Related-Key Cryptanalysis of FUTURE

The Full Round Distinguishing Attack

Amit Jana[1], Smita Das[1], Ayantika Chatterjee[1], Debdeep Mukhopadhyay[1], and Yu Sasaki[2]

[1] Indian Institute of Technology, Kharagpur
{janaamit001,smita1995star,cayantika,debdeep.mukhopadhyay}@gmail.com
[2] NTT Social Informatics Laboratories and NIST Associate
{yusasaki0930}@gmail.com

Abstract. At Africacrypt 2022, Gupta et al. introduced FUTURE, a 64-bit lightweight block cipher based on an MDS matrix and designed in an SPN structure, with a focus on achieving single-cycle encryption and low implementation cost, especially in unrolled architectures. While the designers evaluated its security under various attack models, they did not consider related-key cryptanalysis. In this work, we address this gap by analyzing the security of FUTURE in the related-key setting using techniques based on Mixed Integer Linear Programming (MILP). We first propose a simplified and generalizable approach for applying MILP to model any MDS- or near-MDS-based cipher that follows the substitution-permutation paradigm. Using our MILP framework, we construct an 8-round related-key distinguisher on FUTURE, requiring 2^{58} plaintexts, 2^{58} XOR operations, and negligible memory. Leveraging this distinguisher, we propose a 9-round key recovery attack with data, memory, and time complexities of 2^{59}, 2^{59}, and 2^{69}, respectively. We further identify a full-round (i.e., 10 rounds) boomerang distinguisher with a probability of $2^{-45.8}$, enabling a distinguishing attack with 2^{48} data and time complexity. In addition, we develop a full-round key recovery attack on FUTURE with data, time, and memory complexities of 2^{18}, 2^{70}, and 2^{64}, respectively. Although all known single-key attacks remain impractical with time complexities of at least 2^{112}, our results demonstrate a full-round cryptanalysis of FUTURE in the related-key setting, highlighting the absence of related-key security guarantees in the original design and suggesting that additional considerations may be necessary for deployments involving related keys.

Keywords: Related-Key Cryptanalysis, Boomerang Attack, FUTURE.

1 Introduction

In recent years, the demand for cryptographic solutions optimized for resource-constrained environments–such as RFID tags, sensor networks, and contactless smart cards–has led to the development of lightweight cryptographic primitives. Unlike traditional cryptographic methods like AES [26] and SHA-256 [42],

which are designed for systems with substantial processing power and memory, lightweight cryptography prioritizes efficiency across various metrics, including hardware cost, power utilization, and latency. Block ciphers, which can be thought of as a pseudo-random permutations to transform plaintext into ciphertext blocks of fixed lengths, are mainly categorized into Feistel structures and substitution-permutation networks (SPNs). Feistel structures, used in ciphers like TWINE [55] and PICCOLO [48], are cost-effective but require more rounds to ensure security, while SPNs offer robust security but can be more resource-intensive. The field of lightweight cryptography has expanded significantly, with ciphers such as PRESENT [20], KATAN [24], SIMON & SPECK [7], PRINCE [22], MANTIS [8], LED [30], MIDORI [5], and GIFT [6] being optimized for parameters like code size, latency, and energy consumption. Moreover, tweakable block ciphers like SKINNY [8], CRAFT [9], and QARMA [4], enhance encryption and authentication modes. Additionally, CRAFT addresses challenges such as resistance to Differential Fault Analysis (DFA) attacks.

Several lightweight block ciphers, including LED, MIDORI, and SKINNY, build on the fundamental structure of the AES round function, modifying its components to enhance performance. AES employs MDS (Maximum Distance Separable) matrices in its round function to achieve strong diffusion, which is essential for robust security against various cryptographic threats. However, incorporating MDS matrices into lightweight block ciphers poses a challenge due to their high implementation cost. This often necessitates additional rounds in these ciphers to maintain security against attacks such as differential and linear attacks. As a result, many lightweight block ciphers opt for lighter components, such as near-MDS matrices and bit-permutations, to avoid the high costs associated with MDS matrices. This approach helps manage implementation costs while still aiming to achieve effective diffusion, even though MDS matrices offer superior diffusion benefits.

Mixed Integer Linear Programming (MILP) is a well-established optimization technique used to find the optimal solution for a linear objective function subject to a set of linear constraints. In 2011, Mouha *et al.* introduced an automated differential path search method utilizing MILP [40], which helps generate lower bounds for the number of active S-boxes. At that time, the method could not account for the differential properties of the S-box, limiting its application to bit-oriented ciphers like PRESENT and LS-designs [39]. This limitation was later addressed by Sun *et al.* [54,53], who developed two distinct methods to model the differential propagation of S-boxes using systems of inequalities. The first approach uses logical conditions to represent differential properties through linear inequalities. The second approach employs a geometric method to capture all possible input-output difference transitions through an S-box, computing the H-representation (convex hull) of this set using the SageMath inequality generator function and simplifying constraints with a greedy approach. Sasaki and Todo [46] further advanced this technique by incorporating a MILP-based optimization phase to achieve a more compact representation of S-boxes with fewer constraints. Additionally, Boura and Coggia [23] enhanced the method by re-

ducing the number of constraints needed to capture the differential properties of an S-box by adding related constraints from the set of constraints generated by the SageMath inequality generator function.

In 2022, Gupta *et al.* introduced a new 64-bit lightweight block cipher known as FUTURE [32], which stands out for its exceptionally low implementation cost compared to other block ciphers, particularly when implemented in an unrolled fashion. Notably, FUTURE is one of the few lightweight ciphers where all the round components are new, and it employs an MDS matrix for its diffusion layer. The internal functions of the cipher are designed for high hardware efficiency, with the MDS matrix and S-box being specifically optimized to minimize hardware costs. The S-box used is reported to match the cryptographic quality of those in SKINNY and PICCOLO. Hardware benchmarks on FPGA and ASIC platforms demonstrated that FUTURE outperforms several well-known lightweight ciphers in terms of size, critical path, and throughput, achieving superior results across multiple metrics.

Researchers have explored various attack methods on the FUTURE cipher in single-key settings. In [33], a bit-based mixed-integer linear programming (MILP) approach was used to identify both differential and linear characteristics, revealing distinguishers up to five rounds with probabilities of 2^{-58} and 2^{-62}, respectively. In [47], a full-round key recovery attack was presented by combining the meet-in-the-middle (MITM) technique with MILP, resulting in data, time, and memory complexities of 2^{64}, 2^{126}, and 2^{34}, respectively. Similarly, Lin *et al.* [38] applied a MILP-assisted MITM strategy to the full-round cipher, achieving complexities of 2^{64} for data, 2^{124} for time, and 2^{48} for memory. Roy *et al.* [44] proposed an attack leveraging biclique structures, achieving data, time, and memory complexities of 2^{48}, $2^{125.54}$, and 2^{32}, respectively. In another work, Mondal *et al.* [39] utilized the Yoyo technique in the secret-key setting, successfully distinguishing up to five and six rounds with data complexities of $2^{9.83}$ and $2^{58.83}$, respectively. More recently, Xu *et al.* [58] introduced a full-round key recovery attack using integral cryptanalysis, offering improved complexities compared to an exhaustive search over the full codebook.

In the context of related-key cryptanalysis, it is important to note that block ciphers are often employed within various modes of operation, where the security proofs may assume that block ciphers are ideal ciphers. Hence block ciphers are required to resist related-key attacks in such modes. Also, related-key attacks can be effectively applied to hashing modes such as the Davies-Meyer mode. In these constructions, encryption keys may act as message blocks, while plaintexts serve as chaining values. This configuration makes hash functions susceptible to related-key attacks. By selecting message pairs with specific differences, an attacker can effectively generate related keys and observe how these differences propagate through the cipher to influence the hash output. This setup has been exploited in attacks on several lightweight block ciphers, such as KASUMI [27,14], demonstrating practical impact even in highly constrained environments like RFID. As highlighted in prior studies [31,34,21], analyzing related-key resistance not only strengthens our understanding of a cipher's design but also aids in

refining its key schedule and structural choices. This means that using block ciphers in hash functions can open up unexpected weaknesses. So, it is important to carefully check their security when the inputs (like keys) are related in some way.

In the case of FUTURE, the designers have not explicitly discussed its security against related-key attacks, nor have they limited its use to scenarios where such attacks are not relevant. This raises concerns about its potential use in broader contexts where related-key security may be critical. Therefore, evaluating its resistance in this model is not only necessary but also timely. Our motivation stems from three key observations:

1. Since FUTURE does not specify a dedicated mode of operation, it may be deployed with standard encryption modes that implicitly assume the cipher is secure even in related-key scenarios. Analyzing its resistance helps assess its broader applicability.
2. Many comparable ciphers, such as LED, PRESENT, and SKINNY, explicitly claim resistance to related-key attacks. Since FUTURE performs very well but has a weaker security margin in the related-key setting, our study shows that it is important to improve its security while keeping it efficient.
3. Since there is no standard way to measure a cipher's security margin, focusing only on single-key attacks may miss real-world weaknesses. Including related-key attacks gives a deeper understanding of the cipher's design and can even support known attacks. For instance, related-key differentials have been useful in building single-key meet-in-the-middle attacks in earlier study [19]. This kind of broad analysis is important, as future attack methods are hard to predict.

Beyond the general applicability of related-key attacks and their security implications [36,28], related keys naturally arise in practical scenarios such as key updates and device-specific key derivation, where per-device keys are derived from a master key and a device identifier (e.g., via XOR). Moreover, as noted in [3], modern standards such as NIST avoid primitives that are weak in related-key settings, showing the practical relevance of related-key security. Although some designs trade related-key security for implementation efficiency, such trade-offs must be carefully evaluated. In the case of FUTURE, its strong implementation performance is accompanied by weak related-key security, suggesting that this trade-off may not be justified. Highlighting this limitation provides useful insight for the design of symmetric-key ciphers.

Although theoretical attacks on FUTURE have been explored in the single-key setting, no related-key cryptanalysis has been conducted so far, and the original design specification did not address this model. This paper aims to fill that gap by presenting a thorough analysis of related-key cryptanalysis on FUTURE cipher. We construct a bit-oriented MILP framework to identify enhanced differential characteristics. While [33] introduces a bit-oriented MILP model for analyzing FUTURE in the single-key context, it does not provide sufficient details for related-key scenarios. In this work, we offer a complete description of

how to formulate a bit-oriented MILP model tailored to the related-key setting of the FUTURE cipher. This is particularly important since block ciphers used in real-world systems, whether in encryption modes or as components of hash functions, must be secure even when related keys are involved. This framework also offers a simplified approach for formulating constraints for the individual components of each round, making it adaptable to analyze other SPN ciphers with matrix-based linear layers. Leveraging our proposed MILP framework, we present a full cryptanalysis of the FUTURE cipher in the related-key setting. We believe this analysis provides meaningful insights into the security of FUTURE and helps inform its safe and effective use in practical applications.

Our Contributions. Our contributions are three-fold, as follows:

- We present a comprehensive bit based related key MILP model for the FUTURE cipher, which can also serve as a general methodology for constructing MILP models for SPN ciphers employing a MixColumn matrix in their diffusion layers. In addition, we revisit the work of Boura and Coggia [23], where certain implementation details required for generating an optimal set of constraints to model the DDT are not explicitly specified. Based on our understanding, we propose a revised algorithm that achieves results comparable to those of Boura and Coggia [23, Algorithm 1], while producing a larger final set of constraints. Furthermore, we present a detailed and systematic description of how to model MixColumn operation into a binary matrix using a companion matrix based approach that is well aligned with the cipher structure.
- Utilizing this technique, we demonstrate an 8-round related-key differential characteristic for FUTURE with a probability of $2^{-56.4}$, which leads to a distinguisher with 2^{58} data, 2^{58} time, and negligible memory complexities. Moreover, we extend this distinguisher by one round and propose a 9-round key recovery attack with data, time, and memory complexities of 2^{59}, 2^{69}, and 2^{59}, respectively.
- In addition, we construct a full-round related-key boomerang distinguisher with practical complexity, followed by a key recovery attack that also operates within practical bounds, demonstrating a full-round break of the cipher. A comprehensive comparison of existing attack techniques and their respective complexities is presented in Table 1 (p. 6).

