# Differential Cryptanalysis of WARP

Je Sen Teh* and Alex Biryukov

***Abstract*—WARP is an energy-efficient lightweight block cipher that is currently the smallest 128-bit block cipher in terms of hardware. This paper proposes the first key-recovery attacks on WARP based on differential cryptanalysis in single and related-key settings. We searched for differential trails for up to 20 rounds of WARP, with the first 19 having optimal differential probabilities. We also found that WARP has a strong differential effect, whereby 16 to 20-round differentials have substantially higher probabilities than their corresponding individual trails. A 23-round key-recovery attack was then realized using an 18-round differential distinguisher. Next, we formulated an automatic boomerang search using SMT that relies on the Feistel Boomerang Connectivity Table to identify valid switches. We designed the search as an add-on to the CryptoSMT tool, making it applicable to other Feistel-like ciphers such as TWINE and LBlock-s. For WARP, we found a 21-round boomerang distinguisher which was used in a 24-round rectangle attack. In the related-key setting, we describe a family of 2-round iterative differential trails, we used in a practical related-key attack on the full 41-round WARP.**

***Index Terms*—Symmetric-key, Block ciphers, Differential cryptanalysis, Boomerang distinguisher, Rectangle attack, Related-key, WARP, GFN**

## I. Introduction

Lightweight cryptography is currently one of the most heavily researched areas in recent years. This is due in part to the proliferation of resource-constrained devices such as smart devices which transmit sensitive information on a daily basis. Compared to other symmetric-key primitives, lightweight block ciphers have received the most attention in terms of development and cryptanalytic efforts. Although most lightweight block ciphers have block sizes of 64 bits to minimize their hardware footprint, there were a number of 128-bit block ciphers with lower area and/or power requirements than AES such as MIDORI [1] and GIFT-128 [2]. The majority of these 128-bit lightweight block ciphers are based on the Substitution-Permutation Network (SPN) design paradigm. However, SPN ciphers generally take up more hardware space because they require the inversion of the confusion and diffusion layers.

To overcome this hurdle, Banik et al. adopted the Type-2 Generalized Feistel Network [3] in their 128-bit block cipher called WARP which was proposed in SAC 2020 [4]. WARP's design team consists of the minds behind multiple well-known lightweight block ciphers such as GIFT, MIDORI and TWINE [5]. The motivation behind designing WARP as a 128-bit cipher with a 128-bit key was to realize a direct replacement for AES-128 without having to change the underlying mode of operation, making it applicable to a wide range of applications such as in the development of lightweight authenticated encryption (AE) schemes [6]. GFN-based ciphers can reuse the same circuit for encryption and decryption due to their involutory nature but are known to have slow diffusion. The designers addressed this problem by using a permutation pattern found using Mixed Integer Linear Programming (MILP) that optimizes both diffusion and the number of active S-boxes. By adopting MIDORI's S-box (for reduced latency and area) and a simple alternating key schedule, the designers found that WARP only requires 763 Gate Equivalents (GE) for a bit-serial encryption-only circuit and has better energy consumption than MIDORI, which is widely considered the current state-of-the-art in terms of 128-bit low-energy ciphers.

Manuscript received xx, 2021; revised xx, 2021.

Alex Biryukov is with DCS and SnT, University of Luxembourg.

Je Sen Teh is with SnT, University of Luxembourg and with the School of Computer Sciences, Universiti Sains Malaysia (e-mail:jesen_teh@usm.my)

*Corresponding Author

**Related Work.** To the best of our knowledge, the only prior third-party cryptanalysis result for WARP was an attack reported by Kumar and Yadav [7]. By using an 18-round differential trail, the authors were able to perform a key recovery attack on 21 rounds. WARP's designers analyzed its security against differential and linear cryptanalysis based on the number of active S-boxes. They found that WARP has more than 64 active S-boxes after 19 rounds. They also found a 21-round impossible differential distinguisher and a 20-round integral distinguisher for the cipher. A meet-in-the-middle attack is expected to be feasible for at most 32 rounds of WARP. Although no concrete attacks were described, 41 rounds of WARP is expected to be secure against these attacks.

**Our Contributions.** In this paper, we focused our efforts on cryptanalyzing WARP from several perspectives, all of which are based on differential cryptanalysis. By using an SMT-aided differential search, we found differential trails for up to 20 rounds of WARP, with the first 19 guaranteed to be optimal. These differential trails confirm that the lower bounds provided by the designers cannot be improved. We then performed a differential cluster search for each of these trails and found that WARP has a strong differential effect from round 13 onward. Notably, differentials for 16 to 20 rounds have higher probabilities than their corresponding individual trails by at least a factor of $2^{13}$.

Next, we implemented an automatic search for boomerang (or more specifically, rectangle) distinguishers that utilizes the recently proposed Feistel Boomerang Connectivity Table (FBCT) [8]. The boomerang search was written as a new module for the CryptoSMT tool [9] rather than one that was specifically catered to WARP[1]. Thus, it can be used for other Feistel-like ciphers and modified for other design paradigms like SPN. We showcase its flexibility by searching for boomerang distinguishers for TWINE and LBlock-s [10] apart from WARP. Using our tool, we were able to find a 21-round boomerang distinguisher for WARP with a differential probability, $\mathsf{DP} = 2^{-121.11}$.

[1]The implementation is publicly available under an open-source license at https://github.com/jesenteh/cryptosmt-boomerang.

We also performed a search for related-key differential trails for WARP. As a result, we found that WARP has a family of 2-round iterative related-key differential trails with low weight. These iterative trails can be concatenated to form distinguishers for the full 41-round WARP with $\mathsf{DP} = 2^{-40}$. These trails exist due to an interaction between the cipher's nibble-wise permutation, simple alternating key schedule and subkey XOR operation performed *after* the S-box. The interaction between these design elements also led to another interesting observation whereby knowledge of the input difference for a Feistel-subround can be propagated to the next round without having to guess its corresponding subkey. This property was leveraged in all of our key recovery attacks to target specific subkeys, which then allows the filtering of wrong pairs and key bits.

Finally, we proposed key-recovery attacks on WARP based on the differential distinguishers that were found. In the single-key setting, we have a 23-round differential attack using an 18-round differential distinguisher that has time/data/memory complexities of $2^{108.68}/2^{108.62}/2^{108.62}$, followed by a 24-round rectangle attack using a 21-round boomerang distinguisher with time/data/memory complexities of $2^{122.49}/2^{126.06}/2^{127.06}$. In the related-key setting we show a practical attack on the full 41 rounds of WARP using a 35-round related-key differential distinguisher with time/data/memory complexities of $2^{37}/2^{37}/2^{9.59}$. We computationally verified this attack by using shorter 19-round related-key differential distinguisher to break 25-round WARP and it worked as expected. Our cryptanalytic results are summarized in Table I.

## II. Preliminaries

Notations and abbreviations used in this paper are summarized in Table II. In all of our notations, the rightmost (least significant) bits or nibbles have an index of 0.

| R | Method | Time | Data | Memory | Sect. |
|---|---|---|---|---|---|
| 23 | SK Diff. | $2^{108.62}$ | $2^{108.62}$ | $2^{107}$ | IV-A |
| 24 | SK Rect. | $2^{122.49}$ | $2^{126.06}$ | $2^{127.06}$ | IV-B |
| 41 | RK Diff.[a] | $2^{37}$ | $2^{37}$ | $2^{9.59}$ | V-B |

TABLE I
SUMMARY OF KEY-RECOVERY ATTACKS ON WARP (SK/RK DENOTES SINGLE-KEY/RELATED-KEY)

[a]Complexity involved in recovering 60 bits of the key

| Symbol | Meaning |
|---|---|
| $n$ | Block size in bits |
| $k$ | Key size in bits |
| $\Delta P$ | An XOR difference between two $n$-bit plaintexts, $P_1$ and $P_2$ |
| $\Delta C$ | An XOR difference between two $n$-bit ciphertexts, $C_1$ and $C_2$ |
| $\alpha, \beta, \delta, \gamma$ | $n$-bit input and output differences of differential trails |
| $\alpha_i^j$ | The $i$-th nibble of an $n$-bit XOR difference, $\alpha$ in round $j$ |
| $X_i^j$ | The $i$-th nibble of an $n$-bit binary variable, $X$ in round $j$ |
| $\#AS$ | Number of active S-boxes for a differential trail |
| $\oplus$ | Binary exclusive-OR (XOR) |
| $\|\|$ | Binary concatenation |
| DP | Differential probability |
| R | Number of encryption/decryption rounds |
| $DDT(x,y)$/ $FBCT(x,y)$ | An entry in the DDT/FBCT corresponding to an S-box input $x$ and output $y$ |

TABLE II
SYMBOLS AND NOTATION

## A. Differential Cryptanalysis

A block cipher is a family of key-dependent permutations that map a set of plaintexts to a set of ciphertexts. This mapping is performed by applying a key-dependent round function, $f_j$ on a plaintext in an iterative manner to produce a ciphertext, where $j \in \mathsf{R}$ and $\mathsf{R}$ denotes the total number of rounds of a cipher. The goal of differential cryptanalysis is to find pairs of plaintexts $(P_1, P_2)$ and their corresponding ciphertexts $(C_1, C_2)$ with a strong correlation between their differences $\alpha = P_1 \oplus P_2$ and $\beta = C_1 \oplus C_2$. The propagation pattern of a specific input difference $\alpha$ to a specific output difference $\beta$ is known as a differential characterisitic or trail. A differential trail consists of a sequence of differences,

$$\alpha \xrightarrow{f_1} \alpha^1 \xrightarrow{f_2} ... \xrightarrow{f_{\mathsf{R}-2}} \alpha^{\mathsf{R}-2} \xrightarrow{f_{\mathsf{R}-1}} \beta. \quad (1)$$

Generally, the success of differential cryptanalysis relies on the identification of a differential trail with sufficiently high differential probability,

$$\mathsf{DP} = \mathsf{Pr}(\alpha \xrightarrow{f_1} ... \xrightarrow{f_{\mathsf{R}-1}} \beta). \quad (2)$$

Since it is computationally infeasible to calculate the exact value for differential probability, cryptanalysts rely on the Markov assumption [11] which allows treating the rounds as independent from one another. As such, differential probability can be computed as

$$\mathsf{DP} \approx \prod_{j=1}^{\mathsf{R\text{-}1}} \mathsf{Pr}(\alpha^{j-1} \xrightarrow{f_j} \alpha^j), \quad (3)$$

where $\alpha_0 = \alpha$ and $\alpha_{\mathsf{R\text{-}1}} = \beta$. A better estimate of the differential probability can be obtained by collecting differential trails that share the same input and output differences and summing up their individual probabilities,

$$\mathsf{DP} = \mathsf{Pr}(\alpha \to \beta) = \sum_{\alpha^1 ... \alpha^{\mathsf{R\text{-}2}}} (\alpha \xrightarrow{f_1} ... \xrightarrow{f_{\mathsf{R}-1}} \beta). \quad (4)$$

In the related-key setting, an adversary is allowed to also have a difference in the encryption key, and not only in the plaintext [12]. However, the adversary cannot specify the value of the key itself and the attack must be valid for any pair of keys with the given difference. The related-key model has been used to theoretically cryptanalyze the full rounds of various block ciphers over the years [13]–[19]. In the past, there have also been practical attacks that rely on the related-key property [20], [21]. Ciphers that are vulnerable to related-key attacks are not recommended for use in protocols where key integrity is not guaranteed [22], [23].

