Truncated and Multiple Differential Cryptanalysis of Reduced Round Midori128

Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef(✉)

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Quebéc, Canada
youssef@ciise.concordia.ca

Abstract. Midori is a family of SPN-based lightweight block ciphers designed to optimize the hardware energy consumption per bit during the encryption and decryption operations. At ASIACRYPT 2015, two variants of the cipher, namely Midori128 and Midori64, which support a 128-bit secret key and a 64/128-bit block, respectively, were proposed. Recently, a meet-in-the-middle attack and an invariant subspace attack were presented against Midori64 but both attacks cannot be applied to Midori128. In this paper, we present truncated and multiple differential cryptanalysis of round reduced Midori128. Our analysis utilizes the special structure of the S-boxes and binary linear transformation layer in order to minimize the number of active S-boxes. In particular, we consider differentials that contain only single bit differences in the input and output of the active S-boxes. To keep this single bit per S-box patterns after the $MixColumn$ operation, we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position. Using these restrictions, we were able to find 10-round differential which holds with probability 2^{-118}. By adding two rounds above and one round below this differential, we obtain a 13 round truncated differential and use it to perform a key recovery attack on the 13-round reduced Midori128. The time and data complexities of the 13-round attack are 2^{119} encryptions and 2^{119} chosen plaintext, respectively. We also present a multiple differential attack on the 13-round Midori128, with time and data complexities of $2^{125.7}$ encryptions and $2^{115.7}$ chosen plaintext, respectively.

Keywords: Truncated differential cryptanalysis · Midori128 · Multiple differential cryptanalysis

1 Introduction

Over the past few years, many lightweight block ciphers such as HIGHT [6], mCrypton [10], DESL/DESXL [9], PRESENT [2], KATAN/KTANTAN [3], Piccolo [12] and PRINTcipher [7] were proposed. On the other hand, there has been little work that focuses on determining the design choices that lead to the most energy efficient architecture. While power and energy are clearly correlated, optimizing the power consumption for block ciphers does not necessarily lead to the

M. Bishop and A.C.A. Nascimento (Eds.): ISC 2016, LNCS 9866, pp. 3–17, 2016.
DOI: 10.1007/978-3-319-45871-7_1

most energy efficient designs since a low power optimized cipher may have high latency, i.e., it takes longer to perform the encryption and decryption operations and hence the required energy increases. In other words, there is no guarantee that low power block cipher designs would lead to low energy designs and vice versa.

By identifying some design choices that are energy efficient and by choosing components specifically tailored to meet the requirements of low energy design, at ASIACRYPT 2015, Banik *et al.* [1] proposed an SPN-based lightweight block cipher, Midori. In particular, Midori is designed to optimize the hardware energy consumption per bit during the encryption and decryption operations. Two variants of the cipher, namely, Midori128 and Midori64 which support a 128-bit secret key and a 64/128-bit block, respectively, were proposed. The linear and non-linear operations of both versions were selected to optimize this objective.

The state in Midori is represented as 4×4 matrix, where the size of each cell depends on the version of cipher, e.g., the cell size in Midori128 is 8 bits. Midori uses 4×4 almost MDS binary matrix because, compared to other MDS matrices, this almost MDS matrix is more efficient in terms of area and signal-delay. To compensate for the low branch number of the almost MDS matrix (4 as compared to 5 in the case of MDS), the designers utilized an optimal cell-permutation layer in order to improve the diffusion speed and increase the number of active S-boxes.

Recently, Midori64 has been analyzed by two different techniques. The first is a meet-in-the-middle with differential enumeration and key dependent sieving [11] which attacks 11 rounds (resp. 12 rounds) with time, memory, and data complexities of 2^{122} (resp. $2^{125.5}$) encryptions, $2^{89.2}$ (resp. 2^{106}) 64-bit blocks, and 2^{53} (resp. $2^{55.5}$) chosen plaintext. The second is an invariant subspace attack [5] against the full cipher. The latter attack proves that the security margin of Midori64 is 96 bits instead of 128 bits. Both of these attacks are not applicable to Midori128.

In this paper, we present truncated differential cryptanalysis of round reduced Midori128. Our attack utilizes the following two observations. First, Midori128 uses four different 8-bit S-boxes, namely SSb_0, SSb_1, SSb_2 and SSb_3, where each one is composed of two 4-bit S-boxes Sb_1 in addition to input and output bit permutation. Consequently, in order to minimize the number of active S-boxes, we consider only single bit differences (i.e., $1, 2, 4, 8, 16, 32, 64, 128$) in the input and output of the 8-bit S-boxes. Second, given the binary nature of the almost MDS transformation, and the fact that the Hamming weight of each row is 3, it follows that the active bytes in a column after the *MixColumn* operation have a single bit difference if and only if the active bytes in the input column are all equal and each one has a single active bit. Hence, to maintain the pattern of single bit differences after the *MixColumn* operation, we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position.

