IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2017, No. 1, pp. 215–239. DOI:10.13154/tosc.v2017.i1.215-239

# Meet-in-the-Middle Attacks on Reduced-Round Midori64

Li Lin and Wenling Wu

[1] TCA Labaratory, State Key Laboratory of Computer Science (SKLCS), Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[3] Graduate University of Chinese Academy of Sciences, Beijing 100190, China
{linli,wwl}@tca.iscas.ac.cn

**Abstract.** Midori is a lightweight block cipher designed by Banik et al. at ASIACRYPT 2015 to achieve low energy consumption. One version of Midori uses a 64-bit state, another uses a 128-bit state and we denote these versions Midori64 and Midori128. Each of these versions uses a 128-bit key. In this paper, we focus on the key-recovery attacks on reduced-round Midori64 with meet-in-the-middle method. We use the differential enumeration, key-bridging and key-dependent sieve techniques which are popular to analyze AES to attack Midori64. Using key-bridging and key-dependent sieve techniques directly to achieve the complexity lower bound is almost impossible, we give the model on how to achieve the complexity lower bound using these techniques. We also propose the state-bridge technique to use some key relations that are quite complicated and divided by some rounds. With a 6-round distinguisher, we achieve a 10-round attack. After that, by adding one round at the end, we get an 11-round attack. Finally, with a 7-round distinguisher, we get an attack on 12-round Midori64. To the best of our knowledge, these are recently the best attacks on Midori64 in the single-key setting.

**Keywords:** Block Cipher · Cryptanalysis · Meet-in-the-Middle Attack · Midori64

## 1 Introduction

In the past few years, lightweight cryptography has become a popular research discipline with a number of ciphers and hash functions proposed. The goals of these ciphers range from minimizing the hardware area [BKL+07, SMMK12, WZ11] to low latency [BCG+12]. However, the optimization goal of low energy for block cipher design has not attached much attention. At ASIACRYPT 2015, Banik et al. present a new lightweight block cipher Midori that is optimized with respect to the energy consumed by the circuit per bit in encryption or decryption operation [BBI+15a, BBI+15b]. Midori is based on the Substitution-Permutation Network (SPN). One version of Midori uses a 64-bit state, another uses a 128-bit state and we denote these versions Midori64 and Midori128. Each of these versions uses a 128-bit key.

Meet-in-the-middle attack is first proposed by Diffie and Hellman to attack DES [DH77]. In recent years, it is widely researched due to its effectiveness against block cipher AES [DR02]. For AES, Gilbert and Minier show in [GM00] some collision attacks on 7-round AES. At FSE 2008, Demirci and Selçuk improve the Gilbert and Minier attacks using meet-in-the-middle method instead of collision idea. More specifically, they show that the value of each byte of 4-round AES ciphertext can be described by a function of the $\delta$-*set*, i.e., a set of 256 plaintexts where a byte (called active byte) can take all values and the other 15 bytes are constant, parameterized by 25 [DS08] and 24 [DTÇB09] 8-bit parameters. The last improvement is due to storing differences instead of values. This function is used to build a distinguisher in the precomputation phase, i.e., they build a lookup table containing all the possible sequences constructed from a $\delta$-set. In the


Received: 2016-09-01, Revised: 2016-11-23, Accepted: 2017-01-24, Published: 2017-03-0

online phase, they identify a $\delta$-set, then partially decrypt the $\delta$-set through some rounds and check whether it belongs to the table. At ASIACRYPT 2010, Dunkelman et al. develop many new ideas to solve the memory problems of the Demirci and Selçuk attacks [DKS10]. First of all, they only store multiset, i.e., an unordered sequence with multiplicity, rather than the ordered sequence. Secondly, they propose the key-bridging technique which uses the relations of sub-key cells to improve the complexity in the online phase. The third and main idea is the differential enumeration technique which uses a special property on a truncated differential trail to reduce the number of parameters that describes the set of functions from 24 to 16. Furthermore, Derbez et al. present a significant improvement to the Dunkelman et al.'s differential enumeration technique at EUROCRYPT 2013 [DFJ13], called efficient tabulation. Using this rebound-like idea, they show that many values in the precomputation table are not reached at all under the constraint of a special truncated differential trail. Actually, the size of the precomputation table is determined by 10 byte-parameters only. At FSE 2014, Li et al. introduce the key-dependent sieve technique, which filters the wrong states based on the key relations, to further reduce the complexity in the precomputaion phase [LJW14]. Then they give an attack on 9-round AES-192. In [LJ16], Li and Jin give a meet-in-the-middle attack on 10-round AES-256.

## Our contributions.

In this paper, we carefully study and apply the variant of Derbez et al. attack on Midori64. Differential enumeration, key-dependent sieve and key-bridging techniques are used to achieve a better complexity. Using key-dependent sieve and key-bridging techniques directly to achieve the complexity lower bound is almost impossible, we give the model on how to achieve the complexity lower bound with these techniques. We also propose the state-bridge technique to use some key relations that are quite complicated and divided by some rounds. For Midori64, we present a 6-round distinguisher. Based on this distinguisher, we add one round at the beginning and three rounds at the end to present a 10-round meet-in-the-middle attack. The time complexity of this attack is $2^{99.5}$ 10-round Midori64 encryptions, the data complexity is $2^{61.5}$ chosen-plaintexts and the memory complexity is $2^{92.7}$ 64-bit blocks. After that, by adding one round at the end, we get an 11-round attack with time complexity of $2^{122}$ 11-round Midori64 encryptions, data complexity of $2^{53}$ chosen-plaintexts and memory complexity of $2^{89.2}$ 64-bit blocks. Finally, with a 7-round distinguisher, we get an attack on 12-round Midori64 with time complexity of $2^{125.5}$ 12-round Midori64 encryptions, data complexity of $2^{55.5}$ chosen-plaintexts and memory complexity of $2^{106}$ 64-bit blocks. It is worthy to mention that we ran the automatic search tool of [DF16], and found one potential attack that may provide slightly better time complexity[1]. Unfortunately, this attack can hardly be transferred into a valid attack due to the too low distinguishing probability.

After this paper, some results on Midori64 have been proposed. In [GJN+15] and [TLS16], Guo et al. and Todo et al. give invariant subspace attack and nonlinear invariant attack on full-round Midori64 in the weak-key setting, respectively. In [CW16], Chen and Wang give impossible differential attack on 10-round Midori64. And in [DS16], Dong and Shen give differential attack on 14-round Midori64 in the related-key setting. We present here a summary of our attack results, and compare them to the best attacks known for it. This summary is given in Table 1.

**Organizations of this paper.** The rest of this paper is organized as follows. In Section 2, we provide a brief description of Midori64, some definitions and properties, a brief recall of the previous meet-in-the-middle distinguishers, the attack model and the way to use key relations to improve the complexity. In Section 3, we give our attack on 10-round Midori64. In Section 4, we give our attack on 11-round Midori64. In Section 5, we give our attack on 12-round Midori64. In Section 6, we conclude this paper.

[1] Note that the tool in [DF16] produces potential attacks, but cannot always guarantee the complexity will be actually reached.

**Table 1:** Summary of the best attacks on Midori64.

| Attack type | Rounds | Data | Memory (Bytes) | Time (Enc) | Source |
|---|---|---|---|---|---|
| IDA (SK) | 10 | $2^{62.4}$ CPs | $2^{68.13}$ | $2^{80.81}$ | [CW16] |
| DA (RK) | 14 | $2^{59}$ CPs | $2^{115}$ | $2^{116}$ | [DS16] |
| ISA (WK) | 16 | 2 CPs | - | $2^{16}$ | [GJN+15] |
| NLA (WK) | 16 | $33h$ CPs | - | $32^3 \times h$ | [TLS16] |
| MITM (SK) | 10 | $2^{59.5}$ CPs | $2^{95.7}$ | $2^{99.5}$ | Sec. 3 |
| MITM (SK) | 11 | $2^{53}$ CPs | $2^{92.2}$ | $2^{122}$ | Sec. 4 |
| MITM (SK) | 12 | $2^{55.5}$ CPs | $2^{109}$ | $2^{125.5}$ | Sec. 5 |

CPs: Chosen-Plaintexts, MITM: meet-in-the-middle, IDA: impossible differential attack, DA: differential attack, ISA: invariant subspace attack, NSA: nonlinear invariant attack, SK: single-key, RK: related-key, WK: weak-key, $h$: the number of blocks in the mode of operation.

# 2 Preliminaries

In this section, we give a short description of Midori64 and give some definitions and propositions used throughout this paper. Then we briefly recall the previous meet-in-the-middle distinguishers on AES. Finally, the basic attack model and the way to use key relations to improve the complexity are given.

## 2.1 Description of Midori64

Midori is a lightweight block cipher designed by Banik et al. at ASIACRYPT 2015 [BBI+15b] and is based on the Substitution-Permutation Network (SPN). One version of Midori uses a 64-bit state, another uses a 128-bit state and we denote these versions Midori64 and Midori128. Each of these versions uses a 128-bit key. In this paper, we focus on the 64-bit version of Midori, so we describe it here. The Midori64 block cipher operates on 64-bit state, and uses the following $4 \times 4$ array called state as a data expression:

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

where the size of each cell is 4 bits.

A Midori64 round applies the following four operations to the state matrix:

- **SubCell:** Apply the non-linear $4 \times 4$ S-box in parallel on each nibble of the state.
- **ShuffleCell:** Each nibble of the state is permuted as follows:

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

- **MixColumn:** Midori64 utilizes an involutive binary matrix $\mathbf{M}$ defined as follows:

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$\mathbf{M}$ is applied to every 4-nibble column of the state $S$, i.e.,

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

- **KeyAdd:** The $i^{th}$ 64-bit round key $rk_i$ is xored to a state $S$.

Before the first round, an additional KeyAdd operation is applied, and in the last round the ShuffleCell and MixColumn operations are omitted. The total round number of Midori64 is 16.

The key-schedule of Midori64 is quite simple. A 128-bit secret key $K$ is denoted as two 64-bit keys $k_0$ and $k_1$ as $K = k_0||k_1$. Suppose we focus on Midori64 reduced to $R$-round, the whiten key and the last sub-key are $rk_{-1} = rk_{R-1} = k_0 \oplus k_1$, and the sub-key for round $i$ is $rk_i = k_{(i \bmod 2)} \oplus \alpha_i$, where $0 \leq i \leq R-2$ and $\alpha_i$ is a constant.

In this paper, the plaintext is denoted by $P$, the ciphertext is denoted by $C$. The intermediate state at the beginning of round $i$ is denoted by $x_i$, and the intermediate state after the SubCell, ShuffleCell, MixColumn operations of round $i$ are denoted by $y_i$, $z_i$ and $w_i$. $x_i[j]$ denotes the $j^{th}$ nibble of $x_i$. $x_i^k[j]$ denotes the $k^{th}$ element of a set of some $x_i[j]$. $\Delta x_i^k[j]$ denotes the difference of the $k^{th}$ element and $0^{th}$ element of a set, i.e., $\Delta x_i^k[j] = x_i^k[j] \oplus x_i^0[j]$. $x - x[j]$ denotes a state including all the nibbles of $x$ except the $j^{th}$ nibble.

In some cases, we are interested in interchanging the order of the MixColumn and KeyAdd operations. As these operations are linear, they can be interchanged by first xoring the data with an equivalent key $ru_i = MixColumn^{-1}(rk_i)$ and then applying the MixColumn operation. And we denote the intermediate state after xoring with $ru_i$ by $\overline{w_i}$. We also let $u_i = MixColumn^{-1}(k_i)$, where $i = 0, 1$.

## 2.2 Definitions and Propositions

In [DR02], Daemen and Rijmen first proposed the definition of $\delta$-set of byte, which is a structured set of 256 plaintexts $\{P_0, \cdots, P_{255}\}$ in which one active byte assumes each one of the 256 possible values exactly once, and each one of the other 15 bytes is a constant. After that, $\delta$-set was used in the meet-in-the-middle attacks on AES and other ciphers. In [LWWZ13], Lin et al. extended the definition of $\delta$-set to $T$ active cells, and got $T$-$\delta$-set. In this paper, we use 2-$\delta$-set which is defined as follows.