When cryptanalyzing a block cipher, an adversary is mainly concerned with maximizing the probability of the differential by enumerating as many differential trails as possible. Since this is a time-consuming task, many automated methods have been proposed based on branch-and-bound algorithms [24]–[26], MILP [27]–[29], boolean satisfiability problem (SAT) [30], [31] and satisfiability modulo theory (SMT) solvers [32] as well as a graph-based approach [33]. In this paper, we use CryptoSMT, a differential search tool proposed by Ankele and Kölbl [32]. Due to its highly modular nature, we can code our automatic boomerang search as an additional module of CryptoSMT, taking advantage of its search functions and existing cipher suite.

## B. Boomerang and Rectangle Attacks

The boomerang attack proposed by Wagner [34] is a variant of differential cryptanalysis that concatenates two shorter differentials to form a longer distinguisher. The classical boomerang attack involves decomposing a target cipher, $E$ into two subciphers, $E = E_1 \circ E_0$. We denote the input and output differences of the first or top half of the cipher, $E_1$ as $\alpha$ and $\beta$ while for the lower half, these differences are denoted as $\gamma$ and $\delta$. We denote the probability that $\alpha \xrightarrow{E_0} \beta$ as $p$ and $\gamma \xrightarrow{E_1} \delta$ as $q$. The expected probability of a boomerang differential is $p^2q^2$, which requires an adversary to make $(pq)^2$ adaptive chosen plaintext and ciphertext queries to distinguish $E$ from an ideal cipher.

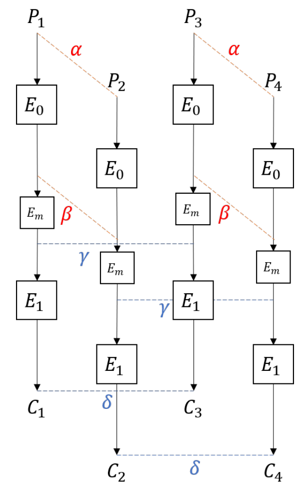


Fig. 1. Sandwich attack

The boomerang attack was later reformulated as a chosen plaintext attack called the amplified boomerang [35] or rectangle attack [36] by encrypting many pairs with the input difference $\alpha$ and searching for a quartet which satisfies $C_1 \oplus C_3 = C_2 \oplus C_4 = \delta$ when $P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha$. Although the probability of a quartet to be a right quartet is reduced to $2^{-n}p^2q^2$, counting over all possible $\beta$'s and $\delta$'s as long as $\beta \neq \delta$ improves the probability to $2^{-n}\hat{p}^2\hat{q}^2$, where $\hat{p} = (\sum_i \mathsf{Pr}^2(\alpha \xrightarrow{E_0} \beta_i)^{\frac{1}{2}}$ and $\hat{q} = (\sum_j \mathsf{Pr}^2(\gamma \xrightarrow{E_1} \delta_j))^{\frac{1}{2}}$.

Independently chosen $E_1$ and $E_0$ trails may turn out to be incompatible [37]. With the introduction of the sandwich attack [15], [16], the boomerang connectivity table (BCT) [38] and its Feistel counterpart [8], we have a systematic means of enumerating these differential trails while guaranteeing their compatibility. The sandwich attack decomposes the cipher into 3 components, $E = E_1 \circ E_m \circ E_0$, where $E_m$ is the transition in the middle round with a switching probability denoted by $r$. We can calculate $r$ with the help of BCT or FBCT, similar to how the differential probability can be calculated based on the differential distribution table (DDT). The connectivity tables already cover the various switches that have been used in the past to improve the probability of boomerang distinguishers such as the ladder, S-box and Feistel switches [15]. The probability of obtaining a right quartet is now

$$\hat{p}^2\hat{q}^2 = \sum_{i,j}(\hat{p_i}^2\hat{q_j}^2 r_{i,j}), \tag{5}$$

where $p_i = \mathsf{Pr}(\alpha \xrightarrow{E_0} \beta_i)$, $q_j = \mathsf{Pr}(\gamma_j \xrightarrow{E_1} \delta)$ and $r_{i,j} = \mathsf{Pr}(\beta_i \xrightarrow{E_m} \gamma_j)$. This sandwich distinguisher is illustrated in Figure 1. It is also possible to evaluate $E_m$ that consists of more than 1 round as described by Song et al. [39].

## C. Specification of WARP

The block cipher WARP is a 41-round, 128-bit block cipher with a 128-bit key designed based on a

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

TABLE III
WARP 4-BIT S-BOX

32-nibble Type-2 GFN. The $i$-th round's state is divided into 32 nibbles, $X^i = X^i_{31}||X^i_{30}||...||X^i_1||X^i_0$, where $X^i_j \in \{0,1\}^4$. It has a simple key schedule that first divides the secret key into two 64-bit round keys, $K = K^1||K^0$, then alternates between them (starting from $K^0$). Each 64-bit round key is divided into 16 nibbles, $K^i = K^i_{15}||K^i_{14}||...||K^i_1||K^i_0$, where $K^i_j \in \{0,1\}^4$, $i \in \{0,1\}$. The round function consists of calls to S-boxes, XOR of the subkeys, and finally a nibble-wise permutation, $\pi$ as shown in Figure 2. An XOR with round constants is also performed on $X^i_1$ and $X^i_3$ prior to the permutation operation. The S-box and permutation pattern are shown in Tables III and IV. Apart from using the inverse permutation, $\pi^{-1}$, the decryption algorithm is the same.

To avoid the complement property of Feistel-type ciphers [40], the designers of WARP opted for the key XOR operation to be after the S-box.However, this design decision leads to the following differential propagation property which we will use as filters in our key recovery attacks:

**Property 1 (Subround Filters).** *Since XOR with the key is done after the S-box in the Feistel-subround which works on two nibbles, it allows to partially decrypt and propagate the knowledge of the difference to the next round. This can be done for both the top and bottom rounds.*

Based on Figure 3, we can see that partially encrypting $P_1$ and $P_2$ that correspond to the input difference, $\alpha$ allows to immediately check if the given pair is valid if the left nibble of the output difference, $\beta_L$ is known. We can do this without having to guess the corresponding key nibble, $K^j_i$ because the output difference of the S-box, which we denote as $\gamma$, can be directly computed from the known values of $x^1_R$ and $x^2_R$. Thus, we can check if $\alpha_L \oplus \gamma = \beta_L$ because the effect of the round key has been negated by the XOR operation. The same property exists for the bottom rounds whereby partially decrypting known values of $C_1$ and $C_2$ when $\alpha_L$ is a known difference allows to check if $\beta_L \oplus \gamma = \alpha_L$.

## III. Searching for WARP Distinguishers

### A. Differential Distinguishers

We use CryptoSMT [9] to search for both differential trails and differentials for WARP. CryptoSMT is a differential search tool based on an SMT solver known as STP [41], CryptoMinisat [42] as its underlying SAT solver. First, a Python script was written to generate the SMT model that describes the differential propagation of WARP. Then, we use the existing functionalities of CryptoSMT to find optimal differential trails for each round and perform differential clustering. Our findings are summarized in Table VI where $\#AS$ refers to the number of active S-boxes and the weight of a differential trail is calculated as $W = -\log_2$ DP.

To find optimal trails (lowest possible weight) for R rounds, we first set the target weight to $\#AS \cdot 2$ for each round, where $\#AS$ is set to the minimum value according to WARP's specification and 2 is the smallest weight for a single S-box,

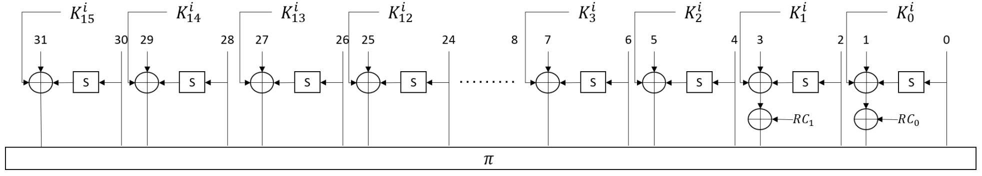
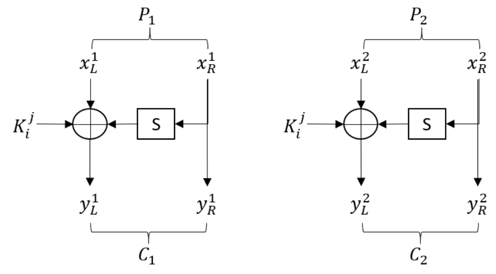


