are 2^{64} different pairs and $2^{32} \times 2^{32}$ items (called cells as well) at each DDT. For fixed input and output differences, one input-output pair can be found on average.

In the precomputation phase, we need to build 33 DDTs. For each DDT, all possible input pairs should be traversed, so the total time complexity is $(33 \times 2^{32} \times 2^{32} \times 2)/9 \approx 2^{66.9}$ 9-round encryptions. We can only store one input-output pair instead of the input pair and output pair into the DDT because of the fixed input and output difference, the memory complexity is about $33 \times 2^{64} \times 8 \approx 2^{72.0}$ bytes.

Next, we will use these DDTs to implement the key recovery phase step by step. Before giving the detailed attack, we need to define some notations. As illustrated in Fig. 4, (I_{L_i}, I_{R_i}) and (O_{L_i}, O_{R_i}) are the input and output values of round i respectively. D_i is input value of round function, and I_i is input value of S-boxes layer. X_i is the result of different operations on D_i and K_{m_i}. And Y_i is the output value of round function.

The overall key recovery phase is shown in Fig. 4, and the detailed attack is as follows:

- **Step 1.** Loop for all possible values of K_{r_2}. In data collection phase, we choose 2^n structures. Each of structures involves 2^{32} plaintexts with the same value of the right half I_r. To satisfy the special difference $f7e00000 \ggg K_{r_2}||00000000$, each structure can produce 2^{31} plaintext pairs, so in total we can obtain 2^{n+31} pairs. Ask for encryption of the pairs, and we can obtain 2^{n+31} ciphertext pairs as well.
- **Step 2.** Guess 5-bit value of K_{r_9} and 5-bit value of K_{r_8}. Then create a vector counter V of size 2^{64}, and initialize all its elements to zero.

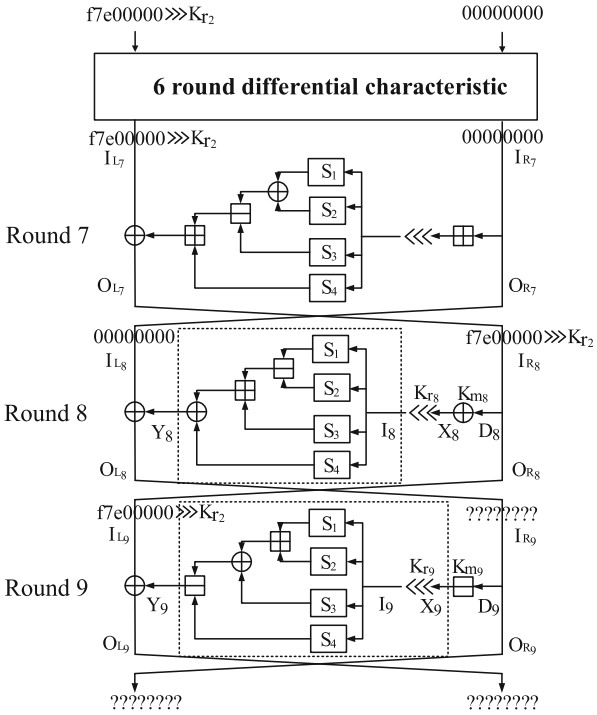

Fig. 4. Key recovery procedure of CAST-128

– **Step 3.** Loop for 2^{n+31} ciphertext pairs, and we can compute the output xor difference $\Delta Y_9 = (C_{L_9} \oplus C'_{L_9}) \oplus (f7e00000 \ggg K_{r_2})$ and the input substractive difference $\Delta X_9 = (C_{R_9} - C'_{R_9}) mod\ 2^{32}$ of f'_3 in round 9 since we already knew the output values of round 9 (i.e. ciphertexts). Then through accessing the corresponding DDT, we can get input-output pairs (X_9, Y_9) and (X'_9, Y'_9) of f'_3 in round 9. Note that $X_9 = (K_{m_9} - C_{R_9}) mod\ 2^{32}$, we can further compute the value of K_{m_9}.
Since the input xor difference ΔI_8 of f'_2 in round 8 equals $(f7e00000 \ggg K_{r_2}) \lll K_{r_8}$ and the output xor difference ΔY_8 equals $C_{R_9} \oplus C'_{R_9}$, we can get the input-output pairs (I_8, Y_8) and (I'_8, Y'_8) through accessing DDT[2]. Further, we can compute $K_{m_8} = (I_8 \ggg K_{r_8}) \oplus I_{L_9}$.
Then the corresponding $K_{m_8}||K_{m_9}$ counter are increased by 1. When all the ciphertext pairs are utilized, the maximum entry in the counter and the corresponding $(K_{r_2}, K_{r_9}, K_{r_8}, K_{m_9}, K_{m_8})$ are stored. In the end, there are 2^{15} candidates as there are 2^{15} values of $K_{r_2}||K_{r_9}||K_{r_8}$, and we consider the subkey with the most happen times as the right key.
– **Step 4.** We have obtained the subkeys of round 8, round 9 and K_{r_2}. In order to recover the remaining subkeys from round 1 to round 7, we truncate the first 5 rounds from the 6-round differential characteristic, and decrypt ciphertexts in the last two rounds. Then we use a very similar way to recover the subkeys