Outline of the Paper. The paper is structured as follows: Section 2 provides an overview of the FUTURE cipher. In Section 3, we present a brief introduction to related-key differential and boomerang cryptanalysis. Section 4 explains the bit-oriented MILP model used for the analysis. In Section 5, we apply this model to construct related-key differential, boomerang distinguishes, and then the key recovery for the FUTURE cipher. Finally, Section 6 concludes the paper with remarks and suggestions for future work. Our implementations are available at [2].

2 Description of Future

FUTURE is an SPN-based 64-bit lightweight block cipher designed to have applications on low hardware cost and latency. It has a key size of 128-bit.

The Round Function The round structure of the FUTURE cipher consists of four operations: SubCell, MixColumn, ShiftRow, and AddRoundKey.Notably, the MixColumn operation is omitted in the final round. The cipher processes a 64-bit input state S arranged as a 4×4 matrix, where each cell is a nibble (i.e., $s_i \in \{0,1\}^4$ for $0 \leq i \leq 15$), as shown in Figure 1a (p. 7).

SubCell The nonlinear transformation in the round function is defined by the SubCell operation, which applies a 4-bit S-box to each cell of the state matrix. This transformation is depicted in Figure 1b (p. 7).

MixColumn. The linear operation is represented by the finite field matrix multiplication involving the MDS (maximum distance separable) matrix (μ) and the state matrix, where the matrix elements are in $GF(2^4)$. The MDS matrix is illustrated in Figure 1c (p. 7). Matrix and vector multiplications are performed in the field $\mathbb{F}_{2^4}$, defined by the primitive polynomial $x^4 + x + 1$.

ShiftRow Each row $(row(i), i = 0, 1, 2, 3)$ of the state matrix is rotated to the right by i positions following the MixColumn operation.

AddRoundKey The 64-bit round keys (sub-keys) $k_i, i = 0, 1, \ldots, 10$ are XORed to the state S in each round. Additionally, the final round sub-key k_{10} is XORed with the state before producing the ciphertext.

Key Schedule In FUTURE encryption, a 128-bit secret key K is divided into two halves k_0 and k_1 for generating round and whitening keys. k_0 acts as the whitening key and generates each round sub-keys $k_i, i = 0, 1, \ldots, 10$ depending on whether i is even or odd. If i is even, k_0 is left-rotated by $5 \cdot \frac{i}{2}$ bits; if i is odd, k_1 undergoes the same left rotation. Left rotation involves circularly shifting bits. Additionally, except for the 5^{th} and 10^{th} rounds, a single '1' bit is XORed into specific positions within 4-bit cells during each encryption round, with these operations defined by round constants.

Attack	Types	#Rounds	Settings	Prob.	Complexity			Reference
					Data	Time	Memory	
Differential	Characteristic	5	Single-key	2^{-58}	–	–	–	[33]
MITM	Key Recovery	10	Single-key	–	2^{64}	2^{126}	2^{34}	[47]
MITM	Key Recovery	10	Single-key	–	2^{64}	2^{124}	2^{48}	[38]
Biclique	Key Recovery	10	Single-key	–	$\leq 2^{48}$	$2^{125.53}$	2^{32}	[44]
Yoyo	Distinguisher	6	Single-key	–	$2^{58.83}$	–	–	[39]
		8	Known-key	–	2^{15}	–	–	
Integral	Key Recovery	10	Single-key	–	2^{63}	$2^{123.7}$ Enc.	2^{16}	[58]
				–	2^{64}	2^{112} Enc.	$2^{16.1}$	
Differential	Distinguisher	8	Related-key	$2^{-56.4}$	2^{58}	2^{58} XOR	Negligible	This Work
	Key Recovery	9		–	2^{59}	2^{69}	2^{59}	This Work
Boomerang	Distinguisher	10	Related-key	$2^{-45.8}$	2^{48}	2^{48} XOR	Negligible	This Work
	Key Recovery	10	Related-key	–	2^{48}	2^{112} Enc./Dec.	Negligible	This Work
				–	2^{18}	2^{70} Enc./Dec.	2^{64}	This Work

Table 1: A Comparison of Different Attacks on FUTURE

3 Related-Key Cryptanalysis

A related-key attack [10] involves analyzing a cipher using multiple keys with known mathematical relationships between them. The attacker has access to encryption or decryption functions with these keys and aims to determine the actual secret keys. The simplest form uses a constant (ΔK) XOR relation between keys, such as $K_1 = K_0 \oplus \Delta K$. Related-key attacks offer more freedom compared to other attacks but can be more challenging to implement. Resistance to such attacks is crucial, as exemplified by the design goals of AES cipher. This paper employs differential attacks and boomerang attacks in a related-key context.

3.1 Related-Key Differential Cryptanalysis

Differential cryptanalysis is an effective method for analyzing and attacking symmetric-key ciphers by examining the differences between pairs of plaintexts and ciphertexts, known as "differential". Introduced by Biham and Shamir [15] in 1990, this technique seeks to identify specific patterns in these differences, termed "differential characteristics", that are unique to the encryption algorithm. By studying these patterns, cryptanalysts can infer the internal state of the cipher and, with sufficient data, uncover the secret key. This approach can lead to distinguishing attacks and then key-recovery attacks.

In related-key attack settings, an attacker can establish or enforce a relationship between multiple keys and has access to the corresponding encryption and decryption functions for all these keys. Consider a tuple $(\Delta_{\text{in}}, \Delta K, \Delta_{\text{out}})$ as an n-round related-key differential for a keyed round function f_K, where f_K^i represents the output after the i-th round for $i = 0, 1, \ldots, n-1$. This differential is valid if, for some plaintext P and key K, the equation $f_K^{n-1}(P) \oplus f_{K\oplus\Delta K}^{n-1}(P \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}$ holds. Let $S_{P,K}^i$ denote the internal state of the round function at round i with inputs P and K. The tuple $(\Delta_{\text{in}}, \Delta K, \Delta S_0, \ldots, \Delta S_{n-1} = \Delta_{\text{out}})$ is an n-round related-key differential characteristic if $(\Delta_{\text{in}}, \Delta_{\text{out}}, \Delta K)$ is an n-round related-key differential and for all i, $S_{P,K}^i \oplus S_{P\oplus\Delta_{\text{in}},K\oplus\Delta K}^i = \Delta S_i$.

Let $p = \Pr[(\Delta_{\text{in}}, \Delta K) \to \Delta_{\text{out}}]$ represent the probability that the related-key differential $(\Delta_{\text{in}}, \Delta K, \Delta_{\text{out}})$ holds. This implies that if $\frac{1}{p}$ number of plaintexts P and keys K are selected uniformly at random, the equation $f_K(P) \oplus f_{K\oplus\Delta K}(P \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}$ will be satisfied once on average with probability more than 60%.

3.2 Related-Key Boomerang Attack.

The boomerang attack, introduced by Wagner in [57], is a differential cryptanalysis method that combines two high-probability differentials to enhance the chances of breaking a cipher. This is described in Figure 2a (p. 8). For a block cipher $E = E_1 \circ E_0$, with differentials $\Delta_0 \xrightarrow{p} \Delta_1$ for E_0 and $\nabla_0 \xrightarrow{q} \nabla_1$ for E_1,

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

(a) The FUTURE State

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	1	3	0	2	7	E	4	D	9	A	C	6	F	5	8	B

(b) The S-box Table

$$\mu = \begin{bmatrix} 8 & 9 & 1 & 8 \\ 3 & 2 & 9 & 9 \\ 2 & 3 & 8 & 9 \\ 9 & 9 & 8 & 1 \end{bmatrix}$$

(c) The MixColumn Matrix

Fig. 1: The State Representation and S-box Table of FUTURE Cipher

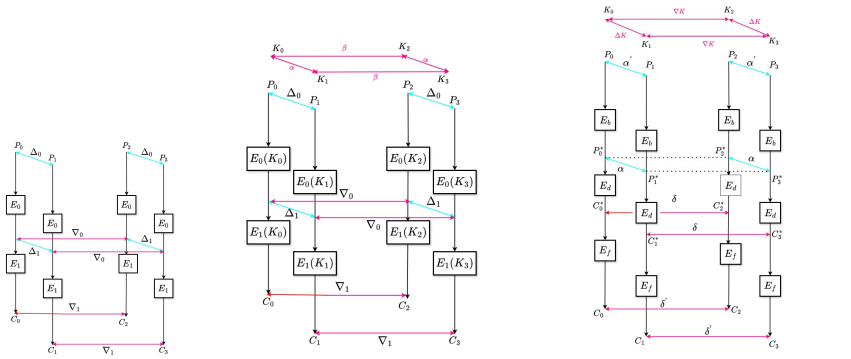
the attack checks if differential relationships hold, with an expected probability of success given by: $\Pr[E^{-1}(E(x) \oplus \nabla_1) \oplus E^{-1}(E(x \oplus \Delta_0) \oplus \nabla_1) = \Delta_0] = p^2 \cdot q^2$. The procedure for mounting the distinguisher in adaptive settings is as follows:

1. Request the ciphertexts $C_0 = E(P_0)$ and $C_1 = E(P_1)$, where $P_1 = P_0 \oplus \Delta_0$.
2. Request the plaintexts $P_2 = E^{-1}(C_2)$ and $P_3 = E^{-1}(C_3)$, where $C_2 = C_0 \oplus \nabla_1$ and $C_3 = C_1 \oplus \nabla_1$.
3. Verify if $P_2 \oplus P_3 = \Delta_0$?

To amplify this attack, the amplified boomerang attack [35] was proposed which works in a non-adaptive (chosen-plaintext attack) scenario. In this attack, the expected probability to get a right quartet will be $p^2 \cdot q^2 \cdot 2^{-n}$. Furthermore, in [11,12], they have pointed out that any value of Δ_1 and ∇_0 can be considered as long as $\Delta_1 \neq \nabla_0$. As a result, the probability of the right quartet is increased to $2^{-n} \cdot \hat{p}^2 \cdot \hat{q}^2$, where $\hat{p} = \sqrt{\sum_i \Pr^2(\Delta_0 \to \Delta_1^i)}$ and $\hat{q} = \sqrt{\sum_j \Pr^2(\nabla_0^j \to \nabla_1)}$. Note that this amplification can be done in the adaptive setting to increase the probability to $\hat{p}^2 \cdot \hat{q}^2$ from $p^2 \cdot q^2$. The sandwich attack [27] further refines this approach by decomposing the cipher into three parts and using the Boomerang Connectivity Table [25] (BCT) to systematically analyze the connections between input and output differences, improving the probability approximation of the distinguisher.

The related-key boomerang attack [13], depicted in Figure 2b (p. 8), utilizes both key and plaintext differences. It assumes that the upper sub-cipher E_0 follows a differential characteristic $\Delta_0 \xrightarrow{p} \Delta_1$ under a key difference $\alpha = K_0 \oplus K_1 = K_2 \oplus K_3$, while the lower sub-cipher E_1 has a differential characteristic $\nabla_0 \xrightarrow{q} \nabla_1$ under a key difference $\beta = K_0 \oplus K_2 = K_1 \oplus K_3$. A related-key distinguisher is built using four different unknown keys: K_0, $K_1 = K_0 \oplus \alpha$, $K_2 = K_0 \oplus \beta$, and $K_3 = K_1 \oplus \beta$. The related-key boomerang distinguisher in the adaptive scenario is executed as follows:

1. Request the ciphertext pairs (C_0, C_1), where $C_0 = E_{K_0}(P_0)$ and $C_1 = E_{K_1}(P_1)$, with $P_0 \oplus P_1 = \Delta_0, K_0 \oplus K_1 = \alpha$.



(a) Boomerang (b) Related-key Boomerang (c) Boomerang Key Recovery

Fig. 2: The Boomerang Framework

2. Request the plaintexts pairs (P_2, P_3), where $P_2 = E_{K_2}^{-1}(C_2)$ and $P_3 = E_{K_3}^{-1}(C_3)$, with $K_0 \oplus K_2 = \beta, K_1 \oplus K_3 = \beta, C_2 = C_0 \oplus \nabla_1$ and $C_3 = C_1 \oplus \nabla_1$.
3. Verify if $P_2 \oplus P_3 = \Delta_0$?