Based on these observations, we are able to find a 10-round differential that holds with probability 2^{-118}. Then, we added two rounds above and one round below this 10-round differential to obtain a 13-round truncated differential [8] that holds with probability 2^{-230}. Using this truncated differential, we can recover the master key of the 13-round reduced cipher with time and data complexities of 2^{119} encryptions, and 2^{119} chosen plaintext, respectively. We also present a multiple differential attack [4] on the 13-round reduced cipher with time and data complexities of $2^{125.7}$ encryptions and $2^{115.7}$ chosen plaintext, respectively.

The rest of the paper is organized as follows. In Sect. 2, we provide the notations used throughout the paper and a brief description of Midori128. In Sect. 3, we describe and analyze the algorithm we use to efficiently search for long differentials with small number of active S-boxes. Details of our truncated differential attack on Midori128 reduced to 13 rounds are presented in Sect. 4. Section 5 presents our multiple differential attack. Finally, the paper is concluded in Sect. 6.

2 Specifications of Midori128

2.1 Notations

The following notations are used throughout the rest of the paper:

- a^t: Transposition of the vector or the matrix a.
- K: The master key.
- RK_i: The 128-bit round key used in round i.
- WK: The 128-bit whitening key.
- x_i: The 128-bit input to the *SubCell* operation at round i.
- y_i: The 128-bit input to the *ShuffleCell* operation at round i.
- z_i: The 128-bit input to the *MixColumn* operation at round i.
- w_i: The 128-bit input to the *KeyAdd* operation at round i.
- $x_i[j]$: The j^{th} byte of x_i, where $0 \leq j < 16$.
- $x_i[j \cdots l]$: The bytes from j to l of x_i, where $j < l$.
- $x_i[j, l]$: The bytes j and l of x_i.
- $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.

2.2 Specifications

Midori128 can be considered as a variant of Substitution Permutation Networks (SPNs). The state in Midori128 is represented as a 4×4 array of bytes as follow:

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

Midori128 iterates over 20 rounds. Each round, except the last one, has 3 layers: S-layer (*SubCell*) which maps $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$, P-layer (*ShuffleCell* and *MixColumn*) which maps $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$ and a key-addition layer (*KeyAdd*) which maps $\{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$. The last round contains only the S-layer. Moreover, before the first and after the last rounds, prewhitening and postwhitening are performed using *WK*. In what follows, we show how these operations update the 128-bit state S:

- *SubCell*: A nonlinear layer applies 4 8-bit S-boxes, namely SSb_0, SSb_1, SSb_2, and SSb_3, on each byte of the state S in parallel, where $s_i \leftarrow SSb_{(i \bmod 4)}\ [s_i]$ and $0 \leq i \leq 15$. As shown in Fig. 1, each 8-bit S-box SSb_i is composed of input and output bit permutations and 2 4-bit S-box Sb_1, where Sb_1 is 4-bit S-box (see Table 1).
- *ShuffleCell*: The bytes of the state S is permuted as follow: $(s_0, s_1, \cdots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$
- *MixColumn*: Each column in the internal state is multiplied by a binary matrix M, where

$$M = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix}$$

  Hence, the internal state is updated as follows:

$$(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t \leftarrow M(s_i, s_{i+1}, s_{i+2}, s_{i+3})^t, i = 0, 4, 8, 12.$$

- *KeyAdd*: Where 128-bit round key RK_i is XORed with the state S.

Table 1. 4-bit bijective S-box Sb_1 in hexadecimal form [1]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$Sb_1[x]$	1	0	5	3	e	2	f	7	d	a	9	b	c	8	4	6

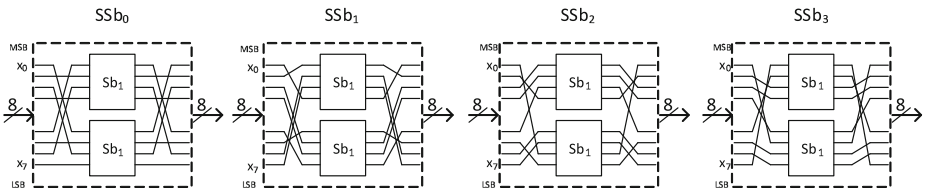

Fig. 1. SSb_0, SSb_1, SSb_2, and SSb_3 [1]

The data encryption procedure of Midori128 is illustrated in Algorithm 1 where $R = 20$ denotes the number of rounds.

Algorithm 1. Data Encryption Algorithm 1

Data: $X, WK, RK_0, ..., RK_{R-2}$
Result: Y
$S \leftarrow KeyAdd(X, WK)$;
for $i \leftarrow 0$ *to* $R-2$ **do**
  $S \leftarrow SubCell(S)$;
  $S \leftarrow ShuffleCell(S)$;
  $S \leftarrow MixColumn(S)$;
  $S \leftarrow KeyAdd(S, RK_i)$;
$S \leftarrow SubCell(S)$;
$Y \leftarrow KeyAdd(S,\ WK)$;

The prewhitening and postwhitening key WK in Midori128 are equal to the master key K, the rounds keys $RK_i = K \oplus \beta_i$, $0 \leq i \leq 18$, where β_i is a constant, X is the plaintext, and Y is the ciphertext. Throughout our analysis, we measure the time complexity of our attack in terms of the equivalent number of reduced-round Midori128 encryptions. For further details about the design rational of the cipher, the reader is referred to [1].

