More Brisés in Ballet: Extending Differential and Linear Cryptanalysis

Emanuele Bellini[1], Gabriele Bellini[2], Alessandro De Piccoli[2], Michela Gallone[2], David Gerault[1], Yun Ju Huang, Paul Huynh[1], Matteo Onger[2], Simone Pelizzola[2] and Andrea Visconti[2]

[1] Technology Innovation Institute, Abu Dhabi, UAE.
{name.lastname}@tii.ae cs@crypto.tw
[2] Università degli Studi di Milano, Milan, Italy,
{alessandro.depiccoli,simone.pelizzola,andrea.visconti}@unimi.it
{gabriele.bellini,michela.gallone,matteo.onger}@studenti.unimi.it

Abstract. In this work, we present new cryptanalytic results on the Ballet block cipher family, a simplified Lay-Massey ARX construction with a linear key schedule, winner of the symmetric algorithm category in the 2018–2020 Chinese National Cryptographic Algorithm Competition. Despite winning the competition, the cipher has received limited attention outside the Chinese Association for Cryptologic Research (CACR) community. We provide the first classical key recovery attacks in the literature, new explicit differential and linear trails (up to 16 rounds for differential, and 16 for linear, while the original paper only provided a bound for 9 rounds), improved impossible differential trails (8 rounds instead of 7), and the first differential-linear analysis of Ballet (up to 20 rounds). Our results lead to key recovery attacks on up to 16 rounds of Ballet-128/128/46, 17 rounds of Ballet-128/256/48 and 22 rounds of Ballet-256/256/74, extending the cryptanalytic understanding of this ARX-based design and contributing new insight into its security margin, an area that the designers themselves note warrants further study.

Keywords: Cryptanalysis · Symmetric cipher · Ballet

1 Introduction

In 2018, the Chinese Association for Cryptologic Research (CACR) launched the National Cryptographic Algorithm Design Competition [7], aiming to develop cryptographic algorithms primarily for China's national cryptography standards. The CACR competition aims to identify and promote robust post-quantum asymmetric algorithms and efficient lightweight symmetric cryptographic algorithms. Consequently, it plays a pivotal role in shaping China's domestic infrastructure and security frameworks.

Following a two-year evaluation period, the Ballet block cipher family [10], was awarded first prize in the symmetric-algorithm category [8]. This block cipher is a simplified Lay-Massey ARX construction with a linear key schedule and XOR key injection, a unique design combination that makes it an interesting use case for cryptanalysis, beyond its original context. Despite its recognition, which poses Ballet as a central cryptographic primitive for the CACR, and original design, this block cipher has received limited attention from the broader research community. The official design documents and full specifications of the cipher were published exclusively in Chinese, a factor that has contributed to the cipher remaining understudied outside the CACR ecosystem.

This work provides a new analysis of Ballet and expands what is known about its security. In particular, we describe the first classical key recovery attacks in the literature, concrete trails for which the design document only provided bounds, and new distinguishers that have not been studied before.

The design document ([10, Section 6]) also highlights the high number of rounds in Ballet as a limitation that impacts performance. The authors made this conservative choice due to the difficulty of analyzing ARX constructions. They suggest that future analysis could tighten the bounds and potentially allow a reduction in the number of roundsOur work takes a step in that direction by providing explicit trails for more rounds than were published in the initial design document. In addition, we provide the first differential-linear cryptanalysis of Ballet, and use the resulting distinguishers to mount key recovery attacks on up to 16 rounds of Ballet-128/128/46, 17 rounds of Ballet-128/256/48, and 22 rounds of Ballet-256/256/74.

1.1 Related works

Since its introduction in 2019 [10], the Ballet family of block ciphers has received surprisingly little attention in international cryptography venues, despite winning the CACR competition for symmetric ciphers. To the best of our knowledge, only two papers have analyzed its security beyond the original design document.

The designers of Ballet provided bounds for differential, linear, impossible differential, zero correlation, and integral cryptanalysis. In particular, they reported that the probability of a 9-round differential trail does not exceed 2^{-43}, but did not give the corresponding trail; from this bound, they concluded that 26 rounds of Ballet-128/128 and 53 rounds for Ballet-256/256 were respectively sufficient for differential cryptanalysis resistance.

Similarly, the authors bounded the correlation of the best linear trail for 8 rounds to 2^{-19}, and to 2^{-23} for 9 rounds, and estimated the number of rounds needed for resistance to linear cryptanalysis to 26 and 56 rounds for Ballet-128/128 and Ballet-256/256.

In addition to these bounds, the authors found impossible differentials for up to 7 rounds, and zero-correlation linear approximations for 6 rounds. Finally, they identified 7-round integral distinguishers using the division property.

Besides this analysis, Mao et al. [15] analyzed Ballet in the related-cipher setting, and identified a key recovery using a single chosen plaintext pair and one encryption query, given access to a Ballet-128/128 oracle using the target key, and a Ballet-128/256 oracle using the same key, padded with zeros. For Ballet-128/256, similar attacks were shown for a weak key class.

Finally, in [19], Zhang et al. introduced an automatic SAT/SMT-based framework for searching rotational-XOR (RX) differential characteristics and applied it to Ballet. The authors obtained a 9-round RX differential characteristic for Ballet-128/256, with probability $2^{-43.47}$.

1.2 Our contribution

In this work, we modify the differential and linear trail search procedure (Section 3.3) of the CLAASP [1, 3] open-source framework, providing a 50–70% performance improvement across multiple ciphers. We also depict a time-based cutoff heuristic (Section 3.3.1), that reduces the reported trails search complexity from days to minutes. In addition, we propose several effective heuristic choices in the application of established cryptanalytic techniques, including avoiding the longest core trail in the splicing-and-extension approach for differential and linear trail search (Section 3.2), extending the Hamming weight of both input and output differences from 1 to 2 in impossible differential analysis (Section 3.6), and decomposing the cipher structure in the construction of differential-linear distinguishers (Section 3.7). We further present an ad hoc strategy for differential-linear key recovery

Table 1: Key recovery attacks on Ballet

Cryptanalysis	Ballet version	Number of rounds	Time complexity	Data complexity
differential-linear	128/128	16	$2^{104.46}$	$2^{72.46}$
differential-linear	128/128	16	$2^{124.26}$	$2^{117.60}$
differential-linear	128/256	17	$2^{192.00}$	$2^{111.46}$
differential-linear	256/256	22	$2^{242.06}$	$2^{178.06}$

(Section 4) that constrains selected bits of the output mask, thereby significantly reducing the complexity of the key-guessing phase. The optimized procedure and cutoff heuristic have been integrated into the open-source framework CLAASP, constituting a concrete contribution to the tool itself rather than a straightforward use of it. Although our attacks rely on known techniques, their effective application required substantial experimentation and careful tuning, both in trail search and in key-recovery phases.

Our cryptanalytic results are summarized in Table 1 for key recovery and in Table 2, Table 3 for distinguishers. In the latter case, results are given in the single-key setting unless specified otherwise. The notation $\leq 2^w$ indicates an upper bound on the probability of the distinguisher, while $\geq 2^w$ indicates a lower bound. Parameters for each technique are described in the corresponding sections in this manuscript (see Section 3). We also report the time and number of cores used to find each trail.

1.3 Outline

In Section 2, we briefly present the Ballet cipher, the encryption process, and the key schedule. In Section 3, the experimental setup is delineated, along with two general methodologies for extending linear and differential trail searches. Moreover, we present the distinguishers found for each type of cryptanalysis as a result of an intensive testing activity. In Section 4, we present key-recovery attacks on Ballet-128/128, Ballet-128/256 and Ballet-256/256. Finally, conclusions are provided in Section 5.

2 Preliminaries

2.1 The cipher

The Ballet block cipher is a lightweight, software-oriented ARX (Addition, Rotation, XOR) based construction. Ballet has three versions: Ballet-128/128/46, Ballet-128/256/48, and Ballet-256/256/74, where the notation denotes block length, key length, and number of rounds, respectively. The symbols used throughout the cipher specifications are as follows: $X_0^i \parallel X_1^i \parallel X_2^i \parallel X_3^i$ indicate the n-bit input at round i, $X_*^i \in \mathbb{F}_2^{n/4}$, $sk_i^L \parallel sk_i^R$ indicate the round key at round i, $sk_i^* \in \mathbb{F}_2^{n/4}$, $\lll$ and $\ggg$ are left and right rotation, $\boxplus$ and $\boxminus$ are the modular addition and subtraction, $\oplus$ is the XOR operation, and, in $x = x_{n-1}x_{n-2}...x_1x_0$, x_{n-1} is the most significant bit and x_0 is the least significant bit.

2.1.1 Encryption process

The cipher operates on n-bit blocks partitioned into four words (X_0, X_1, X_2, X_3) of $n/4$ bits each. At each round i, a pair of subkeys (sk_i^L, sk_i^R) of length $n/4$ bits each is injected.

[1] hw = Hamming weight

[2] This is only for the 128-128 case.

[3] The adopted notation indicates that, in a first step, four rounds of the linear part are fixed. These are then extended by appending the best seven-round trail that is compatible with the required linking conditions. Finally, five rounds corresponding to the differential part are searched.

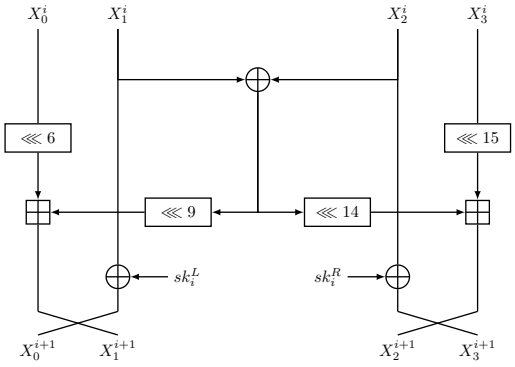
Table 2: Ballet-128/128 and Ballet-128/256: state-of-the-art distinguishers.