Algorithm 1. Key Recovery Procedure of 9-round CAST-128

for 2^5 *values of* K_{r_2} **do**
  Collect 2^{57} plaintext pairs whose differences equal $f7e00000 \lll K_{r_2}||00000000$.
  Ask for encryption of these pairs and get their ciphertext pairs.
  for 2^{10} *values of* K_{r_9} *and* K_{r_8} **do**
    Allocate a vector counter $V[2^{64}]$, and initialize it to zero.
    for 2^{57} *ciphertext pairs* **do**
      Compute $\Delta X_9 = (C_{R_9} - C'_{R_9}) mod\ 2^{32}$.
      Compute $\Delta Y_9 = (C_{L_9} \oplus C'_{L_9}) \oplus (f7e00000 \lll K_{r_2})$.
      Look up the corresponding DDT, and get (X_9, Y_9) and (X'_9, Y'_9).
      Compute K_{m_9} with $X_9 = (K_{m_9} - C_{R_9}) mod\ 2^{32}$.
      Compute $\Delta I_8 = (f7e00000 \ggg K_{r_2}) \lll K_{r_8}$.
      Compute $\Delta Y_8 = C_{R_9} \oplus C'_{R_9}$ where $C_{R_8} = C_{L_9} \oplus Y_9$ and $C'_{R_8} = C'_{L_9} \oplus Y'_9$.
      Look up corresponding DDT, and get (I_8, Y_8) and (I'_8, Y'_8).
      Compute K_{m_8} with $K_{m_8} = (I_8 \ggg K_{r_8}) \oplus C_{R_8}$.
      $V[K_{m_9}||K_{m_8}] + +$.
    Store the maximum value of counter V and corresponding $(K_{r_2}, K_{m_8}, K_{r_8}, K_{m_9}, K_{r_9})$.
$(K_{r_2}, K_{m_8}, K_{r_8}, K_{m_9}, K_{r_9})$ with the maximum happen times is considered as the right key.

of round 6 and round 7. The complexity is smaller than what we need to recover $K_{m_8}, K_{r_8}, K_{m_9}, K_{r_9}$ and K_{r_2}, which can be ignored. Subkeys of the first 5 rounds also can be recovered with similar way.

In order to illustrate the key recovery phase more succinctly, the whole key recovery phase is summarized in Algorithm 1.

3.3 Complexity Evaluation

The ratio of signal to noise is proposed by Selçuk *et al.* in [15], which is calculated as follows:

$$S/N = \frac{p \times 2^k}{\alpha \times \beta} \tag{1}$$

where α is the average count of keys per analyzed pair, β is the ratio of analyzed pairs to all the pairs, k is the number of key bits we are searching and p is the probability of the differential characteristic.

In our attack, $p = 2^{-53}$. Since we can not filter the ciphertext pairs by the known condition, $\beta = 1$. The number of the all subkeys in round 8, round 9 and K_{r_2} is 79, so $k = 79$. For every analyzed pair, if we fix the value of $(K_{r_2}, K_{r_9}, K_{r_8})$, one possible value of $K_{m_8}||K_{m_9}$ can be deduced on average.