Fig. 2. Round Function of WARP

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\pi(x)$ | 31 | 6 | 29 | 14 | 1 | 12 | 21 | 8 | 27 | 2 | 3 | 0 | 25 | 4 | 23 | 10 |
| $\pi^{-1}(x)$ | 11 | 4 | 9 | 10 | 13 | 22 | 1 | 30 | 7 | 28 | 15 | 24 | 5 | 18 | 3 | 16 |
| $x$ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| $\pi(x)$ | 15 | 22 | 13 | 30 | 17 | 28 | 5 | 24 | 11 | 18 | 19 | 16 | 9 | 20 | 7 | 26 |
| $\pi^{-1}(x)$ | 27 | 20 | 25 | 26 | 29 | 6 | 17 | 14 | 23 | 12 | 31 | 8 | 21 | 2 | 19 | 0 |

TABLE IV
WARP PERMUTATION



$$\alpha = (\alpha_L, \alpha_R) = (x_L^1 \oplus x_L^2, x_R^1 \oplus x_R^2)$$
$$\beta = (\beta_L, \beta_R) = (y_L^1 \oplus y_L^2, y_R^1 \oplus y_R^2)$$

Fig. 3. Difference propagation for a pair of nibbles

| | 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 | 2 | 4 | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 2 | 0 | 4 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 4 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 2 | 2 | 2 | 0 | 0 | 0 | 2 | 0 | 2 |
| 4 | 0 | 2 | 4 | 2 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| 5 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 2 | 4 | 0 | 2 | 0 | 0 | 0 |
| 6 | 0 | 2 | 0 | 4 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 2 | 2 | 0 | 2 |
| 7 | 0 | 0 | 0 | 2 | 0 | 4 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 2 | 0 |
| 8 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 0 |
| 9 | 0 | 0 | 4 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0 |
| A | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 4 | 0 | 4 |
| B | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 4 | 0 | 2 | 0 | 2 |
| C | 0 | 0 | 4 | 0 | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 2 | 0 |
| D | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 4 | 0 | 0 | 4 | 2 | 0 | 0 | 2 | 0 |
| E | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 2 | 2 | 4 | 2 |
| F | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 2 | 4 |

TABLE V
WARP'S DIFFERENTIAL DISTRIBUTION TABLE

calculated from its DDT in Table V. If a trail was found, we repeat the search by reducing the weight by 1 to confirm its optimality. If no other solution with lower weight can be found by the solver (unsatisfiable), the R-round trail is already optimal. If no trail was found with the minimum weight, we increment the target weight and look for another trail. The first trail found is guaranteed to be optimal. For up to 19 rounds, we verified that the minimum number of active S-boxes mentioned in WARP's design specification was indeed the lower bound and also found the optimal differential trails for each of these rounds. The time required to find differential trails increased sharply with the number of rounds, compounded with the fact that we are dealing with a 128-bit block size. Finding a trail for Round 17 onward would take between 5 to 12 hours (on a PC with an Intel Core i7-9700K 3.60GHz processor and 32GB of RAM), longer if a trail did not exist for a particular weight. We managed to find a differential trail for 20 rounds of WARP with a weight of 140 but could not verify its optimality. The search for a 20-round trail with a weight of 139 did not manage to complete within a reasonable amount of time (one week), which suggests that the solver could not find a valid solution.

We then fixed the inputs and outputs corresponding to these optimal trails and performed differential clustering. We enumerate other trails for these differentials with a time limit of 24 hours. The results in Table VI show that for the first 9 rounds, all differentials consisted of only 1 trail each. Very few trails were found for the 10 to 12-round differentials, which had a minimal impact on their differential probabilities. From Round 13 onward, however, there was a sharp increase in the number of trails. We managed to find all trails for the 13-round to 15-round differentials, which ranged from 1600 to just under 500000 trails. The number of trails for 16 rounds onward was not exhaustive as they were bounded by the 24-hour search limit. Since the differential search increases the target weight incrementally, we no longer expect significant improvements after 24 hours as each trail would have an immeasurably small differential probability. The results show that WARP has a significant differential effect at higher rounds, whereby rounds 16 to 20 have an improvement to their differential probabilities by a factor of $2^{13.48}$, $2^{18.34}$, $2^{17.38}$, $2^{13.93}$ and $2^{17.29}$, respectively. After the differential effect has been taken into consideration, all 20 of our differentials can potentially be used as differential distinguishers as they have $\mathsf{DP} > 2^{-128}$.

## *B. Boomerang Distinguishers*

To find boomerang distinguishers for WARP, we formulated an automatic boomerang search based on CryptoSMT's differential search functionality. The overall goal of the automatic search is to maximize $\hat{p}^2\hat{q}^2 = \sum_{i,j}(\hat{p_i}^2\hat{q_j}^2 r_{i,j})$ by finding as many $E_0$ and $E_1$ trails that are compatible. The compatibility of the upper and lower trails is determined using the FBCT which describes the propagation of differences and their corresponding probabilities in $E_m$. For more information about the FBCT, readers can refer to the work by Boukerrou et al. [8]. WARP's FBCT shown in Table IX already includes scenarios such as the ladder switch (first row/column) and the Feistel switch (diagonal) where the switching occurs with a probability of 1. The proposed boomerang search procedure is as follows:

1) Search for an $E_0$ trail with $\mathsf{R}_{E_0}$ rounds for up to a weight limit of $W_{upper}$.
2) Search for an $E_1$ trail with $\mathsf{R}_{E_1}$ rounds for up to a weight limit of $W_{lower}$. Limit the search to only compatible trails by propagating $\beta$ from $E_0$ through $E_m$, then including blocking constraints in the SMT model for each of its S-boxes based on entries in the FBCT. If a valid $E_1$ trail is found then:
    a) If this is the first iteration, fix the input and output differences of the boomerang distinguisher to $\alpha$ and $\delta$ for all future iterations.
    b) Calculate the switching probability, $r_{i,j}$ based on $\beta$, $\gamma$, the linear layer, $\pi$ and FBCT as
    $$r_{i,j} = \prod_{\substack{k=\\ \{2,4,\ldots,\\ 28,30\}}} \frac{FBCT(\beta_k, \pi^{-1}(\gamma_k)-1)}{16}. \tag{6}$$
    c) For the clustering process, limit the search to $W_{init} + \frac{n}{l}$ where $W_{init}$ is the weight of the initial trail and $l$ controls the upper limit of the search, e.g. for $l = 64$, the upper weight limit of the clustering process is $W_{init} + 2$. Set individual limits for $E_0$ and $E_1$.
    d) Perform differential clustering for $E_0$ if it has not yet been done. Denote the resulting differential probability as $\hat{p}_i$.
    e) Perform differential clustering for $E_1$. Denote the resulting differential probability as $\hat{q}_j$
    f) Update the current boomerang probability with $\hat{p_i}^2\hat{q_j}^2 r_{i,j}$.
    g) Add blocking constraints to the SMT model to prevent the current $E_1$ trail from being found again, then repeat Step 2.
3) If no more valid $E_1$ trails can be found, clear all blocking constraints for $E_1$, add constraints to the SMT model to block the current $E_0$ trail from being found again, then repeat Step 1.

|  |  | Trail |  |  | Differential |  |
|---|---|---|---|---|---|---|
| R | #$AS$ | $W_{opt}$ ($-\log_2$ DP) | $\alpha$ | $\beta$ | $W_{diff}$ ($-\log_2$ DP) | #Trails |
| 1 | 0 | 0 | `0000 0000 0000 0000 0000 0000 0000 0010` | `0000 0000 0000 0000 0000 0000 0100 0000` | 0 | 1 |
| 2 | 1 | 2 | `0000 0000 0000 0000 0000 0000 4000 0000` | `0000 4000 0000 0000 0000 0000 0000 0200` | 2 | 1 |
| 3 | 2 | 4 | `0000 2000 0021 0000 0000 0000 0000 0000` | `0000 0000 0000 0000 0000 0200 0000 0000` | 4 | 1 |
| 4 | 3 | 6 | `0000 0000 002C 0000 0000 0000 0000 0000` | `0000 0000 0000 0400 000C 2000 0000 0000` | 6 | 1 |
| 5 | 4 | 8 | `0000 A000 00AA 0000 0000 0000 0000 0000` | `F000 0000 0000 0000 0A00 0000 0F00 0000` | 8 | 1 |
| 6 | 6 | 12 | `0092 0000 0000 0000 0000 0000 9000 0042` | `0000 0000 0000 9002 0000 0002 0000 0000` | 12 | 1 |
| 7 | 8 | 16 | `0000 7DA0 FF00 0000 0000 0000 0000 0000` | `0000 0000 0000 A005 000A 000F 0000 00F0` | 16 | 1 |
| 8 | 11 | 22 | `0000 00FA 5A00 0000 00A0 0000 0000 0000` | `0000 0000 0000 A705 000A 500A 0000 A005` | 22 | 1 |
| 9 | 14 | 28 | `0000 0000 0000 1000 0000 C2C0 4200 0012` | `2900 0020 0000 0000 0120 0000 0104 0001` | 28 | 1 |
| 10 | 17 | 34 | `E000 00EE EE00 0000 00E0 00EE 0000 0000` | `E000 0000 0E0E 0000 0E0E 00E0 0E00 E00E` | 33.19 | 7 |
| 11 | 22 | 44 | `0012 0000 1000 1290 1212 0000 1000 0042` | `2000 0000 0101 0000 0101 0020 0100 2004` | 43.19 | 7 |
| 12 | 28 | 56 | `1212 0000 4000 0042 0012 0000 4000 4240` | `0200 0202 0212 0200 1002 0212 4040 0010` | 55.42 | 5 |
| 13 | 34 | 68 | `0020 2000 0024 2000 0000 0020 2121 0021` | `0010 0202 1000 0000 1200 0240 4000 1202` | 62.37 | 1600 |
| 14 | 40 | 80 | `0000 0010 1292 0012 0010 1000 0042 C000` | `0000 1002 0200 4202 40C0 0002 C002 4202` | 72.14 | 21528 |
| 15 | 47 | 94 | `0000 00A0 5A5A 005A 00A0 5000 0057 5000` | `A500 A005 000A 0700 0AA5 55A5 057A 0AA0` | 85.54 | 497248 |
| 16 | 52 | 104 | `A000 5AAA 0000 0000 0000 A05A 005A 0000` | `0A00 000A 000A 0000 0057 0A50 005A 500A` | 90.52 | 800152 |
| 17 | 57 | 114 | `0000 A000 0000 0075 0000 A500 0000 7000` | `000A 5000 0550 0000 AA00 000A 0000 0A00` | 95.66 | 734494 |
| 18 | 91 | 122 | `0000 A0AF 005A 0000 A000 AA75 0000 0000` | `000A 5000 0AA0 0000 5A00 000A 0000 0A00` | 104.62 | 626723 |
| 19 | 66 | 132 | `5000 A55A 0000 0000 0000 70AA 00A5 0000` | `0500 0050 00A0 0A00 00A5 A00A 5007 000A` | 118.07 | 594111 |
| 20 | 70* | 140* | `0000 50AA 0057 0000 F000 5AAF 0000 0000` | `0A00 A000 0000 500A 0000 050A 0000 F50A` | 122.71 | 545045 |