Cryptanalysis	**Rounds**	**Probability**	**Parameters**	**Cores**	**Time** (≤)	**Report trail**	**Reference**
differential distinguisher (see Table 7)	9	$\leq 2^{-43}$	NA	NA	NA	✗	[10]
	9	2^{-46}	NA	80	49 days	✓	this work
differential distinguisher via splicing	10	$\geq 2^{-63}$	8 fixed rounds	50	2 min	✓	this work
	11	$\geq 2^{-81}$		50	3 min	✓	this work
	12	$\geq 2^{-90}$		50	4 min	✓	this work
	13	$\geq 2^{-105}$		50	9 min	✓	this work
	14	$\geq 2^{-116}$		50	4 hours	✓	this work
differential distinguisher via optimized SAT and time-based cutoff heuristic	10	$\geq 2^{-57}$	time cutoff = 4d start = 127	8	4.6 days	✓	this work
	11	$\geq 2^{-64}$	time cutoff = 5d start = 127	8	5.3 days	✓	this work
	12	$\geq 2^{-75}$	time cutoff = 5d start = 127	8	5.5 days	✓	this work
	13	$\geq 2^{-84}$	time cutoff = 2d start = 127	8	5.4 days	✓	this work
	14	$\geq 2^{-91}$	time cutoff = 3d start = 108	30	12.6 days	✓	this work
	15	$\geq 2^{-105}$	time cutoff = 36h start = 123	50	5 days	✓	this work
	16	$\geq 2^{-127}$	time cutoff = 8d start = 127	50	9.9 days	✓	this work
linear distinguisher (correlation) (see Table 9)	9	2^{-23}	NA	NA	NA	✗	[10]
	9	2^{-23}	NA	100	7 hours	✓	this work
	10	2^{-26}	NA	80	4 days	✓	this work
	11	2^{-30}	NA	80	16 days	✓	this work
linear distinguisher via optimized SAT and time-based cutoff heuristic	10	$\geq 2^{-26}$	time cutoff = 2h start = 80	40	15 hours	✓	this work
	11	$\geq 2^{-30}$		40	3 days	✓	this work
	12	$\geq 2^{-36}$		40	2 days	✓	this work
	13	$\geq 2^{-43}$		40	2 days	✓	this work
	14	$\geq 2^{-49}$		40	6 days	✓	this work
	15	$\geq 2^{-56}$		40	10 days	✓	this work
	16	$\geq 2^{-63}$	time cutoff = 36h start = 68	50	4 days	✓	this work
differential-linear distinguisher (see Table 11)	9	2^{-12}	4-1-4	4	2 min	✓	this work
	10	2^{-16}	4-1-5	4	5 min	✓	this work
	11	2^{-22}	5-1-5	4	54 min	✓	this work
	12	2^{-30}	5-1-6	4	7 hours	✓	this work
	13	2^{-40}	6-1-6	50	2 hours	✓	this work
	14	2^{-52}	7-1-6 and 6-1-7	50	3 days	✓	this work
differential-linear distinguisher with splicing	15	$\geq 2^{-64}$	7-1-4+3	16	2 min	✓	this work
	16	$\geq 2^{-78}$	7-1-4+4	16	3 min	✓	this work
	17	$\geq 2^{-96}$	5+0-1-4+f7[3]	1	39 sec	✓	this work
	18	$\geq 2^{-106}$	6-1-4+7	32	5 hours	✓	this work
	19	$\geq 2^{-112}$	6-1-4+8	32	15 hours	✓	this work
	20	$\geq 2^{-124}$	7-1-4+8	16	14 hours	✓	this work
differential-linear distinguisher via optimized SAT and time-based cutoff heuristic	15	$\geq 2^{-61}$	time cutoff = 10d start = 63	4	14.4 days	✓	this work
	16	$\geq 2^{-75}$	time cutoff = 4d start = 77	20	4.4 days	✓	this work
	17	$\geq 2^{-84}$	time cutoff = 22d start = 87	1	30 days	✓	this work
	18	$\geq 2^{-94}$	time cutoff = 7d start = 98	40	11.2 days	✓	this work
	19	$\geq 2^{-106}$	time cutoff = 7d start = 110	40	10.9 days	✓	this work
	20	$\geq 2^{-118}$	time cutoff = 7d start = 120	100	15.9 days	✓	this work
impossible differential distinguisher(see Table 5)	7		hw[1] pt = 1, ct = 1	NA	NA	✗	[10]
	7		hw pt = 1, ct = 1	1	6 hours	✓	this work
	8		hw pt = 2, ct = 1	100	45 min	✓	this work
	7		hw pt = 1, ct = 2	100	45 min	✓	this work
	7		hw pt = 2, ct = 2	64	3 days	✓	this work
integral distinguisher	7		NA	NA	NA	✗	[10]
zero-correlation trails	6		hw 1	NA	NA	✓	[10]
XOR-Rotational (Related key)[2]	8	$2^{-46.64}$	NA	NA	NA	✓	[19]

Table 3: Ballet-256/256: state-of-the-art distinguishers.

Cryptanalysis	**Rounds**	**Probability**	**Parameters**	**Cores**	**Time** ($\leq$)	**Report trail**	**Reference**
differential distinguisher (see Table 8)	9	$\leq 2^{-43}$	NA	NA	NA	✗	[10]
	8	2^{-38}	NA	10	8 days	✓	this work
differential distinguisher via splicing	9	$\geq 2^{-53}$	8 fixed rounds	50	1 min	✓	this work
	10	$\geq 2^{-65}$		50	3 min	✓	this work
	11	$\geq 2^{-88}$		50	3 min	✓	this work
	12	$\geq 2^{-106}$		50	6 min	✓	this work
	13	$\geq 2^{-123}$		50	42 min	✓	this work
	14	$\geq 2^{-130}$		50	101 min	✓	this work
	15	$\geq 2^{-141}$		50	8 hours	✓	this work
	16	$\geq 2^{-153}$		50	4 days	✓	this work
differential distinguisher via optimized SAT and time-based cutoff heuristic	9	$\geq 2^{-48}$	time cutoff = 1d start = 52	32	1.1 days	✓	this work
	10	$\geq 2^{-58}$	time cutoff = 1d start = 64	32	1.3 days	✓	this work
	11	$\geq 2^{-66}$	time cutoff = 1d start = 72	32	1.2 days	✓	this work
	12	$\geq 2^{-79}$	time cutoff = 1d start = 85	32	1.5 days	✓	this work
	13	$\geq 2^{-89}$	time cutoff = 1d start = 95	32	2.6 days	✓	this work
	14	$\geq 2^{-102}$	time cutoff = 1.2d start = 105	32	2.5 days	✓	this work
linear distinguisher (correlation) (see Table 10)	9	$\leq 2^{-23}$	NA	NA	NA	✗	[10]
	9	2^{-23}	NA	100	2 days	✓	this work
linear distinguisher via optimized SAT and time-based cutoff heuristic	10	$\geq 2^{-26}$	time cutoff = 2h start = 30	32	2 hours	✓	this work
	11	$\geq 2^{-30}$	time cutoff = 2h start = 34	32	2.5 hours	✓	this work
	12	$\geq 2^{-36}$	time cutoff = 4h start = 38	32	9 hours	✓	this work
	13	$\geq 2^{-42}$	time cutoff = 4h start = 46	64	8 hours	✓	this work
	14	$\geq 2^{-48}$	time cutoff = 12h start = 52	36	1.3 days	✓	this work
	15	$\geq 2^{-54}$	time cutoff = 2d start = 56	48	4.2 days	✓	this work
	16	$\geq 2^{-63}$	time cutoff = 2d start = 63	48	4.4 days	✓	this work
differential-linear distinguisher (see Table 12)	9	2^{-14}	4-1-4	4	15 min	✓	this work
	10	2^{-18}	4-1-5	4	44 min	✓	this work
	11	2^{-22}	5-1-5	4	4 hours	✓	this work
	12	2^{-30}	5-1-6	16	2 hours	✓	this work
	13	2^{-39}	6-1-6	50	8 hours	✓	this work
differential-linear distinguisher via optimized SAT and time-based cutoff heuristic	14	2^{-46}	time cutoff = 2d start = 57	16	2.5 days	✓	this work
	15	$\geq 2^{-60}$	time cutoff = 1d start = 67	24	2 days	✓	this work
	16	$\geq 2^{-70}$	time cutoff = 1d start = 80	32	1.3 days	✓	this work
	17	$\geq 2^{-83}$	time cutoff = 1d start = 90	32	1.8 days	✓	this work
	18	$\geq 2^{-91}$	time cutoff = 1.5d start = 100	32	3.5 days	✓	this work
	19	$\geq 2^{-94}$	time cutoff = 2d start = 110	64	4.2 days	✓	this work
	20	$\geq 2^{-107}$	time cutoff = 5d start = 112	64	5.3 days	✓	this work
impossible differential distinguisher(see Table 5)	7		hw pt=1 ct=1	NA	NA	✗	[10]
	7		hw pt=1 ct=1	1	1 day	✓	this work
	8		hw pt=2 ct=1	100	11 hours	✓	this work
	7		hw pt=1 ct=2	100	12 hours	✓	this work
integral cryptanalysis	7		NA	NA	NA	✗	[10]
zero-correlation trails	6		hw 1	NA	NA	✓	[10]

Encryption applies r repetitions of the round function, depicted in Figure 1, followed by a final output transformation that omits the permutation step to maintain symmetry with decryption. The transformations are described in Algorithm 1.



Algorithm 1: Encryption process of the Ballet algorithm

for $i = 0$ **to** $r - 2$ **do**
  $X_0^{i+1} = X_1^i \oplus \mathrm{sk}_i^L$;
  $X_1^{i+1} = (X_0^i \lll 6) \boxplus [(X_1^i \oplus X_2^i) \lll 9]$;
  $X_2^{i+1} = (X_3^i \lll 15) \boxplus [(X_1^i \oplus X_2^i) \lll 14]$;
  $X_3^{i+1} = X_2^i \oplus \mathrm{sk}_i^R$;
end
$X_0^r = (X_0^{r-1} \lll 6) \boxplus [(X_1^{r-1} \oplus X_2^{r-1}) \lll 9]$;
$X_1^r = X_1^{r-1} \oplus \mathrm{sk}_{r-1}^L$;
$X_2^r = X_2^{r-1} \oplus \mathrm{sk}_{r-1}^R$;
$X_3^r = (X_3^{r-1} \lll 15) \boxplus [(X_1^{r-1} \oplus X_2^{r-1}) \lll 14]$;

Figure 1: Round function of Ballet

2.1.2 Key schedule

The Ballet family is defined in two configurations: Ballet-n/n and Ballet-$n/2n$.

- For Ballet-n/n (Algorithm 2), the master key is divided into two $n/2$-bit words (k_0, k_1). Across rounds, the subkeys (sk_i^L, sk_i^R) are drawn from k_0, while k_1 evolves through rotations and XORs with round counters, then swaps with k_0.

- For Ballet-$n/2n$ (Algorithm 3), the master key comprises four $n/4$-bit words (k_0, k_1, t_0, t_1). The schedule alternates k- and t-components through cyclic shifts and XOR operations, periodically swapping roles.

