TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/01 pp1–10
DOI: 10.26599/TST.2022.9010007
Volume x, Number x, xxxx xxxx

# Integral Attack on the Full FUTURE Block Cipher

Zeyu Xu, Jiamin Cui, Kai Hu*, and Meiqin Wang

**Abstract:** FUTURE is a recently proposed lightweight block cipher that achieved a remarkable hardware performance due to careful design decisions. FUTURE is an Advanced Encryption Standard (AES)-like Substitution-Permutation Network (SPN) with 10 rounds, whose round function consists of four components, i.e., SubCell, MixColumn, ShiftRow, and AddRoundKey. Unlike AES, it is a 64-bit-size block cipher with a 128-bit secret key, and the state can be arranged into 16 cells. Therefore, the operations of FUTURE including its S-box is defined over $\boldsymbol{F}_{2^4}$. The previous studies have shown that the integral properties of 4-bit S-boxes are usually weaker than larger-size S-boxes, thus the number of rounds of FUTURE, i.e., 10 rounds only, might be too aggressive to provide enough resistance against integral cryptanalysis. In this paper, we mount the integral cryptanalysis on FUTURE. With state-of-the-art detection techniques, we identify several integral distinguishers of 7 rounds of FUTURE. By extending this 7-round distinguisher by 3 forward rounds, we manage to recover all the 128 bits secret keys from the full FUTURE cipher without the full codebook for the first time. To further achieve better time complexity, we also present a key recovery attack on full FUTURE with full codebook. Both attacks have better time complexity than existing results.

**Key words:** symmetric-key; integral attack; division property; FUTURE

## 1 Introduction

FUTURE is a new Substitution-Permutation Network (SPN)-based lightweight block cipher, recently proposed by Gupta et al.[1] It consists of 10 rounds and is an Advanced Encryption Standard (AES)-like cipher that accepts 128-bit key and has a block size of 64-bit. FUTURE is designed to encrypt data in a single clock cycle, i.e., an unrolled implementation. Therefore, the designers are committed to using very low implementation cost than other block ciphers in unrolled fashion. These lightweight algorithms, due to the high implementation cost, tend not to use the Maximum Distance Separable (MDS) matrix, although the MDS matrix provides better security under the same number of rounds. FUTURE overcomes this challenge by judiciously choosing a very lightweight MDS matrix. The MDS matrix is a composition of four sparse matrices, each matrix has very low implement cost. Meanwhile, the S-box is also obtained by composing four low-hardware-cost S-boxes. The authors benchmarked hardware implementations on Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) and compared FUTURE to several well-known lightweight ciphers in the literature with respect to size, critical path, and throughput. FUTURE ended up giving the

• Zeyu Xu, Jiamin Cui, Kai Hu, and Meiqin Wang are with School of Cyber Science and Technology, Shandong University, Qingdao 266237, China, and Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China. E-mail: xuzeyu@mail.sdu.edu.cn; cuijiamin@mail.sdu.edu.cn; kai.hu@sdu.edu.cn; mqwang@sdu.edu.cn.
• Kai Hu is also with School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798, Singapore.
• Meiqin Wang is also with Quan Cheng Laboratory, Jinan 250100, China.
* To whom correspondence should be addressed.
Manuscript received: 2023-11-05; accepted: 2023-12-22

best results among the compared algorithms in many respects. The designer evaluated the security of FUTURE in the document. For differential and linear cryptanalysis, they expected that there would be no effective 5-round distinguisher. Later, Ilter et al. found effective 5-round differential and linear distinguishers in Ref. [2]. However, they did not further utilize distinguishers for key recovery attacks. Recently, Schrottenloher et al. utilized the Meet-in-the-Middle (MitM) technique to present the first full round key recovery attack of FUTURE[3], breaking the designers' 7-round estimate under this technique.

The idea of integral attack comes from Square attack, which was first proposed by Daemon et al.[4], and then formalized by Knudsen and Wagner[5]. It is one of the most powerful cryptanalysis techniques. Generally, it consists of an integral distinguisher and a key-recovery phase. During the construction of the integral distinguisher, a structure of plaintexts is encrypted to obtain a state set with integral property (e.g., zero-sum) in some positions. Then, the attacker guesses the relevant subkeys and partially decrypts the ciphertexts to the end of the distinguisher. The key space is reduced by checking the integral property.

Several techniques were proposed to optimize the integral attacks. The division property is the most efficient and accurate method of detecting integral distinguishers, which was introduced by Todo[6] at EUROCRYPT 2015. It is word-oriented and can exploit the algebraic degree information of the local components. In particular, it was applied to MISTY1, and the full MISTY1 was broken for the first time[7]. To better exploit the concrete structure of the ciphers, Todo and Morii[8] introduced the Bit-based Division Property (BDP) at FSE 2016. BDP treats the components of the target primitive at the bit level so that more information in the structures can be used. Compared with the word-based division property, the BDP is more likely to find better integral characteristics. Later, Wang et al.[9] presented the automatic methods to search for the three-subset bit-based division property effectively. In Ref. [10], Hao et al. introduced the three-subset bit-based division property without unknown subsets (3SDPwoU). The monomial prediction, proposed by Hu et al.[12], is another language of division property from a pure algebraic perspective.

However, using common programming languages, the time and memory complexities for searching division property in practice are $O(2^n)$ where $n$ is the block size. To overcome this bottleneck, Xiang et al. combines the Mixed Integral Linear Programming (MILP) method and division property to search for the integral distinguishers at ASIACRYPT 2016[13]. As a result, they can deal with ciphers of block sizes much larger than 32 bits efficiently. Subsequently, some automatic searching tools aided by Boolean Satisfiability problem (SAT)[14] and Satisfiability Modulo Theories (SMT)[15] are also proposed[16, 17].

Symmetric cryptographic primitives can be decomposed into a sequence of local building blocks. The propagations of division trails through these local components are modeled and the off-the-shelf solvers are called to find out whether there is a solution. If there is no solution, then balanced properties are found. Components such as S-boxes and basic operations like COPY, AND, and XOR have been modeled well in Ref. [13]. The BDP propagations of the linear layer can be easily handled thanks to these basic operations. However, the problem of how to efficiently model a complex linear layer, e.g., an MDS matrix, has remained. So far three methods have been proposed to solve this problem, they are $\mathcal{S}$ method[18], $\mathcal{ZR}$ method[19], and $\mathcal{H}$ method[20]. Among the three methods, $\mathcal{S}$ method cannot perfectly model the BDP propagations over a complex linear layer, while $\mathcal{ZR}$ method is only applicable to the so-called binary matrix. For the MDS matrix used in FUTURE, $\mathcal{H}$ method is then the only choice for a perfect model.