3 A 10-round Differential of Midori128

As mentioned above, in order to minimize the number of active S-boxes, we consider differentials that have only single bit differences in the input and output of the active 8-bit S-boxes. In this section, we describe and analyze the algorithm we use to efficiently find such differentials whose probabilities are greater than 2^{-128}. In each round, we have 4 operations. The operations that can disturb the single bit difference propagation patterns are the *SubCell* and *MixColumn* operations. From the structure of the S-boxes, it follows that for any active S-box and for a given 1-bit input difference, there are at most 4 possible output differences of 1-bit difference because only 1 4-bit S-box will be active. Furthermore, from the properties of the binary almost MDS matrix, preserving single bit differences propagation patterns requires that we restrict the bit differences of the output of the active S-boxes, which lie in the same column after the shuffle operation, to be in the same position. As a result, the active bytes in each column after the *MixColumn* operation have the same value. Therefore, at each round we have at most $(8 \times 15)^4 \approx 2^{28}$ possible input differences. The term 8 in the previous formula denotes the number of possible values of the difference in each column $(1, 2, 4, \cdots, 128)$. The term 15 denotes the total number of combinations for active bytes within each column and the exponent 4 denotes the total number of columns. As noted above, for a given input difference ΔS_i at the beginning of each round, after the S-box layer the values of the active bytes, which lie in

the same column after the shuffle operation, should be equal. Therefore, for each input difference i, we have a set Ω_i which contains at most 4^4 possible output differences (in each column we have at most four 1-bit differences, and we can have at most four active columns.)

Algorithm 2 describes the procedure used to find the maximum number of rounds r, such that a differential with the above S-box propagation patterns exists and its differential probability is greater than 2^{-128}. The algorithm run time is upper bounded by $2^{28} \times 4^4 \times r$. It utilizes four tables where each table has 2^{28} entries corresponding to the possible input differences at each round. In what follows we describe the use of each table:

1. Each entry i in the table *InputDiffProb* indicates the probability to reach the difference i at the beginning of the considered round.
2. Each entry i in the table *OutputDiffProb* indicates the probability to reach the difference i at the end of the considered round.
3. Each entry i in the table *InputParent* indicates the input difference used at the beginning of round 0 to reach difference i at the beginning of the considered round. The value -1 is used to indicate that the difference i cannot be reached from any input difference.
4. Each entry i in the table *OutputParent* indicates the input difference used at the beginning of round 0 to reach difference i at the end of the considered round. The value -1 indicates that the difference i cannot be reached from any input difference.

As explained in Algorithm 2, we iterate over the input differences that have differential probability > 0 round by round, i.e., at each round we propagate all the input differences and begin with the obtained differences as input to the next round. In our implementation, the 28-bit index i encodes the state difference ΔS as follows: we use 7 bits for each one of the four columns where these 7 bits are divided into two parts. The first 3 bits represent the value of the difference of the active bytes in the column and the remaining 4 bits represent the different 15 combinations of the active bytes within the column. After applying the 3 operations: *SubCell*, *ShuffleCell*, and *MixColumn*, we have 3 cases: (i) this difference did not appear before ($Outparent[j] = -1$). In this case we set its probability and parent, (ii) this difference appeared previously and its previous parent is the same as the new parent. Therefore, we add the probabilities, and (iii) this difference appears previously but with a different parent. In this case, we choose the parent with the higher probability. At the end of each round, we copy the *OutputDiffProb* and *OutParent* into *InputDiffProb* and *InputParent*, respectively, and initialize *OutputDiffProb* and *OutParent* to begin another round.

Using a PC with Intel(R) Xeon(R) CPU E3-1280 V2 @ 3.6 GHz and 32 GB RAM, a non-optimized implementation of Algorithm 2 terminates in about 3 hours and outputs $r = 10$ rounds. A 10-round differential which holds with

probability 2^{-118} is illustrated in Fig. 2. This 10-round differential has 2554 characteristics that are distributed as shown in Table 2. The characteristic that holds with probability 2^{-123} is detailed in Table 3.

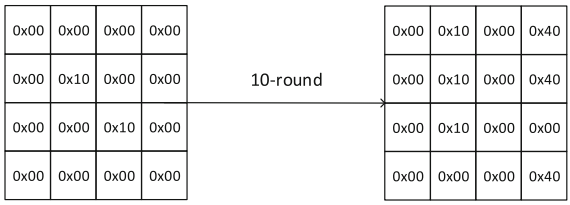

Fig. 2. A 10 rounds differential of Midori128

Table 2. The characteristics distribution of the 10-round differential

# of characteristics	Probability of each characteristic
1	2^{-123}
7	2^{-124}
23	2^{-125}
50	2^{-126}
83	2^{-127}
2390	$\leq 2^{-128}$

4 13-round Truncated Differential Cryptanalysis of Midori128

In this section, we show how we can extend the differential obtained in the previous section by two rounds above, and one round below, to obtain a truncated differential over 13 rounds (see Fig. 3). Then, we present a key recovery attack on Midori128 reduced to 13 rounds using this truncated differential.