Algorithm 2: Key generation for Ballet n/n

$K = k_0 \parallel k_1$;
for $i = 0$ **to** $r - 1$ **do**
  output : $k_0 = sk_i^L \parallel sk_i^R$;
  $k_{temp} = k_1$;
  $k_1 = k_0 \oplus (k_1 \lll 3) \oplus (k_1 \lll 5) \oplus i$;
  $k_0 = k_{temp}$;
end

Algorithm 3: Key generation for Ballet $n/2n$

$K = k_0 \parallel k_1 \parallel t_0 \parallel t_1$;
for $i = 0$ **to** $r - 1$ **do**
  output : $k_0 = sk_i^L \parallel sk_i^R$;
  $t_{temp} = t_1$, $k_{temp} = k_1$;
  $t_1 = t_0 \oplus (t_1 \lll 7) \oplus (t_1 \lll 17)$,;
  $k_1 = k_0 \oplus (k_1 \lll 3) \oplus (k_1 \lll 5)$;
  $t_0 = t_{temp}$, $k_0 = k_{temp}$;
  $k_1 = k_1 \oplus t_1 \oplus i$;
end

3 Distinguishers

In this section, we first describe our experimental setup and two general techniques used to extend linear and differential trail searches. Then, for each type of cryptanalysis, we present the distinguisher we identified, beginning with a brief introduction to the technique employed, followed by the corresponding results.

3.1 Experimental setup

All experiments were conducted on a couple of Intel® Xeon® Gold 6258R CPU@2.7GHz, 56 Cores, 112 Threads, and 2Tb of RAM, using a variable number of cores, up to 100 for the most computationally intensive tasks. On the machine, we install a Debian-based Linux distribution as the operating system. All computations were performed using the latest development version of the CLAASP library [1, 3], together with a set of our own extensions to the library, described later in the paper.

The SAT solvers used in our experiments are ParKissat[4] and CryptoMiniSat[5] (versions 5.11.4 and 5.13.0). ParKissat proved more effective in multi-threaded scenarios, especially for the lowest-weight trail search. CryptoMiniSat was employed for searching for impossible differentials and for the new optimized SAT method with a time-based cutoff heuristic.

3.2 Splicing techniques

3.2.1 Splicing and extending Techniques

The *splicing* and the *extending* heuristics, initially described in [11] and [18], are commonly used to bound the weight of the best differential and linear trails in ARX ciphers.

The splicing technique exploits the fact that it is usually computationally easier to find trails for a small number of rounds than for many rounds. Starting from a fixed middle difference Δ_b, it builds two short optimal differential trails $\Delta_a \to \Delta_b$ and $\Delta_b \to \Delta_c$, which are then combined into $\Delta_a \to \Delta_c$. The extending heuristic, on the other hand, builds upon a known good trail by propagating it forward and/or backward. These heuristics were, for instance, used to find the (at the time) best differential trail for 19 rounds of Speck-128 in [11].

3.2.2 Application of these heuristics

We evaluated the extending technique by measuring its ability to approximate the best-known trails for 8 and 9 rounds. The experiments were conducted by varying the number of initial rounds and using several low-weight trails as starting points.

The results show that the best-weight trail is not always obtained by fixing the maximum possible number of rounds. Consequently, we performed the splicing search using the best trail identified by the classical method, fixing it for 7, 8, and 9 rounds, respectively.

We also examined whether starting from different trails with identical or comparable weights could lead to improved sliced results. However, no significant improvements were observed for the rounds of interest (≥ 10).

For these reasons, we restricted the splicing phase to one representative low-weight trail for each choice of fixed rounds. The best sliced trails were obtained when fixing 8 rounds. The outcomes of these experiments are presented in Table 2 and Table 3 and are further discussed later in this section in comparison with other methods and results.

3.3 Search procedure optimization

The search for optimal differential and linear trails for ARX-based ciphers is known to be difficult, due to the high non-linearity introduced by modular additions, and the difficulty of decomposing the problem into a word-based activity abstraction like the ones used for SPN ciphers, as in [12]. Preliminary experiments showed that SAT solvers, as opposed to MILP and CP, which are also available in the CLAASP library, performed better on these problems, probably due to the binary nature of the search problem.

[4] https://github.com/songfu1983/ParKissat-RS
[5] https://github.com/msoos/cryptominisat

The library's approach for finding the lowest-weight—e.g., differential—trail is to begin at weight zero and increment the weight each time the solver reports the problem as unsatisfiable, until a feasible solution is found. This procedure can easily be proven to return an optimal trail.

However, from a computational perspective, experimental evidence shows that it is generally easier for the solver to find a trail when one exists, often in large numbers, rather than to prove that a given weight results in an unsatisfiable instance. Building on this insight, we improved the library by initiating the search from the weight of a (fast) randomly obtained trail and proceeding downward from above the optimal weight, rather than searching upward from below.

The function searches trails with a weight lower than or equal to the previously found minus one. Doing so, the solver encounters low-cost satisfiable trails until it reaches the optimal weight. At this point, no lower weight trail exists. For this reason, searching for a trail with a lower weight results in an expensive UNSAT computation (compared to the previous SAT instances). This final step, however, certifies the optimality of the previously discovered trail.

Despite these improvements, the overall runtime is still dominated by the cost of the UNSAT search, whose complexity grows exponentially with the target weight. Nevertheless, this strategy yields a measured performance increase of 65% for Ballet-128/128 at 7 rounds, for both differential and linear lowest-weight trail search. In a fast benchmark, outside the scope of this work, we observed improvements ranging from 50% to 70% for the most expensive computations across all tested ciphers[6], again for both differential and linear cryptanalysis.

3.3.1 Time-based cutoff heuristic

To mitigate the limitations discussed in the previous paragraphs, we introduce the following time-based cutoff heuristic.

The search begins from a randomly generated or manually selected weight. The solver then attempts to reduce this weight within the given time limit. If a trail with a lower weight is found, the time counter is reset, and a new search begins for weights smaller than the one obtained in the previous step. If the time limit elapses without improvement, the procedure terminates.

We omit the UNSAT search step. Doing so, we lose the guarantee that the resulting trail is truly optimal. Instead, we gain the ability to evaluate significantly more rounds without incurring the exponential time increase required when moving from one cipher round to the next.

Given enough time, the search would eventually converge to the optimal trail. In our approach, we trade off the closeness to the actual solution and the computational resources used in the search.

This method allowed us to recompute differential and linear trails, the most expensive ones for the standard lowest-weight search function, orders of magnitude faster. Moreover, since the approach can explore trails across more rounds (and does so without fixing intermediate values to reduce search complexity), it produces results superior to the splicing technique, as shown in Table 2 and Table 3.

3.4 Differential trails

3.4.1 Differential cryptanalysis

Differential cryptanalysis [6] examines how differences in pairs of plaintexts propagate through a cipher to produce differences in the corresponding ciphertexts. This propagation

[6] Speck, AES, Ublock, Aradi, Kasumi, Present, Simon

Algorithm 4: Search for a lower-weight XOR linear trail with time-based cutoff heuristic

```
Function find_lower_weight_xor_linear_trail_with_time_cutoff(minutes, start)
    Function find_one_lower_weight_trail(max_weight)
        build_xor_linear_trail_model(weight = max_weight)
        solution ← solve(XOR_LINEAR) // or XOR_DIFFERENTIAL
        return solution
    end
    Function find_lower_weight_xor_linear_trail_within_timeout(start, timeout)
        result ← run_with_timeout(find_one_lower_weight_trail, timeout, max_weight = start)
        return result
    end
    // Search for a better starting weight; essential when the start is unspecified
    a_trail ← find_one_xor_linear_trail()                          // negligible time
    searched_weight ← min(a_trail[total_weight], start)
    // lower the limit until the search gets stuck
    solution ← find_lower_weight_xor_linear_trail_within_timeout( searched_weight, minutes )
    while solution ≠ None and solution[total_weight] ≠ None do
        actual_solution ← solution
        searched_weight ← actual_solution[total_weight] −1
        solution ← find_lower_weight_xor_linear_trail_within_timeout( searched_weight,
          minutes)
    end
    return actual_solution
end
```

is typically studied through the search for high-probability differential trails, which describe the differential transitions and their probabilities for each operation of the cipher.

3.4.2 Experimental results

The scripts used for this experimental phase rely on a SAT-based model implementing the techniques introduced by [11] for ARX ciphers, in which modular addition is encoded using a limited set of Boolean constraints derived from Lipmaa-Moriai's algorithm [14]. The scripts are available in the supplementary material[7].

Unlike the analysis in [10], which provides only a theoretical upper bound on the differential probability for 9 rounds, we identify an explicit optimal differential trail. Although the bound corresponds to a weight of 43, we show that no 9-round trail with weight lower than 46 exists. Consequently, the optimal differential trail has weight 46, indicating that the upper bound in [10] is not tight.

Compared to traditional search strategies, the introduction of the time-based cutoff heuristic substantially improved our ability to efficiently explore the trail space. Using this approach, we recomputed the 9-round trail in 1.36 hours on 4 cores (with a 20-minute time cutoff), compared to the 49 days required by the previous state-of-the-art exhaustive search on 80 cores.

Moreover, this method produced results that surpass those obtained with the splicing technique. As shown in Table 2, splicing reaches round 14 with weight 116 and cannot be pushed further: the 15-round extension (obtained in 7.5 days on 50 cores) has weight 128, and both trails are already optimal once the first eight rounds are fixed.
In contrast, our heuristic approach does not require fixing intermediate values to reduce the search space. With a 36-hour time cutoff on 50 cores, the solver reaches 15 rounds and produces a trail of weight 105 within five days. This result can be further improved: by initializing the search at weight 127 and running the same procedure on 50 cores with an 8-day time cutoff, we obtained a 16-round trail of weight 127 after two days. However, no lower-weight trail was found during the subsequent 8-day period, suggesting that

[7] https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/find_lowest_weight_xor_differential_trail_ballet.py and https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/findLowerDifferentialTrail.py.

the 16-round instance may be too large for the SAT solver to handle effectively without additional constraints on round outputs, as used in splicing techniques.

Applying the same strategy to the 256/256 case, and initializing the search at weight 105 with a 1.2-day cutoff, we obtained a 14-round trail of weight 102 after 2.5 days of computation. The time-cutoff approach yields better results than splicing for the reachable number of rounds; however, for larger round counts, splicing becomes necessary due to computational limitations, as it enables the search to be extended up to 16 rounds within practical time constraints. To reach additional rounds while maintaining a lower trail weight compared to classical splicing, the most effective strategy would be to splice the 14-round trail obtained via the time-cutoff heuristic by fixing some of its intermediate rounds (e.g., eight), and then reapply the optimized SAT solver with a time cutoff to extend the search further.