*Number of active S-boxes and/or differential probability not confirmed to be optimal

TABLE VI
WARP DIFFERENTIALS FOR ROUNDS 1 TO 20

| Cipher | R ($\mathsf{R}_{E_0} + \mathsf{R}_{E_m} + \mathsf{R}_{E_1}$) | $\alpha$ | $\delta$ | $\sum_{i,j}(\hat{p_i}^2 \hat{q_j}^2 r_{i,j})$ |
|---|---|---|---|---|
| TWINE | 15 (7+1+7) | `3890 0000 0097 0000` | `0DB0 0010 0D00 0C00` | $2^{-58.92}$ |
| TWINE | 16 (8+1+7) | `A250 0000 0056 0000` | `A000 0702 0050 0002` | $2^{-61.62}$ |
| LBlock-s | 15 (7+1+7) | `0420 0004 0600 0004` | `6600 0000 4020 0004` | $2^{-58.64}$ |

TABLE VII
BOOMERANG DISTINGUISHERS FOR OTHER CIPHERS

| R ($R_{E_0} + \mathsf{R}_{E_m} + \mathsf{R}_{E_1}$) | $\alpha$ | $\delta$ | $p^2q^2r$ | $\sum_{i,j}(\hat{p_i}^2 \hat{q_j}^2 r_{i,j})$ |
|---|---|---|---|---|
| 20 (9+1+10) | `0000 0000 0000 1000 0000 C2C0 4200 0012` | `0202 0040 0200 1002 4000 0000 0202 0000` | $2^{-124}$ | $2^{-114.24}$ |
| 21 (10+1+10) | `E000 00EE EE00 0000 00E0 00EE 0000 0000` | `2000 0000 0104 0000 0404 0020 0100 2004` | $2^{-142}$ | $2^{-121.11}$ |

TABLE VIII
BOOMERANG DISTINGUISHERS FOR WARP

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 |
| 1 | 16 | 16 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 16 | 4 | 16 | 4 | 4 | 0 | 4 | 0 | 0 | 4 | 0 | 4 | 4 | 0 | 4 | 0 |
| 3 | 16 | 4 | 4 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 16 | 0 | 4 | 0 | 16 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 16 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 8 |
| 6 | 16 | 0 | 4 | 0 | 4 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 8 | 0 | 0 | 8 | 0 | 0 |
| 8 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 16 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 4 | 0 | 0 | 0 | 0 |
| A | 16 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 0 | 0 | 16 | 0 | 0 | 8 | 0 | 8 |
| B | 16 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 16 | 0 | 0 | 0 | 0 |
| C | 16 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 4 | 0 |
| D | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 8 | 0 | 0 | 16 | 0 | 0 |
| E | 16 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 16 | 0 |
| F | 16 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 16 |

TABLE IX
WARP's Feistel Boomerang Connectivity Table

The search itself is generic to Feistel-like ciphers and can be adapted to other design paradigms such as SPN. Since the automated search was written as a new module for CryptoSMT, we can run the boomerang search for other Feistel-like ciphers in its cipher suite. Several examples of boomerang distinguishers for TWINE and LBlock-s found using the proposed boomerang search are shown in Table VII[2]. The best boomerang distinguishers that we could find for WARP are summarized in Table VIII. Both distinguishers have $\mathsf{R}_{E_m} = 1$. The first 20-round boomerang distinguisher with $\mathsf{R}_{E_0} = 9$ and $\mathsf{R}_{E_1} = 10$ has an overall probability of $\mathsf{DP} = 2^{-114.24}$. The boomerang distinguisher is made up of only 1 $E_0$ differential matched with 351 different $E_1$ differentials. The 21-round boomerang distinguisher with $\mathsf{R}_{E_0} = 10$ and $\mathsf{R}_{E_1} = 10$ has an overall probability of $\mathsf{DP} = 2^{-121.11}$. It consists of 5 $E_0$ differentials matched with 316 $E_1$ differentials.

[2]These distinguishers only serve to showcase the flexibility of the proposed boomerang search, and may not be the best boomerang distinguishers found for these ciphers.

## C. Related-key Differential Distinguishers

Although its designers do not claim any security in the related-key setting, WARP could possibly be used to design other primitives such as hash functions or used in certain applications for which resilience against related-key attacks are important[3]. We found that WARP has a family of 2-round iterative related-key differential trails:

**Property 2 (2-round Related-key Trails).** *Let $i$ be an odd-numbered index (1,3,...,29,31) of a nonzero nibble in the input difference and $x$ be the nibble's difference. The input difference $\alpha$ consists of all zero nibbles except $\alpha_i = x$. When $K^1_{\frac{\pi^{-1}(i)}{2}} = y$, $K^0_{\frac{(\pi^{-1})^2(i)-1}{2}} = x$ and $K^0_{\frac{i-1}{2}} = x$, we have a 2-round related-key differential trail from $\alpha \to \alpha$ with $\mathsf{DP} = \frac{DDT(x,y)}{16}$.*

Depending on the DDT (Table V), these trails can either have a differential probability of $\frac{4}{16} = 2^{-2}$ or $\frac{2}{16} = 2^{-3}$. Figure 4 illustrates two examples of trails described in Property 2. For a more concrete example, we set $i = 3$, $x = 1$ and $y = 2$ and have the following differential propagation that follows the red trail in Figure 4:

$$\texttt{0000...0000000000001000} \xrightarrow[2^{-0}]{1r}$$
$$\texttt{0000...0000010000000000}\ (2^{-0}) \xrightarrow[2^{-2}]{1r}$$
$$\texttt{0000...0000000000001000}\ (2^{-2}),$$

where the key difference is $\Delta K = \{\Delta K_1 = \texttt{0000000000200000}, \Delta K_0 = \texttt{0000000010000010}\}$. The trail's differential probability is $\frac{DDT(1,2)}{16} = 2^{-2}$. We can then concatenate this iterative related-key differential trail 20.5 times to obtain a differential distinguisher for the full 41-round WARP that holds with a probability of $2^{-40}$. We estimate that at least $2^{40}$ plaintext pairs with the required input difference, $\alpha$ is sufficient to produce at least one right pair fulfilling the expected output difference, $\beta$ for the given key difference of $\Delta K$.

[3]Other block ciphers such as GIFT have also been extensively cryptanalyzed using related-key attacks despite not claiming any security in this setting [29], [43], [44].

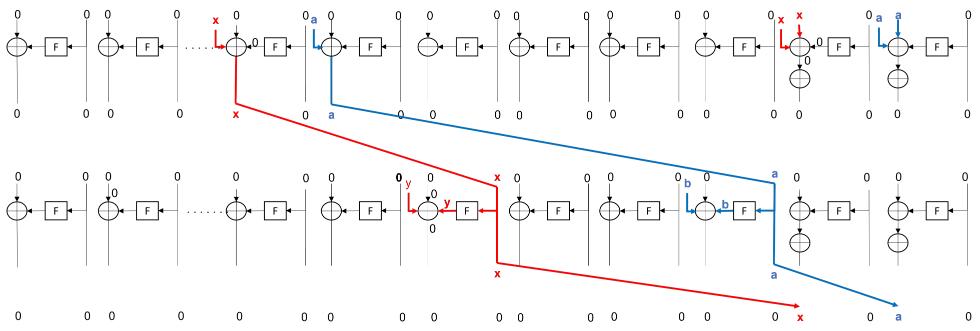


Fig. 4. Two examples of the 2-round iterative related-key differential trails for WARP where $i = 3$ and $i = 1$ are represented by the red and blue trails respectively (Property 2)

## IV. Differential Attacks on WARP

We denote an R-round cipher, $E$ as $E = E_f \circ E' \circ E_b$, where $E'$ is our differential distinguisher. The $\mathsf{R}_b$-round $E_b$ and $\mathsf{R}_f$-round $E_f$ are rounds added before and after the distinguisher, respectively. The input difference of $E_b$ and the output difference of $E_f$ are denoted as $\Delta P$ and $\Delta C$. We denote the number of active or unknown bits of $\Delta P$ as $r_b$ while the $n - r_b$ inactive or fixed bits are denoted as $\hat{r}_b$ and $\bar{r}_b$ for 0s and 1s, respectively. Analogously, these bits are denoted as $r_b$, $\hat{r}_b$ and $\bar{r}_b$ for $\Delta C$. We adopt a targeted approach for the key counting procedure by strategically guessing and filtering keys involved in subround filters described in Property 1. The number of keys that need to be guessed is denoted as $m$.

**Data Preparation.** We collect $y = 2 \cdot 2^{-r_b} \cdot \frac{s}{\mathsf{DP}}$ structures of $2^{r_b}$ plaintexts each, where $s$ is the expected number of right pairs. The plaintexts traverse all possible values for the active $r_b$ bits while the $\hat{r}_b$ and $\bar{r}_b$ bits are assigned suitable constants. Notably, half of the plaintexts should have the $\bar{r}_b$ bits set to 0 while the other half has these set to 1. We encrypt all $2^{r_b}$ plaintexts of each structure by querying an encryption oracle to obtain $2^{r_b}$ corresponding ciphertexts. We store the data in a hash table, $H$ indexed by the $\hat{r}_f$ ciphertext bits. Each structure will have $2^{2r_b-1}$ ciphertext pairs at the beginning.