There are 2^{15} values of K_{r_2}, K_{r_9} and K_{r_8} in total, so we can get that $\alpha = 2^{15}$. Consequently

$$S/N = \frac{p \times 2^k}{\alpha \times \beta} = \frac{2^{-53} \times 2^{79}}{2^{15} \times 1} = 2^{11}. \tag{2}$$

The success probability in [20] is calculated as follows:

$$Ps = \int_{-\frac{\sqrt{\mu S_N} - \Phi^{-1}(1-2^{-k})}{\sqrt{S_N+1}}}^{\infty} \Phi(x)dx \tag{3}$$

where μ is the number of the right pairs (i.e. the pairs that satisfy the differential characteristic). In our attack, $\mu = 2^{n-22}$.

In order to attack the 9-round CAST-128 with the success rate 0.999, we choose $n = 26$. Consequently the data complexity of the attack is $2^{26} \times 2^{32} = 2^{58}$ chosen plaintexts.

The time complexity of Step 1 is $2^5 \times 2^{57} \times 2 = 2^{63}$ 9-round encryptions. Step 2 and Step 3 need about $2^5 \times 2^{10} \times 2^{57} \times 2 = 2^{73}$ memory access. The time complexity of Step 4 is not the domain term, so it can be ignored. We assume one access to the DDT (memory access) equals one 9-round encryption. Consequently the time complexity of the key recovery phase is about $2^{63} + 2^{73} \approx 2^{73}$ 9-round encryptions. And we also need $2^{64.2}$ bytes of memory in the key recovery phase. Besides, in the precomputation stage, we need $2^{66.9}$ encryptions an $2^{72.0}$ bytes of memory, so the complexity of the overall attack is about $2^{74} + 2^{66.9} \approx 2^{74}$ encryptions, and the memory complexity is about $2^{64.2} + 2^{72.0} \approx 2^{72.0}$ bytes of memory.

4 Improved Differential Attack on 10 Quad-Rounds of Modified CAST-256

In this section, we will mount an improved differential attack on 10 quad-rounds of modified CAST-256. In Sect. 4.1, we recall a differential characteristic of 8 quad-rounds proposed by Seki *et al.* [16] firstly. Then in Sect. 4.2, we will give the detailed attack. In this attack, We append two quad-rounds to the end of the differential characteristic and recover 148-bit subkey including all subkeys of round 10 and 74-bit subkey of $K^9_{r_0}$, $K^9_{m_0}$, $K^9_{r_1}$, $K^9_{m_1}$, which needs 2^{123} chosen plaintexts and 2^{217} encryptions under about 2^{221} weak keys.

4.1 A Differential Characteristic of 8 Quad-Rounds for CAST-256

In [16], Seki *et al.* proposed a differential characteristic of 8 quad-rounds for the modified CAST-256, which removed the rotation keys of all f_2 functions. The form of the characteristic is $(00000000||00000000||0000e0f7||00000000) \rightarrow (00000000||00000000||0000e0f7||00000000)$ (see Fig. 5). Its probability is 2^{-120}, and it is satisfied under 2^{-35} of the total key space, i.e. 2^{221} weak keys. With this characteristic, Seki *et al.* gave a differential attack on 9 quad-rounds of modified CAST-256. In Sect. 4.2, we will improve their attack and propose a differential attack on 10 quad-rounds of modified CAST-256.