3.5 Linear trails

3.5.1 Linear Techniques

Linear cryptanalysis, originally introduced by Matsui [16], aims to derive a linear expression of the form

$$P[i_1, i_2, \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c]$$

where i, j and k denote fixed bit positions, and where the equation holds with a probability different from $1/2$ for a random plaintext and its corresponding ciphertext. The quantity $|p - 1/2|$ represents the bias of the linear relation.

Once such a linear expression has been identified, the key bits can be inferred through a two-step procedure.[1]

Step 1. Consider N plaintexts and let T denote the number of them for which the left-hand side of the linear equation evaluates to 0.

Step 2. If $T > N/2$, we guess

$$K[k_1, k_2, \ldots, k_r] = \begin{cases} 0 & \text{if } p > \frac{1}{2}, \\ 1 & \text{if } p < \frac{1}{2}, \end{cases}$$

otherwise we guess

$$K[k_1, k_2, \ldots, k_r] = \begin{cases} 1 & \text{if } p > \frac{1}{2}, \\ 0 & \text{if } p < \frac{1}{2}. \end{cases}$$

The success rate of the method increases with the number of plaintext samples N and with the magnitude of the bias $|p - 1/2|$. The approximation achieving the maximum bias—i.e., the largest value of $|p - 1/2|$—is referred to as the *best* (or *optimal*) linear approximation, and its associated probability p is correspondingly termed the *best probability*.

3.5.2 Experimental results

Similar to Section 3.4, the scripts used for this experimental part apply the techniques introduced by [11] to model linear propagations in ARX constructions. The scripts are available in the supplementary material[9].

[8] This is Matsui's Algorithm 1

[9] https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/find_lowest_weight_xor_linear_trail_ballet.py

For the Ballet-128/128 case in [10], authors reached a 2^{-23} probability for any 9-round linear trail. In our experiments, we independently confirmed this result, specifically obtaining a linear correlation weight of 23 for the 9-round case within approximately 0.3 days of computation. Our method produced an explicit trail achieving this weight. Moreover we reached the 11th round with weight 30.

Similarly, for Ballet-256/256, we match the upper bound of 23 given in [10] and provide the corresponding trail with weight 23.

By applying the optimized SAT-based search with the time-based cutoff heuristic described in Section 3.3 to Ballet-128/128, we were able to extend the analysis up to 16 rounds. Our initial configuration, starting from a weight of 80, using a 2-hour time cutoff and 40 cores, reached round 16 in 7.2 hours, yielding a trail of correlation weight 69. As this correlation remained too high for practical relevance, we repeated the search with a lower starting weight. Beginning from a weight of 68, using 50 cores and a 36-hour time cutoff, we obtained a trail of weight 63 within 3 days, marking a useful improvement for linear cryptanalysis. Applying the same strategy to the 256/256 case, starting from a weight of 63, and employing 48 cores with a two-day time cut-off, we derived a 16-round trail of weight 63 after 4.4 days of computation.

3.6 Impossible differentials

3.6.1 Impossible Differential techniques

Impossible differential cryptanalysis, introduced by Biham, Biryukov, and Shamir [5], exploits differential transitions that are structurally impossible for a given cipher. An impossible differential consists of an input–output difference pair whose propagation has probability zero, independently of the secret key.

3.6.2 Experimental results

The scripts used for the experimental phase follow the approach introduced independently by Sasaki and Todo [17] and by Cui et al. [9]. The input and output differences are fixed in an SAT-based differential model, and the solver checks whether any valid differential trail exists. If the model produces no feasible solution, the corresponding differential is classified as impossible. The scripts are available in the supplemetary material[10].

In [10], authors have already shown that no impossible differentials beyond 7 rounds can be found when restricting the plaintext and ciphertext differences to a Hamming weight of 1. Therefore, we extended our analysis by exploring whether impossible differentials could be identified when relaxing this constraint and varying the Hamming weights of the input and output differences.

To this end, we performed a series of experiments in which the plaintext difference was set to have a Hamming weight of 2, while the ciphertext difference was fixed at a Hamming weight of 1. Our results show that, under these conditions, no impossible differentials can be found beyond 8 rounds.

We also examined additional Hamming-weight pairings: a plaintext difference of 1 with a ciphertext difference of 2, and a plaintext difference of 2 with a ciphertext difference of 2. In both cases, the results were consistent with the previous findings, showing that impossible differentials do not extend beyond 7 rounds.

As the search space grows significantly with increasing Hamming weight, the computational cost becomes substantial. To make the analysis feasible, we parallelized the code and distributed the workload across multiple processing units. This optimization allowed

[10] https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/impossibleDifferentialHammingweightpt2ct1_Ballet.py

us to complete the search in a much shorter time and systematically confirm the absence of impossible differentials up to Hamming weight 2 beyond the 8-round boundary.

3.7 Differential-linear trails

3.7.1 Differential-linear Techniques

Differential-linear cryptanalysis was introduced by Langford and Hellman [13]. The core idea of this attack is to decompose the cipher into three distinct components: a differential component ($F_1(x)$), a linear component ($F_2(x)$), and an intermediate component ($F_m(x)$) that connects and facilitates the interaction between the former two.
The total correlation can be estimated as:

$$\mathrm{Cor}_{x\in\mathbb{F}_2^n}[\langle\Gamma_{\mathrm{out}}, F(x)\rangle \oplus \langle\Gamma_{\mathrm{out}}, F(x\oplus\Delta_{\mathrm{in}})\rangle] = prq^2$$

where:

- $p = \Pr_{x\in\mathbb{F}_2^n}[F_1(x)\oplus F_1(x\oplus\Delta_{\mathrm{in}}) = \Delta_m]$ denotes the probability that the input difference Δ_{in} propagates through the first part F_1 of the cipher to the intermediate difference Δ_m;
- $q = \mathrm{Cor}_{x\in\mathbb{F}_2^n}[\langle\Gamma_m, x\rangle \oplus \langle\Gamma_{\mathrm{out}}, F_2(x)\rangle]$ denotes the correlation of the linear trail from the intermediate mask Γ_m to the output mask Γ_{out} through the second part F_2;
- $r = \mathrm{Cor}_S[\langle\Gamma_m, F_m(x)\rangle \oplus \langle\Gamma_m, F_m(x\oplus\Delta_m)\rangle]$ represents the correlation through the middle layer F_m, estimated experimentally over a set of samples S.

3.7.2 Experimental results

The scripts implementing the following experimental phase are available in the supplementary material[11].

The state of the art for Ballet differential cryptanalysis reaches up to 9 rounds with a weight of 43. Extending the analysis beyond 9 rounds using purely differential techniques would require a prohibitively large amount of computation. Consequently, in our work, we did not build upon previous differential results; instead, we directly initiated the analysis from the 9-round case using the differential-linear approach.

We performed our search using the SAT model implemented in CLAASP, which follows the techniques described in [4] and decomposes the cipher into three distinct subparts:

- A top part encompassing r_t rounds, analyzed through differential propagations,
- A middle part covering r_m rounds, analyzed via bitwise deterministic truncated differential propagations,
- A bottom part spanning r_b rounds, analyzed using linear propagations.

The reported weight is calculated by summing the total weight of the differential component with twice the total weight of the linear component. We focused on single-trail probabilities and did not measure clustering effects. We explored all combinations r_t-r_m-r_b for 9 rounds. After a testing phase, the combinations 5-1-3, 3-1-5, and 4-1-4 yielded the best weight, equal to 12 for the 128-128 case and 14 for the 256-256 case, which is considerably lower than 43, allowing for improved key recovery. To validate the weight of the trail 4-1-4, we heuristically verified it for the given difference-mask pair with a significant number of samples. This confirms that the differential-linear behavior of the trail is consistent with

[11] https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/find_lowest_differential-linear.py

the expected characteristics and supports the soundness of our analysis. Since the 9-round version took a negligible amount of time, the research can be further extended to reach the 14-round version, which has a weight of 52 for the 128-128 case, and weight 46 for the 256-256 case.

Although the number of rounds in the intermediate section is not fixed a priori, experimental results show that the best trails consistently feature a middle section covering a single round. This can be explained by the fact that extending the intermediate section beyond one round introduces an excessive number of unknowns in the state, drastically reducing the available choices for the input mask of the linear part. A closer examination of the results further reveals that the optimal combinations are those in which the differential and linear parts are well balanced, with the intermediate section always limited to one round. The intermediate values of the trail are then used as a starting point to experimentally evaluate larger DL middle sections. For the intermediate round, we rely on the well-established observation that no impossible differentials exist beyond 8 rounds. This implies that the deterministic portion of the differential-linear trail is necessarily limited to a maximum of 4 rounds, ensuring consistency with the underlying structural constraints of the cipher. We introduce a parameter defining the number of unknown variables that remain unconstrained in the differential-linear attack model. It defaults to 1, capturing the minimal realistic case where at least one internal bit/value is not fully determined by the model's equations. If set to k, the attacker can consider an extra search space of 2^k possible assignments, decreasing attack complexity. We observed that increasing this number generally leads to a decrease in the resulting weight, with the optimal value corresponding to the block dimension minus one. The time of the computation does not change when we add it; instead, in some cases, it is even faster.

3.7.3 Differential-Linear trails: splicing and time-based cutoff heuristic.

The scripts for this experimental phase are provided in the supplementary material[12].

Because the search for differential-linear trails becomes computationally expensive from 15 rounds onward in the 128/128 version, the splicing technique was adopted. Its effectiveness was first validated on the best known 14-round trail using the 6-1-7 and 7-1-6 decompositions.

For 15 rounds, the best configuration was 7-1-4+3, where the full differential part is fixed, while only part of the linear trail is fixed and the remaining rounds are searched. The same strategy proved optimal for rounds 16, 18, 19, and 20, consistently fixing the entire differential part and only partially fixing the linear part.

Round 17 is the only exception, as it was obtained by fixing the linear part of the previous 6-1-4+7 decomposition and then searching for the best compatible 5-round differential trail.

Overall, the proposed slicing approach extends the search up to 20 rounds, yielding a 7-1-4+8 trail with weight 124. However, this strategy is less efficient than the time-cutoff approach, which consistently produces lower-weight trails at each round (see Table 2), due to the larger solution space induced by not fixing intermediate portions of the trail. Motivated by these observations, we applied the latter method to both parameter settings in order to further improve the results and extend the search to higher numbers of rounds. This allowed us to reach round 20 for the 256/256 version with a trail of weight 107, and round 20 for the 128/128 version with weight 118.