**Our contributions.** In this paper, we take state-of-the-art techniques for detecting division properties to give a more fine-grained study of the security strength of the block cipher FUTURE against the integral attacks. To find more integral properties, we use the powerful model proposed by Hu et al.[20] to handle the linear layer of FUTURE. 7-round integral distinguishers are established for FUTURE. Based on the 7-round integral distinguisher, exploring the relationship between the round key bits in the key recovery phase, we manage to attack the full FUTURE without the full codebook for the first time. Futher, to achieve a better time complexity, we also present a key recovery attack on full FUTURE with full codebook. All the results are summarized in Table 1.

**Outline.** The paper is organized as follow. In Section 2, we briefly recall the automatic search problems of the BDP and methods to model linear layer. Next, in

**Table 1 Results of key-recovery attacks on FUTURE.**

| Attack | Round | Time (10 rounds) | Memory (bit) | Data | Reference |
|---|---|---|---|---|---|
| MitM | 10/10 | $2^{126}$ | $2^{34}$ | $2^{64}$ | [3] |
| | 10/10 | $2^{124}$ | $2^{84}$ | $2^{64}$ | [3] |
| Integral | 10/10 | $2^{123.7}$ | $2^{16}$ | $2^{63}$ | Section 5.1 |
| | 10/10 | $2^{112}$ | $2^{16.1}$ | $2^{64}$ | Section 5.2 |

Sections 3 and 4 we introduce the block cipher FUTURE and propose a integral distinguisher. We present two key recovery attacks on full FUTURE in Section 5.

## 2 Preliminary

### 2.1 Notations

We list the notations mainly used throughout this paper.

- Blackened italic lowercase letters (i.e., $\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}$): denote vectors in $\boldsymbol{F}_2^n$;
- $\boldsymbol{u}[i]$ or $\boldsymbol{u}_i$: denotes the $i$-th bit of a vector $\boldsymbol{u}\in\boldsymbol{F}_2^n$, where $\boldsymbol{u}=(\boldsymbol{u}_0,\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{n-1})$.
- $\boldsymbol{u}\cdot\boldsymbol{v}$: denotes the inner product operation, $\boldsymbol{u}\cdot\boldsymbol{v}=\bigoplus_{i=0}^{n-1}\boldsymbol{u}[i]\times\boldsymbol{v}[i]$;
- $\boldsymbol{u}[i:j]$: $(j-i+1)$-bit vector starting from $\boldsymbol{u}[i]$ ending at $\boldsymbol{u}[j]$, $i\leqslant j$;
- $M$, $M_1$, $M_2$: denote $n$ by $n$ binary matrix;
- $M[i,j]$: denotes the entry of $M$ located at the $i$-th row and $j$-th column;
- $M[i,*]$, $M[*,j]$: denote the $i$-th row of $M$ and the $j$-th column of $M$.

### 2.2 (Bit-based) division property and automatic search models

The word-based division property[6] was proposed by Todo originally as a generalization of integral attack. Subsequently, by shifting the propagation of the division property to the bit level, Todo and Morii[8] introduced the bit-based division property.

**Definition 1** (Bit-based division property)[8] Let $\boldsymbol{X}$ be a multiset whose elements belong to $\boldsymbol{F}_2^n$. Let $\boldsymbol{K}$ be a set whose elements are n-bit bit vectors. When the multiset $\boldsymbol{X}$ has the division property $\mathcal{D}_{\boldsymbol{K}}^n$, it fulfills the following conditions for any $u\in\boldsymbol{F}_2^n$: $\bigoplus_{\boldsymbol{x}\in\boldsymbol{X}}\pi_{\boldsymbol{u}}(\boldsymbol{x})=$ unknown if there exists a $\boldsymbol{k}\in\boldsymbol{K}$ s.t., $\boldsymbol{u}\succeq\boldsymbol{k}$, and $\bigoplus_{\boldsymbol{x}\in\boldsymbol{X}}\pi_{\boldsymbol{u}}(\boldsymbol{x})=0$ otherwise.

Many symmetric primitives are often composed of bitwise operations like XOR, AND and COPY. When these operations are applied to the elements in $\boldsymbol{X}$, transformations of the division property should also be made following the propagation rules for XOR, AND and COPY which have been proved in Refs. [8, 13].

**XOR**[8] Let the XOR operation create the output $y=x_0\oplus x_1\in\boldsymbol{F}_2$ from the input $x=(x_0,x_1)\in\{0,1\}^2$, where $0\leqslant k_0,\ k_1\leqslant 1$. Assume the input multiset has the division property $\mathcal{D}_{(k_0,k_1)}^2$, then the corresponding output multiset has the division property $\mathcal{D}_{k_0+k_1}^1$, where $0\leqslant i\leqslant k$.

For the BDP $k=(k_0,k_1)$ must satsify $0\leqslant k_0,\ k_1\leqslant 1$. If $k_0=k_1=0$, i.e., the input division property is $\mathcal{D}_{(0,0)}^2$, then the output division property is $\mathcal{D}_0^1$. If the input division property is $\mathcal{D}_{1,0}^2$ or $\mathcal{D}_{0,1}^2$, then the output division property is $\mathcal{D}_1^1$. Moreover, $y$ takes a value in $\boldsymbol{F}_2$, thus $0\leqslant k_0+k_1\leqslant 1$ must hold, i.e., if $(k_0,\ k_1)=(1,\ 1)$, the division property propagation will abort. We denote the division property propagation of XOR operation as $(x_0,\ x_1)\xrightarrow{\text{XOR}}y$.

**COPY**[8] Let the COPY operation create the output $y=(y_0,y_1)\in\{0,1\}^2$ from $x\in\boldsymbol{F}_1$ as $y_0=x$ and $y_1=x$. Assume the input multiset has the division property $\mathcal{D}_k^1$, then the corresponding output multiset has the division property $\mathcal{D}_{(i,k-i)}^2$ where $0\leqslant i\leqslant k$.

The input multiset division property $\mathcal{D}_k^1$ must have $0\leqslant k\leqslant 1$. If $k=0$, the output multiset has the division property $\mathcal{D}_{(0,0)}^1$; otherwise, the output multiset has the division property $\mathcal{D}_{(0,1)(1,0)}^2$. We denote the division property propagation of the COPY operation as $x\xrightarrow{\text{COPY}}(y_0,y_1)$.

The attackers need to determine the division property of the chosen plaintexts, denoted by $\mathcal{D}_{\boldsymbol{K}_0}^{1^n}$. Then the division property of the output ciphertexts at round $r$, denoted by $\mathcal{D}_{\boldsymbol{K}_r}^{1^n}$, can be deduced according to the round function and the propagation rules. Specifically, the attackers determine an index set $I\in\{0,1,\ldots,n-1\}$ of the bit indices of the plaintext and prepare $2^{|I|}$ chosen plaintexts where the variables indexed by $I$ take all possible values. The division property of such chosen plaintexts is $\mathcal{D}_{\boldsymbol{k}}^{1^n}$, where $\boldsymbol{k}_i=1$ if $i\in I$ and $\boldsymbol{k}_i=0$ otherwise. Then, the propagation of the division property from $\mathcal{D}_{\boldsymbol{k}}^{1^n}$ is evaluated as

$$\boldsymbol{k} = \boldsymbol{K}_0 \to \boldsymbol{K}_1 \to \cdots \to \boldsymbol{K}_r,$$

where $\mathcal{D}_{\boldsymbol{K}_i}$ is the BDP after the $i$-round propagation. If the division property $\boldsymbol{K}_r$ does not contain a unit vector $\boldsymbol{e}_i$, then the $i$-th bit of the $r$-round ciphertexts is balanced.