The total probability of the 13-round differential can be calculated from its three parts. First, the differential probability of the top two rounds which is 2^{-112} and can be computed as follows: (i) $4 \rightarrow 2, 3 \rightarrow 1, 4 \rightarrow 2$, and $3 \rightarrow 1$ transitions over *MixColumn* ($z_0 \rightarrow w_0$) which happens with probability $2^{-16} \times 2^{-16} \times 2^{-16} \times 2^{-16} = 2^{-64}$, (ii) $3 \rightarrow 1$ and $3 \rightarrow 1$ transitions over *MixColumn* ($z_1 \rightarrow w_1$) which happens with probability $2^{-16} \times 2^{-16} = 2^{-32}$ and (iii) $w_1[5]$

and $w_1[10]$ equal difference 16 which happens with probability $2^{-8} \times 2^{-8} = 2^{-16}$. Second, the differential probability of 10-round is 2^{-118}, calculated by Algorithm 2. Third, the bottom 1 round has a differential probability equals to 1. Therefore, the differential probability of the 13-round is 2^{-230}.

Table 3. The 2^{-123} 10-round characteristic of Midori128

i	Input difference at round i (in hexadecimal)
0	00000000 00100000 00001000 00000000
1	00000000 00000000 00000000 00080800
2	40004040 00000000 00101010 00000000
3	40400040 40400000 08080008 08080000
4	00400040 00000040 08000000 08000800
5	40000040 00404000 08000008 00000000
6	40404000 00400040 08080008 08000800
7	00400000 40004000 00080000 08000800
8	00000000 00400000 00000800 00000000
9	00000000 00000000 00000000 00080800
10	40004040 00000000 00101010 00000000

In what follows, we show how we can perform our key recovery attack on Midori128 reduced to 13 rounds using the above differential. The attack is decomposed of two steps. The first step is the *data collection* in which we collect many pairs of messages to guarantee that at least one of them confirms to the 13-round truncated differential in Fig. 3. The second step is the *key recovery* in which the collected data pairs are used to identify the key candidates.

Proposition 1. *(Differential Property of bijective S-boxes) Given two non-zero differences, Δi and Δo, in $\mathbb{F}256$, the equation: $S(x) + S(x + \Delta i) = \Delta o$ has one solution on average. This property also applies to S^{-1}.*

Data Collection. To reduce the number of required chosen plaintext and get enough pairs to launch the attack, we use the structure technique. Here, our structure takes all the possible values in all the bytes except bytes 2 and 11. These bytes take fixed value. Therefore, one structure generates $2^{14\times8} \times (2^{14\times8} - 1)/2 \approx 2^{223}$ possible pairs. We need to collect 2^{230} message pairs because the total probability of the 13-round differential is 2^{-230}. Since each structure contains 2^{223} message pairs, we need to collect 2^7 structures to find the right pair. Therefore, we ask the encryption oracle for the encryption of 2^{119} messages.