[12] https://github.com/Crypto-TII/claasping_ballet/blob/main/Ballet/scripts/find_lowest_differential-linear_splicing.py

4 Key Recovery Attacks

In this section, we present key-recovery attacks on Ballet-128/128, Ballet-128/256 and Ballet-256/256 that use the differential–linear distinguishers presented in Section 3.7. A summary of the key recovery results can be found in Table 1.

Given a DL distinguisher covering n_r rounds, we can turn it into a $(n_r + 2)$-round key-recovery attack by prepending one round at the beginning and appending one round at the end.

The initial round can be added at no extra cost because the key is injected at the end of the round. Indeed, let us suppose that we have a pair of states $X^1, X^{1'}$ after one round of encryption such that $X^1 \oplus X^{1'} = \Delta$ and let F_K denote the Ballet round function under the key K, in case the key needs to be specified. Then the equality $F_K(F_0^{-1}(X^1)) \oplus F_K(F_0^{-1}(X^{1'})) = \Delta$ holds for any possible key K.

This follows from the following relations:

$$\begin{aligned}
F_0^{-1}(X^1)_1 &= X_0^1 \\
F_0^{-1}(X^1)_2 &= X_3^1 \\
F_0^{-1}(X^1)_0 &= (X_1^1 \boxminus ((X_0^1 \oplus X_3^1) \lll 9)) \lll 26 \\
F_0^{-1}(X^1)_3 &= (X_2^1 \boxminus ((X_0^1 \oplus X_3^1) \lll 14)) \lll 17
\end{aligned}$$

which imply

$$\begin{aligned}
F_K(F_0^{-1}(X^1))_1 &= X_1^1 \\
F_K(F_0^{-1}(X^1))_2 &= X_2^1 \\
F_K(F_0^{-1}(X^1))_0 &= X_0^1 \oplus K \\
F_K(F_0^{-1}(X^1))_3 &= X_2^1 \oplus K
\end{aligned}$$

Analogous relations can be deduced for $X^{1'}$, from which we can derive that the difference is invariant.

Appending one additional round to the output of the distinguisher allows us to recover 32 bits of information about the key for Ballet-128/128 and Ballet-128/256. Indeed, from the update rules of the last round

$$\begin{aligned}
X_0^{n_r+2} &= (X_0^{n_r+1} \lll 6) \boxplus ((X_1^{n_r+1} \oplus X_2^{n_r+1}) \lll 9), \\
X_1^{n_r+2} &= X_1^{n_r+1} \oplus sk_{n_r+1}^L, \\
X_2^{n_r+2} &= X_2^{n_r+1} \oplus sk_{n_r+1}^R, \\
X_3^{n_r+2} &= (X_3^{n_r+1} \lll 15) \boxplus ((X_1^{n_r+1} \oplus X_2^{n_r+1}) \lll 14),
\end{aligned}$$

we obtain the corresponding inverse relations:

$$X_1^{n_r+1} = X_1^{n_r+2} \oplus sk_{n_r+1}^L \tag{1}$$

$$X_2^{n_r+1} = X_2^{n_r+2} \oplus sk_{n_r+1}^R \tag{2}$$

$$X_0^{n_r+1} = \left(X_0^{n_r+2} \boxminus ((X_1^{n_r+2} \oplus sk_{n_r+1}^L \oplus X_2^{n_r+2} \oplus sk_{n_r+1}^R) \lll 9)\right) \lll 26 \tag{3}$$

$$X_3^{n_r+1} = \left(X_3^{n_r+2} \boxminus ((X_1^{n_r+2} \oplus sk_{n_r+1}^L \oplus X_2^{n_r+2} \oplus sk_{n_r+1}^R) \lll 14)\right) \lll 17. \tag{4}$$

Two observations about the influence of the round key follow immediately:

1. the computations of $X_0^{n_r+1}$ and $X_3^{n_r+1}$ only depend on $sk_{n_r+1}^L \oplus sk_{n_r+1}^R$, through the modular substraction (Equation 3, Equation 4);
2. the computations of $X_1^{n_r+1}$ and $X_2^{n_r+1}$ depend linearly on a single key word each, namely $sk_{n_r+1}^L$ (Equation 1) and $sk_{n_r+1}^R$ (Equation 2), respectively .

This implies that the key pairs $(sk^L_{n_r+1} \| sk^R_{n_r+1})$ $(sk^R_{n_r+1} \| sk^L_{n_r+1})$ induce identical correlations for the linear approximation on $(X_0^{n_r+1} \| X_1^{n_r+1} \| X_2^{n_r+1} \| X_3^{n_r+1})$. More generally, any key pair $(sk'^L_{n_r+1} \| sk'^R_{n_r+1})$ satisfying

$$sk'^L_{n_r+1} \oplus sk'^R_{n_r+1} = sk^L_{n_r+1} \oplus sk^R_{n_r+1}$$

produces the same correlation. Thus, the linear approximation depends only on the XOR of the two key words rather than their individual values. As a result, the key search can be restricted to the 32-bit space of $sk^L_{n_r+1} \oplus sk^R_{n_r+1}$, after which the remaining 32 bits can be recovered by exhaustive search. A similar argument applies to Ballet-256/256, where the attack recovers the 64-bit XOR of the two key halves.

4.1 Differential-linear attack on 16-round Ballet-128/128

In this section, we examine the best 14-round differential–linear distinguisher produced by our tool and listed in Table 2 and Table 11. It achieves a theoretical correlation of 2^{-52} for the input difference Δ_{in} and the output mask Γ_{out} with

$$\Delta_{in} = \texttt{0x00011088000020100000020180041108}$$
$$\Gamma_{out} = \texttt{0x991002193004032000006060 9f103100}.$$

The trail decomposes as follows:

- The differential part covers 6 rounds and holds with probability 2^{-20}.
- The output difference at round 5 propagates deterministically for 1 round.
- The linear part starts at round 6, spans 7 rounds, and has correlation 2^{-16}.

To refine the correlation estimate, we partition the cipher into three subciphers E_1, E_m, and E_2, following the standard approach: E_1 corresponds to the top differential section, E_2 to the bottom linear section, and E_m is the differential–linear middle section, whose correlation is measured experimentally using 2^{37} samples. We evaluated multiple configurations for these sections and found that the highest overall correlation is achieved by assigning 2 rounds to the top, 10 rounds to the middle, and 2 rounds to the bottom:

$$\Delta_{in} \underbrace{\xrightarrow[p=2^{-14}]{\text{2 rounds}}}_{E_1} \Delta_1 \underbrace{\xrightarrow[r=2^{-9.23}]{\text{10 rounds}}}_{E_m} \Gamma_{11} \underbrace{\xrightarrow[q=2^{-10}]{\text{2 rounds}}}_{E_2} \Gamma_{out}$$

with

$$\Delta_1 = \texttt{0x00000400000000000000000000000002}$$
$$\Gamma_{11} = \texttt{0x4400000000000000c0000180044000c00}.$$

This yields an estimated overall distinguisher correlation of approximately $2^{-43.23}$. We present it below in Differential-Linear Distinguisher 1.

Differential-Linear Distinguisher 1. *The following 14-round DL distinguisher holds with a correlation of* $2^{-43.23}$*:*

$$\Delta_{in} = \texttt{0x00011088000020100000020180041108} \xrightarrow{14r} \texttt{0x991002193004032000006060 9f103100} = \Gamma_{out}.$$

A naive attack on 16-round Ballet-128/128 using Differential-Linear Distinguisher 1 would result in a data complexity of $\varepsilon \cdot 2^{87.46}$ and a time complexity of $\varepsilon \cdot 2^{119.46}$ to recover the 32-bit XOR of the two round key halves.

However, *Probabilistic Neutral Bits* may help decrease the overall complexity of differential-linear attacks, as previously investigated in [2]. The idea here is to generate enough good pairs to observe the bias of the linear part, at a much lower cost than p^{-1}, given one pair satisfying the differential.

Definition 1 (Probabilistic Neutral Bit)**.** Let $E : \{0,1\}^n \to \{0,1\}^n$ be a permutation and $(\Delta_{\text{in}} \to \Delta_{\text{out}})$ be a differential. Let $\mathcal{R}$ denote the set of all *comforming pairs* for this differential:

$$\mathcal{R} = \{(P, P') \in (\{0,1\}^n)^2 : P \oplus P' = \Delta_{\text{in}} \text{ and } E(P) \oplus E(P') = \Delta_{\text{out}}\}.$$

A subset of bit indices $\mathcal{S} \subseteq \{0, \ldots, n-1\}$ is a set of p_1-*probabilistic neutral bits* if, for any single bit index $i \in \mathcal{S}$, flipping that bit in a conforming pair results in a new pair that remains in $\mathcal{R}$ with probability p_1. That is, for each $i \in \mathcal{S}$:

$$\Pr_{(P,P') \in \mathcal{R}} \Big[E(P \oplus e_i) \oplus E(P' \oplus e_i) = \Delta_{\text{out}} \Big] = p_1,$$

where e_i denotes the denote the canonical basis of $\mathbb{F}_2^n$. When $p_1 = 1$, the bits in $\mathcal{S}$ are called *deterministic neutral bits*.

If the differential part of the DL distinguisher has l deterministic neutral bits, the data complexity then becomes

$$\varepsilon p^{-1} r^{-2} q^{-4},$$

as long as $2^l > r^{-2}q^{-4}$. When that number is insufficient, one may extend the set with probabilistic neutral bits to decrease the complexity. Given a set of l p_1-PNBs, such that $2^l > r^{-2}q^{-4}$ and $p_1 \gg p$, the probability of obtaining a conforming pair is pp_1 and the data complexity of the improved DL-distinguisher becomes

$$\varepsilon p^{-1} p_1^{-2} r^{-2} q^{-4}.$$

For Differential-Linear Distinguisher 1, in order to reliably distinguish $\Delta_1 \xrightarrow{12r} \Gamma_{out}$, we need structures of size $2^l > r^{-2}q^{-4} = 2^{2 \cdot 9.23 + 4 \cdot 10} = 2^{58.46}$, and we therefore require $l = 59$ neutral bits. Using 2^{30} samples, we found a set $\mathcal{S}$ containing 74 deterministic neutral bits ($p_1 = 1$) for the differential part:

$\{0, 1, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 29, 30, 37, 38, 39$
$40, 41, 48, 50, 51, 52, 54, 55, 58, 59, 61, 62, 63, 69, 70, 71, 72, 73, 80, 82, 83, 84, 86, 87, 90, 91$
$93, 94, 95, 100, 101, 102, 104, 113, 114, 115, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127\}$

With a sufficient number of PNBs, we can proceed with the attack as follows.