**Key Recovery.** We initialize a list of $2^m$ counters then:

1) We filter wrong pairs using inactive bits of $\Delta C$, leaving $\frac{2^{2r_b-1}}{\hat{r}_f+\bar{r}_f}$ pairs.
2) If we have $v_f$ pairs of nibbles in the final round with the difference propagation pattern described in Property 1 (subround filters), we can filter more pairs, leaving $\frac{2^{2r_b-1}}{\hat{r}_f+\bar{r}_f} \cdot 2^{-4\times v_f}$ pairs.
3) Consult the hash table to find plaintext pairs for the remaining ciphertext pairs.
4) If we have $v_b$ subround filters, we can filter more pairs and with $\frac{2^{2r_b-1}}{\hat{r}_f+\bar{r}_f} \cdot 2^{-4\times v_f} \cdot 2^{-4\times v_b}$ remaining. We denote the time complexity of Steps 2, 3 and 4 as $\theta$.
5) For all $\frac{2^{2r_b-1}}{\hat{r}_f+\bar{r}_f} \cdot 2^{-4\times v_f} \cdot 2^{-4\times v_b}$ remaining pairs (per pair of structures), we strategically guess subkeys to perform partial decryptions (or encryptions) to derive the differences involved in the subround filters of the other rounds to discard invalid pairs and keys. We guess a total of $m$ subkey bits such that $m$ is of sufficient size for key counting with respect to the remaining pairs or combinations of pairs associated with the guessed key bits. We denote the time complexity of this step as $\epsilon$.
6) We select the top $2^{m-a}$ hits in the counter to be candidates that deliver an $a$-bit or higher advantage [45], then brute-force the $k - m$ remaining bits of the secret key.

| R | Input Difference ($\Delta P$) | ???? A000 A0?? ???A ?F?? 0000 A0A0 ??50 |
|---|---|---|
| 1 | After S-box and XOR ($\Delta X^1$)<br>After Permutation ($\Delta Y^1$) | 0?0? A000 A07? 0?0A 0F0? 0000 A0A0 0?50<br>00?7 00?A F000 00?A A0?A 00?A ?500 0000 |
| 2 | After S-box and XOR ($\Delta X^2$)<br>After Permutation ($\Delta Y^2$) | 00A7 000A F000 000A A00A 000A 5500 0000<br>0000 A0AF 005A 0000 A000 AA75 0000 0000 |
| 2-20 | Differential distinguisher,<br>$\alpha \to \beta$ | 0000 A0AF 005A 0000 A000 AA75 0000 0000<br>000A 5000 0AA0 0000 5A00 000A 0000 0A00 |
| 21 | After S-box and XOR ($\Delta X^{21}$)<br>After Permutation ($\Delta Y^{21}$) | 00?A 5000 ?AA0 0000 ?A00 00?A 0000 ?A00<br>00AA A00? A00? 0005 0?00 0?A0 00A0 0?00 |
| 22 | After S-box and XOR ($\Delta X^{22}$)<br>After Permutation ($\Delta Y^{22}$) | 00?A A0?? A0?? 00?5 ??00 ??A0 00A0 ??00<br>00?? 000A ??0? 0??A 5?0A ??A0 0000 ?A0? |
| 23 | After S-box and XOR ($\Delta X^{23}$)/<br>Output Difference ($\Delta C$) | 00?? 00?A ???? ???A ???A ??A0 0000 ?A?? |

Additional Notes: ? denotes an undetermined nibble. Red text denotes subround filters based on Property 1.

TABLE X
THE 23-ROUND KEY RECOVERY MODEL FOR WARP USING AN 18-ROUND DIFFERENTIAL

**Complexity Estimation.** The data complexity is around $N = y \cdot 2^{r_b - 1}$ while the memory complexity includes the space required to store the hash table, $H$ and the key counters, which amounts to $N + 2^m \cdot \frac{m}{128}$ 128-bit blocks. The time complexity, which includes the data preparation, filtering based on Property 1 and key recovery is approximately $N + \theta + \epsilon + 2^{n-a}$ encryptions.

## A. 23-round Attack using 18-round Differential

We use the 18-round differential from Table VI with DP $= 2^{-104.62}$ to mount an attack on 23-round WARP by adding 2 rounds at the beginning and 3 rounds at the end. The 23-round key recovery model is depicted in Table X, where we have ($r_b = 56$, $\hat{r}_b = 56$, $\bar{r}_b = 16$) and ($r_f = 72$, $\hat{r}_f = 46$, $\bar{r}_f = 10$). We guess a total of $m = 56$ subkey bits, corresponding to $K_i^0$ and $K_j^1$, where $i = \{0, 1, 4, 5, 7, 8, 9, 11, 14\}$ and $j = \{2, 4, 11, 13, 14\}$. The attack procedure is as follows:

**Data Preparation.** We let $s = 8$ and collect $y = 2 \cdot 2^{-56} \cdot \frac{8}{2^{-104.62}} \approx 2^{52.62}$ structures of $2^{56}$ plaintexts each. We encrypt all $2^{56}$ plaintexts to obtain $2^{56}$ corresponding ciphertexts of all structures that are stored in a hash table $H$, according to the 46 $\hat{r}_f$ bits set to 0. For each pair of structures, we have $2^{111}$ pairs at the beginning.

**Key Recovery.** We initialize a list of $2^{56}$ counters then:

1) We filter wrong pairs using inactive bits of $\Delta C$, leaving $2^{111-56} = 2^{55}$ pairs per structure or $2^{52.62+55} = 2^{107.62}$ pairs in total.
2) The number of filters (Property 1) in the first and final rounds are $v_b = 8$ and $v_f = 6$, respectively as indicated in red in Table X. Thus, the number of valid pairs would be reduced to $2^{107.62-4\times8-4\times6} = 2^{51.62}$.
3) **In Round 23:** We can propagate knowledge of the output difference to Round 22 without having to guess any keys (Property 1).
4) **In Round 22:** In this round, we guess and filter subkeys for each remaining pair based on Property 1. For example, to calculate $\Delta X_2^{22} = \Delta Y_{29}^{22}$, we need to guess $K_{14}^0$ to partially decrypt $\Delta X_{29}^{23}$. On the other hand, $\Delta X_3^{22} = \Delta Y_{14}^{22} = \Delta X_{14}^{23}$ can be directly computed from the ciphertext pairs. Then, we

can check if $\Delta X_3^{22} \oplus S(\Delta X_2^{22}) = \Delta Y_3^{21} = 0$. If the equality holds, then we keep the guessed value of $K_{14}^0$ and the pair, otherwise we discard them. There will be around $2^{49.62} \cdot 2^4 \cdot 2^{-4} = 2^{49.62}$ combinations of the remaining pairs associated with the guessed $K_{14}^0$ values. In other words, there remains around $2^{14} \cdot 2^{-4}$ pairs with $2^4$ candidate values of $K_{14}^0$. We have 6 more of these filters in Round 22, for which we can additionally guess and filter $K_1^0$, $K_4^0$, $K_5^0$, $K_7^0$, $K_8^0$ and $K_{11}^0$ candidates. We expect to have $2^{51.62}$ combinations of the remaining pairs associated with 28-bit key candidates.

5) **In Round 21:** We need to guess $K_{14}^1$, $K_{11}^1$ and $K_4^1$ to calculate $\Delta X_2^{21}$, $\Delta X_{14}^{21}$ and $\Delta X_{28}^{21}$ respectively, while the remaining differences involved in those subround filters can be calculated based on the previous key guesses. After going through these filters, there will be $2^{51.62}$ combinations of the remaining pairs associated with 40-bit key candidates. For the remaining two filters, we need to guess ($K_9^0$,$K_{13}^1$) to calculate $\Delta X_8^{21}$ and ($K_0^0$,$K_2^1$) to calculate $\Delta X_{22}^{21}$. Since there are $2^8$ possible subkey candidates involved in each of these 4-bit filters, each guess will increase the number of combinations of pairs and keys by $2^8 \cdot 2^{-4} = 2^4$. Thus, we will end up with $2^{51.62} \cdot 2^{4\times 2} = 2^{59.62}$ combinations of the remaining pairs associated with 56-bit key candidates.
6) **In Round 1:** For all the remaining pairs, we can propagate knowledge of the input difference to Round 2 without having to guess any keys (Property 1).
7) **In Round 2:** $K^0$ candidates that have been filtered in the earlier steps can be used to filter more pairs based on Property 1. Differences $\Delta Y_6^1$ and $\Delta Y_{24}^1$ can be calculated using $K_0^0$ and $K_{11}^0$ candidates already associated with each remaining pair. The other differences in those Feistel-subrounds, $\Delta Y_7^1$ and $\Delta Y_{25}^1$, can be calculated directly from the plaintext pairs. We can then discard combinations of pairs and keys based on the known differences, $\Delta X_7^2 = 5$ and $\Delta X_{25}^2 = 0$ due to Property 1. This reduces the number of possible combinations to $2^{59.62-4\times 2} = 2^{51.62}$.
8) We increment the key counters based on the $2^{51.62}$ remaining combinations of pairs associated with the 56 bits of guessed keys. We expect that on average, 8 pairs will vote for the right key while the remaining pairs will vote for a random key with a probability of $2^{51.62-56} = 2^{-4.38}$.
9) We select the top $2^{56-52} = 16$ hits in the counter and brute-force the 72 remaining bits of the secret key.

**Complexity Estimation.** The data and memory complexities are $N = 2^{52.62} \cdot 2^{56} \approx 2^{108.62}$ plaintexts and $2^{108.62} + 2^{56} \cdot \frac{56}{128} \approx 2^{108.62}$ 128-bit blocks, respectively. The time complexity of the key recovery is dominated by the final round filtering in Step 2, in which the $2^{107.62}$ pairs need to be partially decrypted. This requires $\theta = 2^{107.62} \cdot \frac{2}{23} \approx 2^{104.09}$ 23-round WARP encryptions. Therefore, the time complexity of the 23-round differential attack is about $2^{108.62} + 2^{104.09} + 2^{76} \approx 2^{108.68}$ 23-round WARP encryptions when $a = 52$.