3.3 Related-Key Boomerang Key Recovery Attack

In general, based on the boomerang distinguisher, the attacker extends the input and/or output by one or two rounds and filters these rounds to enable key recovery in an adaptive setting. In [49,59], the authors proposed a generic and unified framework for key recovery in both rectangle and boomerang attacks. Following their approach, we present the boomerang key-recovery attack in the subsequent section.

4 Mixed-Integer Linear Programming models

Mixed-integer linear programming (MILP) has been successfully utilized to develop automated search algorithms for differential and linear cryptanalysis. Two primary modeling approaches exist for implementing ciphers: the word-oriented model and the bit-oriented model. In the word-oriented model, the cipher state is treated as a sequence of words, with each word represented as a binary variable. In contrast, the bit-oriented model represents each bit of the cipher state as a binary variable, ensuring the generation of the optimal and valid differential characteristics without any inconsistencies in the trail under the independent subkey assumption. The MILP constraints introduced in Mouha *et al.*'s method are insufficient to fully capture the differential propagation behavior in linear diffusion layers built from non-MDS codes. In [52], the authors first proposed a bit-oriented model specifically for SPN ciphers that utilize bit permutation-based linear layers.

In this section, we model the FUTURE cipher components as constraints to construct a bit-based MILP model for analyzing differential characteristics. To build this model, the S-box, permutation, and matrix multiplication over a finite field are represented by linear inequalities with binary variables. In [33], the authors present a bit-based MILP model for the FUTURE cipher aimed at searching for single-key differential and linear characteristics. However, the details provided are incomplete, particularly regarding the generation of linear inequalities for the S-box and the conversion of the MixColumn matrix to a binary matrix. By employing linear inequalities, one can construct a comprehensive bit-based MILP model that automatically identifies differential characteristics.

4.1 Constraints for SubCell Operation

For differential cryptanalysis using bit-based MILP, the goal is to generate a minimal number of constraints involving input and output bits of an S-box to capture the actual behavior of the differences according to the difference distribution table (DDT). Let us assume that $(x_0, \ldots, x_{n-1})$ and $(y_0, \ldots, y_{n-1})$ represent the input and output bit differences of an $n \times n$ S-box respectively. The

problem corresponds to modeling the fact that $(x_0, \ldots, x_{n-1}) \rightarrow (y_0, \ldots, y_{n-1})$ is a possible difference transition in a DDT. In this regard, two different approaches were proposed in 2014 by Sun *et al.* [54,53]. The first is a geometrical one and consists of computing the H-representation of the convex hull of the set of possible transitions. The second one is based on logical condition modeling. The first approach is to use the SageMath inequality generator by taking all the valid difference transition points from the DDT, and it generates a number of linear inequalities that satisfy all the valid difference transition points. However, the number of inequalities using SageMath is typically quite high, with many redundant inequalities. The authors of [54] applied a greedy algorithm to reduce the number of constraints. In this approach, the algorithm adds to the solution set, the best possible inequality that can remove the highest number of impossible difference transition points among those that have not been removed yet. Later, Sasaki and Todo in [46] proposed a new reduction algorithm to further reduce the number of constraints compared to the greedy approach. They proposed to model the problem of minimizing the set of inequalities that remove all the impossible difference transition points as an MILP problem itself and solve it by some solver. More precisely, their method consists of assigning a binary variable z_i to each inequality in which $z_i = 1$ denotes that inequality i is included in the system. Then for each impossible difference transition point j, add the corresponding constraints in the list $\mathcal{L}_j$ which leads to the inequality as $\sum_{i \in \mathcal{L}_j} z_i \geq 1$. Finally, the MILP solver is used for minimizing $\sum_i z_i$ giving a solution to optimize the constraints to capture the DDT of an S-box.

However, there are other works [56,37,43] that further reduce the number of constraints to capture DDT of S-box by proposing new approaches to generate additional inequalities, surpassing the SageMath inequality generator or using Boura and Coggia's approach. In this work, we follow the method outlined by Boura and Coggia [23] to generate the constraints that capture the DDT of a

I\O	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	4	4	0	0	0	0	0	4	4	0	0	0	0	0
2	0	4	0	4	0	2	0	2	0	0	0	0	0	2	0	2	0
3	0	0	0	4	2	0	2	0	0	0	4	0	0	2	0	2
4	0	0	0	0	4	0	4	0	0	0	0	0	0	4	0	4
5	0	0	0	0	2	2	2	2	0	0	0	0	2	2	2	2
6	0	4	0	4	0	2	0	2	0	0	0	0	2	0	2	0
7	0	0	4	0	0	2	0	2	0	4	0	0	2	0	2	0
8	0	0	0	0	2	0	2	0	4	2	0	2	4	0	0	0
9	0	2	2	0	0	2	2	0	0	0	2	2	0	0	2	2
a	0	0	0	0	0	4	0	0	4	2	0	2	0	2	0	2
b	0	2	2	0	0	0	2	2	0	0	2	2	2	0	0	2
c	0	0	0	0	2	0	2	0	4	2	0	2	0	0	4	0
d	0	2	2	0	2	0	0	2	0	0	2	2	2	2	0	0
e	0	0	0	0	0	0	0	4	4	2	0	2	0	2	0	2
f	0	2	2	0	2	2	0	0	0	0	2	2	0	2	2	0

Table 2: DDT of S-box

S-box	# Inequalities		
	[23, Algorithm 1]		Revised Algorithm 1
	Claimed	Actual[3]	
PRESENT	17	22	17
PRINCE	19	24	19
MIDORI S0	16	19	17
MIDORI S1	20	23	20
KLEIN	19	22	19
PICCOLO	16	20	16
LBlock s0	17	21	17
LBlock s1	17	22	17
LBlock s2	17	22	17
LBlock s3	17	22	17
LBlock s4	17	21	17
LBlock s5	17	22	17
LBlock s6	17	22	17
LBlock s7	17	22	17
LBlock s8	17	21	17
LBlock s9	17	21	17
TWINE	19	26	20

Table 3: S-box Inequality Comparison

Algorithm 1 Revised Boura and Coggia's Approach to Compute a Set of Inequalities from DDT of an S-box.

1: **procedure** COMPUTECONSTRAINTS($\mathcal{VDP}, k(\geq 2)$)
2: $\mathcal{H}_{set} \leftarrow$ `inequality_generator(`$\mathcal{VDP}$`)`
3: $\mathcal{C}_{set} \leftarrow \mathcal{H}_{set}$
4: $\mathcal{D}_{set} \leftarrow \{\}$
5: **for** all $\alpha \in \mathcal{VDP}$ **do**
6: $\mathcal{H}^{\alpha}_{set} = \{C \in \mathcal{H}_{set} | C(\alpha) = 0\}$
7: **for** all $\mathcal{H}^{\alpha}_{set}, \alpha \in \mathcal{VDP}$ **do**
8: **if** $k \geq |\mathcal{H}^{\alpha}_{set}|$ **then**
9: **for** all $\{C_1, \cdots, C_k\} \subseteq \mathcal{H}^{\alpha}_{set}$ **do**
10: $C_{new} = C_1 + \cdots + C_k$
11: Add C_{new} into $\mathcal{D}_{set}$
12: **for** all constraints $C_i \in \mathcal{H}_{set}, 1 \leq i \leq |\mathcal{H}_{set}|$ **do**
13: Construct the set $S_i = \{\beta \in \mathcal{IDP} | C_i(\beta) < 0\}$
14: $\mathcal{L} = [S_1, \cdots, S_{|\mathcal{H}_{set}|}]$ ▷ A list of sets
15: **for** all constraints $C_j \in \mathcal{D}_{set}, 1 \leq j \leq |\mathcal{D}_{set}|$ **do**
16: Construct the set $S = \{\beta \in \mathcal{IDP} | C_j(\beta) < 0\}$
17: **if** $\nexists$ any S_i from $\mathcal{L}$ such that $S \subseteq S_i$ **then**
18: Add S in the list $\mathcal{L}$
19: Add C_j in $\mathcal{C}_{set}$
20: Apply Sasaki *et al.*'s [46] approach from $\mathcal{C}_{set}$ to finally get the optimal number of constraints to capture the DDT of S-box

FUTURE S-box. The DDT of the FUTURE S-box is provided in Table 2 (p. 10). The algorithm presented by Boura and Coggia [23, Algorithm 1] for deriving a set of inequalities from the DDT of an S-box lacks certain details. Specifically, in step 7, the authors state that if C_{new} removes a new set of impossible transitions, C_{new} should be added as new constraints to capture the DDT of the S-box. However, the precise meaning of C_{new} is unclear. It is not explicitly explained whether the set S of impossible transitions from C_{new} should contain elements distinct from the elements of S_i, for all i, where each S_i is a set of impossible transitions for every constraint in $\mathcal{C}_{set}$. To clarify this, we revisited their approach and outlined the complete steps necessary to generate the S-box constraints based on our understanding.

Suppose, a difference transition $x \rightarrow y, x, y \in \mathbb{F}_2^n$ through the S-box can be seen as a vector of $\mathbb{F}_2^{2n}$, involving $2n$ binary variables represented as $(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ or as $(x_0, \ldots, x_{2n-1})$. Let $\mathcal{VDP}$ and $\mathcal{IDP}$ denote a set of valid and impossible difference transition points according to the DDT of FUTURE S-box. For example, $0x12 \in \mathcal{VDP}$ and $0x11 \in \mathcal{IDP}$ are the valid and invalid difference points according to the DDT in Table 2 (p. 10). Given $\mathcal{VDP}$ to the `inequality_generator()` function in the `sage.geometry.polyhedron` class of SageMath returns a list of inequalities as the H-set representation of the convex hull of all possible transitions in a DDT. For FUTURE S-box, we get returns 214 inequalities. We denote this list of inequalities as $\mathcal{H}_{set}$. This set $\mathcal{H}_{set}$ has the following properties: (1.) each $\alpha \in \mathcal{VDP}$ must satisfy all the constraints in $\mathcal{H}_{set}$ and (2.) each $\beta \in \mathcal{IDP}$ will not be satisfied by at least one of the constraints in $\mathcal{H}_{set}$. The interesting point here is that the addition of any number of constraints always maintains the above two properties. Although, adding the constraints by choosing all the subsets (of cardinality k) of constraints from $\mathcal{H}_{set}$ and then

append it to $\mathcal{H}_{set}$ can increase the list $\mathcal{H}_{set}$ remarkably high. Instead, for each $\alpha \in \mathcal{VDP}$, the authors choose the constraints from $\mathcal{H}_{set}$ which satisfies by the point α and store them in another list $\mathcal{H}^{\alpha}_{set}$. Then, for each set $\mathcal{H}^{\alpha}_{set}$, choose $\binom{|\mathcal{H}^{\alpha}_{set}|}{k}$ constraints, denoted $C_1, \ldots, C_k$, and add $C_{new} = C_1 + \cdots + C_k$ to a set $\mathcal{D}_{set}$. A list of sets, $\mathcal{L}$, is constructed, where each entry contains the impossible transition points for each constraint in $\mathcal{D}_{set}$. Finally, the constraints from $\mathcal{D}_{set}$ are filtered and added to $\mathcal{L}$, ensuring that the set of impossible transition points S is not a subset of any set already in $\mathcal{L}$. After this filtration, Sasaki and Todo's method is applied to the selected constraints to generate the optimal constraints for capturing the DDT of the S-box. These steps are detailed in Algorithm 1.

Using this algorithm with $k = 2$, we initially generate 971 constraints for C_{set}, which are eventually reduced to 17 constraints by applying the method proposed by Sasaki and Todo. In comparison, [23, Algorithm 1] reports approximately 500 initial constraints for the PRESENT S-box with $k = 2$, which are claimed to be reduced to 17 using the same technique. However, upon verifying their implementation[3], we observed that the number of resulting constraints is around 22, not 17 as stated.

By applying our revised method described in Algorithm 1, we obtained 1138 initial constraints for $\mathcal{C}_{new}$ for the PRESENT S-box, which were similarly reduced to 17 using Sasaki and Todo's method. Using this revised algorithm, we successfully derived 17 constraints for both the FUTURE and PRESENT S-boxes to accurately capture their DDT Additionally, when our revised algorithm is applied to all 4×4 S-boxes, the resulting number of constraints matches the claims made by Boura and Coggia, as summarized in Table 3 (p. 10). However, it is worth noting from Table 3 (p. 10) that while the results match for most S-boxes, the values differ for the MIDORI S0 and TWINE S-boxes when using the revised Algorithm 1.