1. Generate a base pair of plaintexts such that their difference after the first round equals Δ_{in}, by decrypting each element using the all-0 subkey.
2. Compute the set of 2^{59} plaintexts $\mathcal{P}$ using the set of PNBs and request the corresponding set of ciphertexts pairs $\mathcal{C}$, generated after 15 rounds.
3. Initialize a 2^{32} counter to zero. For each pair of $\mathcal{C}$, try all the 2^{32} values for $sk_{15}^L \oplus sk_{15}^R$ and partially decrypt each ciphertext over 1 round, and compute the parity of the resulting difference using the output mask of Differential-Linear Distinguisher 1. This distinguisher should hold with a correlation of $2^{-43.23}$. If the value equals 0, increase the counter.
4. Sort the counter by correlation and if the first value of the list is consistent with the expected correlation, output it as $sk_{15}^L \oplus sk_{15}^R$; otherwise, return to step 1.

The number of input–output pairs required by an attacker is $\varepsilon\, p^{-1} r^{-2} q^{-4} = \varepsilon \cdot 2^{-72.46}$, which yields a data complexity of $\varepsilon \cdot 2^{-73.46}$. Recovering the 32-bit XOR value of the two final round key halves then results in a time complexity of $\varepsilon \cdot 2^{-105.46}$.

4.2 Differential-linear attack on 16-round Ballet-128/128 using a carefully chosen output mask

In the attack described in the previous section, all the bits of the value $sk^L_{n_r+1} \oplus sk^R_{n_r+1}$ need to be guessed in order to check the final linear relation. In this section, we describe another attack that utilizes a carefully chosen distinguisher, allowing only some of the last subkey bits to be guessed, thereby reducing the corresponding part of the time complexity.

From the equations induced by the last round of encryption (Equation 1, Equation 2, Equation 3, Equation 4), two properties can immediately be observed, assuming the ciphertext X^{n_r+2} is known:

1. Any bit i of $X_1^{n_r+1}$ or $X_2^{n_r+1}$ is affected by exactly one bit of the final subkey, in fact $(X_1^{n_r+1})_i = (X_1^{n_r+2})_i \oplus (sk^L_{n_r+1})_i$ and $(X_2^{n_r+1})_i = (X_2^{n_r+2})_i \oplus (sk^R_{n_r+1})_i$. If said bit is not involved in the computation of the other bits involved in the linear relation, then it is possible to use $(X^{n_r+2})_i$ instead of $(X^{n_r+1})_i$ for checking the correlation, and no key guess is necessary. In fact in the relation only the XOR of the key bits involved, which is a constant, would appear and would not impact the correlation.

2. Computing any bit i of $X_0^{n_r+1}$ or $X_3^{n_r+1}$ requires exactly $(6-i) \pmod{32}$ bits of the XOR of the final subkeys. In fact, by the properties of the modular addition, we have that

$$\begin{aligned}(X_0^{n_r+2})_{31} &= (X_0^{n_r+1} \lll 6)_{31} \oplus ((X_1^{n_r+1} \oplus X_2^{n_r+1}) \lll 9)_{31} \\ &= (X_0^{n_r+1})_5 \oplus (X_1^{n_r+1} \oplus X_2^{n_r+1})_8 \\ &= (X_0^{n_r+1})_5 \oplus (X_1^{n_r+2})_8 \oplus (sk^L_{n_r+1})_8 \oplus (X_2^{n_r+2})_8 \oplus (sk^R_{n_r+1})_8\end{aligned}$$

   and therefore

$$(X_0^{n_r+1})_5 = (X_0^{n_r+2})_{31} \oplus (X_1^{n_r+2})_8 \oplus (sk^L_{n_r+1})_8 \oplus (X_2^{n_r+2})_8 \oplus (sk^R_{n_r+1})_8$$

   which means that to retrieve bit 5 of X_0^{15} we need to guess one bit of the XOR of the subkeys, because it is the bit involved in the computation of the least significant bit of the modular addition, which is equivalent to an XOR. For $(X_0^{15})_4$, that corresponds to the second least significant bit of the modular addition, we will also need $(X_0^{15})_5$, which, by the same reasoning, implies 2 total subkeys XOR bits, and so on.

Let us now consider the 14-round differential linear distinguisher, spanning rounds 2 through 15,

$$\Delta = \texttt{0x0140242200088c400020884420040202} \xrightarrow{14r} \texttt{0x0800000000000000000020100040000} = \Gamma$$

This distinguisher has been obtained by our tool from a trail with theoretical correlation of 2^{-66}, which decomposes as follows:

- The differential part covers 7 rounds and holds with probability $p = 2^{-32}$.
- The output difference at round 6 propagates deterministically for 1 round.
- The linear part starts at round 7, spans 6 rounds, and has correlation $q = 2^{-17}$.

Again, we can refine the correlation estimate as before, partitioning the cipher as:

$$\Delta \underbrace{\xrightarrow[p=2^{-11}]{1r}}_{E_n} \Delta_1 \underbrace{\xrightarrow[r=2^{-17}]{4r}}_{E_1} \Delta_5 \underbrace{\xrightarrow[p=2^{-11.72}]{7r}}_{E_m} \Gamma_{12} \underbrace{\xrightarrow[q=2^{-10}]{2r}}_{E_2} \Gamma$$

with

$$\Delta_1 = \texttt{0x00011188000020100000020180041108}$$
$$\Delta_5 = \texttt{0x00010000000040000080000000000010000}$$
$$\Gamma_{12} = \texttt{0x00101800000200000400000000001800}$$

This yields an estimated overall distinguisher correlation of approximately $2^{-59.72}$.

Moreover, we can exploit the neutral bits as in the previous section, on the portion E_n of the distinguisher, meaning that we need a total of approximately $(17+11.72+20)\cdot 2 = 97.44$ neutral bits to make the technique effective. We found 59 neutral bits for the first round of encryption:

$$\{0, 1, 6, 8, 9, 10, 12, 14, 15, 16, 17, 19, 20, 21, 22, 23, 31, 41, 48, 49, 50, 51, 52,$$
$$53, 54, 55, 61, 62, 73, 80, 81, 82, 83, 84, 85, 86, 87, 93, 94, 98, 99, 100, 101, 103,$$
$$104, 112, 113, 114, 115, 116, 117, 118, 120, 121, 122, 123, 124, 125, 127\}$$

and another 39 probabilistic neutral bits with a cumulative probability of more than $2^{-4.08}$:

$$\{2, 3, 4, 7, 24, 25, 27, 28, 29, 32, 36, 37, 38, 43, 44, 45, 46, 47, 56, 57, 58,$$
$$64, 68, 69, 70, 75, 76, 77, 78, 79, 88, 89, 96, 106, 107, 108, 109, 110, 111\}$$

Finally, we can analyze the key recovery. Since the output mask involves only bits $(X_0^{15})_4, (X_2^{15})_{22}, (X_2^{15})_{31}, (X_3^{15})_{13}$, of which $(X_2^{15})_{22}, (X_2^{15})_{31}$ do not require key guesses. By the observations we made before, we can say:

1. $(X_0^{15})_4$ can be deduced by $(sk_{15}^L)_7 \oplus (sk_{15}^R)_7$ and $(sk_{15}^L)_8 \oplus (sk_{15}^R)_8$.
2. $(X_3^{15})_{13}$ can be deduced by $(sk_{15}^L)_{12} \oplus (sk_{15}^R)_{12}$ and $(sk_{15}^L)_{13} \oplus (sk_{15}^R)_{13}$.

The attack proceeds as follows:

1. Generate a pair of values with difference Δ and decrypt them for one round under the 0 subkey, getting a pair of plaintexts such that we are sure that after one round of encryption under the correct key, they will all have difference Δ.
2. Perform the 16 rounds encryption of all the pairs generated by the first one by flipping all the possible subsets of the PNBs under the correct key enough time to obtain $2^{97.44}$ pairs of ciphertexts.
3. For all the possible values of bits $(sk_{15}^L)_7 \oplus (sk_{15}^R)_7$, $(sk_{15}^L)_8 \oplus (sk_{15}^R)_8$, $(sk_{15}^L)_{12} \oplus (sk_{15}^R)_{12}$ and $(sk_{15}^L)_{13} \oplus (sk_{15}^R)_{13}$, decrypt each ciphertext pair for one round to get the bits involved in the output mask and measure the correlation of the linear relation. If there is a guess for which the expected correlation holds, it will correspond to the correct subkey; otherwise, return to step 1.
4. Exhaustively try all the possible master keys for which the 4 key bit conditions retrieved hold, which will be $\frac{2^{128}}{2^4} = 2^{124}$.

Using the notation of the previous section, for this distinguisher, we have

$$p_1 \geq 2^{-4.08},\ p = 2^{-11},\ r \approx 2^{-17} \cdot 2^{-11.72} \approx 2^{-28.72},\ q = 2^{-10},\ kb = 4$$

The data complexity of the attack is

$$D^* \approx p_1^{-2} p^{-1} r^{-2} q^{-4} \times 2 \approx 2^{117.60}$$

The time complexity is the sum of the subkey deduction processes' time complexities, that is $p_1^{-2} p^{-1} r^{-2} q^{-4} \times 2 \times 2^{kb}$ and the final exhaustive search on the key, that is 2^{128-kb}. The final time complexity would then be

$$T^* \approx 2^{117.60} \times 2^4 + 2^{124} \approx 2^{124.26}$$

4.3 Differential-linear attack on Ballet-128/256 and Ballet-256/256

Following the approach of the previous sections, we prepend one round at the beginning and append another at the end of Differential-Linear Distinguisher 2 to obtain a 17-round key-recovery attack against Ballet-128/256. The trail, listed in Table 11, achieves a theoretical correlation of 2^{-61} for the input difference Δ_{in} and the output mask Γ_{out} with

$$\Delta_{in} = \texttt{0x2000002040400004440000000002201002}$$
$$\Gamma_{out} = \texttt{0x0004000000000001000000218004018}0,$$

and decomposes as follows:

- The differential part covers 7 rounds and holds with probability $p = 2^{-27}$.
- The output difference at round 6 propagates deterministically for 1 round.
- The linear part starts at round 7, spans 7 rounds, with correlation $q = 2^{-17}$.