**Definition 1** (**2-$\delta$-set**). Let a 2-$\delta$-set be a set of $2^{2\times4}$ states that are all different in two state nibbles (active nibbles) and all equal in the other state nibbles (inactive nibbles).

In [DR06], Daemen and Rijmen gave the definition of Super-box for AES. For Midori, we can give a similar definition as follows (by interchanging the order of the MixColumn and KeyAdd operations).

**Definition 2** (**Super-box**). For each value of one column of $rk_3$, a Midori Super-box maps one column of $z_3$ to one column of $y_4$ as shown in Fig. 1. It consists of one SubCell operation, one MixColumn operation, one KeyAdd operation and one SubCell operation.

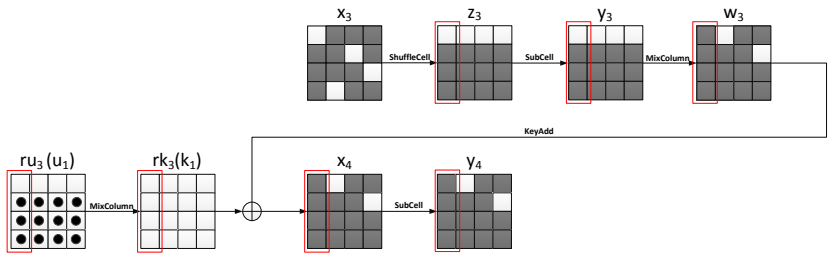



**Figure 1:** Super-box of Midori64.

For one S-box, we have the following proposition.

**Proposition 1** (**Differential Property of S-box, [DFJ13]**). *Given $\Delta_i$ and $\Delta_0$ two non-zero differences, the equation of S-box*

$$S(x) \oplus S(x \oplus \Delta_i) = \Delta_0, \tag{1}$$

*has one solution in average.*

This proposition also applies to Super-box.

**Proposition 2** (**Differential Property of Super-box**). *Given $\Delta_i$ and $\Delta_0$ two non-zero differences in $F_{2^{16}}$, the equation of Super-box*

$$Super-S(x)\oplus Super-S(x\oplus\Delta_i)=\Delta_0, \tag{2}$$

*has one solution in average for each key value.*

For $ru_i$, we have the following proposition.

**Proposition 3.** *As shown in Fig. 1, if the first column of $z_3$ is active only in the last 3 nibbles, $z_3[1,2,3]||y_4[0,1,2,3]$ has one solution in average for each $\Delta z_3[1,2,3]||\Delta y_4[0,1,2,3]||ru_3[1,2,3]$.*

*Proof.* We use the equivalent sub-key in this proof. For each $y_4[0,1,2,3]$ and $ru_3[1,2,3]$, since $\Delta y_4[0,1,2,3]$ is known, one can get $\overline{w_3}[0,1,2,3]$ and $\Delta\overline{w_3}[0,1,2,3]$. With the probability of $2^{-4}$, $y_3[0,1,2,3]$ is active only in the last 3 nibbles. By xoring $ru_3[1,2,3]$, one can get $\Delta z_3[1,2,3]$.

Therefore, for each $\Delta z_3[1,2,3]$ and $\Delta y_4[0,1,2,3]$, the average number of input values of Super-box is $2^{16-12-4}=1$ for each $ru_3[1,2,3]$.

□

## 2.3 Reviews of Former Works

In this section, we review the previous meet-in-the-middle distinguishers on AES in [DS08, DKS10, DFJ13, LJW14].

**Demirci and Selçuk distinguisher.** The distinguishers of Demirci and Selçuk attacks are based on the proposition below.

**Proposition 4** (**Demirci and Selçuk distinguisher, [DS08]**). *Consider the encryption of a $\delta$-set through four full AES rounds. For each of the 16 bytes of the state, the ordered sequence of 256 values of that byte in the corresponding ciphertexts is fully determined by just 25 byte-parameters. Consequently, for any fixed byte position, there are at most $2^{200}$ possible sequences when we consider all the possible choices of keys and $\delta$-sets.*

The table containing all the $2^{200}$ possible sequences is tiny compared with the set of all functions of this type which counts as many as $2^{8\cdot 2^8}=2^{2048}$ elements [DS08]. Considering the differences rather than values, the set of functions can be described by 24 byte-parameters [DTÇB09]. The 24 byte-parameters which map $x_1[0]$ to $\Delta x_5[0]$ are presented as gray cells in Fig. 2. This observation was used in [DTÇB09] to mount attacks on reduced-round AES-256.

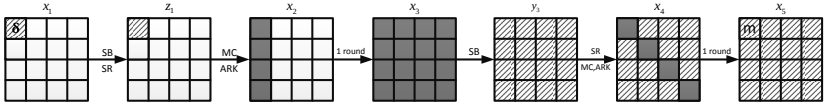



**Figure 2:** The 4-round AES distinguisher used in [DTÇB09]. The gray cells represent 24 byte-parameters, $\delta$ represents the $\delta$-set and $m$ represents the differential sequence to be stored.

**Dunkelman et al. distinguisher.** In [DKS10], Dunkelman et al. introduced three new improvements to further reduce the memory complexity of [DTÇB09]. The first uses multiset which is an unordered sequence with multiplicity to replace ordered sequence in the precomputation phase, since there is enough information so that the attack succeeds. The second uses the relations of sub-key cells to improve the complexity in the online phase called **key-bridging technique**. The third and main improvement uses a novel idea named **differential enumeration technique**. Consider a truncated differential trail for four full AES rounds, in which both the input and the

output differences are non-zero in a single byte as shown in Fig. 3. Since $\Delta x_2$ and $\Delta z_4$ can take $2^{32}$ different values, respectively, then $x_3$ can take only $2^{64}$ different values rather than $2^{128}$ due to Proposition 1. Therefore, the number of parameters which determines the size of the precomputation table reduces from 24 to 16. The probability of this differential trail is expected to be about $2^{-120}$ and thus it is expected that $2^{120}$ randomly chosen pairs with differences would contain one pair that satisfies the trail.

**Derbez et al. distinguisher.** In [DFJ13], Derbez et al. used the efficient tabulation to improve Dunkelman et al.'s differential enumeration technique. Combining with the rebound-like idea, many values in the precomputation table are not reached at all under the constraint of a truncated differential trail.

**Proposition 5 (Differential Enumeration Technique with Efficient Tabulation, [DFJ13]).** *If a message of $\delta$-set belongs to a pair conforming to the 4-round truncated differential trail outlined in Fig. 3, the values of multiset are only determined by 10 byte-parameters of intermediate states $\Delta z_1[0]||x_2[0,1,2,3]||\Delta x_5[0]||z_4[0,1,2,3]$ presented as gray cells in Fig. 3.*

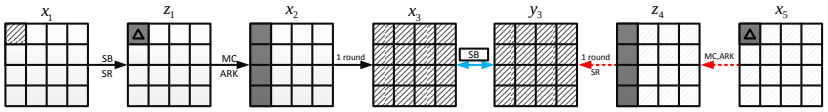



**Figure 3:** The truncated differential trail of 4-round AES used in [DS08], the gray cells represent 10 byte-parameters, $\Delta$ represents difference.

The main idea of their works is that suppose one gets a pair of messages conforming to this truncated differential trail, $\Delta x_3$ is determined by $\Delta z_1[0]||x_2[0,1,2,3]$ and $\Delta y_3$ is determined by $\Delta x_5[0]||z_4[0,1,2,3]$. By Proposition 1, part of the 24 byte-parameters in the Demirci and Selçuk distinguisher, i.e. $x_3$, can be determined. Therefore, the number of parameters which determines the size of the precomputation table reduces from 16 to 10. In the rest of this paper, when we speak of differential enumeration technique, we mean differential enumeration technique with efficient tabulation. The probability of this differential trail is also expected to be about $2^{-120}$.

**Li et al. distinguisher.** At FSE 2014, Li et al. introduced the **key-dependent sieve technique**, which filters the wrong states based on the key relations, to further reduce the complexity in the precomputation phase [LJW14]. More specifically, as shown in Fig. 4, the precomputation procedure allows to deduce $ru_2[3,6,9,12]$ and $rk_3$ independently. Meanwhile, by the key-schedule of AES-192, it is obvious that the knowledge of $rk_3$ allows to deduce Column 0 and Column 1 of $rk_2$. This means that the value of the equivalent sub-key $ru_2[3,6]$ can be deduced from $rk_3$. Thus there exists a contradiction between $ru_2[3,6]$ and $rk_3$ with a probability of $1-2^{-16}$. Therefore, the size of precomputation table is improved by a factor of $2^{16}$.

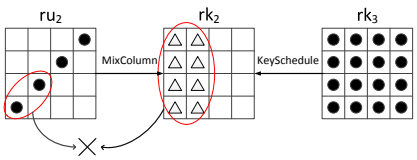



**Figure 4:** Key-dependent sieve technique based on the key-schedule algorithm of AES-192

## 2.4 Basic Attack Model

In this section, we present a unified view of the meet-in-the-middle attack, where $R$ rounds of block cipher can be split into three consecutive parts: $r_0$, $r_1$, and $r_2$, such that a particular set of messages may verify a certain property we denote ★ in the sequel in the middle $r_1$ rounds as shown in Fig. 5.

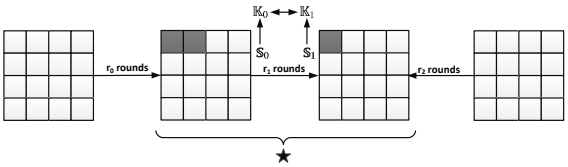



**Figure 5:** General model of meet-in-the-middle attack, where some messages in the middle rounds may verify a certain ★ property used to perform the meet-in-the-middle method.

The general attack model uses two successive phases:

**Precomputation phase**

1. In the precomputation phase, we build a lookup table $T$ containing all the possible sequences constructed from a 2-$\delta$-set such that one message verifies a truncated differential trail.

**Online phase**

1. In the online phase, we need to identify a 2-$\delta$-set containing a message $m$ verifying the desired truncated differential trail of the precomputation phase. This is done by using a large number of plaintexts and ciphertexts, and expecting that for each key candidate, there is one pair of plaintexts satisfying the truncated differential trail.
2. Finally, we partially decrypt the associated 2-$\delta$-set through the last $r_2$ rounds and check whether it belongs to $T$.

## 2.5 Key Relations to Improve the Complexity

From the Derbez et al. distinguisher in Section 2.3, we can deduce many key values from the truncated differential trail. Meanwhile, the online phase also includes some key values. Since Midori64 uses two keys $k_0$ and $k_1$ alternately in every two rounds, a lot of key relations can be found both in the precomputation phase and online phase. However, these key relations cannot be used to improve the complexity directly. As we can see in Fig. 5, $\mathbb{K}_0$ can be deduced from a set of state variables $\mathbb{S}_0$, $\mathbb{K}_1$ can be deduced from a set of state variables $\mathbb{S}_1$ and $\mathbb{K}_0 = \mathbb{K}_1$. If we guess $\mathbb{S}_0$ and $\mathbb{S}_1$ to get the precomputation table $T$, then use $\mathbb{K}_0 = \mathbb{K}_1$ to reduce the size of $T$, all the values in $\mathbb{S}_0 \cup \mathbb{S}_1$ need to be guessed. Hence the time complexity remains unchanged.

To use the key-dependent sieve and key-bridging techniques to improve the time complexity, we can store $\mathbb{S}_0$ in a new table $T_0$ with the index of $\mathbb{K}_0$. In the attack, if we want to use the states in $\mathbb{S}_0$, we can guess $\mathbb{S}_1$ and deduce $\mathbb{K}_1$, then look up $T_0$ to get $\mathbb{S}_0$ with the index of $\mathbb{K}_1$. If there exist relations between $\mathbb{K}_1$ and a new set of key $\mathbb{K}_2$, and we want to get $\mathbb{S}_0$ from $\mathbb{S}_2$. We need to store $\mathbb{S}_1$ in a new table $T_1$ with the index of $\mathbb{K}_1$, then look up $T_1$ with the index of $\mathbb{K}_2$, finally look up $T_0$.