4.2 Constraints for MixColumn Operation

The MixColumn operation is a linear transformation represented by a matrix, defined over an extension field of $\mathbb{F}_2$, and is widely used in block cipher design. In SPN-based block ciphers, MDS matrices, originating from maximum distance separable codes, are commonly employed to achieve optimal diffusion. For more lightweight designs, near-MDS matrices are often used to provide a trade-off between optimal diffusion and implementation cost. In general, most SPN-based block ciphers employ either MDS or near-MDS matrices for the MixColumn operation, typically defined over finite fields. To generate constraints for the matrix-based MixColumn operation, two approaches can be considered: word-based and bit-based MILP modeling. In word-based modeling, we model the MixColumn matrix multiplication using its branch number, i.e., a lower bound

[3] We refer to the code available at https://github.com/dnlcog/efficient_milp_modelings, which accompanies the paper titled "Efficient MILP Modelings for S-boxes and Linear Layers of SPN Ciphers," published in ToSC 2020.

$$\mathbf{1}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},\quad \mathbf{2}=\begin{bmatrix}0&1&0&0\\0&0&1&0\\1&0&0&1\\1&0&0&0\end{bmatrix},\quad \mathbf{3}=\begin{bmatrix}1&1&0&0\\0&1&1&0\\1&0&1&1\\1&0&0&1\end{bmatrix},\quad \mathbf{8}=\begin{bmatrix}1&0&0&1\\1&1&0&0\\0&1&1&0\\0&0&1&0\end{bmatrix},\quad \mathbf{9}=\begin{bmatrix}1&1&0&0\\0&1&1&0\\1&0&1&1\\1&0&0&1\end{bmatrix}$$

Fig. 3: 4×4 binary matrix representation of the field elements in μ

on the number of active S-boxes. In contrast, in bit-based modeling, if the MixColumn operation uses a matrix μ, then it must be converted into a binary matrix over the base field $\mathbb{F}_2$, which is referred to as the primitive representation of μ.

In [50], the authors give a short description of the primitive representation of different linear layers with an example of AES as a 128×128 binary matrix μ for MDS matrix based linear layers. However, in [51], Sun *et al.* provided a method to obtain a primitive representation using linear transformations with matrix representation in $\mathbb{F}_2$. The high level idea is that for a given linear transformation $h : \mathbb{F}_2^n \to \mathbb{F}_2^n$ the aim is to find $n \times n$ matrix M_h^{PR} satisfying $h(x)^T = M_h^{PR} \cdot x^T$. In [33], the authors follow the same approach by Sun *et al.* [51] to utilize the primitive representation. In this work, we thoroughly explore how to efficiently compute a primitive representation of μ using a companion matrix that is compatible with the cipher's bit format, whether it follows a least significant bit (LSB) to the most significant bit (MSB) order or an MSB to LSB order. Finally, to model matrix multiplication in MILP, we might need several binary XOR operations.

For 1-XOR operation, $c = a \oplus b, a, b, c \in \{0, 1\}$, Mouha *et al.* [40] modeled it using 4 constraints and 3 variables as $a + b + c \geq 2d_1, d_1 \geq a, b, c$, where d_1 is a dummy variable. Similarly, for $d = a \oplus b \oplus c, a, b, c, d \in \{0, 1\}$, known as a 2-XOR operation, the approach requires 8 constraints and 5 variables. Yin *et al.* [60] showed a method to model it using 8 constraints and 4 variables. However, Fu *et al.* [29] efficiently model the 1-XOR operation using only one constraint $a + b + c + d = 2d_1, a, b, c, d, d_1 \in \{0, 1\}$. Based on this approach, the authors in [33] extend this approach to model the n-XOR operations $a_0 \oplus \ldots \oplus a_{n-1} = b$, as follows.

$$a_0 + \ldots + a_{n-1} + b = \begin{cases} (n+2)d_1 - (nd_2 + (n-2)d_3 + \ldots + 2d_{\frac{n}{2}+1}) & \text{if } n \text{ is even} \\ (n+1)d_1 - ((n-1)d_2 + (n-3)d_3 + \ldots + 2d_{\frac{n-1}{2}+1}) & \text{if } n \text{ is odd} \end{cases}$$

In our model, we adopt this approach for n-XOR operations for MixColumn matrix multiplication. In FUTURE cipher, the multiplication by 4×4 MDS matrix μ is performed over $GF(2^4)$, defined by the primitive polynomial x^4+x+1. Let, α be a primitive element, serving as a root of the polynomial $x^4 + x + 1$. The MDS matrix μ includes field elements **1, 2, 3, 8**, and **9** from $GF(2^4)$. To model the multiplication by μ for bit-oriented MILP, we need to convert the 4×4 MDS matrix μ over $GF(2^4)$ into a primitive representation of μ, i.e., a 16×16 binary matrix over the base field $\mathbb{F}_2$. Using linear maps with matrix representation, the authors [33] express the corresponding 4×4 binary matrices of these field elements in Figure 3 (p. 13).

Note that, the primitive representation of μ by replacing the corresponding field elements **1, 2, 3, 8**, and **9** is compatible with cipher representation from MSB to LSB. However, this primitive representation would not be compatible

$$M_1 = \begin{bmatrix}
1&0&0&1&0&0&0&1&1&0&0&0&1&0&0&1\\
1&1&0&0&1&0&0&0&0&1&0&0&1&1&0&0\\
0&1&1&0&0&1&0&0&0&0&1&0&0&1&1&0\\
0&0&1&0&0&0&1&1&0&0&0&1&0&0&1&0\\
1&1&0&0&0&1&0&0&0&0&0&1&0&0&0&1\\
0&1&1&0&0&0&1&0&1&0&0&0&1&0&0&0\\
1&0&1&1&1&0&0&1&0&1&0&0&0&1&0&0\\
1&0&0&1&1&0&0&0&0&0&1&1&0&0&1&1\\
0&1&0&0&1&1&0&0&1&0&0&1&0&0&0&1\\
0&0&1&0&0&1&1&0&1&1&0&0&1&0&0&0\\
1&0&0&1&1&0&1&1&0&1&1&0&0&1&0&0\\
1&0&0&0&1&0&0&1&0&0&1&0&0&0&1&1\\
0&0&0&1&0&0&0&1&1&0&0&1&1&0&0&0\\
1&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0\\
0&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0\\
0&0&1&1&0&0&1&1&0&0&1&0&0&0&0&1
\end{bmatrix}$$

(a) For MSB to LSB

$$M_2 = \begin{bmatrix}
0&1&0&0&1&1&0&0&1&0&0&0&0&1&0&0\\
0&1&1&0&0&0&1&0&0&1&0&0&0&1&1&0\\
0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&1\\
1&0&0&1&1&0&0&0&0&0&0&1&1&0&0&1\\
1&0&0&1&0&0&0&1&1&1&0&0&1&1&0&0\\
1&1&0&1&1&0&0&1&0&0&1&0&0&0&1&0\\
0&1&1&0&0&1&0&0&0&0&0&1&0&0&0&1\\
0&0&1&1&0&0&1&0&1&0&0&0&1&0&0&0\\
0&0&0&1&1&0&0&1&0&1&0&0&1&1&0&0\\
1&0&0&1&1&1&0&1&0&1&1&0&0&0&1&0\\
0&1&0&0&0&1&1&0&0&0&1&1&0&0&0&1\\
0&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0\\
1&1&0&0&1&1&0&0&0&1&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&1&1&0&0&1&0&0\\
0&0&0&1&0&0&0&1&0&0&1&1&0&0&1&0\\
1&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1
\end{bmatrix}$$

(b) For LSB to MSB

Fig. 4: The Primitive Representation of μ When the State is Represented from LSB to MSB or from MSB to LSB

with the cipher representation from LSB to MSB. Here we will describe how to construct the primitive representation of μ using a companion matrix to model the cipher which would be compatible in both ways. We know that $\mathbf{2} = 0010 \in GF(2^4)$ is the root α of the primitive polynomial $x^4 + x + 1$ over $GF(2^4)$. Let us assume that, the state of the cipher represented from MSB to LSB, i.e., $S = s_{63}||s_{62}||\ldots||s_0$. In this case, the companion matrix representation of α of the monic primitive polynomial $c_0 + c_1x + c_2x^2 + c_3x^3 + x^4, c_i \in \mathbb{F}_2$ can be written as

$$2 = \alpha = \begin{bmatrix} c_3&1&0&0\\ c_2&0&1&0\\ c_1&0&0&1\\ c_0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 1&0&0&1\\ 1&0&0&0 \end{bmatrix} \text{ with } 1 = \alpha^0 = \begin{bmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{bmatrix}.$$

The other field elements of μ can be computed as $3 = \alpha + \alpha^0, 8 = \alpha^3,$ and $9 = \alpha^3 + \alpha^0$. Thus the 16×16 binary matrix M_1, representing the primitive form of μ, corresponds to the cipher's bit representation from MSB to LSB over $\mathbb{F}_2$ is given in Figure 4a (p. 14). On the other hand, if the cipher is represented from LSB to MSB, i.e., $S = s_0||s_1||\ldots||s_{63}$, then the companion matrix representation of α of the monic primitive polynomial $c_0 + c_1x + c_2x^2 + c_3x^3 + x^4, c_i \in \mathbb{F}_2$ can be written as

$$2 = \alpha = \begin{bmatrix} 0&0&0&c_0\\ 1&0&0&c_1\\ 0&1&0&c_2\\ 0&0&1&c_3 \end{bmatrix} = \begin{bmatrix} 0&0&0&1\\ 1&0&0&1\\ 0&1&0&0\\ 0&0&1&0 \end{bmatrix} \text{ with } 1 = \alpha^0 = \begin{bmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{bmatrix}.$$

Similarly, the other field elements of μ can be computed in the similar way. Then the 16×16 binary matrix M_2, serving as the primitive representation of μ, corresponds to the bit order from LSB to MSB over $\mathbb{F}_2$ is given in Figure 4b (p. 14). Apart from these two companion matrix representations, using any other form of companion matrix in the model either by transposing it or by reordering the rows/columns of the above two matrices would not be compatible with the

cipher representation. This is because altering the companion matrix used to construct the primitive representation M would change the bit sequences. As a result, multiplying M (a 16×16 matrix) by the state (a 16×4 matrix) would not produce a correct state consistent with the cipher's structure.

Finally, the 4×4 state matrix of FUTURE cipher can be further deduced to 16×4 binary matrix. Let, the 16-bit column vectors as $y = (y_0, y_1, \cdots, y_{15})^T$ and $t = (t_0, t_1, \cdots, t_{15})^T$, where $t = M \cdot y$. Therefore, for all four columns of the state, a total of $16 \cdot 4 = 64$ constraints are required to represent the differential propagation through the MixColumn operation.

4.3 Constraints for ShiftRow Operation

The ShiftRow operation performs a row-wise shift at the nibble level, which can be represented as a bit-wise permutation $\pi : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$. To model this operation, the binary variables resulting from the MixColumn step are permuted by ShiftRow. After that, 64 new binary variables are introduced and assigned to these permuted values. If x_i and y_i represent the input and output binary variables, respectively, the constraint $y_i = \pi(x_i)$ is added to the MILP model. To reduce the number of constraints, the output binary variables can be directly permuted according to the ShiftRow bit-wise permutation π while modeling the MixColumn operation.

4.4 Constraints for AddRoundKey Operation

The AddRoundKey operation directly XORs the state bits with the round keys. In the bit-oriented related-key model, the state difference is XORed directly with the sub-key difference. To model the XOR operation between the key and state differences ($c = a \oplus b$), we use the following constraints without introducing dummy variables: $c \geq a - b,\ c \geq b - a,\ c \leq a + b,\ c \leq 2 - a - b$.