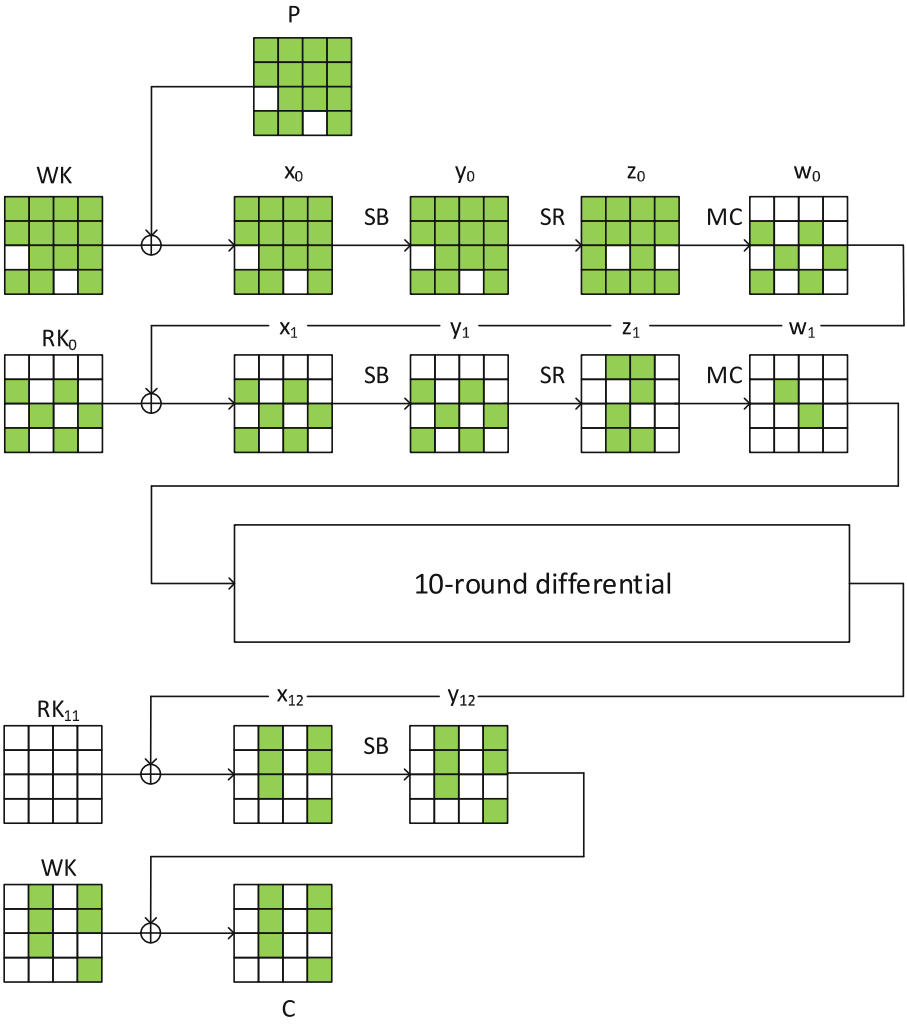

Fig. 3. 13-round truncated differential of Midori128

Algorithm 2. Find the maximum number of rounds, r, that has a differential which holds with probability $> 2^{-128}$ considering only the single bit difference for the active S-boxes

Result: X: Input difference, Y : Output difference, *Probability:* The probability of the differential, r: The number of rounds covered by the differential

for $i \leftarrow 0$ *to* $2^{28} - 1$ **do**
  $InputDiffProb[i] \leftarrow 1$,$OutputDiffProb[i] \leftarrow 0$;
  $InputParent[i] \leftarrow i$,$OutputParent[i] \leftarrow -1$;
$r \leftarrow 0$, *valid*$\leftarrow$ *true*;
while *valid* **do**
  for $i \leftarrow 0$ *to* $2^{28} - 1$ **do**
    if $InputDiffProb[i] = 0$ **then**
      continue;
    map i to state ΔS (see Sect. 3);
    forall the *entries in* Ω_i **do**
      $\Delta S_{temp} \leftarrow nextvalue(\Omega_i)$;
      $prob \leftarrow$ Probability $(\Delta S \rightarrow \Delta S_{temp})$;
      if $prob = 0$ **then**
        continue;
      $\Delta S_{temp} \leftarrow ShuffleCell(\Delta S_{temp})$;
      $\Delta S_{temp} \leftarrow MixColumn(\Delta S_{temp})$;
      map ΔS_{temp} into index j;
      if $OutParent[j] = -1$ **then**
        $OutputDiffProb[j] = InputDiffProb[i] \times prob$;
        $OutputParent[j] = InputParent[i]$;
      else if $OutParent[j] = InputParent[i]$ **then**
        $OutputDiffProb[j] = InputDiffProb[i] \times prob +$
        $OutputDiffProb[j]$;
      else
        if $InputDiffProb[i] \times prob > OutputDiffProb[j]$ **then**
          $OutputDiffProb[j] = InputDiffProb[i] \times prob$;
          $OutputParent[j] = InputParent[i]$;
  $InputDiffProb \leftarrow OutputDiffProb$, $InputParent \leftarrow OutParent$;
  initialize $OutputDiffProb$ to 0 , initialize $OutputParent$ to -1;
  get the index l of the maximum entry in $InputDiffProb$;
  if $InputDiffProb[l] <= 2^{-128}$ **then**
    $valid \leftarrow false$;
  $r \leftarrow r + 1$;
Get the index l of the maximum entry in $InputDiffProb$;
$X \leftarrow InputParent[l]$, $Y \leftarrow l$, $Probability \leftarrow InputDiffProb[l]$;