Sometimes, the relations between $\mathbb{K}_0$ and $\mathbb{K}_1$ are more complicated and we cannot guess all the values of $\mathbb{S}_1$ at the same time (can guess parts of $\mathbb{S}_0$ and $\mathbb{S}_1$ at the same time). Let $\mathbb{K}_0 = f(\mathbb{K}_1)$, $\mathbb{W}_0 \oplus \mathbb{K}_0 = \mathbb{X}_0$ and $\mathbb{W}_1 \oplus \mathbb{K}_1 = \mathbb{X}_1$ ($f$ is a linear function and $(\mathbb{W}_i, \mathbb{X}_i) \subseteq \mathbb{S}_i$), then some states can be stored in a table with the index of $\chi = \mathbb{X}_0 \oplus f(\mathbb{W}_1)$. We can look up this table with the index of $\chi' = \mathbb{W}_0 \oplus f(\mathbb{X}_1)$ to get these states. We call this technique **state-bridge technique**. The similar technique is also used in [DP15] to attack Prince, but ours is more complicate than [DP15] since we use the relations of different nibbles in one state and the relations of different nibbles in states of different rounds as the indexes and contents of the tables.

To improve the time complexity using key relations, we need to build some tables. In this paper, we use figures to show the constructions of these tables. Take Fig. 7 as an example, by guessing $y_6[12]$ and $y_5[2,8,13]$ (nibbles in gray shadow), $x_6[12]$ and $w_5[12]$ can be deduced, then $rk_5[12]$ can be deduced. We store $y_5[2,8,13]$ (nibbles in red) with the index of $y_6[12]$ and $rk_5[12]$ (nibbles

in blue). Since we guess 4 nibbles and take 2 nibbles as index, there are $2^8$ values for each index, the size of this table is $2^{16}$. Sometimes, we add the reason why some nibbles can be deduced above the arrow.

# 3 Meet-in-the-Middle Attack on 10-Round Midori64

In this section, we first propose a 6-round meet-in-the-middle distinguisher with the differential enumeration and key-dependent sieve techniques on Midori64. Then we apply this distinguisher to 10-round Midori64 by adding one round at the beginning and three rounds at the end.

## 3.1 6-Round Distinguisher on Midori64

Since $w_6[9] = z_6[8] \oplus z_6[10] \oplus z_6[11]$ and $w_6[10] = z_6[8] \oplus z_6[9] \oplus z_6[11]$, we have $w_6[9] \oplus w_6[10] = z_6[9] \oplus z_6[10]$. Let $e_{in} = z_6[9] \oplus z_6[10]$ and $e_{out} = x_7[9] \oplus x_7[10]$, then $e_{out} = e_{in} \oplus rk_6[9] \oplus rk_6[10]$, the 6-round distinguisher on Midori64 is based on the proposition below.

**Proposition 6.** *Let* $\{w_0^0, w_0^1, \cdots, w_0^{255}\}$ *be a 2-*$\delta$*-set where* $w_0[5]$ *and* $w_0[10]$ *are the active nibbles. Consider the encryption of the first 33 values* $(w_0^0, w_0^1, \cdots, w_0^{32})$ *of the 2-*$\delta$*-set through 6-round Midori64, in the case of that a message of the 2-*$\delta$*-set belongs to a pair which conforms to the truncated differential trail outlined in Fig. 6(a), then the corresponding 128-bit ordered sequence* $(e_{out}^1 \oplus e_{out}^0, e_{out}^2 \oplus e_{out}^0, \cdots, e_{out}^{32} \oplus e_{out}^0)$ *only takes about* $2^{104}$ *values (out of the* $2^{128}$ *theoretical values).*

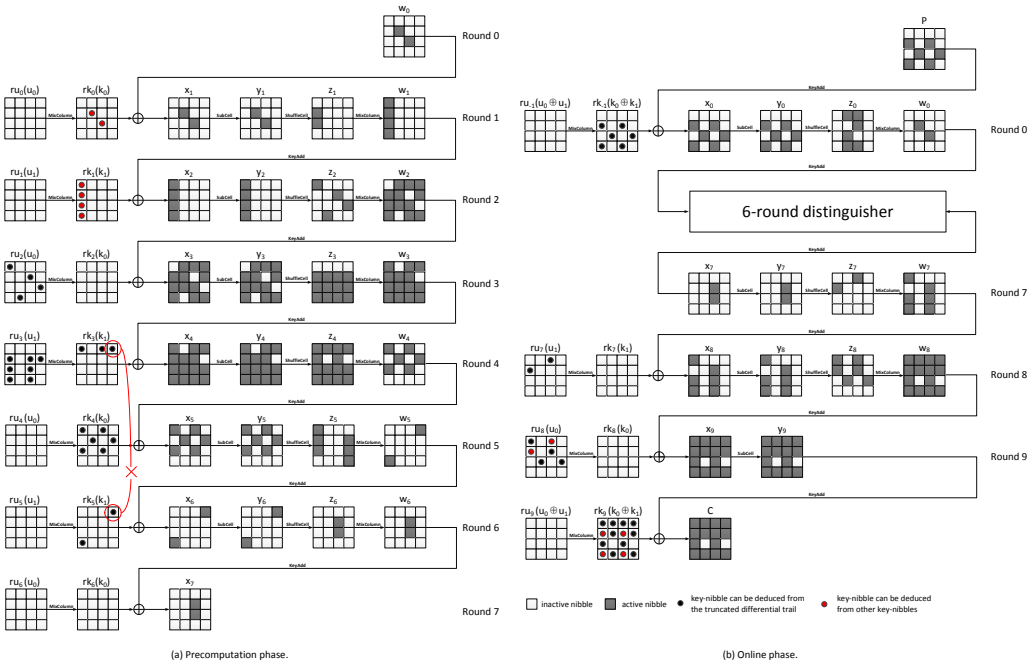



**Figure 6:** The attack on 10-round Midori64. The 6-round distinguisher is shown in (a), the online phase is shown in (b).

*Proof.* As shown in Fig. 6(a), for the encryption of the first 33 values of the 2-$\delta$*-set*, the output

sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ is determined by the 42 nibble-parameters:

$$\begin{aligned} &w_0[5,10]||x_1[5,10]||x_2[0,1,2,3]||x_3 - x_3[0,7,9,14]|| \\ &x_4 - x_4[4,13]||rk_4[0,2,5,8,10,13]||rk_5[3,12] \end{aligned} \tag{3}$$

At round 1, since $\Delta x^m_1[5,10] = \Delta w^m_0[5,10]$ $(0 < m \leq 32)$, we can get $z_1[1,2]$ by the knowledge of $x_1[5,10]$. Since the ShuffleCell, MixColumn and KeyAdd operations are linear, $\Delta x^m_2[0,1,2,3]$ can be deduced. Similarly, $\Delta y^m_2[0,1,2,3]$ can be deduced by the knowledge of $x_2[0,1,2,3]$, $\Delta y^m_3 - \Delta y^m_3[0,7,9,14]$ can be deduced by the knowledge of $x_3 - x_3[0,7,9,14]$, $\Delta y^m_4 - \Delta y^m_4[4,13]$ can be deduced by the knowledge of $x_4 - x_4[4,13]$, $\Delta y^m_5[0,2,5,8,10,13]$ can be deduced by the knowledge of $rk_4[0,2,5,8,10,13]$, and $\Delta z^m_6[9,10]$ can be deduced by the knowledge of $rk_5[3,12]$. Then we get the value of $e^m_{in} \oplus e^0_{in}$. Since $e^m_{out} \oplus e^0_{out} = e^m_{in} \oplus e^0_{in}$, we can get $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$.

However, if a pair of messages conforms to the truncated differential trail outlined in Fig. 6(a), the above 42 nibble-parameters are determined by the 27 nibble-parameters:

$$\Delta z_1[1,2]||x_2[0,1,2,3]||x_3 - x_3[0,7,9,14]||y_5[0,2,5,8,10,13]||y_6[3,12]||\Delta z_6[9] \tag{4}$$

Since $\Delta z_1[1,2]$ is known, we can get $\Delta x_2[0,1,2,3]$. Since $\Delta y_2[0,1,2,3]$ can be deduced by the knowledge of $x_2[0,1,2,3]$, we can get $\Delta x_3 - \Delta x_3[0,7,9,14]$. Also $\Delta x_4 - \Delta x_4[4,13]$ can be deduced by the knowledge of $x_3 - x_3[0,7,9,14]$. For the backward direction, since $\Delta w_6[8] = \Delta z_6[9] \oplus \Delta z_6[10] \oplus \Delta z_6[11]$, $\Delta z_6[11] = 0$ and $\Delta w_6[8] = 0$, we can get that $\Delta z_6[9] = \Delta z_6[10]$. For the same reason as the forward direction, $\Delta y_4 - \Delta y_4[4,13]$ can be deduced by the knowledge of $y_5[0,2,5,8,10,13]||y_6[3,12]||\Delta z_6[9]$. According to Proposition 1, we get one value of intermediate state $x_4 - x_4[4,13]$ in average for the fixed difference $\Delta x_4 - \Delta x_4[4,13]||\Delta y_4 - \Delta y_4[4,13]$. Apparently, $ru_2[0,7,9,14]||rk_4[0,2,5,8,10,13]||rk_5[3,12]$ is also deduced for every 27 nibble-parameters. Since $z_3[13,14,15]$ is known, $w_3[12]$ can be deduced. Then $rk_3[12]$ can be deduced for the reason that $rk_3[12] = x_4[12] \oplus w_3[12]$. According to the key-schedule of Midori64, $rk_3[12]$ and $rk_5[12]$ are affected by the same nibble of $k_1$. By the key-dependent sieve technique, there are $2^{104}$ possible values for the 27 nibble-parameters.

Since $z_3[1,2,3]$ and $x_4[0,1,2,3]$ are known, $ru_3[1,2,3]$ can be deduced. According to the key-schedule, $rk_3[3]$ can be deduced by the knowledge of $rk_5[3]$. Since $rk_3[3] = ru_3[0] \oplus ru_3[1] \oplus ru_3[2]$, $ru_3[0]$ can be deduced. Then $rk_3[0,1,2,3]$ can be deduced from $ru_3[0,1,2,3]$. After that, $rk_1[0,1,2,3]$ can be deduced. We can also deduce $rk_0[5,10]$ from $rk_4[5,10]$. Therefore, $w_0[5,10]$ and $x_1[5,10]$ can be deduced from $x_2[0,1,2,3]$.

So the 42 nibble-parameters (3) are determined by 27 nibble-parameters (4), i.e., the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ can take about $2^{104}$ values (since $rk_3[3]$ can be deduced by the knowledge of $rk_5[3]$).

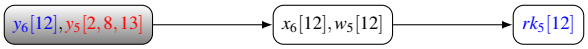
□

## 3.2 Attack on 10-Round Midori64

The attack is made up of two phases: precomputation phase and online phase.

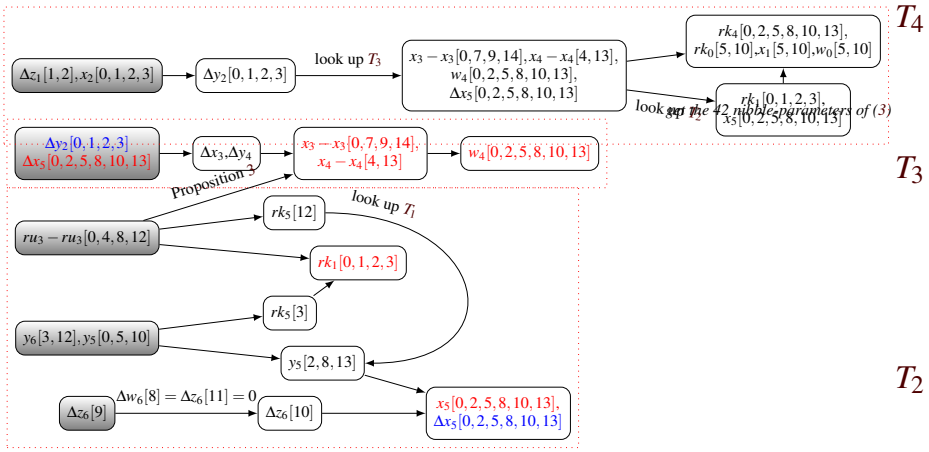
**Precomputation phase:** In the precomputation phase, we need to build a table $T_4$ that contains all the sequences $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ described in Proposition 6. To use the key-dependent sieve technique to improve the complexity, we need to build three more tables $T_1$, $T_2$ and $T_3$. To use the key relation of $rk_3[12]$ and $rk_5[12]$, some nibbles can be stored in a table $T_1$ with the index of $rk_5[12]$. $T_2$ and $T_3$ are built to perform the inbound phases from $\Delta y_1$ to $\Delta x_3$ and $\Delta z_6$ to $\Delta y_4$, then perform the outbound phases to deduce the values step by step. When the values in $T_1$ are needed at the construction of $T_2$, we can look up $T_1$ with the index of $rk_3[12]$. Combining $T_2$, $T_3$ and some extra guessing nibbles, the precomputation table $T_4$ can be deduced.