4.5 Construction of the Objective Function

The objective function of a MILP model can be designed to minimize the number of active S-boxes. In a bit-oriented MILP model, there will be no inconsistencies in the propagation of bit differences through rounds, provided the S-box constraints accurately represent its DDT. To account for an active S-box in a bit-based model, we introduce a dummy variable along with four additional constraints for each S-box. Let the input bit differences of an S-box be represented by $(\delta x_3, \delta x_2, \delta x_1, \delta x_0)$ and define a new binary dummy variable, d_0. This dummy variable d_0 will determine whether the S-box is active or inactive based on the following constraints: $\sum_{i=0}^{n-1} \delta x_i \geq d_0, d_0 \geq \delta x_i, i = 0, 1, 2, 3$. The objective function is then to minimize the sum of the dummy variables d_i for each S-box position in the rounds. To calculate the probability of the differential trail produced by the model, the probability of each active S-box from the DDT must be checked,

and the overall probability of the differential characteristic is obtained by multiplying these values. For a clearer understanding of the MILP model applied to the FUTURE cipher, we provide our MILP model implementation in [2].

5 Results

This section presents an analysis of the differential characteristics of FUTURE in the related-key attack setting. The differential characteristics are determined using the methodology in Section 4.

5.1 Related-Key Differential Distinguishers

To search for differential characteristics of FUTURE[4] in the related-key setting, we constructed a MILP model using the Gurobi Python API [1]. The necessary constraints for building the model across rounds are outlined in Section 4. A summary of the related-key characteristics for different rounds, along with their probabilities, is presented in Table 4 (p. 17). This model enables us to search for related-key differential characteristics up to 7 rounds. However, due to the large number of constraints and variables, the model struggles to complete the search for 6 and 7 rounds. For the 7-round case, we identified several differential characteristics with 22 active S-boxes and a probability of approximately 2^{-48}. The 7-round differential characteristic corresponds to the first seven rounds of the 8-round trail shown in Figure 5 (p. 18). To confirm the individual probabilities for each S-box, the DDT is provided in Table 2 (p. 10). Additionally, we identified three distinct clustering effects for the 7-round differential characteristic (see Table 4 (p. 17)) from 50 different solutions generated by the model, where the characteristics share the same input and output. This clustering further increases the probability of the differential characteristic to $3 \cdot 2^{-48} \approx 2^{-46.4}$. Furthermore, for 8 rounds, the solver could not reach a near-optimal solution due to the large number of constraints and variables. Therefore, we extended the 7-round differential characteristic by adding an additional round. Using the MILP model, we verified that seven S-boxes are active in the final round, with a probability of 2^{-17}. Consequently, the overall probability for the 8-round differential characteristic (see Figure 5 (p. 18)) becomes $2^{-46.4} \cdot 2^{-17} = 2^{-63.4}$. For the distinguishing attack, the number of required plaintext pairs should be more than $2^{63.4}$ to ensure a high enough success probability for the distinguisher. Since the attacker only has 2^{63} plaintext pairs with a fixed input difference, an alternative strategy is to consider truncated differences in the last round to increase the overall trail probability.

In Figure 5 (p. 18), the S-box input differences at the last round are $1, f, 3, 9, a, 2$. According to the DDT (Table 2 (p. 10)), the input difference 1 transitions to output differences 2 or 3 with probability 2^{-1}, i.e., $\Pr[1 \xrightarrow{\text{S-box}} 001*] = 2^{-1}$. Similarly, the probabilities for other differential transitions $2 \rightarrow 00*1,\ 3 \rightarrow 1*10,\ 9 \rightarrow$

[4] In the single-key setting, our MILP framework produces the same results as those reported in [33].

#Rounds	#Active S-box	Differential		Probability
		Input Differences	Output Difference	
4	2	$\Delta P = 0x2300\ 0010\ 0001\ 0000$ $\Delta K_0 = 0x2300\ 0010\ 0001\ 0000$ $\Delta K_1 = 0x0004\ 0000\ 0000\ 0000$	$\Delta C = 0x0000\ 4000\ 0440\ 008c$	2^{-5}
5	6	$\Delta P = 0x1201\ 01c0\ 0000\ 0000$ $\Delta K_0 = 0x0000\ 01c0\ 0000\ 0000$ $\Delta K_1 = 0x0000\ 0200\ 0020\ 0002$	$\Delta C = 0x0008\ 0000\ 8000\ 0802$	2^{-14}
6	11	$\Delta P = 0xc840\ 0000\ 0000\ 0005$ $\Delta K_0 = 0xc840\ 0000\ 0000\ 0005$ $\Delta K_1 = 0x0000\ 0000\ 0000\ 2480$	$\Delta C = 0x0200\ 0010\ 0001\ d420$	2^{-27}
7	22	$\Delta P = 0x0000\ 8000\ 1180\ 0008$ $\Delta K_0 = 0x0000\ 8000\ 1180\ 0008$ $\Delta K_1 = 0x0000\ 0000\ 000b\ 0000$	$\Delta C = 0x0000\ 0007\ 1002\ 0000$	2^{-48}
8	29	$\Delta P = 0x0000\ 8000\ 1180\ 0008$ $\Delta K_0 = 0x0000\ 8000\ 1180\ 0008$ $\Delta K_1 = 0x0000\ 0000\ 000b\ 0000$	$\Delta C = 0x0144\ 2a00\ 0089\ 9108$	$2^{-63.4}$

Table 4: Related-Key Differentials for Different Rounds of FUTURE using Bit-Oriented MILP Model

$111*$, $a \rightarrow 100*$, and $f \rightarrow 101*$ are each improved approximately by a factor of 2. Thus, by considering truncated differentials, the improved probability for the 8-round related-key differential becomes $2^{-56.4}$, which is an improvement approximately by a factor of 2^7.

This can be directly leveraged to mount an attack on the security notion of indistinguishability against FUTURE reduced to 8 rounds. In this distinguisher, the attacker requires 2^{57} plaintext pairs, thus 2^{58} data complexity, effectively exhausting the entire plaintext space. The offline time complexity amounts to 2^{58} XOR operations. The attack does not necessitate storing intermediate values, except for one ciphertext when $C_i \oplus C_i^{'} = \Delta C$ is satisfied. Therefore, the memory complexity is minimal, or effectively negligible. Furthermore, the attacker can reduce the data complexity by restricting the input space along with its associated probability. Additionally, to further minimize the data complexity for the boomerang attack, an effective strategy is to utilize structured input sets. However, in this case, the memory complexity will increase. Finally, we propose a 9-round key-recovery attack based on the 8-round distinguisher, which is described in Section 5.2.

Experimental Verification. As previously mentioned, the bit-oriented MILP model guarantees no inconsistencies in the solutions it returns. During our experiments, we evaluated differential characteristics using automated tools. We were able to successfully verify several characteristics whose probabilities are higher than 2^{-32}, confirming their practical relevance and effectiveness. The implementation used to verify these characteristics is available in [2].

5.2 9-Round Related-Key Differential Key Recovery Attack

Based on an 8-round related-key distinguisher with a probability of $2^{-56.4}$, we extend the attack to 9 rounds by appending one additional round at the top. The overall idea is shown in Figure 6. The probability of reaching the 8-round distinguisher through the first round is 2^{-48} (as there are 22 active S-boxes), so the total probability of the full 9-round differential path is $2^{-104.4}$.

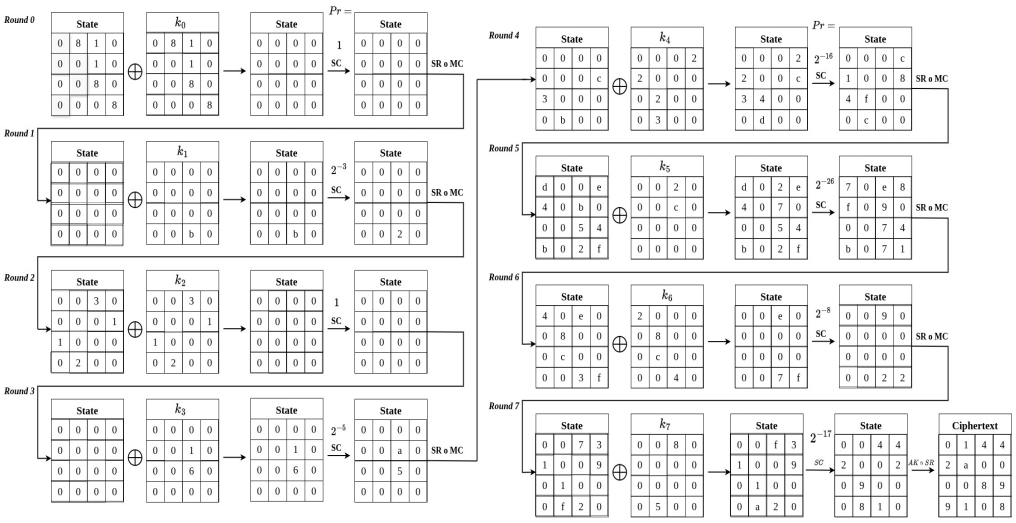


Fig. 5: Eight Round Related-Key Differential Characteristic of FUTURE Cipher

In this attack, we choose a structure of plaintexts that varies over the 12 active nibble positions (marked in green) and keeps the remaining 4 nibbles with fixed difference. This structure gives us 2^{95} plaintext pairs. To gather enough data for the full 9-round path (more than $2^{104.4}$ pairs), we need to repeat the experiment for 2^{10} times.

In each experiment, we collect 2^{95} plaintext pairs and their corresponding ciphertexts, and filter out the pairs that do not match the expected differences at the output (marked in blue).

This attack requires guessing 19 key nibble values, 12 from k_0 and 7 from k_9 (shown in red). For each guess, we maintain a counter, and the correct key nibble values will have the highest counts. Once the number of data pairs reaches around $2^{104.4}$, the right key values are expected to with a counter value more than 1. Also, it should be noted that generating two right pairs is enough to identify the right key when the *signal-to-noise* ratio is greater than 1. Therefore, we run about 2^{10} experiments to recover the correct key nibbles.

The complete steps of the attack are described next.

A. Initialize counters for all possible values of $k_0[1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15]$, $k_9[1, 6, 7, 8, 11, 12, 13]$.
B. Repeat the experiment for 2^{10} times.
   B.1. Ask for a structure of 2^{48} plaintext-ciphertext pairs under the keys K and $K^{'}$.
   B.2. Store the plaintexts P associated to K in a hash table T indexed by $[SR^{-1}(C[i]), i \in \{0, 2, 3, 4, 5, 9, 10, 14, 15\}]$ and the plaintexts $P^{'}$ associated to $K^{'}$ in a hash table $T^{'}$ indexed by $[SR^{-1}(C^{'}[i]), i \in \{0, 2, 3, 4, 5, 9, 10, 14, 15\}]$.
   B.3. The hash tables are used to generate the pairs $(P, P^{'})$ that may follow the differential characteristic. For each 36-bit value x, generate all the pairs $(P, P^{'})$

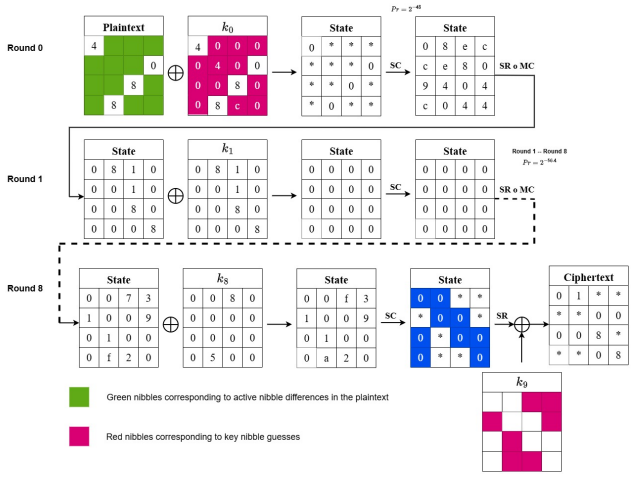


Fig. 6: Differential Key Recovery Attacks on 9 Rounds of FUTURE. Blue bytes indicate fixed nibble differences in the last round. Red bytes represent key nibble values involved in the key recovery process. Green bytes correspond to plaintext nibbles that vary within the structure.

such that $P \in T[x]$ and $P^{'} \in T^{'}[x]$. We expect $2^{2 \cdot 48 - 9 \cdot 4 - 1} = 2^{59}$ such pairs. For each pair, our goal is now to recover the round-key nibble values (shown in red colored rectangles) in Figure 6.