### 2.3 Propagation of BDP in automatic search model

Finding the propagation of BDP is tedious because the size of $\boldsymbol{K}_i$ increases rapidly. At Asiacrypt 2016, Xiang et al.[13] showed that the propagation can be efficiently evaluated by using MILP. Firstly, they introduced the division trail as follows.

**Definition 2** (Division trail)[13] Consider the propagation of the division property $\{\boldsymbol{k}\} = \boldsymbol{K}_0 \to \boldsymbol{K}_1 \to \cdots \to \boldsymbol{K}_r$. Moreover, for any vector $\boldsymbol{k}_{i+1}^* \in \boldsymbol{K}_{i+1}$, there must exist a vector $\boldsymbol{k}_i^* \in \boldsymbol{K}_i$ such that $\boldsymbol{i}_i^*$ can propagate to $\boldsymbol{k}_{i+1}^*$ by the propagation rule of the BDP for the current operation. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in (\boldsymbol{K}_0 \times \boldsymbol{K}_1 \times \cdots \times \boldsymbol{K}_r)$, if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for all $i \in \{0, 1, \cdots, r-1\}$, we call $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r)$ an $r$-round division trail.

The propagation of set $\boldsymbol{K}_i$ was transformed into the propagation of the division trails by this definition. Let $E_k$ be the target $r$-round cipher, if there is no division trail $\boldsymbol{0} \xrightarrow{E_k} \boldsymbol{k}_r = \boldsymbol{e}_i$, then there is no unit vector $\boldsymbol{e}_i$ in $\boldsymbol{K}_r$, i.e., the $i$-th bit is balanced. So once all the unit vectors appear in $\boldsymbol{K}_{r+1}$, there will be no balanced bits at the end of the $(r+1)$-th round of $E_k$, and the maximum number of rounds that integral distinguisher based on BDP can cover is $r$ rounds.

### 2.4 New model for general linear layers

For cipher with a bit-permutation linear layer like PRESENT, GIFT, etc., after the nonlinear layer, there is no cost for the BDP. The block cipher FUTURE has a non-bit-permutation linear layer, i.e., an MDS matrix, which have been considered by the $\mathcal{S}$ method[18] an the $\mathcal{ZR}$ method[19]. In this work, we use the new method that proposed by Hu et al. to trace the BDP propagation[20]. This method is accurate and effective for any type of matrices.

**Proposition 1**[20] For a primitive matrix $M \in F_2^{n\times n}$, a division trail $(\boldsymbol{u}, \boldsymbol{v})$ is valid if and only if $(\boldsymbol{u}, \boldsymbol{v})$ meets the following constraints:

$$E(i,j)\cdot v_j - \sum_{k=0}^{n-1} M(i,k)\cdot v_i \cdot u_k \cdot M_{\boldsymbol{v},\boldsymbol{u}}^{\text{expand}'}(k,j) = 0,$$

for $0 \leqslant i, j \leqslant n-1$, where $E$ is an $n\times n$ identity matrix and $M_{\boldsymbol{v},\boldsymbol{u}}^{\text{expand}'} \in \boldsymbol{F}_2^{n\times n}$ is an auxiliary matrix with $n^2$ elements.

## 3 FUTURE

FUTURE is an AES-like block cipher, the block size is 64 bits, and the master key length is 128 bits. It has a construction of 10 rounds in a fully unrolled fashion. The S-box and the MDS matrix are designed to be efficient in hardware.

**Round function.** The basic operations of each round of FUTURE are SubCell (SC), MixColumn (MC), ShiftRow (SR), and AddRoundKey (ARK). MC operation is removed in the final round. The state of the cipher is denoted by a $4\times4$ matrix $S$ where each entry is a nibble; i.e., $s_i \in \boldsymbol{F}_2^4$ for $0 \leqslant i \leqslant 15$,

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

The round function is presented in Fig. 1.

**SubCell.** SubCell is a nonlinear transformation in which 4-bits S-box $S$ is applied to every nibble of the state. $S$ is a concatenation of four low hardware cost S-boxes. The hexadecimal notation are given by the following Table 2.

**MixColumn.** The MixColumn is a linear operation that operates separately on each column of the four column of the state. FUTURE use an MDS matrix $M_1$ for this operation. The multiplication is over finite field $\mathrm{GF}(2^4) = \mathrm{GF}(2)/\langle x^4 + x + 1\rangle$. Assert $\alpha$ is a root of $x^4 + x + 1$, we have

$$M_1 = \begin{bmatrix} \alpha^3 & \alpha^3+1 & 1 & \alpha^3 \\ \alpha+1 & \alpha & \alpha^3+1 & \alpha^3+1 \\ \alpha & \alpha+1 & \alpha^3 & \alpha^3+1 \\ \alpha^3+1 & \alpha^3+1 & \alpha^3 & 1 \end{bmatrix},$$

and

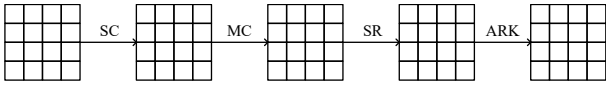


**Fig. 1 Round function.**

**Table 2 S-box of FUTURE.**

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SC($x$) | 1 | 3 | 0 | 2 | 7 | e | 4 | d | 9 | a | c | 6 | f | 5 | 8 | b |

$$\begin{bmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{bmatrix} = M_1 \times \begin{bmatrix} s_i \\ s_{i+1} \\ s_{i+2} \\ s_{i+3} \end{bmatrix},$$

for $i = 0, 4, 8, 12$.

**ShiftRow.** ShiftRow rotates row $i$ of the array state $i$ nibble positions to the right,

$$\begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix} \xrightarrow{\text{SR}} \begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_{13} & s_1 & s_5 & s_9 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_7 & s_{11} & s_{15} & s_3 \end{bmatrix}.$$

**AddRoundKey.** The 64-bit round key $\text{RK}_i$ is XORed to the state at round $i$ of the cipher. And the 64-bit whitening key WK is XORed to the plaintext of the cipher.

**Key scheduling.** Future uses 128-bit master key $K$. It splits $K$ into two 64-bit parts for whitening key and the first round key, i.e., $K = \text{WK}\|\text{RK}_1$. Then the round key $\text{RK}_i$ $(2 \leqslant i \leqslant 10)$ generation is as follow:

$$\text{RK}_i = \begin{cases} \text{WK} \lll \left(5 \times \dfrac{i}{2}\right), & \text{if } 2 \mid i; \\ \text{RK}_1 \lll \left(5 \times \left\lfloor \dfrac{i}{2} \right\rfloor\right), & \text{if } 2 \nmid i. \end{cases}$$

Obviously, $\text{RK}_i$ is a 64-bit permutation in WK or $\text{RK}_1$ without any bit operations.