To use the key-bridging technique in the online phase, we need to build four more tables $T^0_5$, $T^2_5$, $T_6$ and $T_7$. $T^i_5$ $(i = 0,2)$ and $T_6$ are built to use the key-bridges $rk_9[1,3,9,11] = rk_{-1}[1,3,9,11]$ and $(ru_7[1,8], ru_8[1,8]) \Leftrightarrow ru_9[1,8]$, respectively. $T^i_5$ $(i = 0,2)$ connect Round 0 before the distinguisher




■: guessed-nibbles; $x_i[j]$: content of the table; $y_i[j]$: index of the table.

**Figure 7:** Construction of Table $T_1$ on the 10-round attack. There are about $2^8$ values for each index in average.




■: guessed-nibbles; $x_i[j]$: content of the table; $y_i[j]$: index of the table.

**Figure 8:** Constructions of Table $T_2$, Table $T_3$ and Table $T_4$ on the 10-round attack. For $T_2$, there are about $2^8$ values for each index in average. For $T_3$, there are about $2^{24}$ values for each index in average.

and Rounds 8, 9 after the distinguisher, and $T_6$ connects Rounds 7, 8 and 9. When the values in these tables need to be deduced in the online phase, we can look up these tables with the index of sub-key nibbles or state nibble relations. $T_7$ is built to connect Round 8 and $\Delta e_{out}$ of Round 7.

We show the detailed process of the precomputation phase in Appendix A.

1. Table $T_1$ is built to use the key-dependent sieve technique that $rk_3[12]$ and $rk_5[12]$ are affected by the same nibble of $k_1$ as shown in Proposition 6. As Section 2.5 shows, this relation cannot be used directly to improve the complexity, so $y_5[2,8,13]$ is stored in $T_1$ with the index of $rk_5[12]$. When $y_5[2,8,13]$ is needed to use in the construction of Table $T_2$, we can look up $T_1$ with the index of $rk_3[12]$ (deduced from $ru_3 - ru_3[0,4,8,12]$). The construction of this table is shown in Fig. 7.

2. For each 48-bit value $ru_3 - ru_3[0,4,8,12]$, we can build two tables $T_2$ and $T_3$, then use these tables to get the 42 nibble-parameters (3). From the proof of Proposition 6, we need to perform the inbound phases from $\Delta y_1$ to $\Delta x_3$ and $\Delta z_6$ to $\Delta y_4$, then perform the outbound phases to deduce the values. To use the key-dependent sieve technique to improve the time complexity, we deal with these phases step by step. Table $T_2$ is built to perform the inbound phase from $\Delta z_6$ to $\Delta x_5$ in the decryption direction as shown in the proof of Proposition 6 and Fig. 6(a). Table $T_3$ is built to perform the inbound phases from $\Delta y_2$ to $\Delta x_3$ and $\Delta x_5$ to $\Delta y_4$, then perform the outbound phases to get the values of $x_3$ and $y_4$ using Proposition 3. Then we can use the 42 nibble-parameters to compute the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$, and store it along with a 16-bit value $ru_2[0,9,14]||ru_3[1]$ in the precomputation table $T_4$ as Proposition 6. The constructions of these tables are shown in Fig. 8.

3. Table $T_5^0$ (resp. $T_5^2$) is built to use the relation that $rk_9[1,3] = rk_{-1}[1,3]$ (resp. $rk_9[9,11] = rk_{-1}[9,11]$) in the online phase. We cannot use this key-bridge to improve the complexity

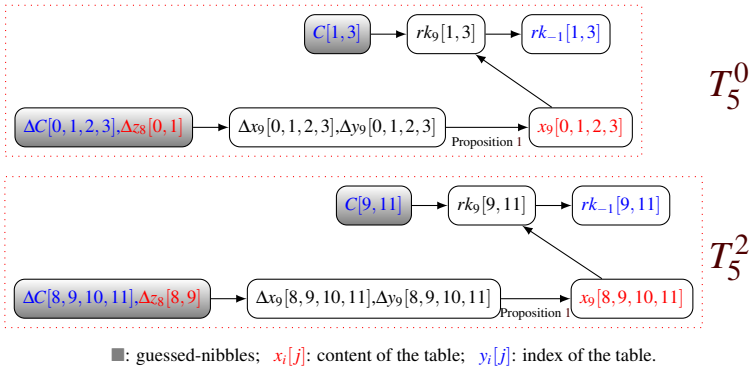



**Figure 9:** Constructions of Table $T_5^0$ and Table $T_5^2$ on the 10-round attack. There is one value for each index in average.

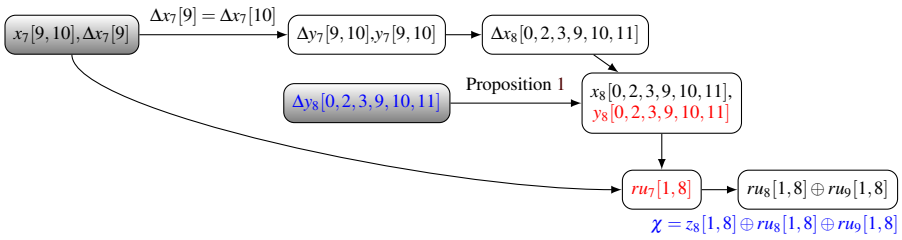



**Figure 10:** Construction of Table $T_6$ on the 10-round attack. There are $2^4$ values for each index in average.

directly, so $T_5^0$ (resp. $T_5^2$) is built. After deducing $rk_{-1}[1,3]$ (resp. $rk_9[1,3]$) from the first round in the online phase, instead of guessing $x_9[0,1,2,3]$ and $\Delta z_8[0,1]$ (resp. $x_9[8,9,10,11]$ and $\Delta z_8[8,9]$), we can look up this table with the index of $rk_{-1}[1,3]$ (resp. $rk_{-1}[9,11]$). The constructions of these tables are shown in Fig. 9.

4. Table $T_6$ is built to use the relation $(ru_7[1,8], ru_8[1,8]) \Leftrightarrow ru_9[1,8]$. This key-bridge cannot be used directly and this relation is more complicated than others, so state-bridge technique is used to improve the complexity. By guessing $x_7[9,10]$, $\Delta x_7[9]$ and $\Delta y_8[0,2,3,9,10,11]$, $z_8[1,8]$ and $ru_7[1,8]$ can be deduced, then $\chi = z_8[1,8] \oplus ru_8[1,8] \oplus ru_9[1,8]$ can be deduced. Store $y_8[0,2,3,9,10,11]$ and $ru_7[1,8]$ in Table $T_6$ with the index of $\chi$. Since $z_8[1,8] \oplus ru_8[1,8] = \overline{w_8}[1,8]$, we can deduce $\chi' = ru_9[1,8] \oplus \overline{w_8}[1,8]$ in the online phase, and look up $T_6$ to get $y_8[0,2,3,9,10,11]$ and $ru_7[1,8]$ with the index of $\chi'$. The construction of $T_6$ is shown in Fig. 10.

5. Table $T_7$ is built to get $e_{out}$ from $\overline{w_8}[0,1,6,8,9,14]$ in the decryption direction. The construction of $T_7$ is shown in Fig. 11.

**Online phase:** In the online phase of the attack, we first find at least one pair which satisfies the truncated differential trail in Fig. 6(a). To find the right pair, instead of guessing the sub-keys and checking whether this pair satisfies the truncated differential trail, we deduce the sub-keys which make it satisfy the truncated differential trail for each pair. Then we identify the 2-$\delta$-set, calculate the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ and check whether it belongs to Table $T_4$. Finally, we use $ru_2[0,9,14]||ru_3[1]$ to filter the remaining keys and retrieve the correct key.

1. **Phase A – Detecting the right pair.**

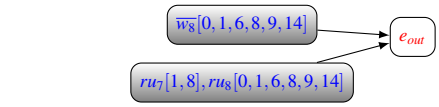



**Figure 11:** Construction of Table $T_7$ on the 10-round attack.

(a) Define a structure of $2^{24}$ plaintexts where $P[1,3,6,9,11,14]$ takes all the possible values, and the remaining 10 nibbles are fixed to some constants. Hence, we can generate $2^{24} \times (2^{24}-1)/2 \approx 2^{47}$ pairs satisfying the plaintext difference.

(b) Choose $2^{29}$ structures to get about $2^{29+47} = 2^{76}$ pairs. As shown in Fig. 6(b), the probability to get the truncated differential trail in the forward and backward directions is $2^{(2-6+1-16)\times 4} = 2^{-76}$ [2], then about 1 pair follows the truncated differential trail for each guess of the key.

Among the $2^{76}$ pairs, we expect about $2^{76-8} = 2^{68}$ pairs to verify that $\Delta C[6,14] = 0$.

2. **Phase B – Checking the $\delta$-set.**

For each of the $2^{68}$ remaining pairs, we do the following sub-steps:

(a) Guess $\Delta w_0[5,10]$, and deduce $\Delta y_0[1,3,6,9,11,14]$. According to Proposition 1, $x_0[1,3,6,9,11,14]$ can be deduced from $\Delta y_0[1,3,6,9,11,14]$ and $\Delta P[1,3,6,9,11,14]$. Then $rk_{-1}[1,3,6,9,11,14]$ can be deduced.

(b) For each of the $2^8$ deduced sub-keys in (a), encrypt the plaintext pairs and get the value $z_0[4,6,7,8,9,11]$. Change the value of $w_0[5,10]$ to be $(0,1,\cdots,32)$ and compute their corresponding plaintexts $(P^0,P^1,\cdots,P^{32})$, then get the corresponding ciphertexts.

(c) For each of the deduced $rk_{-1}[1,3,6,9,11,14]$, compute $rk_9[1,3]$ (resp. $rk_9[9,11]$). Look up Table $T_5^0$ (resp. $T_5^2$) to get about one value $x_9[0,1,2,3]||\Delta z_8[0,1]$ (resp. $x_9[8,9,10,11]||\Delta z_8[8,9]$) with the index of $rk_{-1}[1,3]||\Delta C[0,1,2,3]||C[1,3]$ (resp. $rk_{-1}[9,11]||\Delta C[8,9,10,11]||C[9,11]$). Deduce $rk_9[0,2]$ (resp. $rk_9[8,10]$) from the ciphertext.

(d) Guess $\Delta z_8[6,14]$, and deduce $\Delta x_9[4,5,7,12,13,15]$. Then $rk_9[4,5,7,12,13,15]$ and $x_9[4,5,7,12,13,15]$ can be deduced. Deduce $ru_9[1,8]$ from $rk_9[0,2,3,9,10,11]$, and deduce $\overline{w_8}[1,8]$ from $x_9[0,2,3,9,10,11]$. Then we can get $\chi' = ru_9[1,8] \oplus \overline{w_8}[1,8]$, i.e., $\chi' = z_8[1,8] \oplus ru_8[1,8] \oplus ru_9[1,8]$. Look up Table $T_6$ to get about $2^4$ values $y_8[0,2,3,9,10,11]||ru_7[1,8]$ with the index of $\chi'||\Delta z_8[0,1,6,8,9,14]$. Deduce $ru_8[0,1,6,8,9,14]$ from $y_8[0,2,3,9,10,11]$ and $x_9 - x_9[6,14]$.

(e) For about $2^{20}$ values $rk_{-1}[1,3,6,9,11,14]||rk_9 - rk_9[6,14]||ru_8[0,1,6,8,9,14]||ru_7[1,8]$ we have got, decrypt the corresponding ciphertexts we made in (b) and get $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ using Table $T_7$. Check whether it lies in the precomputation table $T_4$. If not, try another one. If so, we check whether $ru_2[0,9,14]||ru_3[1]$ matches $ru_8[0,9,14]||ru_7[1]$. So the probability for a wrong sub-key to pass this test is $2^{-24-16} = 2^{-40}$.