B.4. For each pair:

B.4.1. Compute the key nibbles of $k_0[1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15]$, $k_9[1, 6, 7, 8, 11, 12, 13]$ from the 0^{th} and 8^{th} rounds' active Sboxes.

B.4.2. Update the key counters.

C. Select the key values with counter values more than 2. Together with the all possible values for another 13 nibbles outside (i.e., which are not guessed yet), test exhaustively to find the right master key.

Therefore, the data complexity of this attack is $2^{49+10} = 2^{59}$, since we perform about 2^{10} experiments, each using two structures of $2 \times 2^{48} = 2^{49}$ plaintext queries under both K and $K^{'}$. The memory complexity is 2^{76}, determined by the maximum of 2^{76}, which comes from either storing the counters for 19 key nibbles ($2^{19 \times 4} = 2^{76}$) or handling 2^{59} data pairs. Finally, the time complexity is approximately $2^{10} \cdot (2^{49} + 2^{59}) + 2^{52} = 2^{69}$. To reduce the memory complexity further, the attacker can guess 12 key nibble of k_0 and sieve the data pairs before starting the whole attack. This can lowering the memory complexity to 2^{59}. Thus, the final attack complexity ($data, time$, and $memory$) would be $2^{59}, 2^{69}$, and 2^{59}, respectively.

Signal-to-Noise Ratio For any key recovery attack, the *signal-to-noise* measures the certainty of obtaining the correct key from the right pairs instead of the wrong pairs. The *signal-to-noise* ratio [16], [45], denoted by S/N, is the number

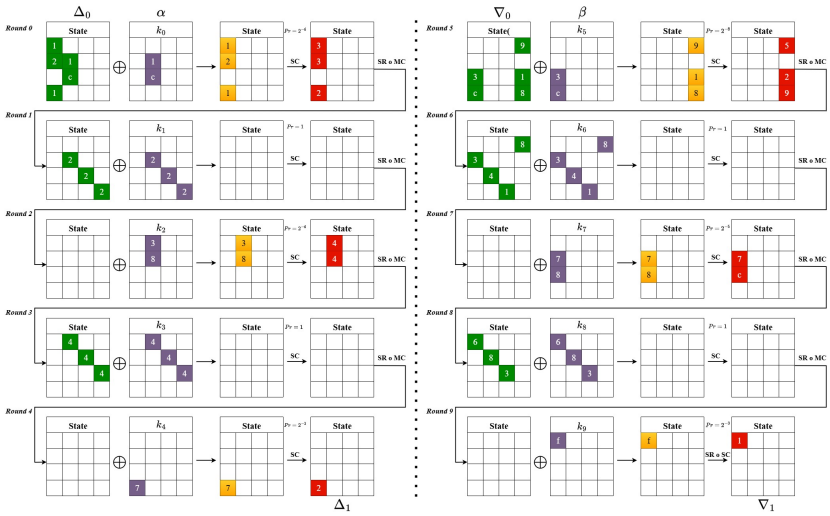


Fig. 7: Full Round Boomerang Distinguisher

of right pairs needed to recover the key in the distinguisher. The value of S/N can be calculated as: $S/N = \frac{2^k \cdot p}{\alpha \cdot \beta}$, where k denotes the size of the guessed key bits, p denotes the probability of distinguisher, β denotes the probability that a randomly generated ciphertext pair is a candidate of the one satisfying the differential characteristic, called the filtering power, and α denotes the number of all combinations of suggestions of guessed key nibbles. When S/N is greater than 1, collecting two right pairs is enough to identify the right key by using the ranking test.

According to the DDT in Table 2 (p. 10), non-zero input differences to the S-box yield either 4, 6, or 8 possible output differences. From Figure 6 (p. 19), the input differences to the S-box at the 8^{th} round are $1, 1, a, 2, f, 3, 9$, and the corresponding numbers of possible output differences (from the DDT) are $4, 4, 6, 6, 8, 6, 8$, respectively. Thus, with these ciphertext nibble positions, the *filtering power* β can be computed as $\beta = (\frac{4}{16})^2 \times (\frac{6}{16})^2 \times (\frac{8}{16})^2 \approx 2^{-10}$.

Also, from the DDT in Table 2 (p. 10), the ratio of entries equal to 2 versus 4 is $24 : 72 = 1 : 3$. Since this attack requires guessing 19 key nibbles, the number of key candidates can be estimated as $\alpha \approx (2^2)^5 \times (2)^{14} = 2^{24}$.

Therefore, in this attack, $k = 76, p = 2^{-56.4}, \beta \approx 2^{-10}$, and $\alpha \approx 2^{24}$. So, we have $S/N = 2^{5.6} \gg 1$.

5.3 Related-Key Boomerang Distinguisher

In this section, we construct boomerang distinguishers for FUTURE over different rounds. Using our automated search model, we identify two distinct related-key differential characteristics for five rounds each, corresponding to the upper and lower halves of the boomerang. These characteristics have probabilities of 2^{-14}

and 2^{-16}, respectively. For clarity, let $\Delta_0 \xrightarrow{Upper\ Trail} \Delta_1$ and $\nabla_0 \xrightarrow{Lower\ Trail} \nabla_1$ denote the differential characteristics for the upper and lower five rounds of the full boomerang, respectively. Additionally, let α and β represent the differences in the round keys of the upper and lower trails. The full round boomerang structure is illustrated in Figure 7 (p. 20). Thus, the distinguishing probability for this boomerang is given by $(2^{-14})^2 \cdot (2^{-16})^2 = 2^{-60}$, which can be utilized to perform a distinguishing attack on the full-round FUTURE cipher under adaptively chosen plaintext and ciphertext (ACPC) settings. In this distinguisher, the attacker needs 2^{60} plaintext-ciphertext quartets, which corresponds to 2^{62} data in total. The offline time complexity is $4 \cdot 2^{60} = 2^{62}$ XOR operations. The attack does not require storing intermediate values, except for one plaintext P_i when $P_2^i \oplus P_3^i = \Delta_0$ is satisfied. As a result, the memory complexity is negligible.

Checking Incompatibilities in the Boomerang In boomerang-style attacks, selecting compatible differential characteristics for E_0 and E_1 is crucial, as independent choices can lead to incompatibility and reduce the probability of generating a right quartet to zero. Murphy [41] highlighted that dependencies between characteristics can benefit attackers. Biryukov *et al.* introduced the middle-round S-box trick [17], and later Biryukov and Khovratovich [18] proposed techniques like the ladder and S-box switch to improve probabilities. These ideas were formalized by Dunkelman et al. as the sandwich attack [27], which divides the cipher into three parts, enhancing the overall probability. Further, to evaluate the middle part efficiently and systematically, the authors [25] introduced a boomerang connectivity table (BCT) for a single round.

Suppose that the middle layer at the fourth round of the given boomerang (Figure 7 (p. 20)) is composed of 16 S-box layers independently. For more clarity, we only chose one S-box layer which is depicted in Figure 9 (p. 23). According to Figure 9 (p. 23), the BCT [25] is defined in the following way.

$$\mathrm{BCT}(\Delta_i, \nabla_o) = \{x \in \{0,1\}^4 : S^{-1}(S(x) \oplus \nabla_o) \oplus S^{-1}(S(x \oplus \Delta_0) \oplus \nabla_o) == \Delta_0\}.$$

This BCT provides a unified representation of existing observations on checking the inconsistency as well as the dependencies to further increase the probability of the boomerang for a single round.

Incompatibility. Incompatibility occurs when, as shown in Figure 9 (p. 23), the boomerang connection table (BCT) entry $\mathrm{BCT}(\Delta_i, \nabla_o) = 0$, meaning the boomerang cannot be formed. If $\mathrm{BCT}(\Delta_i, \nabla_o) \neq 0$, the differential characteristics are compatible to form the quartet for the boomerang.

Ladder Switch. The ladder switch, introduced in [18], occurs when $\Delta_i \neq 0$ and $\nabla_o = 0$, resulting in $\mathrm{BCT}(\Delta_i, \nabla_o) = 2^4$, i.e., $\Pr[\Delta_i^{'} = \Delta_i] = 1$. Geometrically, when $\nabla_o = 0$, the upper planes coincide, and the input pairs (x_3, x_4) on the opposite plane are directly replaced by (x_1, x_2). In a similar fashion, if $\Delta_i = 0$ and $\nabla_o \neq 0$, the lower planes coincide, and the input pairs (y_1, y_3) on the opposite plane are directly replaced by (y_2, y_4).

S-box Switch. The S-box switch, introduced in [18], occurs when $\mathrm{DDT}(\Delta_i, \Delta_o) \neq 0$ and $\Delta_o = \nabla_o$, resulting in $\mathrm{BCT}(\Delta_i, \nabla_o) = \mathrm{DDT}(\Delta_i, \Delta_o)$, i.e., $\Pr[\Delta_i^{'} = \Delta_i] =$

$\frac{\mathrm{DDT}(\Delta_i,\Delta_o)}{2^4}$. Geometrically, when $\Delta_o = \nabla_o$, the upper planes interchange their input pairs, i.e., the input pairs (x_3, x_4) on the opposite plane are directly replaced by (x_2, x_1), consistent with $\mathrm{DDT}(\Delta_i, \Delta_o)$.

Based on the switch techniques and the BCT, we verified the compatibility of the full round boomerang distinguisher shown in Figure 7 (p. 20). In this distinguisher, the S-box layer in the fifth round (Round 4) is chosen as the middle layer. We examine the state difference at the Round 4 S-box layer for the upper differential trail and the state difference at the Round 5 S-box layer for the lower trail. As shown, only the third S-box is active in the upper trail, while all S-box nibbles are active in the lower trail. Consequently, all nibbles except the third in the middle layer fall under the ladder switch category, resulting in a probability of 1. For the third nibble position, $\Delta_i = 0x07$ and $\nabla_o = 0x0c$, where we confirmed that $\mathrm{BCT}(0x07, 0x0c) = 2$, validating the compatibility of our differential characteristics to form the full-round boomerang distinguisher.

Refinements to the Boomerang Distinguisher In the previous paragraph, we demonstrated the compatibility of the two differential characteristics necessary to form a full-round boomerang using middle-round switch effects. Now, we will delve into a more detailed analysis of how these switching effects can be leveraged to significantly enhance the boomerang probability. As shown in Figure 8 (p. 22), there are three active S-boxes at positions 12, 14, and 15 in the lower trail during round 5. Tracing this lower trail backward, the first column $((0,0,0,8)^T)$ contains a single active nibble, $0x08$, at the third position following the inverse ShiftRow operation. This nibble difference, $0x08$, arises from the difference $0x09$ after the inverse S-box operation in round 5. Notably, the other two active S-boxes in round 5 do not impact the first column after the inverse ShiftRow and MixColumn operations, as depicted in Figure 10 (p. 23).