Key Recovery. In this step, we try to identify the key candidates that confirm to the 10-round differential. First, we try to identify the number of key suggestions of 26 bytes $WK[0, 1, 3 \cdots 10, 12 \cdots 15], RK_0[1, 3, 6, 9, 11, 14]$, and $WK[4, 5, 6, 12, 13, 15]$ that correspond to each pair of messages. This can be achieved as follows: to deduce the values of the 6 bytes $WK[4, 5, 6, 12, 13, 15]$, we know that the 6 bytes $\Delta x_{12}[4, 5, 6, 12, 13, 15]$ only take one value, the same difference of the end of the 10-round differential since the key add layer does not change the difference. The knowledge of the ciphertext allows to compute $\Delta y_{12}[4, 5, 6, 12, 13, 15]$. Using the differential property of the S-box, we can evaluate $y_{12}[4, 5, 6, 12, 13, 15]$. The knowledge of the ciphertext with $y_{12}[4, 5, 6, 12, 13, 15]$ allows us to deduce the values of the 6 bytes $WK[4, 5, 6, 12, 13, 15]$. From the other side, we know the difference ΔW_1 since it is the same difference at the beginning of the 10-round differential. Then we propagate this difference linearly trough *MixColumn* and *InvShuffleCell* to get the difference Δy_1. Δy_1 has only 6 active bytes and each, after the *SubCell* operation, has only 6 possible differences. Therefore, we have $6^6 \approx 2^{15.6}$ possible differences at Δx_1. Then after the *MixColumn* and *InvShuffleCell*, we have only $2^{15.6}$ possible differences at Δy_0. The knowledge of the plaintext allows us to compute the difference at Δx_0. Then, guessing the $2^{15.6}$ possible differences of Δy_0 and using the S-box proposition, we get the value of $x_0[0, 1, 3 \cdots 10, 12 \cdots 15]$. From the knowledge of the plaintext we can drive the value of $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$. As a result we have $2^{15.6}$ key candidates for $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$ and $WK[4, 5, 6, 12, 13, 15]$ but we have 6 bytes common; therefore, we have only $2^{15.6-48} = 2^{-32.4}$ key candidates. To derive the 6 bytes value $RK_0[1, 3, 6, 9, 11, 14]$, we know that we have only one difference at Δy_1. Then we get one key candidate for $RK_0[1, 3, 6, 9, 11, 14]$ but also we have 5 bytes filter between $WK[0, 1, 3 \cdots 10, 12 \cdots 15]$ and $RK0[1, 3, 6, 9, 11, 14]$, from the key schedule as the whitening key is the master key K and the round keys $RK_i = K \oplus \beta_i$ and β_i is constant. Therefore, we have only $2^{-32.4} \times 2^{-40} = 2^{-72.4}$ key candidates for each message pair. To identify the remaining key candidates for all the message pairs, we should identify the remaining message pairs after the ciphertext filter. The ciphertext has a filter probability of $2^{-112.3}$ and is computed as follows: we have 10 bytes of zero difference which have probability of 2^{-80} and the remaining 6 bytes have probability of $2^{15.7-48} = 2^{-32.3}$ since we know that each byte of the 6 active bytes in Δx_{12} has only one difference and after the S-box layer 5 bytes out of these 6 active bytes, each, has 6 possible difference and the remaining byte has 7 possible differences. Hence, we have $6^5 \times 7 = 2^{15.7}$ possible differences at Δy_{12} which also the same differences in the 6 bytes in the ciphertext. Therefore, after the ciphertext filter, we have $2^{230} \times 2^{-112.3} = 2^{117.7}$ remaining message pairs to identify the key candidates. As a result, we have $2^{117.7} \times 2^{-72.4} = 2^{45.3}$ remaining key candidates for 15 bytes of the master key K.

In order to determine efficiently the remaining key candidates for all the remaining message pairs after the ciphertext filter, we perform the following steps:

1. From the ciphertext side, we have only one value for the active bytes of Δx_{12}. Then, using the S-box proposition, we can derive the 6 bytes $WK[4, 5, 6, 12, 13, 15]$. Therefore, we have $2^{117.7}$ key candidates for $WK[4, 5, 6, 12, 13, 15] \equiv K[4, 5, 6, 12, 13, 15]$.
2. From the plaintext side, by guessing the 6 possible differences of $\Delta w_0[14]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can know the values of $\Delta y_0[7, 8, 13]$. Hence, we can use the S-box proposition to derive the values of $x_0[7, 8, 13]$. Then, knowing the plaintext and $x_0[7, 8, 13]$ allows to derive $WK[7, 8, 13] \equiv K[7, 8, 13]$. Using the one value of the difference $\Delta y_1[14]$, we can derive the value of $x_1[14]$ using the S-box proposition. Then, we can derive $RK_0[14] \equiv K[14]$. At this stage, we have $2^{117.7}$ key candidates and we guess 6 values to derive $K[7, 8, 13, 14]$ but we have one filter of $K[13]$ between this step and the previous step. Therefore, in total, we have $2^{117.7} \times 6 \times 2^{-8} = 2^{112.3}$ remaining key candidates for $K[4 \cdots 8, 12 \cdots 15]$.
3. By guessing the 6 possible differences of $\Delta w_0[6]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell*, we can determine the values of $\Delta y_0[1, 4, 14]$. Hence, we can use the S-box proposition to derive the values of $x_0[1, 4, 14]$. Then, knowing the plaintext and $x_0[1, 4, 14]$ allows us to derive $WK[1, 4, 14] \equiv K[1, 4, 14]$. Using the one value of the difference $\Delta y_1[6]$, we can derive the value of $x_1[6]$ using the S-box proposition. Then, we can derive $RK_0[6] \equiv K[6]$. At this stage we have $2^{112.3}$ key candidates and we guess 6 values to derive $K[1, 4, 6, 14]$ but we have 3 filters of $K[4, 6, 14]$ between this step and the previous step. Therefore, in total we have $2^{112.3} \times 6 \times 2^{-24} = 2^{90.9}$ remaining key candidates for $K[1, 4 \cdots 8, 12 \cdots 15]$.
4. By guessing the 6^2 possible differences of $\Delta w_0[1, 3]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can determine the values of $\Delta y_0[0, 5, 10, 15]$. Hence, we can use the S-box proposition to derive the values of $x_0[0, 5, 10, 15]$. Then, knowing the plaintext and $x_0[0, 5, 10, 15]$ allows us to derive $WK[0, 5, 10, 15] \equiv K[0, 5, 10, 15]$. Using the one value of the difference $\Delta y_1[1, 3]$, we can derive the value of $x_1[1, 3]$ using the S-box proposition. Then, we can derive $RK_0[1, 3] \equiv K[1, 3]$. At this stage we have $2^{90.9}$ key candidates and we guess 6^2 values to derive $K[0, 1, 3, 5, 10, 15]$ but we have 3 filters of $K[1, 5, 15]$ between this step and the previous step. Therefore, in total we have $2^{90.9} \times 6^2 \times 2^{-24} = 2^{72.1}$ remaining key candidates for $K[0, 1, 3 \cdots 8, 10, 12 \cdots 15]$.
5. By guessing the 6^2 possible differences of $\Delta w_0[9, 11]$ and propagating them backward through the linear operations *MixColumn* and *InvShuffleCell* we can determine the values of $\Delta y_0[3, 6, 9, 12]$. Hence, we can use the S-box proposition to derive the values of $x_0[3, 6, 9, 12]$. Then, knowing the plaintext and $x_0[3, 6, 9, 12]$ allows us to derive $WK[3, 6, 9, 12] \equiv K[3, 6, 9, 12]$. Using the one value of the difference $\Delta y_1[9, 11]$, we can derive the value of $x_1[9, 11]$ using the S-box proposition. Then, we can derive $RK_0[9, 11] \equiv K[9, 11]$. At this stage we have $2^{72.1}$ key candidates and we guess 6^2 values to derive $K[3, 6, 9, 11, 12]$ but we have one filter in this step of $K[9]$ and 3 of $K[3, 6, 12]$ between this step and the previous step. Therefore, in total we have $2^{72.1} \times 6^2 \times 2^{-32} = 2^{45.3}$ remaining key candidates for $K[0, 1, 3 \cdots 15]$.