3. **Exhaustively search the rest of the key:** In the end, there are about $2^{22\times 4-40} = 2^{44}$ sub-keys remaining. Then exhaustively search for the $2^{44}$ sub-keys and 10 unknown key-nibbles to recover the master key.

**Complexity analysis.** In the precomputation phase, in order to construct $T_4$, we need to perform

[2] $P$ is active in 6 nibbles, $w_0$ is active in 2 nibbles, $C$ is active in 16 nibbles (reduce to 14 afterward) and $x_7$ is active in 1 nibble relation.

$2^{104}$ partial encryptions on 33 messages. The time complexity of this phase is about $2^{104+5-2} = 2^{107}$ 10-round Midori64 encryptions, the memory complexity is about $2^{104+7.2-6} = 2^{105.2}$ 64-bit blocks. In the online phase, we need to perform $2^{20+68}$ partial encryptions on 33 messages. The time complexity of this phase is about $2^{88+5-3} = 2^{90}$ 10-round Midori64 encryptions, the data complexity is $2^{24+29} = 2^{53}$ chosen-plaintexts and the memory complexity is $2^{53}$ 64-bit blocks. With data/time/memory tradeoff, the adversary only need to precompute a fraction of $2^{-8.5}$ of possible sequences, then the time complexity becomes $2^{107-8.5} = 2^{98.5}$, the memory complexity becomes $2^{96.7}$ 64-bit blocks. But in the online phase, the adversary will repeat the attack $2^{8.5}$ times to offset the probability of the failure. So the data complexity increases to $2^{61.5}$ chosen-plaintexts, and the time complexity increases to $2^{90+8.5} = 2^{98.5}$. Otherwise, we can divide the whole attack into series of weak-key attacks according to the relations between the sub-keys in the online phase and the precomputation phase as Li et al. presented in [LJW14]. Using the relation of $ru_3[1]$ and $ru_7[1]$, the attack can be divided into $2^4$ weak-key attacks, i.e., the precomputation table can be divided into $2^4$ sub-tables with the index of $ru_3[1]$ and we only need to check the sub-table according to the value of $ru_7[1]$ in the online phase. The memory complexity can be reduced by a factor of $2^4$. In total, the time complexity of this attack is $2^{99.5}$ 10-round Midori64 encryptions, the data complexity is $2^{61.5}$ chosen-plaintexts and the memory complexity is $2^{92.7}$ 64-bit blocks.

# 4 Meet-in-the-Middle Attack on 11-Round Midori64

Based on the 10-round attack, we can add one round at the end to mount an 11-round attack on Midori64.

The precomputation phase is almost the same as the 10-round attack except the tables for online phase, i.e., we use new tables $T_5^i$ $(i = 0, \cdots, 3)$ to replace $T_5^i$ $(i = 0, 2)$ of the 10-round attack and a new table $T_6$ to replace $T_6$ of the 10-round attack. We also build two new tables $T_8^i$ $(i = 0, 1)$ to get $\overline{w_8}$ from $\overline{w_9}$ and $ru_9$ for online phase. $T_5^i$ $(i = 0, \cdots, 3)$ and $T_6$ are built to use the key-bridges $rk_{10}[1,3,6,9,11,14] = rk_{-1}[1,3,6,9,11,14]$, $(ru_8[0,1,6,8,9,14], ru_9[0,1,6,8,9,14]) \Leftrightarrow ru_{10}[0,1,6,8,9,14]$ and $(ru_7[1,8], ru_8[1,8]) \Leftrightarrow ru_{10}[1,8]$, respectively. $T_5^i$ $(i = 0, \cdots, 3)$ connect Round 0 before the distinguisher and Rounds 9, 10 after the distinguisher, and $T_6$ connects Rounds 8, 9 and 10. When the values in these tables need to be deduced in the online phase, we can look up these tables with the index of sub-key nibbles or state nibble relations.

We show the detailed process of the precomputation phase in Appendix B.

1. Tables $T_5^i$ $(i = 0, \cdots, 3)$ are built to use the relation that $rk_{10}[1,3,6,9,11,14] = rk_{-1}[1,3,6,9,11,14]$ in the online phase, and Table $T_5^i$ is for Column $i$. We cannot use these key-bridges to improve the complexity directly, so $T_5^i$ $(i = 0, \cdots, 3)$ are built. After deducing $rk_{-1}[1,3,6,9,11,14]$ from the first round in the online phase, instead of guessing $x_{10}$ and $\Delta z_9 - \Delta z_9[4,11]$, we can look up these tables with the index of $rk_{-1}[1,3,6,9,11,14]$. The constructions of these tables are shown in Fig. 12.

2. Table $T_6$ is built to use the key relations $(ru_8[0,1,6,8,9,14], ru_9[0,1,6,8,9,14]) \Leftrightarrow ru_{10}[0,1,6,8,9,14]$ and $(ru_7[1,8], ru_8[1,8]) \Leftrightarrow ru_{10}[1,8]$ as shown in Fig. 15. These key-bridges cannot be used directly and these relations are more complicated than others, so state-bridge technique is used to improve the complexity. By guessing $\Delta y_8[0,2,3,9,10,11]$, $\Delta z_9 - \Delta z_9[4,11]$, $x_7[9,10]$ and $\Delta x_7[9]$, $y_9 - y_9[6,14]$, $ru_8[0,1,6,8,9,14]$ and $ru_7[1,8]$ can be deduced, then $\chi = z_9[0,1,6,8,9,14] \oplus ru_{10}[0,1,6,8,9,14] \oplus ru_9[0,1,6,8,9,14]$ and $ru_{10}[1,8]$ can be deduced. Store $y_9 - y_9[6,14]$, $ru_8[0,1,6,8,9,14]$ and $ru_7[1,8]$ in Table $T_6$ with the index of $\chi$ and $ru_{10}[1,8]$. Since $z_9[0,1,6,8,9,14] \oplus ru_9[0,1,6,8,9,14] = \overline{w_9}[0,1,6,8,9,14]$, we can deduce $\chi' = ru_{10}[0,1,6,8,9,14] \oplus \overline{w_9}[0,1,6,8,9,14]$ in the online phase, and look up $T_6$ to get $y_9 - y_9[6,14]$, $ru_8[0,1,6,8,9,14]$ and $ru_7[1,8]$ with the index of $ru_{10}[1,8]$ and $\chi'$. The construction of Table $T_6$ is shown in Fig. 13.

3. Tables $T_8^i$ $(i = 0, 1)$ are built to get $\overline{w_8}$ from $\overline{w_9}$ in the decryption direction. The constructions

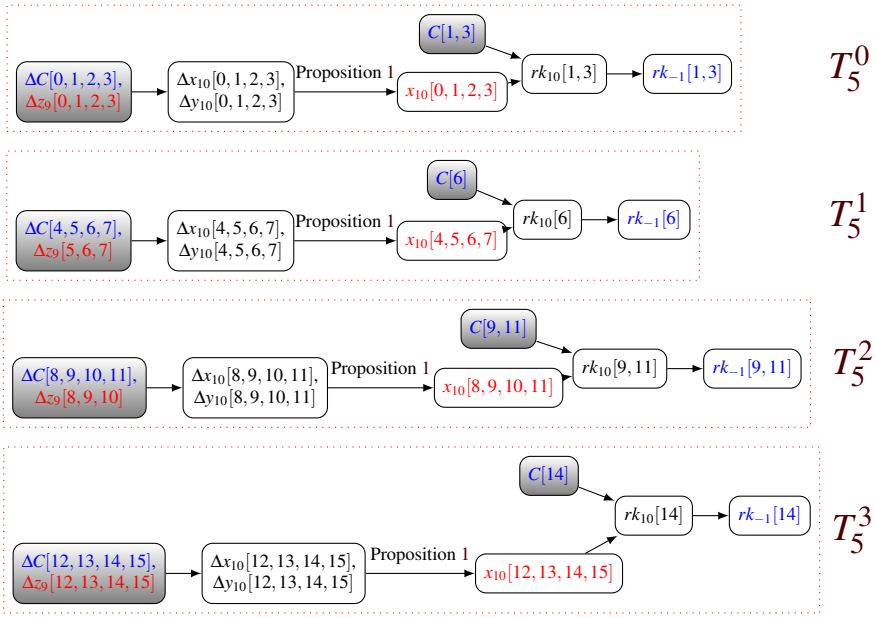



**Figure 12:** Constructions of Tables $T_5^i$ ($i = 0, \cdots, 3$) on the 11-round attack. There are $2^8$, $2^8$, $2^4$, $2^{12}$ values for each index in each table, respectively.

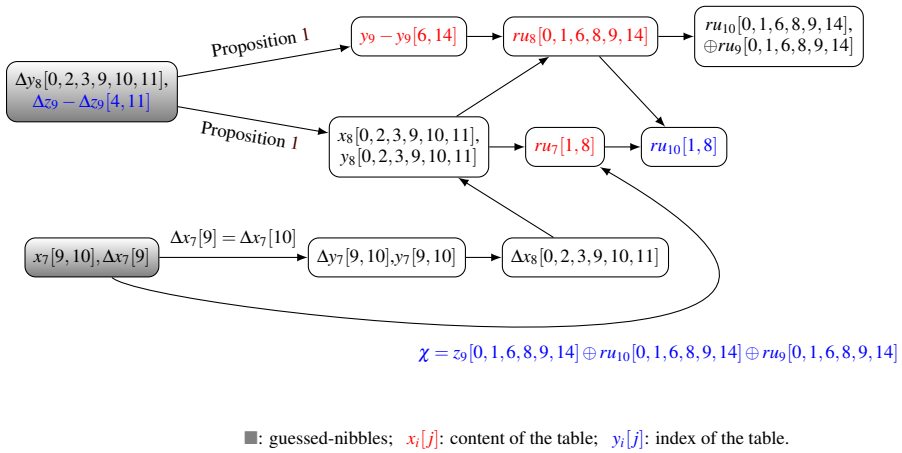



**Figure 13:** Construction of Table $T_6$ on the 11-round attack. There are $2^4$ values for each index in average.

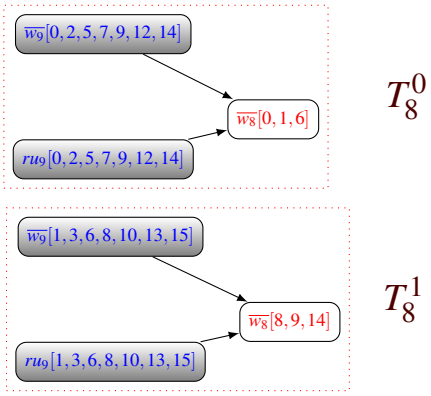



**Figure 14:** Constructions of Table $T_8^0$ and Table $T_8^1$ on the 11-round attack.

of these tables are shown in Fig. 14.

The online phase (Fig. 15) is different from the 10-round attack at Step 2(c), Step 2(d) and Step 2(e). And since all nibbles of ciphertext are active, we should try all the $2^{76}$ pairs.

1. At Step 2(c), for each of the deduced $rk_{-1}[1,3]$, compute $rk_{10}[1,3]$. Look up Table $T_5^0$ to get about $2^8$ values $x_{10}[0,1,2,3]||\Delta z_9[0,1,2,3]$ with the index of $rk_{-1}[1,3]||\Delta C[0,1,2,3]||C[1,3]$. Deduce $rk_{10}[0,2]$ from the ciphertext. Do the same things to Column $i$ and $T_5^i$ $(i=1,2,3)$, and deduce about $2^{32}$ values $rk_{10}||x_{10}||\Delta z_9 - \Delta z_9[4,11]$.

2. At Step 2(d), deduce $ru_{10}[0,1,6,8,9,14]$ from $rk_{10}$, and deduce $\overline{w_9}[0,1,6,8,9,14]$ from $x_{10}$. Then we can get $\chi' = ru_{10}[0,1,6,8,9,14] \oplus \overline{w_9}[0,1,6,8,9,14]$, i.e., $\chi' = z_9[0,1,6,8,9,14] \oplus ru_9[0,1,6,8,9,14] \oplus ru_{10}[0,1,6,8,9,14]$. Look up Table $T_6$ to get about $2^4$ values $y_9 - y_9[6,14]||ru_7[1,8]||ru_8[0,1,6,8,9,14]$ with the index of $ru_{10}[1,8]||\chi'||\Delta z_9 - \Delta z_9[4,11]$. Deduce $ru_9 - ru_9[4,11]$ from $y_9 - y_9[6,14]$ and $x_{10}$.