**Round function (Equivalent key).** Note that both MC and SR are invertible linear operations, so the ARK operation can be moved before the MC operation. The new round key $\text{EK}_i$ is the equivalent round key, which is obtained by performing the inverse operation of SR and MC in turn, i.e., $\text{EK}_i = \text{MC}^{-1} \circ \text{SR}^{-1}(\text{RK}_i)$. Therefore, there exists a 64 order binary invertible matrix $M_2$ such that $\text{EK}_i = M_2 \times \text{RK}_i$. The equivalent round function is shown in Fig. 2.

## 4 Distinguisher

By choosing a proper initial BDP, we find 7-round integral distinguisher. The distinguisher with 63 active bits in the plaintexts and full balanced bits in the ciphertexts, is given as below

$$\begin{bmatrix} \texttt{caaa} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \end{bmatrix} \xrightarrow{7-\text{round}} \begin{bmatrix} \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \end{bmatrix}.$$

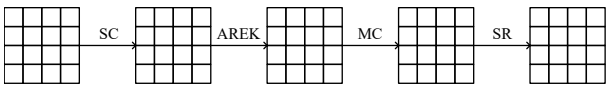


**Fig. 2 Equivalent round function.**

For example, if the plaintext set is $\{1\|A \mid A = \boldsymbol{F}_2^{63}\}$ or $\{0\|A \mid A = \boldsymbol{F}_2^{63}\}$, then the ciphertext set obtained by encrypting 7 rounds is balanced on 64 bits.

## 5 Key-Recovery Attack on Full FUTURE

### 5.1 Key-recovery attack on full future without full data

In this subsection, we propose a key recovery attack on full FUTURE without full data. By adding three rounds after the 7-round distinguisher in Section 4, we can give a key recovery attack on full (10-round) FUTURE, which is presented in Fig. 3. In our attack, we guess round keys' cells in equivalent keys $\text{EK}_8$, $\text{EK}_9$, and $\text{EK}_{10}$ instead of $K_8$, $K_9$, and $K_{10}$, where $\text{EK}_i = \text{MC}^{-1} \circ \text{SR}^{-1}(K_i)$, $i \in \{7, 8, 9\}$.

As shown in Fig. 3, when encrypting a plaintext set with the form of $\mathcal{C}^1\mathcal{A}^{63}$, each cell in $X_8$ is balanced ($X_8[0]$). Therefore, the whole attack includes a data collection phase and a key recovery phase. During the data collection phase, we remain the ciphertext set obtained by encrypting the plaintext set mentioned above During the key recovery phase, we decrypt each ciphertext $C$ in the set to $X_8[0:3]$ by guessing some cells in $\text{EK}_8$, $\text{EK}_9$, and $\text{EK}_{10}$, and then filter out the wrong key based on the balance of $X_8[0:3]$. Detailed steps of attack are given as follows.

**Data collection.** According to the distinguisher, the plaintext set has the form $\mathcal{C}^1\mathcal{A}^{63}$, which means the size of set is $2^{63}$. After accessing all plaintext in the set to the 10-round encryption machine, we remain the returned ciphertext in set $\boldsymbol{C}$ with the size of $2^{63}$. In this phase, the time complexity is $2^{63}$ 10-round encryptions.

**Key recovery.** We focus on checking the balance of $X_8[0:3]$. In order to decrypt ciphertext $C$ to $X_8[0:3]$, we need to know all the gray cells in Fig. 3, that is, we need to guess $\text{EK}_{10}$, $\text{EK}_9[0, 5, 10, 15]$, and $\text{EK}_8[0:3]$. However, according to the key scheduling, we have

$$\text{EK}_{10} \xrightarrow{\text{SR}^{-1} \circ \text{MC}^{-1}} K_{10} \xrightarrow{\ggg 5} K_8 \xrightarrow{\text{MC} \circ \text{SR}} \text{EK}_8,$$

which indicates that $\text{EK}_8$ can be deduced from $\text{EK}_{10}$. Algorithm 1 provides a brief description of the key recovery procedure. The detailed attack procedure is as follows.

(1) **Steps 1 to 6.** Guess $\text{EK}_{10}$ and compute $Z_9[0, 5, 10, 15]$ under $2^{63}$ plaintexts $C$. According to Fig. 3,