**Success Probability.** We calculate the probability of success, $\mathsf{Pr_S}$ of our attack based on the method proposed by Selçuk [45]:

$$\mathsf{Pr_S} = \Phi\left(\frac{\sqrt{s \cdot S_N} - \Phi^{-1}(1-2^{-a})}{\sqrt{S_N + 1}}\right), \qquad (7)$$

where the signal-to-noise ratio is calculated as $S_N = \frac{\mathsf{DP}}{2^{-n}}$. With $a = 52$, the probability that the attack succeeds is 99.76%.

### *B. 24-round Rectangle Attack using 21-round Boomerang Distinguisher*

We use the 21-round boomerang distinguisher from Table VIII where $\mathsf{R}_{E_0} = 10$, $\mathsf{R}_{E_1} = 10$ and $\sum_{i,j} \hat{p_i}^2 \hat{q_j}^2 r_{i,j}) = 2^{-121.11}$ to mount a rectangle attack on 24-round WARP by appending 1 round at the beginning and 2 rounds at the end. Based on this boomerang distinguisher, we expect that the probability of a quartet to be a right rectangle

| R | | |
|---|---|---|
| | Input Difference ($\Delta P$) | `00?E 0000 E000 ?EE0 ?E?E 0000 E000 00?E` |
| 1 | After S-box and XOR ($\Delta X^1$) | `000E 0000 E000 0EE0 0E0E 0000 E000 000E` |
| | After Permutation ($\Delta Y^1$) | `E000 00EE EE00 0000 00E0 00EE 0000 0000` |
| 2-22 | Boomerang distinguisher, | `E000 00EE EE00 0000 00E0 00EE 0000 0000` |
| | $\alpha \rightarrow \delta$ | `2000 0000 0104 0000 0404 0020 0100 2004` |
| 23 | After S-box and XOR ($\Delta X^{23}$) | `2000 0000 ?1?4 0000 ?4?4 0020 ?100 20?4` |
| | After Permutation ($\Delta Y^{23}$) | `400? 024? 4010 0040 0200 0?0? 0?1? 0200` |
| 24 | After S-box and XOR ($\Delta X^{24}$)/ Output Difference ($\Delta C$) | `40?? ?2?? 4010 0040 ?200 ???? ???? ?200` |

? denotes an undetermined nibble. Red text denotes subround filters based on Property 1.

TABLE XI
THE 24-ROUND KEY RECOVERY MODEL OF THE RECTANGLE ATTACK FOR WARP USING A 21-ROUND (10+1+10) BOOMERANG DISTINGUISHER

quartet is $2^{-n} \cdot (\sum_{i,j} \hat{p_i}^2 \hat{q_j}^2 r_{i,j}) = 2^{-249.11}$. The 24-round key recovery model is depicted in Table XI, where we have ($r_b = 20$, $\hat{r}_b = 84$, $\bar{r}_b = 24$) and ($r_f = 60$, $\hat{r}_f = 61$, $\bar{r}_f = 7$). The number of subkey bits that will be guessed are $m_f = 16$, corresponding to $K_j^1$ where $j = \{2, 8, 10, 15\}$. Based on prior work by Biham et al. [36] and Zhao et al. [43], details of our rectangle attack are as follows:

**Data Preparation.** To have $s = 8$ right quartets, we collect $y = \frac{\sqrt{8} \cdot 2^{64-20}}{\sqrt{2^{-121.11}}} = 2^{106.06}$ structures of $2^{20}$ plaintexts each. Similar to subsection IV-A, the plaintexts of each structure are assigned all possible combinations of the $r_b$ active bits while the other bits are assigned suitable constants. We encrypt all $2^{20}$ plaintexts of each structure to obtain $2^{20}$ corresponding ciphertexts. We store the plaintext-ciphertext pairs in a hash table, $H_1$, indexed by the $r_b$ bits of the plaintext.

**Key Recovery.** We initialize a list of $2^{16}$ counters then:

1) We construct a set $S = \{(P_1, C_1, P_2, C_2) : E_b(P_1) \oplus E_b(P_2) = \alpha\}$. We can construct $S$ without having to guess any keys in $E_b$ as follows:
   a) For every plaintext $P_1$ in a structure, we determine the known $\hat{r_b} + \bar{r_b}$ bits in $P_2$ by calculating $P_2 = P_1 \oplus \Delta P$.
   b) The unknown nibbles in $P_2$, which are all left input nibbles to Feistel-subrounds, can be calculated from their corresponding right input nibbles. Let the pairs of nibbles for $P_1$ and $P_2$ be denoted as ($x_L^1$, $x_R^1$) and ($x_L^2$, $x_R^2$) respectively (see Figure 3). We already know the values for the right halves ($x_R^1$, $x_R^2$) after Step 1(a) and we also know the value of $x_L^1$ from $P_1$. We can then calculate the remaining unknown value as
   $$x_L^2 = x_L^1 \oplus S(x_R^1) \oplus S(x_R^2).$$
   For example, if $(\Delta P_1, \Delta P_0) = \texttt{?E}$, and $(x_L^1, x_R^1) = \texttt{7F}$, we can calculate $x_R^2 = x_R^1 \oplus \Delta P_0 = \texttt{F} \oplus \texttt{E} = \texttt{1}$. Then, $x_L^2 = \texttt{7} \oplus S(\texttt{F}) \oplus S(\texttt{1}) = \texttt{7} \oplus \texttt{6} \oplus \texttt{A} = \texttt{B}$. We can then easily verify that $\Delta P_1 \oplus S(\Delta P_0) = \Delta X_1^1 = \texttt{0}$.
   c) After calculating all the unknown bits of $P_2$, we check $H_1$ to find the corresponding plaintext-ciphertext pair indexed by the $r_b$ bits of $P_2$. Since $v_b = 5$, we expect to have $2^{20 \times 2 - 1} \cdot 2^{-4 \times v_b} = 2^{19}$ pairs in $S$.
2) The size of $S$ is $N = 2^{106.06} \cdot 2^{(19+1) \times 2^{-1}} = 2^{116.06}$ chosen plaintexts. Insert $S$ into a hash table $H_2$ indexed by the 61 $\hat{r}_f$ bits of $C_1$

and $C_2$. For each element of $S$, we check $H_2$ to find $(P_1, C_1, P_2, C_2)$ where $(C_1, C_3)$ and $(C_2, C_4)$ collide in the $\hat{r}_f + \bar{r}_f = 68$ known bits. There will be $(2^{116.06})^2 \cdot 2^{-2\times 68} = 2^{96.12}$ quartets remaining.

3) The number of subround filters (Property 1) in the final round is $v_f = 9$ as indicated in red in Table XI. Thus, the number of valid quartets would be reduced to $2^{96.12-8\times 9} = 2^{24.12}$. The filtering effect due to Property 1 is twofold since it is applicable to both pairs in a quartet.
4) **In Round 24:** We propagate the knowledge of the output difference to Round 23 (Property 1).
5) **In Round 23:** Perform the guess-and-filter procedure for subkeys in Round 23. For example, $\Delta X_0^{23}$ can be directly computed from the ciphertext pairs. We guess $K_{15}^1$ and partially decrypt $(C_1, C_3)$ and $(C_2, C_3)$ to obtain $\Delta X_1^{23}$ for each pair. Since the difference $\delta_1$ is known, we can check if each pair in the quartet fulfills $\Delta X_1^{23} \oplus S(\Delta X_0^{23}) = \delta_1 = 0$. If the equality holds for both pairs in the quartet, we keep the guessed key and the quartet, otherwise, we discard them. There will be around $2^{24.12} \cdot 2^4 \cdot 2^{-8} = 2^{-20.12}$ combinations of the remaining quartets associated with the guessed $K_{15}^1$ values. In other words, there remains around $2^{24.12} \cdot 2^{-8}$ quartets with $2^4$ candidate values of $K_{15}^1$ each. We guess and filter three more subkeys, $K_2^1$, $K_8^1$ and $K_{10}^1$, which leaves $2^{20.12} \cdot 2^{-4\times 3} = 2^{8.12}$ combinations of the remaining quartets associated with the guessed keys.
6) We increment the key counters based on the $2^{8.12}$ remaining combinations of quartets associated with the 16 bits of guessed keys. We expect that on average, 8 quartets will vote for the right key while the remaining quartets will vote for a random key with a probability of $2^{6.12-16} = 2^{-7.88}$.
7) We select the top $2^{16-12} = 16$ hits in the counter and brute force the 112 remaining bits of the secret key.

**Complexity Estimation.** The data complexity of the attack is $N = 2^{106.06} \cdot 2^{20} \approx 2^{126.06}$ chosen plaintexts. The memory complexity includes the space required to store the hash tables and the key counters, which is $2 \cdot 2^{126.06} + 2^{24} \cdot \frac{24}{128} \approx 2^{127.06}$ 128-bit blocks. To prepare the quartets, we require around $2N$ memory accesses and the time complexity of the key recovery is dominated by the final round filtering in Step 3, which is approximately $\theta = 2^{96.12} \cdot \frac{4}{24} \approx 2^{93.54}$. The overall time complexity of the 24-round attack is $2 \cdot 2^{126.06} \cdot \frac{1}{24} + 2^{93.54} + 2^{116} \approx 2^{122.49}$ 24-round WARP encryptions when $a = 12$.

**Success Probability.** When $a = 12$ and $S_N = \frac{\sum_{i,j}(\hat{p_i}^2 \hat{q_j}^2 r_{i,j})}{2^{-n}} = 2^{-121.11}$, the probability that the attack succeeds is 99.38%.

## V. Related-key Differential Attacks on WARP

In this section, we show how the 2-round iterative related-key differential trails from subsection III-C can be used in key-recovery attacks against WARP. First, we formulate a 25-round related-key differential attack on WARP which we can verify experimentally. Then, we describe a related-key differential attack on the full WARP using the same attack procedure.