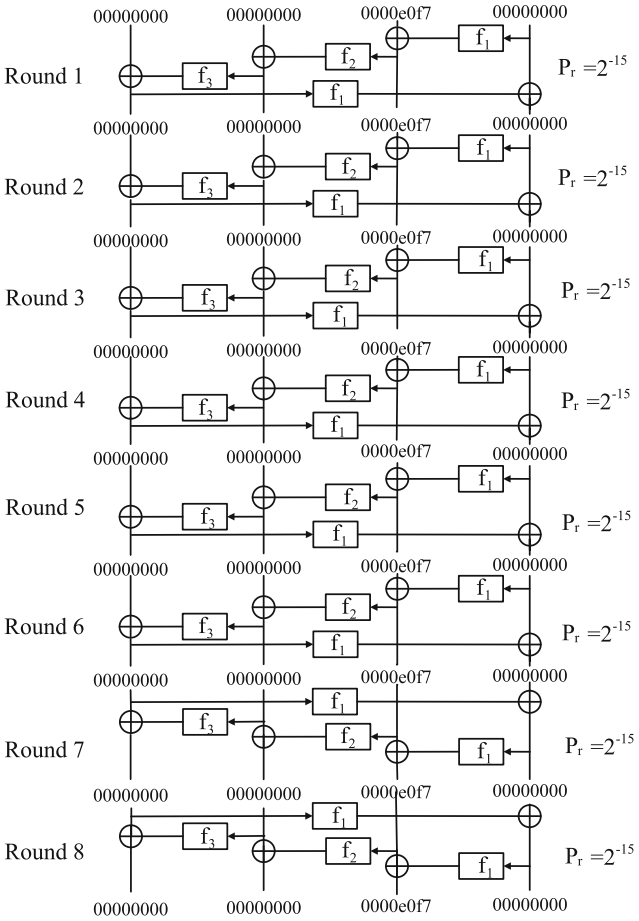

Fig. 5. A differential characteristic of 8 Quad-Rounds for CAST-256

4.2 Key Recovery Attack on CAST-256

In the attack on CAST-256, we also need the "differential distribution tables" of f_2' and f_3' functions. The procedure to produce these DDTs is the same as before. The precomputation stage requires about $(33 \times 2^{32} \times 2^{32} \times 2)/40 \approx 2^{64.6}$ encryptions and $33 \times 2^{64} \times 8 \approx 2^{72.0}$ bytes.

In this attack, we append two quad-rounds to the end of the differential characteristic of 8 quad-rounds. Before giving concrete attack, we define some notations. As described in Fig. 6, $A_i||B_i||C_i||D_i$ is the input of round i, which also is the output of round $(i-1)$, so in our new attack the ciphertext is $A_{11}||B_{11}||C_{11}||D_{11}$. There are four round functions in one quad-round, then we use $D_i^j, X_i^j, I_i^j, Y_i^j (j = 0, 1, 2, 3)$ to denote corresponding values in the j-th round function of the i-th quad-round, which are similar with D_i, X_i, I_i, Y_i in the attack on CAST-128.

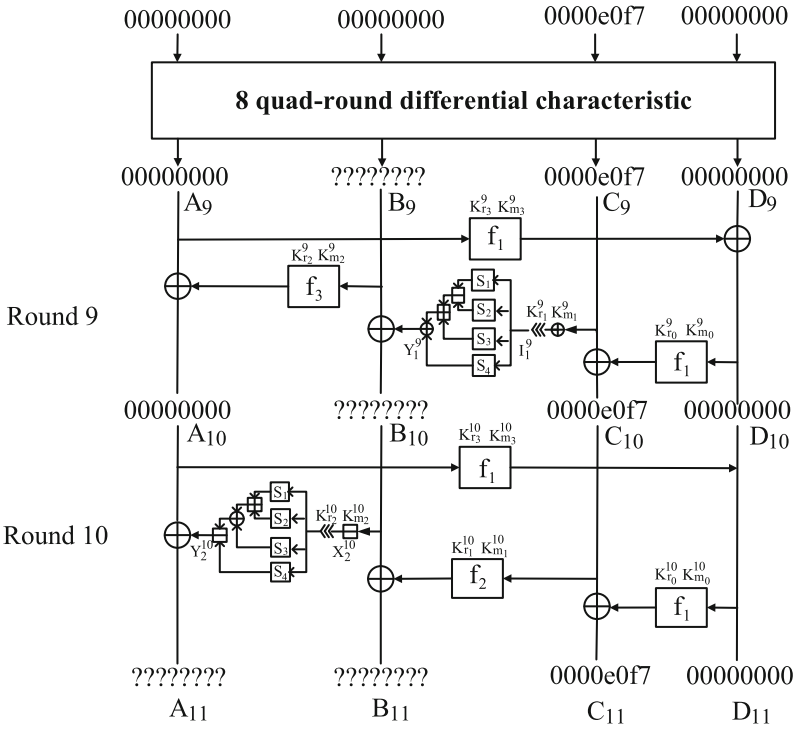

Fig. 6. Key recovery procedure of CAST-256

The overall key recovery phase is shown in Fig. 6, and the detailed attack is as follows:

- **Step 1.** We need to choose 2^n structures. Each of structures involves 2^{32} plaintexts with the same value in the A, B and D parts which can produce 2^{31} plaintext pairs satisfying the special input difference 00000000||00000000||0000e0f7||00000000. So there are 2^{n+31} pairs in total. Ask for encryption of the plaintext pairs, then choose pairs whose ciphertext differences satisfy $C_{11} = 0000e0f7$ and $D_{11} = 00000000$. About $2^{n+31} \times 2^{-64} = 2^{n-33}$ pairs are reserved.
- **Step 2.** Guess 74-bit $K^{10}_{r_0}$, $K^{10}_{m_0}$, $K^{10}_{r_1}$, $K^{10}_{m_1}$. By decrypting the last round, we can get the second and the third part of the input values of round 10. $C_{10} = C_{11} \oplus f_1(D_{11}, K^{10}_{r_0}, K^{10}_{m_0})$, $C'_{10} = C'_{11} \oplus f_1(D'_{11}, K^{10}_{r_0}, K^{10}_{m_0})$, $B_{10} = B_{11} \oplus f_2(C_{10}, K^{10}_{r_1}, K^{10}_{m_1})$, $B'_{10} = B'_{11} \oplus f_2(C'_{10}, K^{10}_{r_1}, K^{10}_{m_1})$. Store $A_{11}||B_{10}||C_{10}||D_{11}$ and $A'_{11}||B'_{10}||C'_{10}||D'_{11}$ instead of ciphertext pairs.
- **Step 3.** Guess the 5-bit value of $K^{10}_{r_2}$, 74-bit value of $K^{10}_{r_3}$, $K^{10}_{m_3}$, $K^{9}_{r_0}$, $K^{9}_{m_0}$ and 5-bit value of $K^{9}_{r_1}$. Then create a vector counter of size 2^{64}, and initialize all elements of it to zero.

– **Step 4.** Loop for the remaining pairs. Compute input and output differences of f_3' in round 10 with $\Delta X_2^{10} = (B_{10} - B'_{10}) mod\ 2^{32}$ and $\Delta Y_2^{10} = A_{11} \oplus A'_{11}$. Access the corresponding DDT of f_3', then get the input-output pairs (X_2^{10}, Y_2^{10}) and $(X_2^{10'}, Y_2^{10'})$ of f_3'. And compute $K_{m_2}^{10}$ with $X_2^{10} = (K_{m_2}^{10} - B_{10}) mod\ 2^{32}$.
Compute the forth part of the input of round 10 and the third part of the input of round 9 with $D_{10} = D_{11} \oplus f_1(A_{10}, K_{r3}^{10}, K_{m3}^{10})$ and $C_9 = C_{10} \oplus f_1(D_{10}, K_{r_0}^9, K_{m_0}^9)$ where A_{10} and D_{10} can be computed by the obtained values. Then compute the input and output difference of f_2' in round 9 with $\Delta I_1^9 = 0000e0f7 \lll K_{r_1}^9$, $\Delta Y_1^9 = B_{10} \oplus B'_{10}$, and get the input-output pairs (I_1^9, Y_1^9) and $(I_1^{9'}, Y_1^{9'})$ through looking up DDT^2 of f_2'. Then compute $K_{m_1}^9$ with $K_{m_1}^9 = (I_1^9 \ggg K_{r_1}^9) \oplus C_9$.
The corresponding $K_{m_2}^{10} || K_{m_1}^9$ counter is increased by 1. When all the remaining pairs are utilized, store the maximum value of the counter and corresponding $(K_{r_2}^{10}, K_{r_3}^{10}, K_{m_3}^{10}, K_{r_0}^9, K_{m_0}^9, K_{r_1}^9, K_{r_0}^{10}, K_{m_0}^{10}, K_{r_1}^{10}, K_{m_1}^{10})$. Compare the selected value with the previous stored one, and preserve the larger one and corresponding subkeys. However, no comparison is needed in the first storage.

In order to illustrate the key recovery phase more succinctly, the key recovery phase can be performed like Algorithm 2.