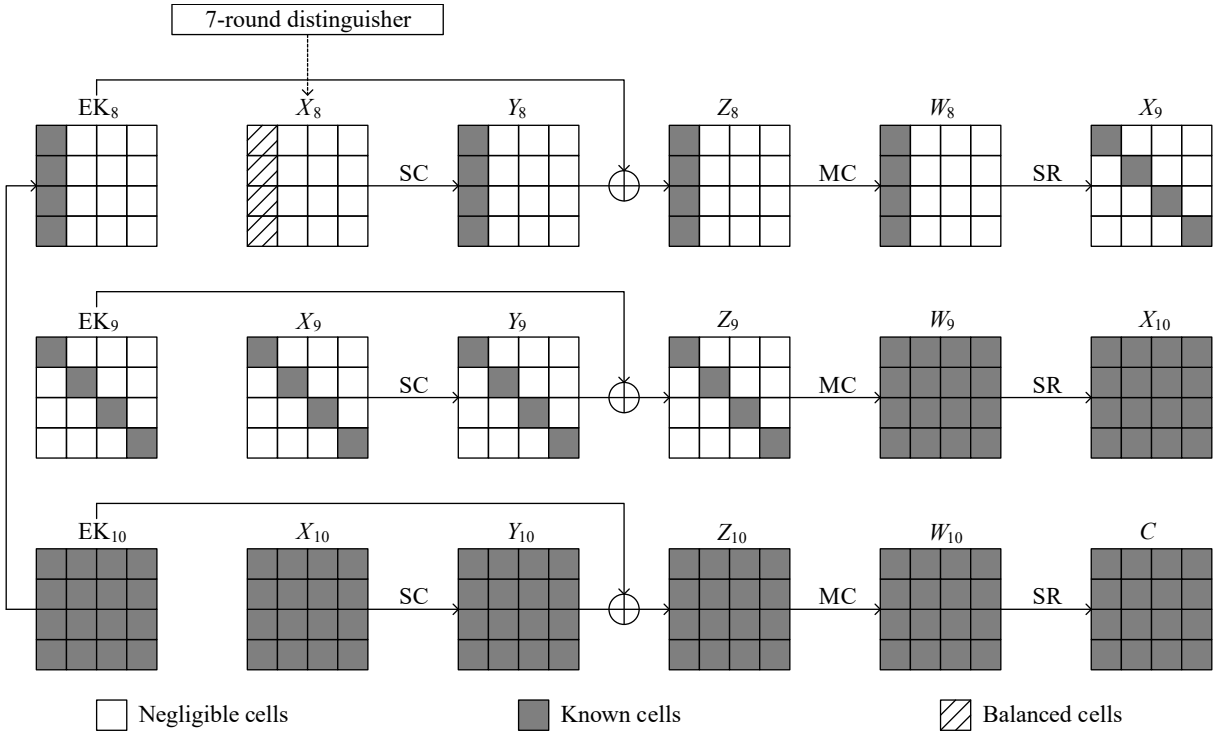


**Fig. 3 10-round key recovery attack.**

**Algorithm 1 10-round key recovery attack**

1 **for** each $\mathrm{EK}_{10}$ **do**
2 Allocate and initialize the arrays $V_1[x]$ with $|x| = 16$;
3 Deduce $\mathrm{EK}_8$ from $\mathrm{EK}_{10}$;
4 **for** $C$ in $\boldsymbol{C}$ **do**
5 Compute $Z_9[0, 5, 10, 15]$;
6 Let $x = Z_9[0, 5, 10, 15]$ and $V_1[x] = V_1[x] \oplus 1$;
7 **for** $2^{16}$ $\mathrm{EK}_9[0, 5, 10, 15]$ **do**
8 Allocate and initialize the 16-bit variable sum;
9 **for** $x = 0$; $x < 2^{16}$; $x{+}{+}$ **do**
10 **if** $V_1[x] == 1$ **then**
11 Compute $X_8[0:3]$ from $Z_9[0, 5, 10, 15] = x$;
12 sum = sum $\oplus X_8[0:3]$;
13 **if** sum == 0 **then**
14 **for** $2^{48}$ $\mathrm{EK}_9[1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14]$ **do**
15 Deduce master key $K$;
16 **if** $E(K, p) == c$ **then**
17 return $K$;

$$W_9 = \mathrm{SR}^{-1}(\mathrm{MC}^{-1} \circ \mathrm{SR}^{-1}(C) \oplus \mathrm{EK}_{10}),$$

$$Z_9[0] \leftarrow \mathrm{MC}^{-1}(W_9[0:3]),$$

$$Z_9[5] \leftarrow \mathrm{MC}^{-1}(W_9[4:7]),$$

$$Z_9[10] \leftarrow \mathrm{MC}^{-1}(W_9[8:11]),$$

$$Z_9[15] \leftarrow \mathrm{MC}^{-1}(W_9[12:15]).$$

Remain the value of $Z_9[0, 5, 10, 15]$ in $V_1$ for the next decryption step. Note that in $V_1$, only the even and odd of the occurrence number of $Z_9[0, 5, 10, 15]$ is recorded (0 refers to even, 1 refers to odd), because even times of a value will not affect the balance. This step requires

$$2^{64} \times 2^{63} \times \frac{1}{10} \approx 2^{123.7}$$

10-round encryptions. And the memory complexity of this part is $2^{16}$ bits.

(2) **Steps 6 to 13.** Guess $\mathrm{EK}_9[0, 5, 10, 15]$ and compute $X_8[0:3]$ under $Z_9[0, 5, 10, 15]$ with $V_1[Z_9[0, 5, 10, 15]] == 1$. Note that $\mathrm{EK}_8$ can be deduced from $\mathrm{EK}_{10}$. Compute sum by operating bit-based XOR for all $X_8[0:3]$. This step requires

$$2^{64} \times 2^{16} \times 2^{16} \times \left(\frac{4}{16} + \frac{4}{16}\right) \times \frac{1}{10} \approx 2^{91.7}$$

10-round encryptions.

(3) **Steps 13 to 17.** From a random perspective, there are $2^{64} \times 2^{16} \times 2^{-16} = 2^{64}$ ($\mathrm{EK}_{10}$, $\mathrm{EK}_9[0, 5, 10, 15]$) that satisfy sum = 0, and the correct key must be among them. Then guess $\mathrm{EK}_9[1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14]$ and deduce master key $K$ under each ($\mathrm{EK}_{10}$,

$EK_9[0, 5, 10, 15]$). Extract plaintext-ciphertext pair $(p,c)$ from the data and obtain the correct key by verifying whether $E(K,p)$ is equal to $c$. This step requires $2^{64} \times 2^{48} = 2^{112}$ 10-round encryptions.

**Complexity.** The 10-round attack requires a data complexity of $2^{63}$ chosen plaintexts. The time complexity is $2^{63} + 2^{123.7} + 2^{91.7} + 2^{112} \approx 2^{123.7}$ 10-round encryptions. The memory complexity is $2^{16}$ bits.

### 5.2 Key-recovery attack on full FUTURE with full data

In this subsection, we propose a key recovery attack on full FUTURE with full data. By adding one round before and two rounds after the 7-round distinguisher in Section 4, we can give a key recovery attack on full (10-round) FUTURE. The time complexity of this attack is currently the lowest.

As shown in Fig. 4, assuming we have a plaintext set, when the $Z_1$ set obtained by encrypting the plaintext set has the form $C^1\mathcal{A}^{63}$, each nibble in $X_9$ is balanced (i.e., $X_9[0]$). Therefore, the whole attack includes a data collection phase and a key recovery phase. During the data collection phase, we remain the ciphertext set obtained by encrypting full plaintext. During the key recovery phase, we choose the corresponding ciphertext set based on guessing the whitening key. Then we decrypt each ciphertext $C$ in the set to $X_9[0:3]$ by guessing some bits in $EK_9$ and $EK_{10}$, and then filter out the wrong key based on the balance of $X_9[0:3]$. Detailed steps of attack are given as follows.

**Data collection.** All ciphertext are used in this attack. The time complexity is $2^{64}$ 10-round encryptions, and the memory complexity is $2^{64} \times 64 = 2^{70}$ bits. Note that the ciphertext are stored in a $2^{16}$ by $2^{48}$ two-dimensional list $S$, The index of the first dimension corresponds to some cells of the plaintext $P$, this is, $S[P[0:3]] = \{E(K, P[0:3]||x)|x \in \boldsymbol{F}_2^{48}\}$.