As done in Section 4.1, we instead split the cipher into three subciphers, allowing us to represent the DL distinguisher as

$$\Delta_{in} \underbrace{\xrightarrow[p=2^{-13}]{3 \text{ rounds}}}_{E_1} \Delta_2 \underbrace{\xrightarrow[r]{8 \text{ rounds}}}_{E_m} \Gamma_{10} \underbrace{\xrightarrow[q=2^{-15}]{4 \text{ rounds}}}_{E_2} \Gamma_{out},$$

where

$$\Delta_2 = \texttt{0x10000000000000000000000000000000}$$
$$\Gamma_{10} = \texttt{0x00000400000c0000000000000000400}0.$$

The middle part $\Delta_2 \xrightarrow{8r} \Gamma_{10}$ was evaluated empirically using 2^{39} samples. The measured value, $r = 2^{-5.11}$, is significantly higher than the expected 2^{-18}, giving an overall correlation $prq^2 = 2^{-48.11}$ for the distinguisher.

Differential-Linear Distinguisher 2. *The following 15-round DL distinguisher holds with a correlation of* $2^{-48.11}$*:*

$$\Delta_{in} = \texttt{0x2000002040400004440000000002201002} \xrightarrow{15r}$$
$$\texttt{0x0004000000000001000000218004018}0 = \Gamma_{out}.$$

The number of input–output pairs required by an attacker is $\varepsilon\, p^{-2} r^{-2} q^{-4} = \varepsilon \cdot 2^{-96.22}$, which yields a data complexity of $\varepsilon \cdot 2^{-96.22}$. Recovering the 32-bit XOR value of the two final round key halves then results in a time complexity of $\varepsilon \cdot 2^{-128.22}$. The time complexity of an attack recovering the full key would be dominated by the cost of the exhaustive search on the remaining 192 bits.

For Ballet-256/256, the 20-round trail of Table 12 gives a DL distinguisher with correlation $2^{-89.03}$ (using 2^{38} samples for the middle section):

$$\Delta_{in} \xrightarrow[p=2^{-17}]{5r} \Delta_4 \xrightarrow[r=2^{-0.03}]{4r} \Gamma_8 \xrightarrow[q=2^{-36}]{11r} \Gamma_{out},$$

$$\Delta_{in} = \texttt{0x0008088040000000000110000000000000001100080000000000808000000000},$$
$$\Gamma_{out} = \texttt{0x18c8ccc0000000001101000000000300200000000000020004ccc000003001}.$$

This yields a data complexity of $\varepsilon \cdot 2^{-178.06}$. Recovering the 64-bit XOR value of the two final round key halves then results in a time complexity of $\varepsilon \cdot 2^{-242.06}$.

Table 4: Summary of bounds, attacks, and round reduction estimates for Ballet.

Scheme	Rounds	Attack	Proven bounds (conservative)		Trail growth estimates	
		diff-lin.	No differential	No linear	No diff./linear	Recommended
		r	$\geq$ r	$\geq$ r	$\geq$ r	**r ($\geq$35% sec.)**
Ballet-128/128	46	16	27	26	20	**27**
Ballet-128/256	48	17	27	26	20	**27**
Ballet-256/256	74	22	54	52	40	**54**

5 Conclusions

Although the Ballet block cipher family was selected as the winning symmetric-cipher design in the CACR competition, its study outside China has remained limited due to the lack of accessible documentation. To advance understanding of its security, we extended the analysis beyond the number of rounds examined in the original design report. Using SAT-based methods implemented in the CLAASP library, together with the optimizations and heuristics introduced in Section 3.3, we provided explicit differential, linear, impossible differential, and differential-linear trails for both Ballet-128 and Ballet-256. Finally, we applied our newly found differential-linear distinguishers to derive key-recovery attacks on reduced-round Ballet-128/128, Ballet-128/256 and Ballet-256/256, reaching up to 15, 17 and 22 rounds respectively. These attacks may still be improved by combining our distinguishers with more refined key-recovery techniques.

Nevertheless, we concur with the designers of Ballet that the number of rounds has been chosen conservatively. In particular, our best attack on Ballet-128 does not exceed 17 rounds. In practice, extending differential-linear key-recovery attacks beyond 16 rounds is constrained by the limited availability of PNBs, leading to prohibitive time complexity. Moreover, considering the growth trend of differential and linear trail weights, we estimate that no viable differential or linear distinguisher exists starting from round 20. This suggests that the round count of Ballet-128 could be safely reduced from 46/48 to approximately 27 rounds while still maintaining a security margin of at least 35% with respect to the best reachable attack. A similar observation holds for Ballet-256. Our best attack reaches 22 rounds, and the projected growth of differential and linear trail weights indicates that no viable differential or linear distinguisher should exist, conservatively, from round 40. Therefore, reducing the number of rounds from 74 to 54 would still maintain a security margin of at least 35% with respect to the best reachable attack. Such reductions would significantly improve performance, representing a substantial gain, especially when targeting lightweight applications.

Overall, our results broaden the current understanding of Ballet's security margin and offer a basis for further analysis. A natural direction for future work is to investigate the related-key setting, as well as other cryptanalytic techniques that were not considered in this work, such as boomerang attacks.

As future work, we plan to evaluate the optimized SAT-based search and time-based cutoff heuristic against other methods and across additional ciphers, to assess their impact on improving state-of-the-art trails and upper bounds; both techniques will also be integrated into future CLAASP releases (beyond version 3.2.0).

A Test Vectors

We generated test vectors from our CLAASP implementation to support reproducibility and verification of our results, and to serve as an independent reference. Official test vectors were not publicly available; however, our generated vectors match when plugged in the reference implementation provided by the authors.

```
[128/128]
pt:  e60e830ca56ec84814fbd2579993d435
key: cd52c514213c9632514fb60a64840881
ct:  c1c2e89c1581d166f3c87b5999f87a9f
[128/256]
pt:  c419afdd747886b9f8e6890a3db19fa3
key: 8e1d7bede15b5fae9e67b09c734829149b5e7f8d02f49fccaa1437574d9f792b
ct:  636f07e9df66d2ec34d0ad3bb87e0f79
[256/256]
pt:  fdc0bf9c6bfeb2ffd160128e5190af6cdad291114d953986de472ad8be6ea8c7
key: 19f29ab90c31da41d2013ed7128338ad7eacb494fae0572801c30948454cb1ca
ct:  2d07ee91d634c27f3155f9e575bdc634acaa611e3654c4ce06ea130e9bc394ee
```

B Sample trails

In this section, we report the best input/output values found for impossible (Table 5, Table 6), differential (Table 7, Table 8), linear (Table 9, Table 10) and differential-linear (Table 11, Table 12) cryptanalysis. All complete trails are available in the project's GitHub repository[13].

Note that in our CLAASP implementation of Ballet, the cipher can be instantiated with an arbitrary number of rounds. The final round is treated slightly differently from the others: it omits the Feistel branch swap that appears after every other round. Consequently, the output difference returned by our trail search always corresponds to the state after the omitted final swap, regardless of the number of rounds instantiated.

Table 5: Impossible differentials Ballet– SAT exhaustive search up to Hamming weight 2

128 bit block

Rounds	Plaintext	Ciphertext
2	10^{127}	10^{127}
3	10^{127}	10^{127}
4	10^{127}	10^{127}
5	10^{127}	10^{127}
6	10^{127}	$0^{96}10^{31}$
7	$0^{6}10^{121}$	$0^{96}10^{31}$
8	$0^{6}10^{104}10^{16}$	$0^{96}10^{31}$

256 bit block

Rounds	Plaintext	Ciphertext
2	10^{255}	10^{255}
3	10^{255}	10^{255}
4	10^{255}	10^{255}
5	10^{255}	10^{255}
6	10^{255}	$0^{192}10^{63}$
7	$0^{6}10^{249}$	$0^{192}10^{63}$
8	$0^{6}10^{200}10^{48}$	$0^{192}10^{63}$

[13] https://github.com/Crypto-TII/claasping_ballet/tree/main/Ballet/results

Table 6: Impossible differential – Ballet-128/128 – MILP fully automatic

Rounds		Trail
2	pt:	0000000000000000000000000000000000000000000010000000000000000000000000000010000000000000000000000000000000000000000000000000
	1r:	0000000000010000000000000000000000000000000010000000000000000000000000000010000000000000000000000000000000000000000000000000
	1r:	??????????????????????????????????????1?????????0??0?????????????1??111?00?0000000?11?0001O??00?000?000??????????????????????????????????????
	ct:	0000000001000001000100000000000?1?????????0??0?????????????1??111?00?0000000?11?0001O??00?000?000000000000000?00???0?000000000000
3	pt:	0000000001100000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000
	2r:	????00000000000000000000000000000???0
	1r:	?????1?????????????????????????????????1?????1???1O?111111111111?1????0000000?????????000?000000000000??????????????????????????????
	ct:	1001000001000000000000001000000??1?????1???1O?111111111111?1????0000000?????????000?000000000000???1?????????0?0000?00000?00010
4	pt:	0000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
	1r:	0000000000000000000000000000000100000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000
	3r:	???1000?????????????????????????????0000001000000000000000000000000000000000000000000000000000000000000000?????????0000000??????????????????
	ct:	0000000000000000000000000000000100000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000
5	pt:	0000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
	2r:	1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
	3r:	???100?????????????????????????????0000001000000000000000000000000000000000000000000000000000000000000000?????????0000000??????????????????
	ct:	0000000000000000000000000000000100000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000
6	pt:	0000001000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
	2r:	0000000000000000000000000000000?????????????????????????????100000?????????????????1000000000000000000000000000000000000000
	3r:	???100?????????????????????????????0000001000000000000000000000000000000000000000000000000000000000000000?????????0000000??????????????????
	ct:	0000000000000000000000000000000100000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000

Table 7: Differential (single-key) – Ballet-128/128

# Rounds	Best weight	Input difference	Output difference
1	0	0x00000000800000008000000000000000	0x00000000800000008000000000000000
2	0	0x02000000000000000000000000010000	0x00000000800000008000000000000000
3	2	0x02000000000000000000000000010000	0x00000020000000000000000000004000
4	6	0x10080000020000000001000001008000	0x00000020000000000000000000004000
5	10	0x10080000020000000001000001008000	0x00804000000000200004000010080000
6	19	0x10100000022020000022000001011008	0x80010800000000040000008000210000
7	26	0x02022010004400000004400200202000	0x04804010004200000840400080000200
8	37	0x11111080020220100020200081150000	0x04804010004200000840400080000200
9	46	0x00200804000001008004000040020080	0x14000120800000100002001080002000
10	57	0x01100000002000000002000000110008	0x14000120800000100002001080002000
11	64	0x02022010004400000004400200202000	0x04008048200000440000000400100800
12	75	0x13111080020220100020200081150000	0x04008048200000440000000400100800
13	84	0x02000004000201008042010040200100	0x40000080422100800201008080001008
14	91	0x2200004c040000088040020046220000c	0x80000100844201000402010000002011
15	105	0x402884841111108081150000008020800	0x80440000000020000040020008800010
16	127	0x00080404020000040201008002008040	0x84002000000200111040000a0800050