Attack Complexity. The time complexity of the key recovery phase can be derived from the previous steps as follows: step 1 needs $2 \times 2 \times 2^{117.7}/(4 \times 13) = 2^{114}$ encryptions, step 2 needs $2 \times 2^{117.7} \times 6/(4 \times 13) = 2^{115.6}$ encryptions, step 3 needs $2 \times 2^{112.3} \times 6/(4 \times 13) = 2^{110.2}$ encryptions, step 4 needs $2 \times 2^{90.9} \times 6^2/(4 \times 13) = 2^{91.4}$ encryptions, step 5 needs $2 \times 2^{72.1} \times 6^2/(4 \times 13) = 2^{72.6}$ encryptions. Therefore, the time complexity to find $2^{45.3}$ key candidates for $K[0, 1, 3 \cdots 15]$ is $2^{114} + 2^{115.6} + 2^{110.2} + 2^{91.4} + 2^{72.6} \approx 2^{115.6}$. To retrieve the master key we make an exhaustive search for the remaining key candidates with $K[2]$ which needs $2^8 \times 2^{45.3} = 2^{53.3}$ encryptions. Therefore, the time complexity of the attack is dominated by the time needed to build the required structures which is 2^{119} encryptions. The data complexity of the attack is 2^{119} chosen plaintext.

5 Multiple Differential Cryptanalysis of Midori128

In this section, we describe a multiple differential attack that offers some time-data trade-off compared to the previous attack. Using Algorithm 2 and by enumerating all the 10-round differentials that hold with probability $> 2^{-124}$, we found 700 such differentials. In here, we show how to exploit 16 of them to launch a multiple differential attack on Midori128. These 16 differentials are shown in Table 4. As shown in the table, all these differentials have the same input difference, but have different output differences that are all active in the same bytes. The first differential in Table 4 is the differential that is used in the attack described in the previous section. Therefore, the previous attack can be applied with multiple differentials with small modifications.