### A. 25-round Related-key Differential Attack

We concatenate 9 instances of the 2-round iterative related-key differential trail described in subsection III-C and append 1 more round to form a 19-round distinguisher with DP$= 2^{-18}$. After appending 6 rounds to the end of this distinguisher, we have a 25-round key-recovery model depicted in Table XII where $r = 19$. We guess a total of 16 subkey bits, corresponding to $K_i^0$ where $i = \{4, 7, 10, 14\}$. Although it may be possible to guess more key bits to reduce the computational complexity of the final brute force step, we stick with 16 bits so we can computationally verify the attack efficiently. The attack procedure is as follows:

| R $(\Delta K)$ | | |
|---|---|---|
| $0 - r$ | RK differential distinguisher,<br>$\alpha \rightarrow \beta$ | `0000 0000 0000 0000 0000 0000 0000 1000`<br>`0000 0000 0000 0000 0000 0100 0000 0000` |
| $r + 1$ $(\Delta K_1)$ | After S-box and XOR $(\Delta X^{r+1})$<br>After Permutation $(\Delta Y^{r+1})$ | `0000 0000 0000 0000 0000 ?100 0000 0000`<br>`0000 0000 0000 0000 0000 0000 0000 100?` |
| $r + 2$ $(\Delta K_0)$ | After S-box and XOR $(\Delta X^{r+2})$<br>After Permutation $(\Delta Y^{r+2})$ | `0000 0000 0000 0000 1000 0000 0000 00??`<br>`?000 0000 0000 0000 0000 0100 0?00 0000` |
| $r + 3$ $(\Delta K_1)$ | After S-box and XOR $(\Delta X^{r+3})$<br>After Permutation $(\Delta Y^{r+3})$ | `?000 0000 0000 0000 0000 ?100 ??00 0000`<br>`0000 0?00 00?0 0000 0000 000? 0000 100?` |
| $r + 4$ $(\Delta K_0)$ | After S-box and XOR $(\Delta X^{r+4})$<br>After Permutation $(\Delta Y^{r+4})$ | `0000 ??00 00?0 0000 1000 00?? 0000 00??`<br>`?00? ?000 0000 ?00? 0000 0100 0?00 0?00` |
| $r + 5$ $(\Delta K_1)$ | After S-box and XOR $(\Delta X^{r+5})$<br>After Permutation $(\Delta Y^{r+5})$ | `?0?? ?000 0000 ?0?? 0000 ?100 ??00 ??00`<br>`0??0 0?00 0??? 000? ??00 00?? 0000 100?` |
| $r + 6$ $(\Delta K_0)$ | After S-box and XOR $(\Delta X^{r+6})$<br>Output Difference $(\Delta C)$ | `???0 ??00 ???? 00?? ??00 00?? 0000 00??` |

? denotes an undetermined nibble. Red text denotes subround filters based on Property 1.

TABLE XII
AN $r + 1$-ROUND KEY RECOVERY MODEL OF THE DIFFERENTIAL ATTACK FOR WARP USING AN $r$-ROUND RELATED KEY DIFFERENTIAL DISTINGUISHER WHERE $r \in \{3, 5, ..., 31\}$

**Data Preparation.** We encrypt $2^{20}$ pairs of plaintexts, $(P_1, P_2)$ with the required input difference $P_1 \oplus P_2 = \alpha$ using a pair of related keys, $(K, K \oplus \Delta K)$. We expect $s = 2^2$ right pairs. There is a strong filtering effect at the output difference $\Delta C$, which has 60 inactive bits and 5 subround filters (Property 1). The probability of a wrong pair surviving is $2^{20} \cdot 2^{-60} \cdot 2^{-4\times 5} = 2^{-60}$, which implies that only the right pairs remain.

**Key Recovery.** For all the remaining (right) pairs:

1) **In Round 25:** We propagate knowledge of the output difference to Round 24 without having to guess any keys (Property 1).
2) **In Round 24:** We guess $K^0_{14}$ to calculate $2^{24}_2$, then derive $2^{24}_3$ from the ciphertext pairs. Since all pairs are valid, none of them will be discarded. Instead, each pair will be associated with at least one possible 4-bit key candidate, one of which is the correct subkey. We repeat the procedure by guessing $K^0_{10}$, $K^0_7$ and $K^0_4$ to calculate $\Delta X^{24}_6$, $\Delta X^{24}_{16}$ and $\Delta X^{24}_{28}$ respectively, while the remaining differences involved in those subround filters can be calculated from the ciphertexts.

**Complexity Estimation.** The data complexity of the attack is $N = 2 \cdot 2^{20} = 2^{21}$ chosen plaintexts and it finds correctly 16-bits of the key, which is sufficient for verification of the attack correctness. The memory requirement for the attack is negligible because only 4-bit counters are required for each subkey.

**Computational Verification.** We execute the entire attack 10 times with randomly selected keys and plaintexts on a PC with an Intel Core i7-9700K 3.60GHz processor and 32GB of RAM. The correct subkey candidates always have the highest count of $s$, and will be among the top $2^4$ key candidates 70% of the time. On average, the attack completes in under 5 minutes using an unoptimized implementation. Codes associated with the 25-round attack as well as supplementary codes to verify the correctness of our WARP implementation is available at https://github.com/jesenteh/cryptosmt-boomerang/.

## *B. 41-round (Full) Related-key Differential Attack*

The key recovery model for a 41-attack on WARP using a 35-round distinguisher with $\mathsf{DP} = 2^{-34}$ is also depicted in Table XII where $r = 35$. We generate $2^{36}$ pairs and expect $2^2$ right pairs. After filtering, we expect that only the right pairs remain. From Round 40 to Round 36, we guess a total of 60 subkey bits (16 in Round 40, 12 in Round 39, 16 bits in 38, 12 in Round 37 and 4 in Round 36). There are subround filters in Rounds 37 and 38 that require guessing 12 key bits, so we need key counters that can accommodate $2^{12}$ possibilities for these instances. The memory requirement is $(6 \cdot 2^4 \cdot \frac{4}{128} + 2 \cdot 2^{12} \cdot \frac{12}{128}) \approx 2^{9.59}$ 128-bit blocks. The time complexity for the guess-and-determine procedure is negligible, therefore recovering 60 bits of the key comes mainly from encrypting the $2^{37}$ chosen plaintexts. We can either brute force the remaining 68 bits, which would then dominate the time complexity or use faster auxiliary techniques to find the rest of the key.

## VI. Conclusion

This paper described cryptanalytic attacks on the energy-efficient lightweight block cipher WARP, which is currently the smallest 128-bit block cipher in terms of hardware requirements. All attacks were based on differential cryptanalysis and considered both the single-key and related-key settings. We found efficient differential distinguishers for the first 20 rounds of WARP. Differential clustering for these trails revealed that WARP has a strong differential effect from round 13 onward. We also introduced an automatic search for boomerang distinguishers which is applicable to Feistel-like ciphers. We found boomerang distinguishers for WARP for up to 21 rounds and also demonstrated its flexibility on other lightweight block ciphers, TWINE and LBlock-s. Next, we described a family of 2-round iterative related-key differential trails with only 1 active S-box. These trails can be concatenated to construct a full 41-round distinguisher for WARP with a probability of $2^{-40}$. We then proposed key-recovery attacks on WARP using the various distinguishers that have been identified. In the single-key setting, we attacked 23 and 24 rounds of WARP using an 18-round differential and 21-round boomerang distinguisher, respectively. We proposed and computationally verified a 25-round related-key attack on WARP using a 19-round related-key differential distinguisher, whereby 16 subkey bits can be recovered in under 5 minutes. Using the same framework, we finally introduce a related-key attack on the full 41 rounds of WARP. All attack complexities were summarized in Table I. To the best of our knowledge, this is the first (valid) 3rd party cryptanalysis results for WARP.

## Acknowledgements

Je Sen Teh was supported by the Luxembourgish Fonds National de la Recherche (FNR) and the German Research Foundation (DFG) project APLICA (C19/IS/13641232).

## References

[1] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, "Midori: A block cipher for low energy," in *ASIACRYPT 2015, Part II*, ser. LNCS, T. Iwata and J. H. Cheon, Eds., vol. 9453. Springer, Heidelberg, Nov. / Dec. 2015, pp. 411–436.

[2] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A small present - towards reaching the limit of lightweight encryption," in *CHES 2017*, ser. LNCS, W. Fischer and N. Homma, Eds., vol. 10529. Springer, Heidelberg, Sep. 2017, pp. 321–345.

[3] Y. Zheng, T. Matsumoto, and H. Imai, "On the construction of block ciphers provably secure and not relying on any unproved hypotheses," in *CRYPTO'89*, ser. LNCS, G. Brassard, Ed., vol. 435. Springer, Heidelberg, Aug. 1990, pp. 461–480.

[4] S. Banik, Z. Bao, T. Isobe, H. Kubo, F. Liu, K. Minematsu, K. Sakamoto, N. Shibata, and M. Shigeri, "WARP : Revisiting GFN for lightweight 128-bit block cipher," in *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, ser. Lecture Notes in Computer Science, O. Dunkelman, M. J. J. Jr., and C. O'Flynn, Eds., vol. 12804. Springer, 2020, pp. 535–564. [Online]. Available: https://doi.org/10.1007/978-3-030-81652-0_21

[5] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE : A lightweight block cipher for multiple platforms," in *SAC 2012*, ser. LNCS, L. R. Knudsen and H. Wu, Eds., vol. 7707. Springer, Heidelberg, Aug. 2013, pp. 339–354.

[6] S. Banik, A. Bogdanov, A. Luykx, and E. Tischhauser, "SUNDAE: Small universal deterministic authenticated encryption for the internet of things," *IACR Trans. Symm. Cryptol.*, vol. 2018, no. 3, pp. 1–35, 2018.

[7] M. Kumar and T. Yadav, “MILP based differential attack on round reduced WARP,” Cryptology ePrint Archive, Report 2020/1598, Version 20210802:090929, 2020, https://ia.cr/2020/1598.

[8] H. Boukerrou, P. Huynh, V. Lallemand, B. Mandal, and M. Minier, “On the feistel counterpart of the boomerang connectivity table introduction and analysis of the FBCT,” *IACR Trans. Symmetric Cryptol.*, vol. 2020, no. 1, pp. 331–362, 2020. [Online]. Available: https://doi.org/10.13154/tosc.v2020.i1.331-362

[9] Stefan Kölbl, “CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives,” https://github.com/kste/cryptosmt.