3. At Step 2(e), for about $2^{44}$ values $rk_{-1}[1,3,6,9,11,14]||rk_{10}||ru_9 - ru_9[4,11]||ru_8[0,1,6,8,9,14]||ru_7[1,8]$ we have got, decrypt the corresponding ciphertexts we made in (b) and get $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ using $T_7$, $T_8^0$ and $T_8^1$. Check whether it lies in the precomputation table $T_4$. If not, try another one. If so, we check whether $ru_2[0,9,14]||ru_3[1]$ matches $ru_8[0,9,14]||ru_7[1]$. So the probability for a wrong sub-key to pass this test is $2^{-24-16} = 2^{-40}$.

**Complexity analysis.** The time complexity of the precomputation phase is the same as the 10-round attack. In the online phase, we need to perform $2^{44+76}$ partial encryptions on 33 messages. The time complexity of this phase is about $2^{120+5-3} = 2^{122}$ 11-round Midori64 encryptions, the data complexity is $2^{24+29} = 2^{53}$ chosen-plaintexts and the memory complexity is $2^{53}$ 64-bit blocks. Otherwise, we can divide the whole attack into series of weak-key attacks according to the relations between the sub-keys in the online phase and the precomputation phase as Li et al. presented in [LJW14]. Using the relation of $ru_2[0,9,14]||ru_3[1]$ (precomputation phase) and $ru_8[0,9,14]||ru_7[1]$ (online phase), the attack can be divided into $2^{16}$ weak-key attacks. The memory complexity can be reduced by a factor of $2^{16}$. In total, the time complexity of this attack is $2^{122}$ 11-round Midori64 encryptions, the data complexity is $2^{53}$ chosen-plaintexts and the memory complexity is $2^{89.2}$ 64-bit blocks.

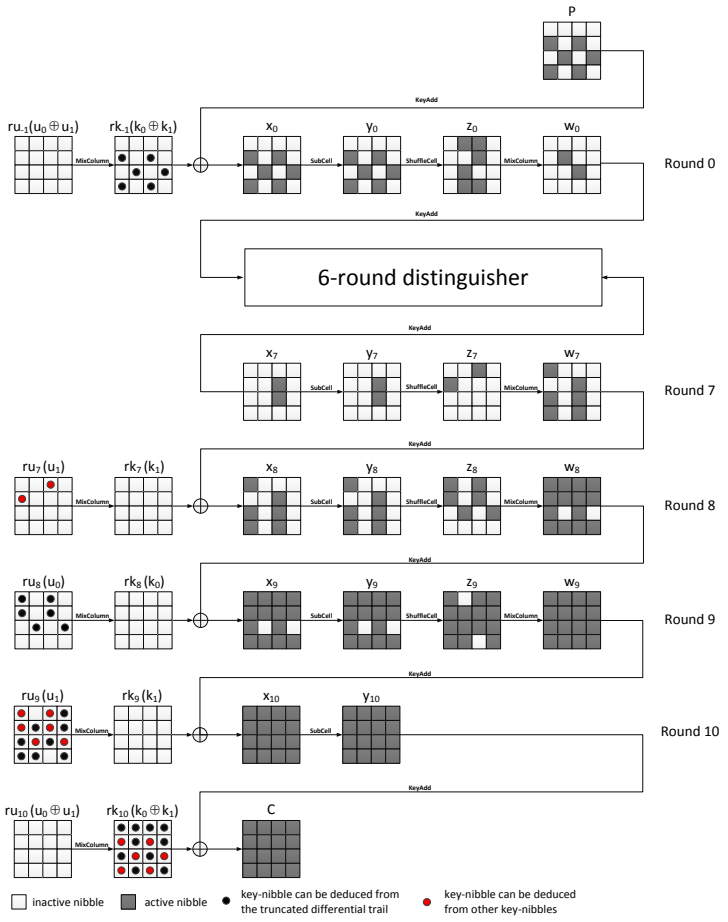



**Figure 15:** Online phase of the attack on 11-round Midori64.

# 5 Meet-in-the-Middle Attack on 12-round Midori64

In this section, we first propose a 7-round meet-in-the-middle distinguisher with the differential enumeration and key-dependent sieve techniques on Midori64. Then we apply this distinguisher to 12-round Midori64 by adding one round at the beginning and four rounds at the end.

## 5.1 7-Round Distinguisher on Midori64

Since $w_7[5] = z_7[4] \oplus z_7[6] \oplus z_7[7]$ and $w_7[6] = z_7[4] \oplus z_7[5] \oplus z_7[7]$, we have $w_7[5] \oplus w_7[6] = z_7[5] \oplus z_7[6]$. Let $e_{in} = z_7[5] \oplus z_7[6]$ and $e_{out} = x_8[5] \oplus x_8[6]$, then $e_{out} = e_{in} \oplus rk_7[5] \oplus rk_7[6]$, the 7-round distinguisher on Midori64 is based on the proposition below.

**Proposition 7.** *Let $\{w_0^0, w_0^1, \cdots, w_0^{255}\}$ be a 2-$\delta$-set where $w_0[5]$ and $w_0[10]$ are the active nibbles. Consider the encryption of the first 33 values $(w_0^0, w_0^1, \cdots, w_0^{32})$ of the 2-$\delta$-set through 7-round Midori64, in the case of that a message of the 2-$\delta$-set belongs to a pair which conforms to the truncated differential trail outlined in Fig. 16(a), then the corresponding 128-bit ordered sequence $(e_{out}^1 \oplus e_{out}^0, e_{out}^2 \oplus e_{out}^0, \cdots, e_{out}^{32} \oplus e_{out}^0)$ only takes about $2^{124}$ values (out of the $2^{128}$ theoretical values).*

*Proof.* As shown in Fig. 16(a), for the encryption of the first 33 values of the 2-$\delta$-*set*, the output

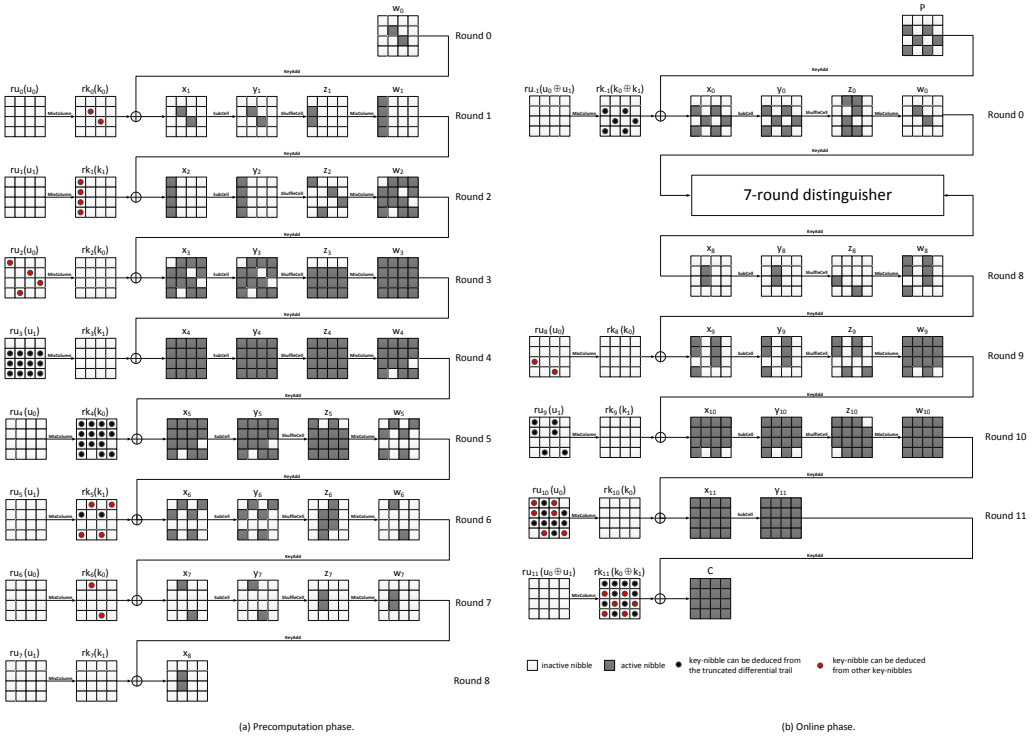



**Figure 16:** The attack on 12-round Midori64. The 7-round distinguisher is shown in (a), the online phase is shown in (b).

sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ is determined by the 58 nibble-parameters:

$$\begin{aligned} &w_0[5,10]||x_1[5,10]||x_2[0,1,2,3]||x_3 - x_3[0,7,9,14]||x_4|| \\ &rk_4 - rk_4[7,14]||rk_5[1,3,4,9,11,12]||rk_6[4,11] \end{aligned} \tag{5}$$

However, if a pair of messages conforms to the truncated differential trail outlined in Fig. 16(a), the above 58 nibble-parameters are determined by the 41 nibble-parameters:

$$\begin{aligned} &\Delta z_1[1,2]||x_2[0,1,2,3]||x_3 - x_3[0,7,9,14]|| \\ &y_5 - y_5[7,14]||y_6[1,3,4,9,11,12]||y_7[4,11]||\Delta z_7[5] \end{aligned} \tag{6}$$

Meanwhile, $ru_2[0,7,9,14]||ru_3 - ru_3[0,4,8,12]||rk_4 - rk_4[7,14]||rk_5[1,3,4,9,11,12]||rk_6[4,11]$ can be determined by the above 41 nibble-parameters. Since $ru_4[0,7,9,14]$ can be deduced from $rk_4 - rk_4[7,14]$, $rk_3[4,12]$ can de deduced from $ru_3[5,6,7,13,14,15]$ and $rk_3[3,11]$ can be deduced from $ru_3[1,2,3,9,10,11]||ru_5[1,9]$, according to the key-schedule of Midori64, $ru_2[0,7,9,14]||rk_3[3,4,11,12]||rk_6[4,11]$[3] and $ru_4[0,7,9,14]||rk_5[3,4,11,12]||rk_4[4,11]$ are affected by the same nibbles of the master key. By the key-dependent sieve technique, there are $2^{124}$ possible values for the 58 nibble-parameters (5).

So the 58 nibble-parameters (5) are determined by 41 nibble-parameters (6), i.e., the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ can take about $2^{124}$ values.

□

[3] $rk_3[3,11]$ can be deduced from $ru_3[1,2,3,9,10,11]$ and $rk_5[1,9]$.

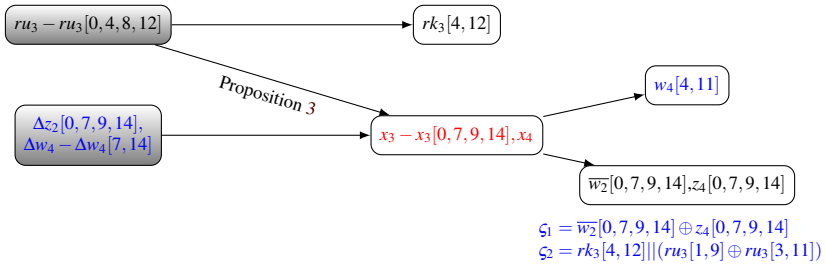



**Figure 17:** Construction of Table $T_1$ on the 12-round attack. There are $2^8$ values for each index in average.

## 5.2 12-Round Attack on Midori64

The attack is made up of two phases: precomputation phase and online phase.