According to Figure 10 (p. 23), if we chose all possible differences δ from $0x9$ through S-box inverse, i.e., $0x9 \xrightarrow{inverse\ \mathrm{DDT}} \{0x1, 0x7, 0x8, 0xa, 0xc, 0xe\}$ (see Table 2 (p. 10)), we get different $\delta_3(= \eta_3) \in \{0xc, 0x8, 0xd, 0xf, 0xa, 0x9\}$ through inverse ShiftRow and MixColumn operations. Finally, we checked that if $\delta_3(= \eta_3) \in \{0xc, 0xa, 0x9\}$, then $\mathrm{BCT}(\zeta(= 0x7), \eta) \neq 0$. This demonstrates that the two differential characteristics are compatible for forming the quartet

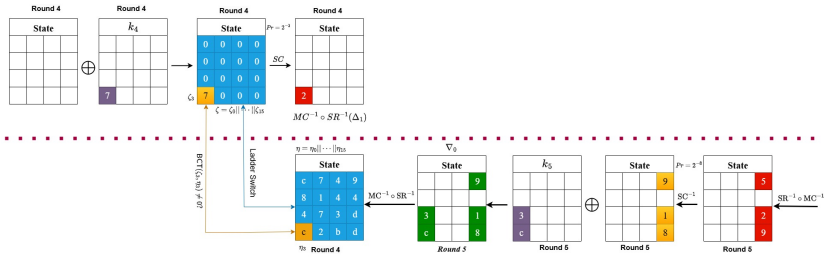


Fig. 8: Middle Round Switching Effects

in the boomerang if the output differences are $\{0x8, 0xc, 0xe\}$ from the input difference $0x09$ through the inverse S-box operation at round 5 (at position 15 in the lower half). This increases the probability from 2^{-3} ($= \Pr[0x9 \xrightarrow{S^{-1}} 0x8]$) to 3/8. Furthermore, any possible output differences from the input differences $0x05$ and $0x02$ (i.e., $\Pr[0x5 \xrightarrow{S^{-1}} *] = 1, \Pr[0x2 \xrightarrow{S^{-1}} *] = 1$) do not affect the first column after the ShiftRow and MixColumn inverse operations, increasing the probability from 2^{-5} to 1. Similarly, the output difference corresponding to the active S-box at round 4 for the upper half can be arbitrary, i.e., $0x7 \xrightarrow{\text{DDT}} *$. This also increases the probability from 2^{-2} to 1. As a result, for the one lower half, the probability improves by a factor of $2^5 \cdot 3$. Thus, for the two parallel lower halves, the probability improves by a factor of $(2^{6.6})^2 = 2^{13.2}$. For upper halves, the probability improves by a factor of $(2^2)^2 = 2^4$. Additionally, we account for the probability that $\text{BCT}(7, \delta_3) \neq 0$ for the middle-round switch at the round 4 S-box operation. Since $\delta_3 \in \{0xc, 0xa, 0x9\}$, the probability of $\text{BCT}(7, \delta_3) \neq 0$ is lower bounded by the minimum of their respective probabilities, i.e.,

$$\Pr[\text{BCT}(7, \delta_3) \neq 0] \geq \frac{\min\{\text{BCT}(7, 0xc), \text{BCT}(7, 0xa), \text{BCT}(7, 0x9)\}}{2^4} = \frac{\min\{2, 4, 4\}}{2^4} = 2^{-3},$$

where $\text{BCT}(7, 0xc) = 2, \text{BCT}(7, 0xa) = 4,$ and $\text{BCT}(7, 0xc) = 4$. Finally, the refined probability for the boomerang becomes $2^4 \cdot 2^{13.2} \cdot 2^{-60} \cdot 2^{-3} = 2^{-45.8}$. This scenario can be further mapped to a Sandwich attack ($E = E_1 \circ E_m \circ E_0$) with probability $\bar{p}^2 \cdot r \cdot \bar{q}^2$, where $\bar{p} = 2^{-12.4}$, $r = 2^{-3}$, and $\bar{q} = 2^{-9}$. As a result, the data, time, and memory complexities of the distinguishing attack are reduced to $2^{48} (= 4 \cdot 2^{46})$ plaintexts, 2^{48} XOR operations, and negligible memory, respectively.

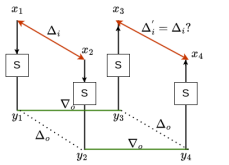


Fig. 9: Single S-box Layer in the Middle of the Boomerang

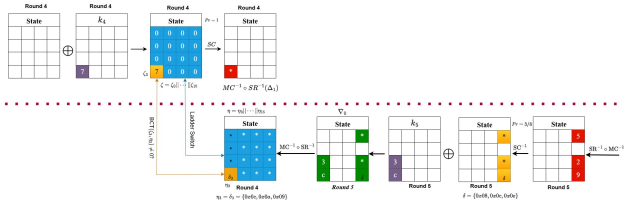


Fig. 10: Middle Round Switching Effects using Truncated Differences

Experimental Verification For this boomerang distinguisher, we have experimentally verified both the upper and lower differential characteristics along with their corresponding probabilities. The implementation used for verification is provided in [2].

Key Recovery Attack In this section, we present the full-round boomerang key recovery attack. First, we demonstrate how four key nibbles can be recovered using the full-round boomerang distinguisher. Next, we select an 8-round boomerang distinguisher and extend the lower part of its differential trail by two additional rounds to facilitate key recovery.

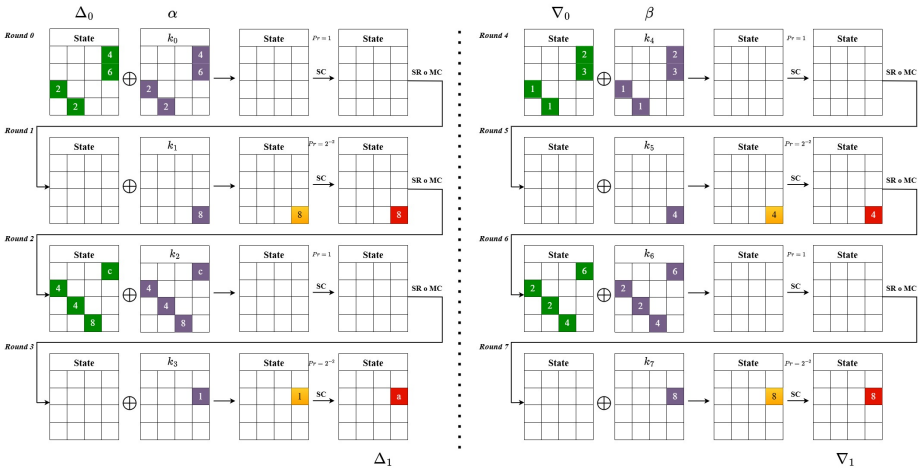


Fig. 11: An 8-Round Boomerang Distinguisher

Key Recovery from the Full Round Distinguisher Recalling the full-round boomerang distinguisher (see Figure 7 (p. 20)) described in the previous section, the data, time, and memory complexities to satisfy the boomerang trail are 2^{48}, 2^{48}, and a negligible amount of memory, respectively. The idea is that once an attacker obtains the satisfying quartets of plaintexts and ciphertexts as (P_0, P_1, P_2, P_3) and (C_0, C_1, C_2, C_3), they can immediately filter the active S-box differentials from the first and last rounds of the upper and lower parts of the differential trails. According to Figure 7 (p. 20), three active S-box differentials (at positions 0, 1, and 3) in the first round of both the upper differential trails can be directly filtered, and the respective key nibbles $k_0[5]$ and $k_0[6]$ are uniquely retrieved from the plaintext nibble pairs $(P_0[5], P_2[5])$ and $(P_0[6], P_2[6])$. Similarly, from the last round of the lower differential trail, an attacker can filter the 0^{th} S-box differential and retrieve the key nibble $k_{10}[0]$ from $C_0[0]$ and $C_2[0]$.

The remaining task is a straightforward exhaustive search over the remaining 28 nibbles, requiring 2^{112} computations. Thus, the data, time, and memory complexities for the full-round key recovery are 2^{48}, 2^{112}, and a negligible amount of memory, respectively.

Key Recovery from an 8-Round Distinguisher In this attack, we choose an 8-round boomerang distinguisher with four round upper and lower differential trails with probabilities 2^{-4} each. Thus, the probability of the boomerang distinguisher would be 2^{-16}. The 8-round boomerang distinguisher is depicted in Figure 11 (p. 24). For this full round related-key boomerang key recovery attack, as described in Section 3, we treat the cipher as $E = E_f \circ E_d$, where E_d is an 8-round distinguisher of probability $p^2 = 2^{-16}$, and the number of rounds in E_f is 2. This is depicted in Figure 7 (p. 20). In this case, the other parameters are $n = 128, k = 128, m_f = 24 \times 4 = 96, r_f = 15 \times 4 = 60$.

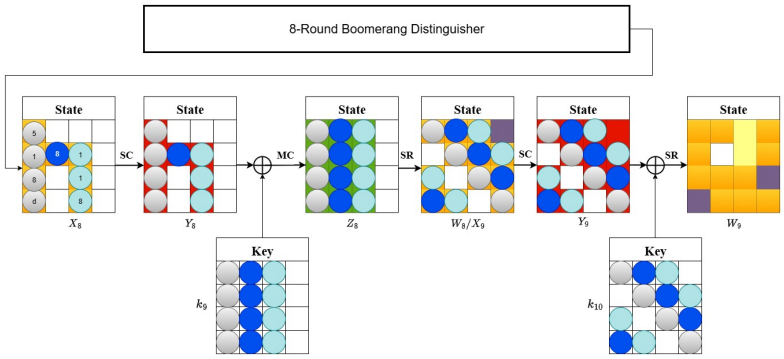


Fig. 12: The Full Round Boomerang Attack of FUTURE

The best guessing parameters are $m_f^{'} = 8 \times 4 = 32, m_f^* = 16 = 64, r_f^{'} = 4 \times 4 = 16$, and $r_f^* = 11 \times 4 = 44$. Therefore, using the unified and generic boomerang key recovery attack (described in Section 3.3), the complexities will be as follows.

- The data complexity is $D^{'} = \frac{4 \cdot s}{p^2} = s \cdot 2^{18}$.
- The memory complexity is $M = D^{'} + D + 2^{t+m_f^*} = s \cdot 2^{18} + s \cdot 2^{16} + 2^{64+t}$.
- For time complexity:
  - $T_1 = 2^{m_f^{'}} \cdot D^{'} = s \cdot 2^{18} \cdot 2^{32} = s \cdot 2^{50}$, $T_2 = 2^{m_f^{'}} \cdot D = s \cdot 2^{48}$.
  - $T_3 = 2^{m_f} \cdot D \cdot 2^{2 \cdot r_f^* - n - 1} \cdot \epsilon = 2^{32} \cdot s \cdot 2^{16} \cdot 2^{88-64-1} \cdot \epsilon = s \cdot 2^{71} \cdot \epsilon$.
  - $T_4 = 2^{128-h}$, where $h \leq t + m_f^*$.

Note that ϵ represents the number of partial decryption, which is around $2/10 = 2^{-2.25}$ encryptions. Thus, if we set $s = a, h = 60, t = 0$, then the data, time, and memory complexities will be $2^{18}, 2^{70}$, and 2^{64} respectively.

6 Conclusion and Future Works

In this work, we present a comprehensive implementation of bit-oriented MILP models for the FUTURE lightweight block cipher in related-key settings. This approach can be extended to model MDS (or near-MDS) based SPN ciphers in the future. Utilizing this model, we explored related-key differential characteristics across different rounds, identifying a seven-round differential characteristic with a probability of $2^{-46.4}$. We further extended this characteristic by adding an extra round, providing a distinguisher with data complexity of 2^{58}, time complexity of 2^{58} XOR operations, and negligible memory requirements. Based on this distinguisher, we propose a 9 round key recovery attack with data, time, and memory complexities of 2^{59}, 2^{69}, and 2^{59} respectively. Additionally, we developed a full-round boomerang distinguisher with a probability of 2^{-60} based on the round-reduced differential characteristics. By applying a one-round middle switch effect, we refined the boomerang probability from 2^{-60} to $2^{-45.8}$. Consequently, the complexities of the attack are improved to 2^{48} plaintexts, 2^{48} XOR

operations, and negligible memory. Furthermore, we present a key recovery attack with data, time, and memory complexities of 2^{18}, 2^{70}, and 2^{64}, respectively.

In future work, it would be valuable to explore optimizing the probability of the distinguisher, rather than focusing solely on the number of active S-boxes. This could potentially enhance the overall probability of the distinguisher. Additionally, recent advancements in automated tools for cryptanalysis present an opportunity to develop a tool for conducting truncated differential and sandwich attacks, capturing more dependencies in the middle rounds, and further improving the probabilities of differential and boomerang distinguishers. Lastly, another interesting direction for future research would be to propose an efficient key recovery attack based on the distinguishers presented in this work.

Acknowledgements The authors thank the anonymous reviewers for their valuable feedback. This work was partially supported by the Centre on Hardware-Security Entrepreneurship Research and Development (C-HERD) and the Information Security Education and Awareness (ISEA) initiative, MeitY, Government of India.