4.3 Complexity Evaluation

In order to calculate the ratio of signal to noise, $p = 2^{-120}$ and $\beta = 2^{-64}$ in our attack. k is number of the total subkeys in round 10 and partial subkeys in round 9 which equals 222. For fixed value of $K_{r_0}^{10}$, $K_{m_0}^{10}$, $K_{r_1}^{10}$, $K_{m_1}^{10}$, $K_{r_2}^{10}$, $K_{r_3}^{10}$, $K_{m_3}^{10}, K_{r_0}^9, K_{m_0}^9$, $K_{r_1}^9$, one possible value of $K_{m_2}^{10} || K_{m_1}^9$ can be deduced on average. There are 2^{158} values of $K_{r_0}^{10}$, $K_{m_0}^{10}$, $K_{r_1}^{10}$, $K_{m_1}^{10}$, $K_{r_2}^{10}$, $K_{m_2}^{10}$, $K_{r_3}^{10}$, $K_{m_3}^{10}, K_{r_0}^9, K_{r_1}^9$ in total, so $\alpha = 2^{158}$. Consequently

$$S/N = \frac{p \times 2^k}{\alpha \times \beta} = \frac{2^{-120} \times 2^{222}}{2^{158} \times 2^{-64}} = 2^8. \tag{4}$$

In the formula of success probability, we need to know the number of the right pairs, which $\mu = 2^{n-89}$ in our attack. In order to let the success rate be close to 1, we choose $n = 91$. Consequently the data complexity of the attack is $2^{91} \times 2^{31} \times 2 = 2^{123}$ chosen plaintexts.

The time complexity of Step 1 is 2^{123} 10 quad-rounds of encryptions. The time complexity of Step 2 is $2^{58} \times 2^{74} \times 2 = 2^{133}$ half quad-round of encryptions. Then Step 3 and Step 4 need about $2^{58} \times 2^{74} \times 2^{84} \times 2 = 2^{217}$ memory accesses. Consequently the overall complexity of the key recovery phase is about $2^{123} + 2^{133}/20 + 2^{217} \approx 2^{217}$ encryptions. And we need $2^{58} \times 16 + 2^{64} = 2^{64.3}$ bytes of memory in this phase. What's more, in the precomputation stage, we need $2^{64.6}$ encryptions and $2^{72.0}$ bytes of memory. So the overall time complexity of this attack is about $2^{217} + 2^{64.6} \approx 2^{217}$ encryptions, and the memory complexity is about $2^{64.3} + 2^{72.0} \approx 2^{72.0}$ bytes of memory.

Algorithm 2. Key Recovery Procedure of 10 Quad-Rounds CAST-256

Collect 2^{122} plaintext pairs whose difference equal $00000000||00000000||0000e0f7||00000000$.
Ask for ciphertext pairs, and filter them with ciphertext difference. //About 2^{58} pairs remained.
for 2^{74} *values of* $K^{10}_{r_0}$, $K^{10}_{m_0}$, $K^{10}_{r_1}$, $K^{10}_{m_1}$ **do**
    Decrypt ciphertext pairs in the last round, and get C_{10}, C'_{10} and B_{10}, B'_{10}. Store $(A_{11}||B_{10}||C_{10}||D_{11})$ and $(A'_{11}||B'_{10}||C'_{10}||D'_{11})$ instead of ciphertext pairs.
    for 2^5 *values of* $K^{10}_{r_2}$, 2^{74} *values of* $K^{10}_{r_3}$, $K^{10}_{m_3}$, $K^{9}_{r_0}$, $K^{9}_{m_0}$ *and* 2^5 *values of* $K^{9}_{r_1}$ **do**
        Allocate a vector counter $V[2^{64}]$, and initialize it to zero.
        for 2^{58} *remained pairs* **do**
            Compute $\Delta X^{10}_2 = (B_{10} - B'_{10}) mod\ 2^{32}$.
            Compute $\Delta Y^{10}_2 = A_{11} \oplus A'_{11}$.
            Look up the corresponding DDT, and get (X^{10}_2, Y^{10}_2) and $(X^{10'}_2, Y^{10'}_2)$.
            Compute $K^{10}_{m_2}$ with $X^{10}_2 = (K^{10}_{m_2} - B_{10}) mod\ 2^{32}$.
            Compute $D_{10} = D_{11} \oplus f_1(A_{10}, K^{10}_{r_3}, K^{10}_{m_3})$.
            Compute $C_9 = C_{10} \oplus f_1(D_{10}, K^{9}_{r_0}, K^{9}_{m_0})$.
            Compute $\Delta I^9_1 = 0000e0f7 \lll K^9_{r_1}$.
            Compute $\Delta Y^9_1 = B_{10} \oplus B'_{10}$.
            Look up the corresponding DDT, and get (I^9_1, Y^9_1) and $(I^{9'}_1, Y^{9'}_1)$.
            Compute $K^9_{m_1}$ where $K^9_{m_1} = (I^9_1 \ggg K^9_{r_1}) \oplus C_9$.
            $V[K^{10}_{m_2}||K^9_{m_1}] + +$.
        Find the maximum value of the counter and the corresponding $(K^{10}_{r_2}, K^{10}_{r_3}, K^{10}_{m_3}, K^9_{r_0}, K^9_{m_0}, K^9_{r_1}, K^{10}_{r_0}, K^{10}_{m_0}, K^{10}_{r_1}, K^{10}_{m_1})$.
        Compare the maximum value of the counter with the previous stored one, and preserve the larger one. //No comparison is needed in the first storage.
The subkey corresponding to the final maximum entry is considered as the right key.