[10] L. Zhang, W. Wu, Y. Wang, S. Wu, and J. Zhang, “LAC,” Tech. Rep., 2014, available at https://competitions.cr.yp.to/round1/lacv1.pdf.

[11] X. Lai, J. L. Massey, and S. Murphy, “Markov ciphers and differential cryptanalysis,” in *EUROCRYPT’91*, ser. LNCS, D. W. Davies, Ed., vol. 547. Springer, Heidelberg, Apr. 1991, pp. 17–38.

[12] E. Biham, “New types of cryptanalytic attacks using related keys (extended abstract),” in *EUROCRYPT’93*, ser. LNCS, T. Helleseth, Ed., vol. 765. Springer, Heidelberg, May 1994, pp. 398–409.

[13] E. Biham, O. Dunkelman, and N. Keller, “A related-key rectangle attack on the full KASUMI,” in *ASIACRYPT 2005*, ser. LNCS, B. K. Roy, Ed., vol. 3788. Springer, Heidelberg, Dec. 2005, pp. 443–461.

[14] K. Jeong, C. Lee, J. Sung, S. Hong, and J. Lim, “Related-key amplified boomerang attacks on the full-round Eagle-64 and Eagle-128,” in *ACISP 07*, ser. LNCS, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., vol. 4586. Springer, Heidelberg, Jul. 2007, pp. 143–157.

[15] A. Biryukov and D. Khovratovich, “Related-key cryptanalysis of the full AES-192 and AES-256,” in *ASIACRYPT 2009*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer, Heidelberg, Dec. 2009, pp. 1–18.

[16] O. Dunkelman, N. Keller, and A. Shamir, “A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony,” in *CRYPTO 2010*, ser. LNCS, T. Rabin, Ed., vol. 6223. Springer, Heidelberg, Aug. 2010, pp. 393–410.

[17] T. Ashur and O. Dunkelman, “A practical related-key boomerang attack for the full MMB block cipher,” in *CANS 13*, ser. LNCS, M. Abdalla, C. Nita-Rotaru, and R. Dahab, Eds., vol. 8257. Springer, Heidelberg, Nov. 2013, pp. 271–290.

[18] J. Lu, W.-S. Yap, and Y. Wei, “Weak keys of the full MISTY1 block cipher for related-key differential cryptanalysis,” in *CT-RSA 2013*, ser. LNCS, E. Dawson, Ed., vol. 7779. Springer, Heidelberg, Feb. / Mar. 2013, pp. 389–404.

[19] Y. Sasaki, “Related-key boomerang attacks on full ANU lightweight block cipher,” in *ACNS 18*, ser. LNCS, B. Preneel and F. Vercauteren, Eds., vol. 10892. Springer, Heidelberg, Jul. 2018, pp. 421–439.

[20] G. Tsudik and E. Van Herreweghen, “On simple and secure key distribution,” in *ACM CCS 93*, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds. ACM Press, Nov. 1993, pp. 49–57.

[21] M. Broersma, “New Xbox security cracked by Linux fans,” 2002, https://www.zdnet.com/article/new-xbox-security-cracked-by-linux-fans/.

[22] J. Kelsey, B. Schneier, and D. Wagner, “Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA,” in *ICICS 97*, ser. LNCS, Y. Han, T. Okamoto, and S. Qing, Eds., vol. 1334. Springer, Heidelberg, Nov. 1997, pp. 233–246.

[23] E. Biham, O. Dunkelman, and N. Keller, “A unified approach to related-key attacks,” in *FSE 2008*, ser. LNCS, K. Nyberg, Ed., vol. 5086. Springer, Heidelberg, Feb. 2008, pp. 73–96.

[24] M. Matsui, “On correlation between the order of S-boxes and the strength of DES,” in *EUROCRYPT’94*, ser. LNCS, A. D. Santis, Ed., vol. 950. Springer, Heidelberg, May 1995, pp. 366–375.

[25] A. Biryukov and V. Velichkov, “Automatic search for differential trails in ARX ciphers,” in *CT-RSA 2014*, ser. LNCS, J. Benaloh, Ed., vol. 8366. Springer, Heidelberg, Feb. 2014, pp. 227–250.

[26] J. Chen, J. Teh, Z. Liu, C. Su, A. Samsudin, and Y. Xiang, “Towards accurate statistical analysis of security margins: New searching strategies for differential attacks,” *IEEE Trans. Computers*, vol. 66, no. 10, pp. 1763–1777, 2017. [Online]. Available: https://doi.org/10.1109/TC.2017.2699190

[27] N. Mouha, Q. Wang, D. Gu, and B. Preneel, “Differential and linear cryptanalysis using mixed-integer linear programming,” in *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, ser. Lecture Notes in Computer Science, C. Wu, M. Yung, and D. Lin, Eds., vol. 7537. Springer, 2011, pp. 57–76. [Online]. Available: https://doi.org/10.1007/978-3-642-34704-7_5

[28] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song, “Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers,” in *ASIACRYPT 2014, Part I*, ser. LNCS, P. Sarkar and T. Iwata, Eds., vol. 8873. Springer, Heidelberg, Dec. 2014, pp. 158–178.

[29] B. Zhu, X. Dong, and H. Yu, “MILP-based differential attack on round-reduced GIFT,” in *CT-RSA 2019*, ser. LNCS, M. Matsui, Ed., vol. 11405. Springer, Heidelberg, Mar. 2019, pp. 372–390.

[30] N. Mouha and B. Preneel, “Towards finding optimal differential characteristics for ARX: Application to Salsa20,” Cryptology ePrint Archive, Report 2013/328, 2013, https://eprint.iacr.org/2013/328.

[31] L. Sun, W. Wang, and M. Wang, “Accelerating the search of differential and linear characteristics with the SAT method,” *IACR Trans. Symmetric Cryptol.*, vol. 2021, no. 1, pp. 269–315, 2021. [Online]. Available: https://doi.org/10.46586/tosc.v2021.i1.269-315

[32] R. Ankele and S. Kölbl, “Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis,” in *SAC 2018*, ser. LNCS, C. Cid and M. J. Jacobson Jr:, Eds., vol. 11349. Springer, Heidelberg, Aug. 2019, pp. 163–190.

[33] M. Hall-Andersen and P. S. Vejre, “Generating graphs packed with paths,” *IACR Trans. Symm. Cryptol.*, vol. 2018, no. 3, pp. 265–289, 2018.
[34] D. Wagner, “The boomerang attack,” in *FSE’99*, ser. LNCS, L. R. Knudsen, Ed., vol. 1636. Springer, Heidelberg, Mar. 1999, pp. 156–170.
[35] J. Kelsey, T. Kohno, and B. Schneier, “Amplified boomerang attacks against reduced-round MARS and Serpent,” in *FSE 2000*, ser. LNCS, B. Schneier, Ed., vol. 1978. Springer, Heidelberg, Apr. 2001, pp. 75–93.
[36] E. Biham, O. Dunkelman, and N. Keller, “New results on boomerang and rectangle attacks,” in *FSE 2002*, ser. LNCS, J. Daemen and V. Rijmen, Eds., vol. 2365. Springer, Heidelberg, Feb. 2002, pp. 1–16.
[37] S. Murphy, “The return of the cryptographic boomerang,” *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2517–2521, 2011. [Online]. Available: https://doi.org/10.1109/TIT.2011.2111091
[38] C. Cid, T. Huang, T. Peyrin, Y. Sasaki, and L. Song, “Boomerang connectivity table: A new cryptanalysis tool,” in *EUROCRYPT 2018, Part II*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10821. Springer, Heidelberg, Apr. / May 2018, pp. 683–714.
[39] L. Song, X. Qin, and L. Hu, “Boomerang connectivity table revisited,” *IACR Trans. Symm. Cryptol.*, vol. 2019, no. 1, pp. 118–141, 2019.
[40] A. Biryukov and I. Nikolic, “Complementing Feistel ciphers,” in *FSE 2013*, ser. LNCS, S. Moriai, Ed., vol. 8424. Springer, Heidelberg, Mar. 2014, pp. 3–18.
[41] V. Ganesh and D. L. Dill, “A decision procedure for bit-vectors and arrays,” in *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings*, ser. Lecture Notes in Computer Science, W. Damm and H. Hermanns, Eds., vol. 4590. Springer, 2007, pp. 519–531. [Online]. Available: https://doi.org/10.1007/978-3-540-73368-3_52
[42] M. Soos, K. Nohl, and C. Castelluccia, “Extending SAT solvers to cryptographic problems,” in *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, ser. Lecture Notes in Computer Science, O. Kullmann, Ed., vol. 5584. Springer, 2009, pp. 244–257. [Online]. Available: https://doi.org/10.1007/978-3-642-02777-2_24
[43] B. Zhao, X. Dong, W. Meier, K. Jia, and G. Wang, “Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT,” *Des. Codes Cryptogr.*, vol. 88, no. 6, pp. 1103–1126, 2020. [Online]. Available: https://doi.org/10.1007/s10623-020-00730-1
[44] R. Zong, X. Dong, H. Chen, Y. Luo, S. Wang, and Z. Li, “Towards key-recovery-attack friendly distinguishers: Application to GIFT-128,” *IACR Trans. Symmetric Cryptol.*, vol. 2021, no. 1, pp. 156–184, 2021. [Online]. Available: https://doi.org/10.46586/tosc.v2021.i1.156-184
[45] A. A. Selçuk, “On probability of success in linear and differential cryptanalysis,” *Journal of Cryptology*, vol. 21, no. 1, pp. 131–147, Jan. 2008.

**Je Sen Teh** received his Ph.D. in Computer Science from Universiti Sains Malaysia 2017. He is currently a research associate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. He also holds a position as a Senior Lecturer in Universiti Sains Malaysia under the School of Computer Sciences.

**Alex Biryukov** is a full professor at the Department of Computer Science and Security and Trust Center, University of Luxembourg. He is an expert in cryptanalysis of symmetric cryptographic primitives and is interested in applied cryptography in general.