**Key recovery.** We focus on checking the balance of $X_9[0:3]$ under the assumption that the set of $X_2$ has the form $C^1\mathcal{A}^{63}$. Therefore, we need to ensure that the $X_2$ set obtained by encrypting the plaintext set has form $C^1\mathcal{A}^{63}$, then decrypt the corresponding ciphertext set to

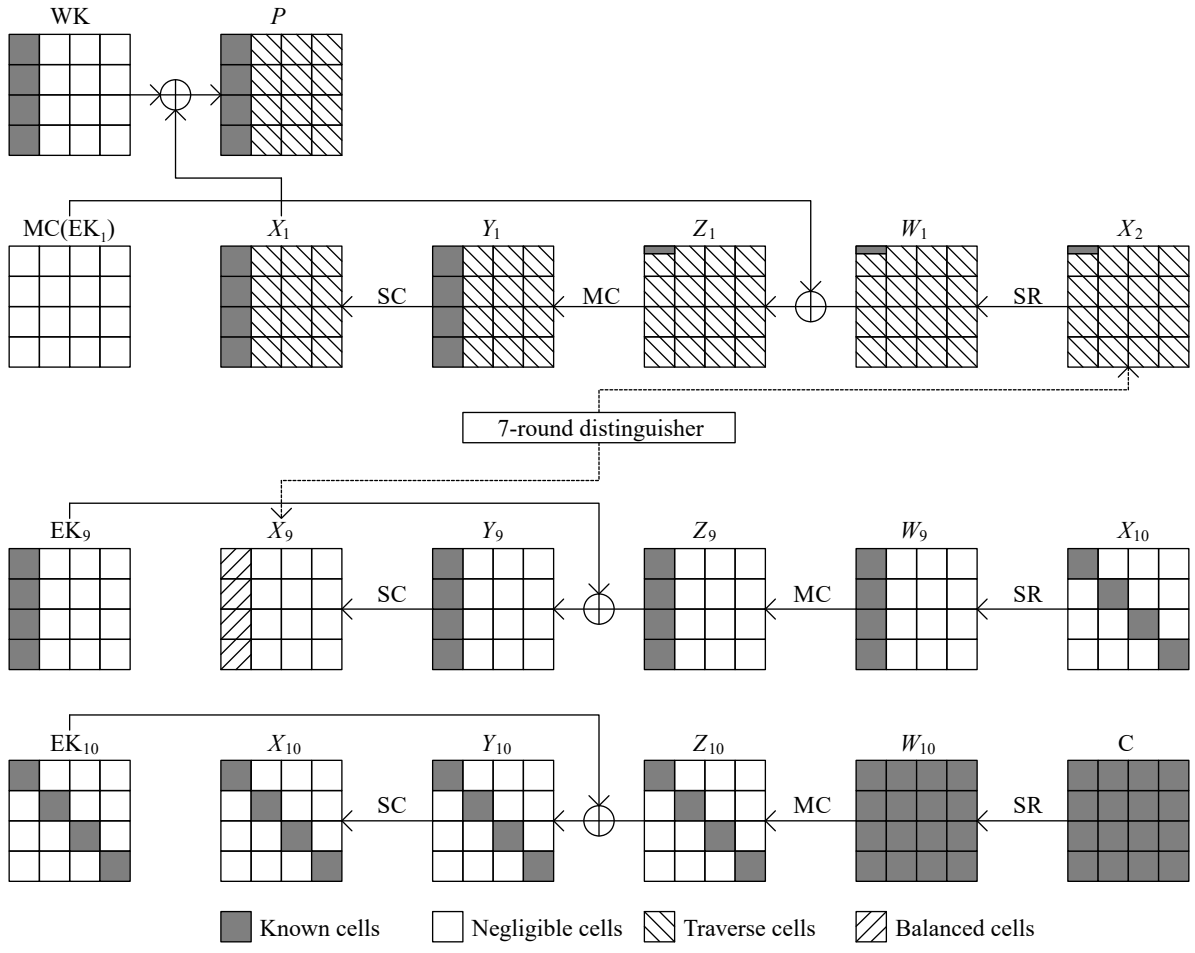


**Fig. 4 10-round key recovery attack.**

get the $X_9$ set. Note that $Z_1 = \mathrm{MC}(\mathrm{EK}_1) \oplus \mathrm{SR}^{-1}(X_2)$, where $\mathrm{SR}^{-1}$ does not affect the first block of $X_2$ and $\mathrm{MC}(\mathrm{EK}_1)$ is a constant. Thus, the $Z_1$ set also has the form $C^1\mathcal{A}^{63}$, denoted by

$$\{0||A||B \mid A = \boldsymbol{F}_2^{15}, B = \boldsymbol{F}_2^{48}\}.$$

Furthermore, we obtain the plaintext set

$$\{(\mathrm{WK}[0:3] \oplus \mathrm{SC}^{-1} \circ \mathrm{MC}^{-1}(0||A))||B\}.$$

This indicates that given the $Z_1$ set, the plaintext set is determined by WK[0 : 3]. And WK[0 : 3] only denotes $P[0:3]$, and each $P[0:3]$ corresponds to $2^{48}$ $P[4:15]$. The ciphertext set under each $P[0:3]$ can be obtained by using the list $S$ obtained during the data collection phase.

After obtaining the ciphertext set, we guess $\mathrm{EK}_{10}$[0, 5, 10, 15] and $\mathrm{EK}_9[0:3]$ to obtain the set of $X_9[0:3]$. According to the key scheduling, there exists a 64th order full rank binary matrix $M$ such that $\mathrm{EK}_{10} = M \times \mathrm{WK}$, while $\mathrm{EK}_9$ is independent of WK. Set the unknown bit in WK(WK[4 : 15]) to 48 variables $\boldsymbol{x} = [x_1, x_2, \ldots, x_{48}]^{\mathrm{T}}$, and use $[M_L|M_R]$ to record the following matrix:

$$\left[\begin{array}{ccc|ccc} M_{0,0} & \cdots & M_{0,15} & M_{0,16} & \cdots & M_{0,63} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{3,0} & \cdots & M_{3,15} & M_{3,16} & \cdots & M_{3,63} \\ M_{20,0} & \cdots & M_{20,15} & M_{20,16} & \cdots & M_{20,63} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{23,0} & \cdots & M_{23,15} & M_{23,16} & \cdots & M_{23,63} \\ M_{40,0} & \cdots & M_{40,15} & M_{40,16} & \cdots & M_{40,63} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{43,0} & \cdots & M_{43,15} & M_{43,16} & \cdots & M_{43,63} \\ M_{60,0} & \cdots & M_{60,15} & M_{60,16} & \cdots & M_{60,63} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{63,0} & \cdots & M_{63,15} & M_{63,16} & \cdots & M_{63,63} \end{array}\right]_{16\times 64},$$

we can get the following system of linear equations containing 16 equations and 48 variables:

$$[M_L|M_R] \times \left[\begin{array}{c} \mathrm{WK}[0] \\ \mathrm{WK}[1] \\ \mathrm{WK}[2] \\ \mathrm{WK}[3] \\ \hline \boldsymbol{x} \end{array}\right]_{64\times 1} = \left[\begin{array}{c} \mathrm{EK}_{10}[0] \\ \mathrm{EK}_{10}[5] \\ \mathrm{EK}_{10}[10] \\ \mathrm{EK}_{10}[15] \end{array}\right]_{16\times 1},$$

Then, we have

$$M_R \times \boldsymbol{x} = \left[\begin{array}{c} \mathrm{EK}_{10}[0] \\ \mathrm{EK}_{10}[5] \\ \mathrm{EK}_{10}[10] \\ \mathrm{EK}_{10}[15] \end{array}\right] \oplus M_L \times \left[\begin{array}{c} \mathrm{EK}_{10}[0] \\ \mathrm{EK}_{10}[5] \\ \mathrm{EK}_{10}[10] \\ \mathrm{EK}_{10}[15] \end{array}\right] = \boldsymbol{b}.$$