In this attack, we retrieve the same key bytes as in the previous attack. The total differential probability of the 16 differential is $2^{-114.7}$. Therefore, the total differential probability of the 13-round differential is $2^{-112} \times 2^{-114.7} = 2^{-226.7}$. Consequently, we need $2^{226.7-223} = 2^{3.7}$ structures with $2^{3.7} \times 2^{112} = 2^{115.7}$ chosen plaintext. As shown in Table 4, $\Delta x_{12}[4, 5, 6]$ have the following possible differences $\{4, 8, 16, 32\}$ and $\Delta x_{12}[12, 13, 15]$ have the following possible differences $\{8, 16, 32, 64\}$. Therefore, after the S-box layer, we have the following: each one of $\Delta y_{12}[4, 6, 15]$ has 19 possible differences, $\Delta y_{12}[5]$ has 15 possible differences, $\Delta y_{12}[12]$ has 17 possible differences, and $\Delta y_{12}[13]$ has 20 possible differences. As a result, we have $19^3 \times 15 \times 17 \times 20 = 2^{25.1}$ possible differences at the ciphertext. Consequently, we have $2^{226.7} \times 2^{-80} \times 2^{25.1-48} = 2^{123.8}$ remaining message pairs after the ciphertext filter. The remaining key candidates for each pair are $16 \times 2^{15.6} \times 2^{-88} = 2^{-68.4}$. Therefore, the number of remaining key candidates for all the remaining message pairs, after the ciphertext filter, is given by $2^{123.8} \times 2^{-68.4} = 2^{55.4}$ which can be exhaustively searched with the remaining key byte. We can use the same steps that are used in the previous attack to determine theses $2^{55.4}$ remaining key candidates. The time complexity of the attack is dominated by step 2. At the beginning of step 2, we have $2^{123.8} \times 16 = 2^{127.8}$ key candidates for $K[4, 5, 6, 12, 13, 15]$. Therefore, the time complexity of step 2 is $2 \times 2^{127.8} \times 6/(4 \times 13) = 2^{125.7}$ encryptions. The data complexity is $2^{115.7}$ chosen plaintext.

Table 4. 10-round differentials of Midori128

Input difference (in hexadecimal)	Output difference (in hexadecimal)	Probability
00000000 00100000 00001000 00000000	40004040 00000000 00101010 00000000	$2^{-118.09}$
00000000 00100000 00001000 00000000	08000808 00000000 00080808 00000000	$2^{-118.12}$
00000000 00100000 00001000 00000000	08000808 00000000 00202020 00000000	$2^{-118.12}$
00000000 00100000 00001000 00000000	40004040 00000000 00080808 00000000	$2^{-118.18}$
00000000 00100000 00001000 00000000	40004040 00000000 00202020 00000000	$2^{-118.18}$
00000000 00100000 00001000 00000000	10001010 00000000 00080808 00000000	$2^{-118.36}$
00000000 00100000 00001000 00000000	10001010 00000000 00202020 00000000	$2^{-118.36}$
00000000 00100000 00001000 00000000	10001010 00000000 00101010 00000000	$2^{-118.43}$
00000000 00100000 00001000 00000000	10001010 00000000 00040404 00000000	$2^{-118.8}$
00000000 00100000 00001000 00000000	40004040 00000000 00040404 00000000	$2^{-118.8}$
00000000 00100000 00001000 00000000	20002020 00000000 00101010 00000000	$2^{-118.81}$
00000000 00100000 00001000 00000000	08000808 00000000 00040404 00000000	$2^{-119.29}$
00000000 00100000 00001000 00000000	20002020 00000000 00080808 00000000	$2^{-119.81}$
00000000 00100000 00001000 00000000	20002020 00000000 00202020 00000000	$2^{-119.81}$
00000000 00100000 00001000 00000000	08000808 00000000 00101010 00000000	$2^{-120.32}$
00000000 00100000 00001000 00000000	20002020 00000000 00040404 00000000	$2^{-120.43}$

6 Conclusion

We showed how to exploit the structure of the S-boxes and the *MixColumn* operations of Midori128 in order to obtain long differentials that use single bit difference for the inputs and outputs of the active S-boxes. Then, we developed an algorithm that can be used to efficiently enumerate all such differentials for a given number of rounds. Using this algorithm, we obtained a 10-round differential that holds with probability 2^{-118}. By appending 2 rounds above and one round below this 10-round differential, we obtained a 13 round truncated differential and used it to launch a key recovery attack attack on 13-round reduced Midori128. The time and data complexities of the attack are 2^{119} encryptions and 2^{119} chosen plaintext. Moreover, we presented a multiple differential attack on the 13-round reduced cipher with time and data complexities of $2^{125.7}$ encryptions and $2^{115.7}$ chosen plaintext, respectively.

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: a block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 411–436. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_17
2. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

3. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
4. Canteaut, A., Fuhr, T., Gilbert, H., Naya-Plasencia, M., Reinhard, J.-R.: Multiple differential cryptanalysis of round-reduced PRINCE. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 591–610. Springer, Heidelberg (2015)
5. Guo, J., Jean, J., Nikolić, I., Qiao, K., Sasaki, Y., Sim, S.M.: Invariant Subspace Attack Against Full Midori64. IACR Cryptology ePrint Archive, 2015/1189 (2015). https://eprint.iacr.org/2015/1189.pdf
6. Hong, D., Sung, J., Hong, S.H., Lim, J.-I., Lee, S.-J., Koo, B.-S., Lee, C.-H., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J.-S., Chee, S.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
7. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: a block cipher for IC-printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
8. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
9. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New lightweight DES variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
10. Lim, C.H., Korkishko, T.: mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
11. Lin, L., Wu, W.: Meet-in-the-Middle Attacks on Reduced-Round Midori-64. IACR Cryptology ePrint Archive, 2015/1165 (2015). https://eprint.iacr.org/2015/1165.pdf
12. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)