5 Conclusion

In this paper, we have proposed an improved differential attack on reduced 9-round CAST-128 cipher which needs 2^{58} chosen plaintexts and 2^{73} 9-round encryptions based on the known 6-round differential characteristic. Besides, we also proposed a differential attack on 10 quad-rounds of modified CAST-256. The data and time complexity of the attack are 2^{123} chosen plaintexts and 2^{217} encryptions respectively. Our attacks on CAST-128 and CAST-256 are the best known ones under the weak key assumption.

References

1. Adams, C.: Constructing symmetric ciphers using the CAST design procedure. Des. Codes Crypt. **9**, 283–316 (1997)
2. Adams, C.: The CAST-128 Encryption Algorithm. RFC 2144 (1997)
3. Adams, C., Cilchist, J.: The CAST-256 Encryption Algorithm. RFC 2612 (1997)
4. Adams, C., Heys, H.: An analysis of the CAST-256 cipher. In: IEEE Canadian Conference on Electrical and Computer Engineering, pp. 9–12. IEEE Press, Canada (1999)
5. Biham, E.: New types of cryptanalytic attacks using related keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994). doi:10.1007/3-540-48285-7_34
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991). doi:10.1007/3-540-38424-3_1
7. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer New York, New York (1993)
8. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_16
9. Cui, T., Chen, H., Wen, L., Wang, M.: Statistic integral attack on CAST-256 and IDEA. In: ArcticCrypt 2016, Longyearbyen (2016)
10. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the key scheduling algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001). doi:10.1007/3-540-45537-X_1
11. 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. LNCS, vol. 6498, pp. 282–291. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17401-8_20
12. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993). doi:10.1007/3-540-47555-9_7
13. National Institute of Standards and Technology: Advanced Encryption Standard(AES). crsc.nist.gov/encryption/aes
14. Nakahara, J., Rasmussen, M.: Linear analysis of reduced-round CAST-128 and CAST-256. In: SBSEG2007, pp. 45–55. Brazil (2007)
15. Selçuk, A., Bicak, A.: On probability of success in linear and differential cryptanalysis. J. Crypt. **21**, 131–147 (2008)
16. Seki, H., Kaneko, T.: Differential cryptanalysis of CAST-256 reduced to nine quad-rounds. IEICE Trans. Fundam. **E84–A**, 913–918 (2001)
17. Wagner, D.: The boomerang attack. In: Knudsen, L. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999). doi:10.1007/3-540-48519-8_12
18. Wang, M., Wang, X., Chow, K.: New differential cryptanalysis results for reduced-round CAST-128. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E93–A**, 2744–2754 (2010)
19. Wang, M., Wang, X., Hu, C.: New linear cryptanalytic results of reduced-round of CAST-128 and CAST-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 429–441. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04159-4_28
20. Zhao, J., Wang, M., Wen, L.: Improved linear cryptanalysis of CAST-256. J. Comput. Sci. Technol. **537**, 2–21 (2001)

http://www.springer.com/978-3-319-54704-6

Information Security and Cryptology
12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers
Chen, K.; Lin, D.; Yung, M. (Eds.)
2017, XIII, 544 p. 78 illus., Softcover
ISBN: 978-3-319-54704-6