According to the theory of nonhomogeneous linear equations, only when $\mathrm{rank}(M_R) = \mathrm{rank}([M_R|\boldsymbol{b}])$, the equations have solutions, and the size of the solution space is $2^{48-\mathrm{rank}(M_R)}$.

$$M_L = \left[\begin{array}{cccccccccccccccc} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&1&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&0&0&0&1&0&1&0&0&0&1&1&0 \\ 1&0&0&1&1&0&0&1&0&0&1&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0 \\ 0&1&1&0&0&1&0&0&0&0&1&0&0&1&1&0 \\ 0&0&1&1&0&0&1&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{array}\right],$$

and

$$M_R = \left[\begin{array}{cccccccccccccccc} 1&1&0&1&0&1&0&0&1&0&0&0&0&0&0&1 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 1&1&0&1&1&1&0&1&0&0&0&1&1&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \mathbf{0}^{32} \\ 1&0&0&0&0&0&0&1&1&0&0&1&1&0&0&0 & \mathbf{0}^{32} \\ 1&0&0&0&1&1&0&0&0&1&0&0&0&1&0&0 & \mathbf{0}^{32} \\ 0&1&0&0&0&1&1&0&0&0&1&0&0&0&1&0 & \mathbf{0}^{32} \\ 0&0&1&0&0&0&1&1&0&0&0&1&0&0&0&1 & \mathbf{0}^{32} \\ \mathbf{0}^{17} & 0&0&1&1&0&0&0&1&0&0&1&1&0&0&1 & \mathbf{0}^{16} \\ 0&0&1&0&0&1&1&0&1&1&0&0&1&0&0&0 & \mathbf{0}^{32} \\ 0&0&0&1&0&0&1&1&0&1&1&0&0&1&0&0 & \mathbf{0}^{32} \\ 1&0&0&1&1&0&0&0&0&0&1&1&0&0&1&0 & \mathbf{0}^{32} \end{array}\right],$$

where $\mathbf{0}^i$ is $i$-bit zero vector. Obviously, $\mathrm{rank}(M_L) = 6$ and $\mathrm{rank}(M_R) = 10$. There are 6 zero lines in $M_R$, which means $\boldsymbol{b}[i] = 0, \quad i = 1,2,3,5,6,7$. Therefore, given WK[0 : 3], $\mathrm{EK}_{10}$[0, 5, 10, 15] has 6 bits constraints, then $2^{10}$ $\mathrm{EK}_{10}$[0, 5, 10, 15] satisfy the equation with solutions. And the solution space size of $\boldsymbol{x}$ is $2^{48-10} = 2^{38}$. Based on the key relationship obtained above, we briefly introduced the key recovery procedure in Algorithm 2. The detailed attack procedure is as follows.

(1) **Steps 1 to 5.** Guess WK[0 : 3] and compute $P[0:3]$ under $2^{15}$ $Z_1[0:3]$. Remain the value of $P[0:3]$ in $S_1$ as indexes for ciphertext list $S$. This step requires

$$2^{16} \times 2^{15} \times \frac{4}{16} \times \frac{1}{10} \approx 2^{25.7}$$

10-round encryptions. And the memory complexity of this part is $2^{15} \times 16 = 2^{19}$ bits.

(2) **Steps 7 to 13.** Guess $\mathrm{EK}_{10}$[0, 5, 10, 15] and check whether satisfy the constraint. The complexity of matrix-vector multiplication is $2^{16} \times 2^{16} \times 6 \times 16 = 2^{41.6}$. This step will remain $2^{16-6} = 2^{10}$ $\mathrm{EK}_{10}$[0, 5, 10, 15]. Then compute $Z_9[0:3]$ under ciphertext set obtained by $S_1$ and $S$. As in Algorithm 2, $V_1$ record the parity of the number of occurrences of $Z_9[0:3]$. Meanwhile,

**Algorithm 2 10-round key recovery attack with full data**

**for** $2^{16}$ WK[0 : 3] **do**
  Allocate and initialize the set $S_1$;
  **for** $2^{15}$ $Z_1[0:3]$ **do**
    Compute $P[0:3] = \mathrm{WK}[0:3] \oplus \mathrm{SC}^{-1} \circ \mathrm{MC}^{-1}(Z_1[0:3])$;
    Append $P[0:3]$ to $S_1$;
  **for** $2^{16}$ $\mathrm{EK}_{10}$[0, 5, 10, 15] **do**
    **if** $\mathrm{EK}_{10}$[0, 5, 10, 15] and WK[0 : 3] satisfy the constraints **then**
      Allocate the set $S_2$ containing the basic solutions of $2^{38}$ WK[4 : 15] deduced by WK[0 : 3] and $\mathrm{EK}_{10}$[0, 5, 10, 15];
      Allocate and initialize the arrays $V_1[x]$ with $|x| = 16$;
      **for** $2^{15}$ $P[0:3]$ in $S_1$ **do**
        **for** $2^{48}$ $C$ related to $P[0:3]$ **do**
          Compute $Z_9[0:3]$;
          Let $x = Z_9[0:3]$ and $V_1[x] = V_1[x] \oplus 1$;
      **for** $2^{16}$ $\mathrm{EK}_9[0:3]$ **do**
        Allocate and initialize the 16-bit variable sum;
        **for** $x = 0$; $x < 2^{16}$; $x{+}{+}$ **do**
          **if** $V_1[x] == 1$ **then**
            Compute $X_9[0:3]$ from $Z_9[0:3] = x$;
            $\mathrm{sum} = \mathrm{sum} \oplus X_9[0:3]$;
        **if** sum == 0 **then**
          **for** $2^{48}$ $\mathrm{EK}_9[4:15]$ **do**
            **for** $2^{32}$ WK[4 : 15] generated by the basic solutions in $S_2$ **do**
              Deduce master key $K$;
              **if** $E(K, p) == c$ **then**
                return $K$;

$S_2$ contains $2^{38}$ possible values of WK, which are obtained by solving the equation system. This step requires

$$2^{16} \times 2^{10} \times 2^{15} \times 2^{48} \times \frac{4}{16} \times \frac{1}{10} \approx 2^{83.7}$$

10-round encryptions. The complexity of Gaussian Elimination is $O(2^{16} \times 2^{10} \times 10^3) = O(2^{36})$. And the memory complexity of this part is $|V_1| + |S_2| = 2^{16} + 39 \times 48 = 2^{16.1}$ bits.

(3) **Steps 14 to 19.** Guess $\mathrm{EK}_9[0:3]$ and compute $X_9[0:3]$ under $Z_9[0:3]$ with $V_1[Z_9[0:3]] == 1$. Compute sum by operating bit-based XOR for all $X_9[0:3]$. This step requires

$$2^{16} \times 2^{16} \times 2^{16} \times 2^{16} \times \frac{4}{16} \times \frac{1}{10} \approx 2^{58.7}$$

10-round encryptions.