References

1. Linear Programming Formulation With Gurobi Python API. https://www.gurobi.com/resources/ch4-linear-programming-with-python
2. Unoptimized MILP Codes and Their Verifications using C. https://github.com/janaamit001/RK-Cryptanalysis-of-FUTURE
3. Final steps of the nist lightweight cryptography standardization. https://csrc.nist.gov/csrc/media/Presentations/2023/final-steps-of-the-nist-lightweight-cryptography-s/images-media/talk-cesar-2023-meltem-nov2023.pdf (2023)
4. Avanzi, R.: The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/TOSC.V2017.I1.4-44
5. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Advances in Cryptology - ASIACRYPT, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. LNCS, vol. 9453, pp. 411–436. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_17
6. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Cryptographic Hardware and Embedded Systems - CHES, Taipei, Taiwan, September 25-28, 2017, Proceedings. LNCS, vol. 10529, pp. 321–345. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_16
7. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015. pp. 175:1–175:6. ACM (2015). https://doi.org/10.1145/2744769.2747946

8. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Advances in Cryptology - CRYPTO, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. LNCS, vol. 9815, pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
9. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. **2019**(1), 5–45 (2019). https://doi.org/10.13154/TOSC.V2019.I1.5-45
10. Biham, E.: New types of cryptanalytic attacks using related keys. J. Cryptol. **7**(4), 229–246 (1994). https://doi.org/10.1007/BF00203965
11. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the serpent. In: Advances in Cryptology - EUROCRYPT, Innsbruck, Austria, May 6-10, 2001, Proceeding. LNCS, vol. 2045, pp. 340–357. Springer (2001). https://doi.org/10.1007/3-540-44987-6_21
12. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Fast Software Encryption–FSE, Leuven, Belgium, February 4-6, 2002, Revised Papers. LNCS, vol. 2365, pp. 1–16. Springer (2002). https://doi.org/10.1007/3-540-45661-9_1
13. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Advances in Cryptology - EUROCRYPT, Aarhus, Denmark, May 22-26, 2005, Proceedings. LNCS, vol. 3494, pp. 507–525. Springer (2005). https://doi.org/10.1007/11426639_30
14. Biham, E., Dunkelman, O., Keller, N.: A related-key rectangle attack on the full KASUMI. In: Advances in Cryptology - ASIACRYPT, Chennai, India, December 4-8, 2005, Proceedings. LNCS, vol. 3788, pp. 443–461. Springer (2005). https://doi.org/10.1007/11593447_24
15. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Advances in Cryptology - CRYPTO, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. LNCS, vol. 537, pp. 2–21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1
16. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer (1993). https://doi.org/10.1007/978-1-4613-9314-6
17. Biryukov, A., Cannière, C.D., Dellkrantz, G.: Cryptanalysis of SAFER++. In: Advances in Cryptology - CRYPTO, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. LNCS, vol. 2729, pp. 195–211. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_12
18. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Advances in Cryptology - ASIACRYPT, Tokyo, Japan, December 6-10, 2009. Proceedings. LNCS, vol. 5912, pp. 1–18. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_1
19. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Advances in Cryptology - ASIACRYPT, Seoul, South Korea, December 4-8, 2011. Proceedings. LNCS, vol. 7073, pp. 344–371. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_19
20. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems - CHES, Vienna, Austria, September 10-13, 2007, Proceedings. LNCS, vol. 4727, pp. 450–466. Springer (2007). https://doi.org/10.1007/978-3-540-74735-2_31

21. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In: Selected Areas in Cryptography–SAC, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers. LNCS, vol. 6544, pp. 229–240. Springer (2010). https://doi.org/10.1007/978-3-642-19574-7_16
22. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Advances in Cryptology - ASIACRYPT, Beijing, China, December 2-6, 2012. Proceedings. LNCS, vol. 7658, pp. 208–225. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_14
23. Boura, C., Coggia, D.: Efficient milp modelings for sboxes and linear layers of spn ciphers. IACR Transactions on Symmetric Cryptology **2020**(3), 327–361 (2020)
24. Cannière, C.D., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In: Cryptographic Hardware and Embedded Systems - CHES, Lausanne, Switzerland, September 6-9, 2009, Proceedings. LNCS, vol. 5747, pp. 272–288. Springer (2009). https://doi.org/10.1007/978-3-642-04138-9_20
25. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Advances in Cryptology - EUROCRYPT, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II. LNCS, vol. 10821, pp. 683–714. Springer (2018). https://doi.org/10.1007/978-3-319-78375-8_22
26. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). https://doi.org/10.1007/978-3-662-04722-4
27. Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3g telephony. J. Cryptol. **27**(4), 824–849 (2014). https://doi.org/10.1007/S00145-013-9154-9
28. Faust, S., Krämer, J., Orlt, M., Struck, P.: On the related-key attack security of authenticated encryption schemes. In: Security and Cryptography for Networks–SCN, Amalfi, Italy, September 12-14, 2022, Proceedings. pp. 362–386. LNCS, Springer (2022). https://doi.org/10.1007/978-3-031-14791-3_16
29. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: Milp-based automatic search algorithms for differential and linear trails for speck. In: Fast Software Encryption–FSE, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23. pp. 268–288. Springer (2016)
30. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Cryptographic Hardware and Embedded Systems - CHES, Nara, Japan, September 28 - October 1, 2011. Proceedings. LNCS, vol. 6917, pp. 326–341. Springer (2011). https://doi.org/10.1007/978-3-642-23951-9_22
31. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Cryptographic Hardware and Embedded Systems - CHES, Nara, Japan, September 28 - October 1, 2011. Proceedings. LNCS, vol. 6917, pp. 326–341. Springer (2011). https://doi.org/10.1007/978-3-642-23951-9_22
32. Gupta, K.C., Pandey, S.K., Samanta, S.: Future: a lightweight block cipher using an optimal diffusion matrix. In: International Conference on Cryptology in Africa. pp. 28–52. Springer (2022)
33. İlter, M.B., Selçuk, A.A.: Milp-aided cryptanalysis of the future block cipher. In: International Conference on Information Technology and Communications Security. pp. 153–167. Springer (2022)

34. Iwata, T., Kurosawa, K.: How to enhance the security of the 3gpp confidentiality and integrity algorithms. In: Fast Software Encryption–FSE, Paris, France, February 21-23, 2005, Revised Selected Papers. LNCS, vol. 3557, pp. 268–283. Springer (2005). https://doi.org/10.1007/11502760_18
35. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and serpent. In: Fast Software Encryption–FSE, New York, NY, USA, April 10-12, 2000, Proceedings. LNCS, vol. 1978, pp. 75–93. Springer (2000). https://doi.org/10.1007/3-540-44706-7_6
36. Kelsey, J., Schneier, B., Wagner, D.A.: Related-key cryptanalysis of 3-way, biham-des, cast, des-x, newdes, rc2, and TEA. In: Information and Communication Security–ICICS, Beijing, China, November 11-14, 1997, Proceedings. pp. 233–246. LNCS, Springer (1997). https://doi.org/10.1007/BFB0028479
37. Li, T., Sun, Y.: Superball: A new approach for MILP modelings of boolean functions. IACR Trans. Symmetric Cryptol. **2022**(3), 341–367 (2022). https://doi.org/10.46586/TOSC.V2022.I3.341-367
38. Lin, H., Zou, J., Li, J.: The differential meet-in-the-middle attack on FUTURE and CRAFT. In: Proceedings of the 2023 13th International Conference on Communication and Network Security, ICCNS, Fuzhou, China, December 6-8, 2023. pp. 151–158. ACM (2023). https://doi.org/10.1145/3638782.3638805
39. Mondal, S.K., Rahman, M., Sarkar, S., Adhikari, A.: Yoyo cryptanalysis on future. Int. J. Appl. Cryptogr. **4**(3/4), 238–249 (2024). https://doi.org/10.1504/IJACT.2024.138453
40. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Information Security and Cryptology–Inscrypt, Beijing, China, November 30–December 3, 2011. Revised Selected Papers 7. pp. 57–76. Springer (2012)
41. Murphy, S.: The return of the cryptographic boomerang. IEEE Trans. Inf. Theory **57**(4), 2517–2521 (2011). https://doi.org/10.1109/TIT.2011.2111091
42. National Institute of Standards and Technology: Secure Hash Standard (SHS). Federal Information Processing Standards (FIPS) Publication 180-4 (August 2015). https://doi.org/https://doi.org/10.6028/NIST.FIPS.180-4
43. Pal, D., Chandratreya, V.P., Chowdhury, D.R.: New techniques for modeling sboxes: An MILP approach. In: Cryptology and Network Security–CANS , Augusta, GA, USA, October 31 - November 2, 2023, Proceedings. LNCS, vol. 14342, pp. 318–340. Springer (2023). https://doi.org/10.1007/978-981-99-7563-1_15
44. Roy, H.S., Dey, P., Mondal, S.K., Adhikari, A.: Cryptanalysis of full round FUTURE with multiple biclique structures. Peer Peer Netw. Appl. **17**(1), 397–409 (2024). https://doi.org/10.1007/S12083-023-01600-Y
45. Sakiyama, K., Sasaki, Y., Li, Y.: Security of Block Ciphers: From Algorithm Design to Hardware Implementation. Wiley Publishing, 1st edn. (2015)
46. Sasaki, Y., Todo, Y.: New algorithm for modeling s-box in milp based differential and division trail search. In: Innovative Security Solutions for Information Technology and Communications–SecITC, Bucharest, Romania, June 8–9, 2017, Revised Selected Papers 10. pp. 150–165. Springer (2017)
47. Schrottenloher, A., Stevens, M.: Simplified modeling of MITM attacks for block ciphers: New (quantum) attacks. IACR Trans. Symmetric Cryptol. **2023**(3), 146–183 (2023). https://doi.org/10.46586/TOSC.V2023.I3.146-183
48. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An ultra-lightweight blockcipher. In: Cryptographic Hardware and Embedded Systems - CHES, Nara, Japan, September 28 - October 1, 2011. Proceed-

ings. LNCS, vol. 6917, pp. 342–357. Springer (2011). https://doi.org/10.1007/978-3-642-23951-9_23
49. Song, L., Zhang, N., Yang, Q., Shi, D., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle attacks: A unified and generic framework for key recovery. In: Advances in Cryptology - ASIACRYPT, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I. LNCS, vol. 13791, pp. 410–440. Springer (2022). https://doi.org/10.1007/978-3-031-22963-3_14
50. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., Alkhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Advances in Cryptology - CRYPTO, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. LNCS, vol. 9215, pp. 95–115. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_5
51. Sun, L., Wang, W., Wang, M.: Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**(1), 12–20 (2020). https://doi.org/10.1049/IET-IFS.2018.5283
52. Sun, S., Hu, L., Song, L., Xie, Y., Wang, P.: Automatic security evaluation of block ciphers with s-bp structures against related-key differential attacks. In: Information Security and Cryptology–Inscrypt, Guangzhou, China, November 27-30, 2013, Revised Selected Papers. LNCS, vol. 8567, pp. 39–51. Springer (2013). https://doi.org/10.1007/978-3-319-12087-4_3
53. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Paper 2014/747 (2014), https://eprint.iacr.org/2014/747
54. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des (l) and other bit-oriented block ciphers. In: Advances in Cryptology–ASIACRYPT, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20. pp. 158–178. Springer (2014)
55. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A lightweight block cipher for multiple platforms. In: Selected Areas in Cryptography–SAC, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. LNCS, vol. 7707, pp. 339–354. Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_22
56. Udovenko, A.: MILP modeling of boolean functions by minimum number of inequalities. IACR Cryptol. ePrint Arch. p. 1099 (2021), https://eprint.iacr.org/2021/1099
57. Wagner, D.A.: The boomerang attack. In: Fast Software Encryption–FSE, Rome, Italy, March 24-26, 1999, Proceedings. LNCS, vol. 1636, pp. 156–170. Springer (1999). https://doi.org/10.1007/3-540-48519-8_12
58. Xu, Z., Cui, J., Hu, K., Wang, M.: Integral attack on the full FUTURE block cipher. IACR Cryptol. ePrint Arch. p. 549 (2024), https://eprint.iacr.org/2024/549
59. Yang, Q., Song, L., Zhang, N., Shi, D., Wang, L., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. J. Cryptol. **37**(2), 19 (2024). https://doi.org/10.1007/S00145-024-09499-1
60. Yin, J., Ma, C., Lyu, L., Song, J., Zeng, G., Ma, C., Wei, F.: Improved cryptanalysis of an ISO standard lightweight block cipher with refined MILP modelling. In: Information Security and Cryptology–Inscrypt, Xi'an, China, November 3-5, 2017, Revised Selected Papers. LNCS, vol. 10726, pp. 404–426. Springer (2017). https://doi.org/10.1007/978-3-319-75160-3_24