**Precomputation phase:** In the precomputation phase, we need to build a table $T_4$ that contains all the sequences $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ described in Proposition 7. To use the key-dependent sieve technique to save some time, we need to build three more tables $T_1$, $T_2$ and $T_3$. These tables are built to perform the inbound phases from $\Delta y_1$ to $\Delta x_3$ and $\Delta z_7$ to $\Delta y_4$, then perform the outbound phases to deduce the values step by step. $T_1$ is also built to use the key relation of $ru_2[0,7,9,14]||rk_3[3,4,11,12]||rk_6[4,11]$ and $ru_4[0,7,9,14]||rk_5[3,4,11,12]||rk_4[4,11]$. Some nibbles can be stored in the table $T_1$ with the index of a linear relation concerning $ru_2[0,7,9,14]||rk_3[3,4,11,12]||rk_6[4,11]$. After building $T_2$ and $T_3$ and combining all the values in these two tables to construct $T_4$, the index of $T_1$, i.e. a linear relation concerning $ru_4[0,7,9,14]||rk_5[3,4,11,12]||rk_4[4,11]$, can be deduced, then the values in $T_1$ can be deduced. The precomputation table $T_4$ can be deduced with the help of $T_1$, $T_2$ and $T_3$.

We show the detailed process of the precomputation phase in Appendix C.

1. Table $T_1$ is built to perform the inbound phases from $\Delta z_2$ to $\Delta x_3$ and $\Delta w_4$ to $\Delta y_4$, then perform the outbound phases to get the values of $x_3$ and $y_4$ using Proposition 3. Table $T_1$ is also built to use the key relation that $ru_2[0,7,9,14]||rk_3[3,4,11,12]||rk_6[4,11]$ and $ru_4[0,7,9,14]||rk_5[3,4,11,12]||rk_4[4,11]$ are affected by the same nibbles of the master key. The state-bridge technique is used to improve the complexity. By guessing $ru_3 - ru_3[0,4,8,12]$, $\Delta z_2[0,7,9,14]$ and $\Delta w_4 - \Delta w_4[7,14]$, $x_3 - x_3[0,7,9,14]$, $x_4$, $rk_3[4,12]$, $\overline{w_2}[0,7,$
$9,14]$ and $z_4[0,7,9,14]$ can be deduced, then $\varsigma_1 = \overline{w_2}[0,7,9,14] \oplus z_4[0,7,9,14]$ and $\varsigma_2 = rk_3[4,12]||(ru_3[1,9] \oplus ru_3[3,11])$ can be deduced. Store $x_3 - x_3[0,7,9,14]$ and $x_4$ in Table $T_1$ with the index of $\varsigma_1$ and $\varsigma_2$. Let $\varsigma'_1 = \overline{w_2}[0,7,9,14] \oplus z_4[0,7,9,14] = z_2[0,7,9,14] \oplus \overline{w_4}[0,7,9,14]$ and we can deduce $\varsigma_2$ from $T_2$, so we can look up $T_1$ to get $x_3 - x_3[0,7,9,14]$ and $x_4$ with the index of $\varsigma'_1$ and $\varsigma_2$. The construction of Table $T_1$ is shown in Fig. 17.

2. Table $T_2$ and Table $T_3$ are built to perform the inbound phases from $\Delta z_7$ to $\Delta w_4$ and $\Delta z_1$ to $\Delta y_2$, respectively. For each value of $T_2$ and $T_3$, we can get the corresponding index of Table $T_1$. Look up $T_1$ to get all the 58 nibble-parameters of (5) in Proposition 7, then compute the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$, and store it along with a 60-bit value $ru_4[0,1,2,7,8,9,10,11,14]||ru_3[0,1,7,8,9,15]$ in a table $T_4$. The constructions of these tables are shown in Fig. 18.

The online phase and the constructions of Tables $T^i_5$ $(i = 0, \cdots, 3)$, $T_6$, $T_7$, $T^0_8$ and $T^1_8$ are almost the same as the 11-round attack except the positions of nibbles. The process of this phase is shown in Fig. 16(b).

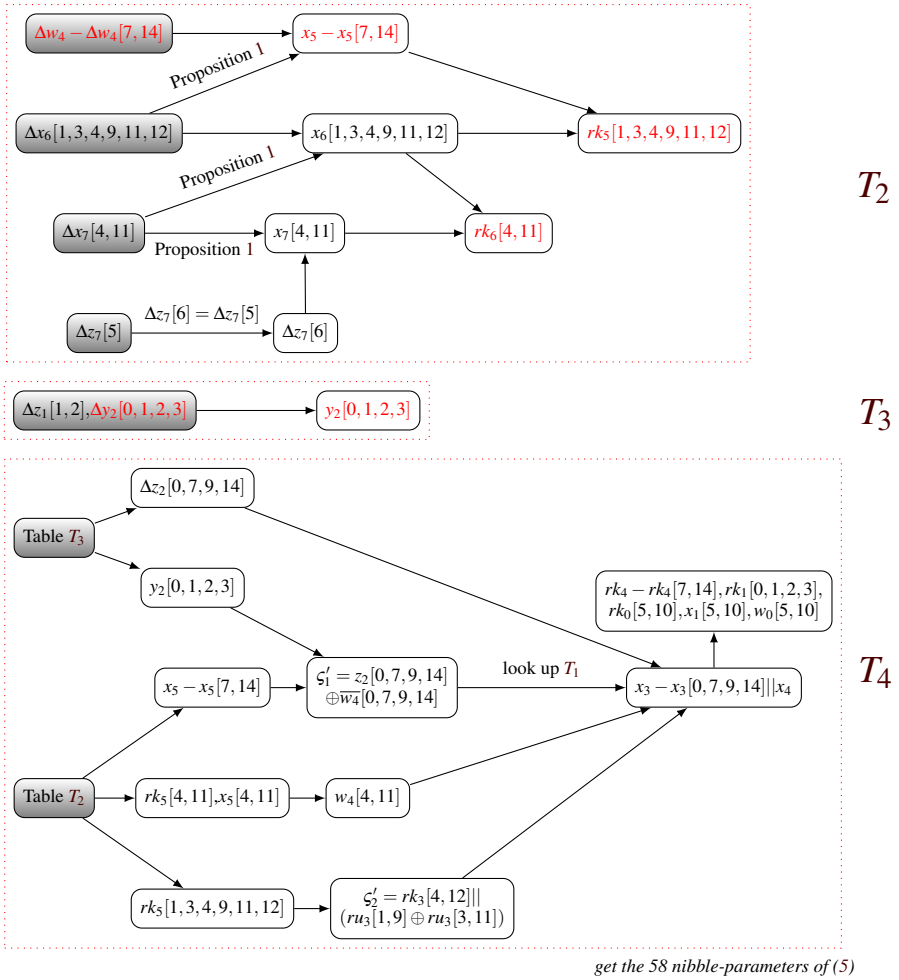



**Figure 18:** Constructions of Table $T_2$, Table $T_3$ and Table $T_4$ on the 12-round attack.

Therefore, the time complexity of the precomputation phase is $2^{124+5-2} = 2^{127}$ 12-round Midori64 encryptions, the memory complexity is $2^{124+7.2-6} = 2^{125.2}$ 64-bit blocks. The time complexity of the online phase is about $2^{120+5-3} = 2^{122}$ 12-round Midori64 encryptions, the data complexity is $2^{24+29} = 2^{53}$ chosen-plaintexts and the memory complexity is $2^{53}$ 64-bit blocks. By data/time/memory tradeoff and weak-key attacks, the time complexity of this attack is about $2^{125.5}$ 12-round Midori64 encryptions, the data complexity is $2^{55.5}$ chosen-plaintexts and the memory complexity is $2^{106}$ 64-bit blocks[4].

# 6 Conclusions and Further Works

In this paper, we discussed the security of Midori64 against meet-in-the-middle attacks. Using the differential enumeration, key-bridging and key-dependent sieve techniques, we proposed a 6-round meet-in-the-middle distinguisher on Midori64. Based on this distinguisher, we added one round at the beginning and three rounds at the end to present a 10-round attack with time complexity of $2^{99.5}$ 10-round Midori64 encryptions, data complexity of $2^{61.5}$ chosen-plaintexts and memory complexity of $2^{92.7}$ 64-bit blocks. After that, by adding one round at the end, we got an 11-round attack with time complexity of $2^{122}$ 11-round Midori64 encryptions, data complexity of $2^{53}$ chosen-plaintexts and memory complexity of $2^{89.2}$ 64-bit blocks. Finally, with a 7-round distinguisher, we got an attack on 12-round Midori64 with time complexity of $2^{125.5}$ 12-round Midori64 encryptions, data complexity of $2^{55.5}$ chosen-plaintexts and memory complexity of $2^{106}$ 64-bit blocks. There are many further works possible: the way to apply this kind of attacks to Midori128, the way to get better attack complexity with meet-in-the-middle method and the security level against other cryptanalytic methods (e.g., impossible differential and zero-correlation linear) for Midori.

# Acknowledgements

We would like to thank the anonymous reviewers for providing valuable comments. The research presented in this paper is supported by the National Basic Research Program of China (No. 2013CB338002) and National Natural Science Foundation of China (No. 61672509 and 61232009).

# References

[BBI+15a] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 411–436, 2015.

[BBI+15b] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy (extended version). *IACR Cryptology ePrint Archive*, 2015:1142, 2015.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 208–225, 2012.

[4] The memory complexity comes from the construction of Table $T_1$

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 450–466, 2007.

[CW16] Zhan Chen and Xiaoyun Wang. Impossible differential cryptanalysis of midori. *IACR Cryptology ePrint Archive*, 2016:535, 2016.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 157–184, 2016.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 371–387, 2013.

[DH77] Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 158–176, 2010.

[DP15] Patrick Derbez and Léo Perrin. Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 190–216, 2015.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DR06] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, pages 78–94, 2006.

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pages 116–126, 2008.

[DS16] Xiaoyang Dong and Yanzhao Shen. Cryptanalysis of reduced-round midori64 block cipher. *IACR Cryptology ePrint Archive*, 2016:676, 2016.

[DTÇB09] Hüseyin Demirci, Ihsan Taskin, Mustafa Çoban, and Adnan Baysal. Improved meet-in-the-middle attacks on AES. In *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, pages 144–156, 2009.

[GJN+15] Jian Guo, Jérémy Jean, Ivica Nikolic, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant subspace attack against full midori64. *IACR Cryptology ePrint Archive*, 2015:1189, 2015.

[GM00] Henri Gilbert and Marine Minier. A collision attack on 7 rounds of rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[LJ16] Rongjia Li and Chenhui Jin. Meet-in-the-middle attacks on 10-round AES-256. *Des. Codes Cryptography*, 80(3):459–471, 2016.

[LJW14] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved single-key attacks on 9-round AES-192/256. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 127–146, 2014.

[LWWZ13] Li Lin, Wenling Wu, Yanfeng Wang, and Lei Zhang. General model of the single-key meet-in-the-middle distinguisher on the word-oriented block cipher. In *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, pages 203–223, 2013.

[SMMK12] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 339–354, 2012.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack -practical attack on full scream, iscream, and midori64. *IACR Cryptology ePrint Archive*, 2016:732, 2016.

[WZ11] Wenling Wu and Lei Zhang. Lblock: A lightweight block cipher. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, pages 327–344, 2011.

# Appendices

In this section, we give the supplementary material about the detailed processes of the precomputation phases on 10-round, 11-round and 12-round Midori64.

## A Precomputation Phase of 10-Round Attack

**Precomputation phase:** In the precomputation phase, we need to build a table that contains all the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$ described in Proposition 6.

1. Guess $y_6[12]||y_5[2,8,13]$, and compute $x_6[12]$ and $w_5[12]$. Deduce $rk_5[12]$ from $x_6[12]||w_5[12]$. Store $y_5[2,8,13]$ in a table $T_1$ with the index of $rk_5[12]||y_6[12]$. There are about $2^8$ values of $y_5[2,8,13]$ for each index in average.

2. For each 48-bit $ru_3 - ru_3[0,4,8,12]$, do the following steps:

   (a) Guess $\Delta z_6[9]$. Since $\Delta w_6[8] = \Delta z_6[11] = 0$, we can deduce $\Delta z_6[10]$. Deduce $rk_5[12]$ from $ru_3[13,14,15]$. Guess $y_6[3,12]||y_5[0,5,10]$, look up Table $T_1$ to get about $2^8$ values of $y_5[2,8,13]$ with the index of $rk_5[12]||y_6[12]$. Then compute $x_5[0,2,5,8,10,13]||\Delta x_5[0,2,5,8,10,13]$. Deduce $rk_5[3]$ from $y_6[3]$ and $y_5[0,5,10]$, then deduce $rk_1[0,1,2,3]$ from $rk_5[3]$ and $ru_3[1,2,3]$. Store $rk_1[0,1,2,3]||x_5[0,2,5,8,10,13]$ in a table $T_2$ with the index of $\Delta x_5[0,2,5,8,10,13]$. There are about $2^8$ values for each index in average.