(4) **Steps 20 to 25.** From a random perspective, there are $2^{16} \times 2^{-16} = 1$, $\mathrm{EK}_9[0:3]$ that satisfy $\mathrm{sum} = 0$, and the correct key must be among them. Then guess $\mathrm{EK}_9[4:15]$ and deduce master key $K$ under each WK generated by the basic solutions in $S_2$. Extract plaintext-ciphertext pair $(p, c)$ from the data and obtain the correct key by verifying whether $E(K, p)$ is equal to $c$. This step requires

$$2^{16} \times 2^{10} \times 2^{16} \times 2^{-16} \times 2^{48} \times 2^{38} = 2^{112}$$

10-round encryptions.

**Complexity.** This 10-round attack requires a data complexity of $2^{64}$. The time complexity is $2^{112}$ 10-round encryptions. The memory complexity is $2^{16.1}$ bits.

## Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62032014), the National Key Research and Development Program of China (No. 2018YFA0704702), and the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (No. ZR202010220025).

## References

[1] K. C. Gupta, S. K. Pandey, and S. Samanta, FUTURE: A lightweight block Cipher using an optimal diffusion matrix, in *Progress in Cryptology - AFRICACRYPT 2022*, L. Batina and J. Daemen, eds. Cham, Switzerland: Springer, vol. 13503, 2022, pp. 28–52.

[2] M. B. İlter and A. A. Selçuk, MILP-aided cryptanalysis of the FUTURE block cipher, in *Innovative Security Solutions for Information Technology and Communications*, G. Bella, M. Doinea, and H. Janicke, eds. Cham, Switzerland: Springer Nature, 2023, pp. 153–167.

[3] A. Schrottenloher and M. Stevens, Simplified modeling of MITM attacks for block ciphers: New (quantum) attacks, *IACR Trans. Symmetric Cryptol.*, pp. 146–183, 2023.

[4] J. Daemen, L. Knudsen, and V. Rijmen, The block cipher Square, in *Fast Software Encryption*, E. Biham, ed. Haifa, Israel: Springer, vol. 1267, 1997.

[5] L. R. Knudsen and D. A. Wanger, Integral Cryptanalysis, in *Fast Software Encryption*, J. Daemen and V. Rijmen, eds. leuven, Belgium: Springer, vol. 2365, 2002.

[6] Y. Todo, Structural evaluation by generalized integral property, in *Advances in Cryptology -- EUROCRYPT 2015*, E. Oswald and M. Fischlin, eds. Sofia, Bulgaria: Springer, 2015, pp. 287–314.

[7] Y. Todo, Integral Cryptanalysis on Full MISTY1, *J. Cryptol.*, vol. 30, pp. 920–959, 2017.

[8] Y. Todo and M. Morii, Bit-based division property and application to Simon family, in *Fast Software Encryption*, T. Peyrin, ed. Bochum, Germany: Springer, vol. 9783,

2016.
[9] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, and W. Meier, Improved division property based cube attacks exploiting algebraic properties of superpoly, in *Advances in Cryptology – CRYPTO 2018*, H. Shacham and A. Boldyreva, eds. Cham, Switzerland: Springer, 2018, pp. 275–305.
[10] Y. Hao, G. Leander, W. Meier, Y. Todo, and Q. J. Wang, Modeling for three-subset division property without unknown subset, *J. Cryptol.*, vol. 34, no. 22, 2021.
[11] P. Hebborn, B. Lambin, G. Leander, and Y. Todo, Lower bounds on the degree of block ciphers, in *Advances in Cryptology – ASIACRYPT 2020*, S. Moriai and H. Wang, eds. Cham, Switzerland: Springer, 2020, pp. 537–566.
[12] K. Hu, S. Sun, M. Wang, and Q. Wang, An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums, in *Advances in Cryptology – ASIACRYPT 2020*, S. Moriai and H. Wang, eds. Cham, Switzerland: Springer, 2020, pp. 446–476.
[13] Z. Xiang, W. Zhang, Z. Bao, and D. Lin, Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers, in *Advances in Cryptology – ASIACRYPT 2016*, J. Cheon and T. Takagi, eds. Berlin, Germany: Springer, 2016, pp. 648–678.
[14] S. A. Cook, M. A. Harrison, R. B. Banerji, J. D. Ullman, The complexity of theorem-proving procedure, in *Proc. 3rd Annual ACM Symposium on Theory of Computing*, *Shaker Heights*, New York, NY, USA, pp. 151–158, 1971.
[15] C. W. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability modulo theories, in *Handbook of Model Checking*, E. Clarke, T. Henzinger, H. Veith, and R. Bloem, eds. Cham, Switzerland: Springer, vol. 185, pp. 825–885
[16] K. Hu and M. Wang, Automatic search for a variant of division property using three subsets, in *Topics in Cryptology – CT-RSA 2019*, M. Matsui, ed. Cham, Switzerland: Springer, 2019, pp. 412–432.
[17] L. Sun, W. Wang, and M. Wang, Automatic search of bit-based division property for ARX ciphers and word-based division property, in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, eds. Cham, Switzerland: Springer, 2017, pp. 128–157.
[18] L. Sun, W. Wang, and M. Q. Wang, MILP-aided bit-based division property for primitives with non-bit-permutation linear layers, *IET Inf. Secur.*, vol. 14, no. 1, pp. 12–20, 2020.
[19] W. Zhang and V. Rijmen, Division cryptanalysis of block ciphers with a binary diffusion layer, *IET Inf. Secur.*, vol. 13, no. 2, pp. 87–95, 2019.
[20] K. Hu, Q. Wang, and M. Wang, Finding bit-based division property for ciphers with complex linear layers, *IACR Trans. Symmetric Cryptol.*, pp. 396–424, 2020.

**Zeyu Xu** received the BS and MS degrees from Shandong University, in 2017 and 2020, respectively. He is working toward the PhD degree in cyberspace security. His current research focuses on analysis of symmetric-key algorithms.

**Jiamin Cui** received the BS degree from Shandong University in information security, in 2019. She is working toward the PhD degree in cyberspace security. Her current research focuses on design and analysis of symmetric-key algorithms.

**Kai Hu** received the BS and PhD degrees from in cyberspace security from Shandong University, China, in 2016 and 2021, respectively. He was a postdoctoral researcher at Nanyang Technological University from 2021 to 2023. He is currently a researcher with School of Cyber Science and Technology, Shandong University. His research interest is cryptography, including the design and analysis of symmetric-key algorithms.

**Meiqin Wang** received the BS and MS degrees from Xi'an Jiaotong University, in 1996 and 1999, respectively, and the PhD degree from Shandong University, in 2007. She was a guest researcher of The University of Hong Kong, in 2005 and 2008, the and guest researcher of KU Leuven, Belgium from 2010 to 2011. She is currently a professor with School of Cyber Science and Technology, Shandong University. She has coauthored more than 100 research peer reviewed journal and conference papers. She was the general co-chair of FSE 2023. Her research interest is cryptography, including the design and analysis of symmetric-key algorithms.