   (b) For all $2^{40}$ values of $\Delta y_2[0,1,2,3]$ and $\Delta x_5[0,2,5,8,10,13]$, deduce $\Delta x_3$ and $\Delta y_4$. According to Proposition 3, we can get $x_3 - x_3[0,7,9,14]$ and $y_4 - y_4[4,13]$. Then compute $w_4[0,2,5,8,10,13]$, and store $x_3 - x_3[0,7,9,14]||x_4 - x_4[4,13]||w_4[0,2,5,8,10,13]||\Delta x_5[0,2,5,8,10,13]$ in a table $T_3$ with the index of $\Delta y_2[0,1,2,3]$. There are about $2^{24}$ values for each index in average.

   (c) For each $\Delta z_1[1,2]||x_2[0,1,2,3]$, do the following sub-steps:

      i. Deduce $\Delta y_2[0,1,2,3]$ from $\Delta z_1[1,2]$ and $x_2[0,1,2,3]$. Then look up Table $T_3$ to get about $2^{24}$ values $x_3[1,2,3,4,5,6,8,10,11,12,13,15]||x_4 - x_4[4,13]||w_4[0,2,5,8,10,13]||\Delta x_5[0,2,5,8,10,13]$. For each of these values, look up Table $T_2$ to get about $2^8$ values $rk_1[0,1,2,3]||x_5[0,2,5,8,10,13]$. Deduce $rk_4[0,2,5,8,10,13]$ from $x_5[0,2,5,8,10,13]$ and $w_4[0,2,5,8,10,13]$, then deduce $rk_0[5,10]$ from $rk_4[5,10]$. Compute $x_1[5,10]$ from $rk_1[0,1,2,3]$ and $x_2[0,1,2,3]$, then compute $w_0[5,10]$ from $x_1[5,10]$ and $rk_0[5,10]$. Therefore, we get the 42 nibble-parameters (3).

      ii. Compute the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$, and store them along with a 16-bit value $ru_2[0,9,14]||ru_3[1]$ in a table $T_4$.

3. We build two tables $T^0_5$ and $T^2_5$ for online phase. As shown in Fig. 6(b), for Column 0, guess $\Delta C[0,1,2,3]||\Delta z_8[0,1]$, and deduce $\Delta x_9[0,1,2,3]$ and $\Delta y_9[0,1,2,3]$. By Proposition 1, we can deduce $y_9[0,1,2,3]$. Guess $C[1,3]$, $rk_9[1,3]$ can be deduced. One can deduce $rk_{-1}[1,3]$ from $rk_9[1,3]$, and store $x_9[0,1,2,3]||\Delta z_8[0,1]$ in a table $T^0_5$ with the index of $rk_{-1}[1,3]||\Delta C[0,1,2,3]||C[1,3]$. There is one value for each index in average. Similarly, we can get a table $T^2_5$ for Column 2.

4. We build a table $T_6$ for online phase. Guess $x_7[9,10]||\Delta x_7[9]$, one can deduce $\Delta y_7[9,10]$ and $y_7[9,10]$ since $\Delta x_7[9] = \Delta x_7[10]$. Then $\Delta x_8[0,2,3,9,10,11]$ can be deduced. Guess $\Delta y_8[0,2,3,9,10,11]$, then $x_8[0,2,3,9,10,11]$ and $y_8[0,2,3,9,10,11]$ can be deduced by Proposition 1. Deduce $ru_7[1,8]$ from $x_7[9,10]$ and $x_8[0,2,3,9,10,11]$, and deduce $ru_8[1,8] \oplus ru_9[1,8]$ from $ru_7[1,8]$. Let $\chi$ denote $z_8[1,8] \oplus ru_8[1,8] \oplus ru_9[1,8]$. Store $y_8[0,2,3,9,10,11]||ru_7[1,8]$ in a table $T_6$ with the index of $\chi||\Delta z_8[0,1,6,8,9,14]$. There are $2^4$ values for each index in average.

5. We build another table $T_7$ for online phase. For all 36-bit sub-keys $ru_7[1,8]||ru_8[0,1,6,8,9,14]$, decrypt all 24-bit values $\overline{w_8}[0,1,6,8,9,14]$ and obtain the values $e_{out}$. Store $e_{out}$ with the index of $ru_7[1,8]||ru_8[0,1,6,8,9,14]||\overline{w_8}[0,1,6,8,9,14]$ in a table $T_7$.

# B Precomputation Phase of 11-Round Attack

The precomputation phase is almost the same as the 10-round attack except the following steps.

1. At Step 3, we need to build four tables $T_5^i$ ($i = 0,\cdots,3$). As shown in Fig. 15, for Column 0, guess $\Delta C[0,1,2,3]||\Delta z_9[0,1,2,3]$, and deduce $\Delta x_{10}[0,1,2,3]$ and $\Delta y_{10}[0,1,2,3]$. By Proposition 1, we can deduce $y_{10}[0,1,2,3]$. Guess $C[1,3]$, $rk_{10}[1,3]$ can be deduced. One can deduce $rk_{-1}[1,3]$ from $rk_{10}[1,3]$, and store $x_{10}[0,1,2,3]||\Delta z_9[0,1,2,3]$ in a table $T_5^0$ with the index of $rk_{-1}[1,3]||\Delta C[0,1,2,3]||C[1,3]$. There are $2^8$ values for each index in average. Similarly, we can get one table $T_5^i$ for Column $i$ ($i = 1,\cdots,3$), and there are $2^8$, $2^8$, $2^4$ and $2^{12}$ values for each index in $T_5^i$ ($i = 0,\cdots,3$), respectively.

2. At Step 4, guess $x_7[9,10]||\Delta x_7[9]$, one can deduce $\Delta y_7[9,10]$ and $y_7[9,10]$ since $\Delta x_7[9] = \Delta x_7[10]$. Then $\Delta x_8[0,2,3,9,10,11]$ can be deduced. Guess $\Delta y_8[0,2,3,9,10,11]||\Delta y_9 - \Delta y_9[6,14]$, then $x_8[0,2,3,9,10,11]$ and $y_8[0,2,3,9,10,11]$ can be deduced by Proposition 1, and $x_9 - x_9[6,14]$ and $y_9 - y_9[6,14]$ can be also deduced by Proposition 1. Deduce $ru_7[1,8]$ from $x_7[9,10]$ and $x_8[0,2,3,9,10,11]$, and deduce $ru_8[0,1,6,8,9,14]$ from $y_8[0,2,3,9,10,11]$ and $x_9 - x_9[6,14]$. Deduce $ru_{10}[1,8]$ from $ru_7[1,8]$ and $ru_8[1,8]$, and deduce $ru_{10}[0,1,6,8,9,14] \oplus ru_9[0,1,6,8,9,14]$ from $ru_8[0,1,6,8,9,14]$. Let $\chi$ denote $z_9[0,1,6,8,9,14] \oplus ru_{10}[0,1,6,8,9,14] \oplus ru_9[0,1,6,8,9,14]$. Store $y_9 - y_9[6,14]||ru_7[1,8]||ru_8[0,1,6,8,9,14]$ in a table $T_6$ with the index of $ru_{10}[1,8]||\chi||\Delta z_9 - \Delta z_9[4,11]$. There are $2^4$ values for each index in average. We can also reduce the size of $T_6$ by dividing it into small tables.

3. Besides, we need to build two more tables for online phase. For all 28-bit sub-keys $ru_9[0,2,5,7,9,12,14]$, decrypt all 28-bit values $\overline{w_9}[0,2,5,7,9,12,14]$ and obtain $\overline{w_8}[0,1,6]$. Store $\overline{w_8}[0,1,6]$ with the index of $ru_9[0,2,5,7,9,12,14]||\overline{w_9}[0,2,5,7,9,12,14]$ in a table $T_8^0$. For all 28-bit sub-keys $ru_9[1,3,6,8,10,13,15]$, decrypt all 28-bit values $\overline{w_9}[1,3,6,8,10,13,15]$ and obtain $\overline{w_8}[8,9,14]$. Store $\overline{w_8}[8,9,14]$ with the index of $ru_9[1,3,6,8,10,13,15]||\overline{w_9}[1,3,6,8,10,13,15]$ in a table $T_8^1$.

# C Precomputation Phase of 12-Round Attack

**Precomputation phase:** In the precomputation phase, we need to build a table that contains all the sequence $(e_{out}^1 \oplus e_{out}^0, e_{out}^2 \oplus e_{out}^0, \cdots, e_{out}^{32} \oplus e_{out}^0)$ described in Proposition 7.

1. For each 120-bit value $\Delta z_2[0,7,9,14]||\Delta w_4 - \Delta w_4[7,14]||ru_3 - ru_3[0,4,8,12]$, deduce $x_3 - x_3[0,7,9,14]||y_4$ by Proposition 3. Compute $\overline{w_2}[0,7,9,14]$ and $z_4[0,7,9,14]$, and let $\varsigma_1 = \overline{w_2}[0,7,9,14] \oplus z_4[0,7,9,14]$. Deduce $rk_3[4,12]$ from $ru_3[5,6,7,13,14,15]$, and let $\varsigma_2 = rk_3[4,12]||(ru_3[1,9] \oplus ru_3[3,11])$. Store $x_3 - x_3[0,7,9,14]||x_4||w_4 - w_4[7,14]$ in a Table $T_1$ with the index of $w_4[4,11]||\varsigma_1||\varsigma_2||\Delta z_2[0,7,9,14]||\Delta w_4 - \Delta w_4[7,14]$. There are $2^8$ values for each index in average.

2. For each 92-bit value $\Delta z_7[5]||\Delta x_7[4,11]||\Delta x_6[1,3,4,9,11,12]||\Delta w_4 - \Delta w_4[7,14]$, deduce $\Delta z_7[6]$ since $\Delta z_7[6] = \Delta z_7[5]$, then deduce $x_7[4,11]$, $x_6[1,3,4,9,11,12]$ and $x_5 - x_5[7,14]$ by Proposition 1. Deduce $rk_6[4,11]$ and $rk_5[1,3,4,9,11,12]$. Store $x_5 - x_5[7,14]||rk_6[4,11]||rk_5[1,3,4,9,11,12]||\Delta w_4 - \Delta w_4[7,14]$ in a Table $T_2$.

3. For each 24-bit value $\Delta z_1[1,2]||\Delta y_2[0,1,2,3]$, deduce $y_2[0,1,2,3]$. Store $y_2[0,1,2,3]||\Delta z_2[0,7,9,14]$ in a Table $T_3$.

4. For each value of Table $T_2$ and Table $T_3$, do the following steps:

   (a) Compute $w_4[4,11]$ from $rk_5[4,11]$ and $x_5[4,11]$, and compute $\varsigma_1' = z_2[0,7,9,14] \oplus \overline{w_4}[0,7,9,14]$ from $y_2[0,1,2,3]$ and $x_5 - x_5[7,14]$[5]. Deduce $\varsigma_2' = rk_3[4,12]||(ru_3[1,9] \oplus ru_3[3,11])$ from $rk_5[1,3,4,9,11,12]$. Look up Table $T_1$ to get about $2^8$ values of $x_3 - x_3[0,7,9,14]||x_4||w_4 - w_4[7,14]$ with the index of $w_4[4,11]||\varsigma_1'||\varsigma_2'||\Delta z_2[0,7,9,14]||\Delta w_4 - \Delta w_4[7,14]$. Deduce $rk_4 - rk_4[7,14]$, $rk_1[0,1,2,3]$ and $rk_0[5,10]$, then deduce $x_1[5,10]$ and $w_0[5,10]$. Therefore, we get the 58 nibble-parameters (5).

   (b) Compute the sequence $(e^1_{out} \oplus e^0_{out}, e^2_{out} \oplus e^0_{out}, \cdots, e^{32}_{out} \oplus e^0_{out})$, and store them along with a 60-bit value $ru_4[0,1,2,7,8,9,10,11,14]||ru_3[0,1,7,8,9,15]$ in a Table $T_4$.

[5] $\varsigma_1'$ is different from $\varsigma_1$ by a constant.