Table 8: Differential (single-key) – Ballet-256/256

Rounds	Weight	Input difference	Output difference
1	0	0x020000000000000fffffffffffffffffffffffffffffffff0001000000000000	0x800000000000000fffffffffffffffffffffffffffffffff8000000000000000
2	0	0x0200000000000000000000000000000000000000000000001000000000000	0x0000000000000008000000000000080000000000000000000000000000000000
3	2	0x0200000000000000000000000000000000000000000000001000000000000	0x0000000000000020000000000000000000000000000000000000000000004000
4	6	0x0200000000000000000000000000000000000000000000001000000000000	0x0000000000804000000000000000020000000000004000000000010080000
5	10	0x0000201000000000000040000000000000002000000000000020100000000	0x8000000000001000040000000000080000000000000000000000002010
6	19	0x1010000000000000220200000000000022000000000001011008000000000	0x8000000000108000000000000004000000000000080000000000210000
7	26	0x1010000000000000220200000000000022000000000001011008000000000	0x0000000401000080000000001080000000000021000000000802402000
8	38	0x1111108000000000202201000000000020200000000001110008004000000	0x0000001004804000000000000420000000000084040000000020080000000
9	48	0x0000010080040000000002000000000000000001008000000000100800400	0x0200000000800024100000000000002000000000004002000000010000400
10	58	0x0000010000400000000002000800000000000008800000000001000044020	0x8000000000108000000000000004000000000000080000000000210000
11	66	0x0000000002022010000000000440000000000000044002000000000202000	0x0400000000080482000000000000440000000000000040000000000100800
12	79	0x0800000000011111010000000002022000000000002020008004000001110	0x0480400000000080442000000000000400000000000080000000000100
13	89	0x0202201000000000044000000000000044002000000002020000000000	0x0120120100000004001091000000000021001000000000200000000000080
14	102	0x0000000011111080000000002022010000000000202000804000001110000	0x0000000401201201000000000109100000000021001000000080200000000
15	141	0x1311108000000000202201000000000020200000000001110008004000000	0xc0004200000840000000201000000200000000000040000040000108100
16	153	0x1311108000000000202201000000000020200000000001110000800400000	0x0004804000020100800042000000000000040000000100008000000402000

Table 9: Linear – Ballet-128/128

# Rounds	Correlation	Input mask	Output mask
1	0	0x00000000000040000000040000000020000	0x00000000000000000000000000000001
2	0	0x00000000040000000000000000000000	0x0000000100800000008000000000000
3	1	0x00000000000040000000040000000020000	0x00000000000000002000000020000c000
4	2	0x0400000000840000c084000000020000	0x0000000000000c0000000c00020000000
5	4	0x0400000000084000c0084000000020000	0x000018000000c0880000008800300000
6	8	0x0000010010000020100000002008000000	0x008000010e8460000884618000000301
7	12	0x040300000084430c00e4430000020100	0x0804000000462300006623080c000010
8	19	0x0400000000084000c0084000000020000	0x18000000000040000c00401018000002
9	23	0x0400804000841080008410040002000	0x0060000000022000600220000c0000000
10	26	0x0420000004800028008002000020010	0x0000040000000022000400220080000
11	30	0x2013000000334200040142000000180	0x0000040000000033000600330080000
12	36	0x9800010012100011021000300000c0000	0x0044000040231300000313008c001000
13	43	0x980000311300403613000006100000000	0x101000020d88c8190108c80020004000
14	49	0x04c0000808908000805080010000600	0x1080c000884a4740040c474001400200
15	56	0x00000000200604022000040298000201	0x002110000190988a0008188a0000c004
16	63	0x040000000c8060000c80600004002000	0x0000002098000001980030010000400

Table 10: Linear – Ballet-256/256

Rounds	Correlation	Input mask	Output mask
1	0	0x040000000000000ff7bffffffffffff7bffffffffffff0002000000000000	0x0000000000000001ffffffffffffffffffffffffffffffff0000000000000001
2	0	0x0400000000000000080000000000000800000000000000000000000000000000	0x0000000000000000000000000000010000000000000000000000000000000000
3	1	0x0000000000000000300000000000000200000000000000000000000000000000	0x0000000000000000800000000000000000000000000000010000000000000000
4	2	0x0400000000000000840000000000c0840000000000000000200000000000	0x0000000000000000000000000000c000000000000000c0000000000020000000
5	4	0x0400000000000000840000000000800840000000000000002000000000000	0x0000000000010000000000000c0c8000000000000c8000000000300000
6	8	0x0000010000000000000003018000000000030180000000000000008000000	0x00c0000000000010a8440000000000c8440000000018000000000000201
7	12	0x00000000c060000000001801088400000000001c884000000000000603000	0x00c0000000000010e8440000000000088440000000018000000000000301
8	19	0x0400000000000000084000000000c0084000000000000002000000000000	0x0000000018000000000000000040000000100c0040000000000218000000
9	23	0x0400000000000000084000000000c0084000000000000002000000000000	0x0000000001000000000020080880000000230800880000804020000000
10	26	0x0433000000000000480400000000080080400000000000002000000000000	0x0000060000000000000002200000000006002200000000000c000000000000
11	30	0x201b000000000000032420000000004042000000000000001800000000000	0x0000060000000000000002200000000006002200000000000c000000000000
12	36	0x0000018098000000000000112100000000000300a1000000000000000c0000	0x0000000006600000000000403202000000000200202000001800cc000000
13	42	0x0000000000003013430000000000002243000000000060101000000000000	0x1800000000000201 0c080000000001000800000002001000000000000030
14	48	0x800000000000c04d8c000000000008c18c0000000001800040000000000	0xc0c0000000000184e44400000100c808444000000100040000000040030001
15	54	0x0000660600000000100c04c0c6000000000000440c60000000000040003000000	0x0031000000000000190d08000000002000010800000000200000000000c004
16	63	0x0401800000000000020a421000000c00c6494210000000000301200c000000000	0x0000000008000030100000000040000110000008044020010000001180006000

Table 11: Differential-linear – Ballet-128/128. Each row gives the top input/output differences, the middle deterministic output, and the bottom linear input/output masks.

Rounds	Weight	Trail
9 (4–1–4)	12 tot 6 diff. 3 lin. (×2)	0x00000400000000000000000000000002 0x00400000800001000000201080000000 10000000000000000000000100000000??????????????????????10000000??????????????????10000000000000000000000000000000010000000010000 0x00000000000000000000100000000000 0x0080000000046000c004600000000001
10(4-1-5)	16 tot 6 diff. 5 lin. (×2)	0x10080000020000000001000001008000 0x00000000000000200000400000000000 00000000000000000000000000100000??????????????????10000000000000000??????????????10000000000000000000000000000000000001000000000000000 0x00000000001000000001000000000000 0x00000000001000000001000000000000
11 (5–1–5)	22 tot 12 diff. 5 lin. (×2)	0x00010080000020000000001000001008 0x02000000000008040001008000000004 00000000000000000000100000000100??????????????????????10000000000??????????????10000000000000000000000000000000010000000010000000 0x00000000001800000001800000000000 0x00000001808000018080000100004000
12 (5–1–6)	30 tot 10 diff. 10 lin. (×2)	0x00002010000004000000000200000201 0x00400000800001000000201080000000 10000000000000000000000100000000??????????????????????10000000??????????????????10000000000000000000000000000000010000000010000 0x00000000000040000000400000003000 0x00000019c080100cc080080004002300
13 (6–1–6)	40 tot 22 diff. 9 lin. (×2)	0x000884c0000100800000100800208844 0x02000000000008040001008000000004 00000000000000000000100000000100??????????????????????10000000000??????????????10000000000000000000000000000000010000000010000000 0x00000000001000000001000000000000 0x20000060211800022118600300008060
14(7-1-6)	52 tot 32 diff. 10 lin. (×2)	0x40280484000111888004110844008040 0x00400000800001000000201080000000 10000000000000000000000100000000??????????????????????10000000??????????????????10000000000000000000000000000000010000000010000 0x00000000000040000000400000002000 0x00000011c080180cc080000004002200
14(6-1-7)	52 tot 20 diff. 16 lin. (×2)	0x00011088000020100000020180041108 0x00400000800001000000201080000000 00000000000000000000000100000000??????????????????????10000000??????????????????10000000000000000000000000000000010000000010000 0x00000000000040000000400000003000 0x30040320991002199f10310000006060
15(7–1–7)	61 tot 27 diff. 17 lin. (×2)	0x20000020404000044400000002201002 0x02100100200000808040100442000000 00100000000000000000000010000000??????????????????????????1000000??????????????????????????10000100000000100000000010000000000100 0x00000000000020000000000000000000 0x00000001000400008004018000000021
16(8–1–7)	75 tot 37 diff. 19 lin. (×2)	0x13111080020220100020200081150000 0x00420000048040108000020008404000 00000100100000000100000000010000??????????????????????????????????1??????????????????????????00000010000000000000000000001000000000 0x00000000000020000000000000000000 0x18000000000060000c00601010000002
17 (10–1–6	84 tot 64 diff. 10 lin. (×2)	0x00201000080000100804021008020100 0x00008020010848040108000000000400 00000001000010000100100000000100??????????????????????00000000000??????????????10000000000000000000000010000100000000000000000000 0x00000000000c00000000800000000000 0x18000020d88800011888200100006030
18 (7–1–10)	94 tot 30 diff. 32 lin. (×2)	0x00402000000800000000040000040200 0x8c4200200480e0089004012088400010 10000010010000000111000000000001000??????????????????????????????????0??????????????????????????????11001000000000100000000001001000000 0x00000000000020000000000000000000 0x06001000302200cc003b8ccc02200008
19(7–1–11)	106 tot 40 diff. 33 lin. (×2)	0x40288c84111110808115000008020800 0x00000000000001000000200080000000 00000000000000000000000100000000??????????????10000000000000000??????????????????10000000000000000000000000000000010000000000000 0x0000006000100c000000000000000006 0x0c8080100864644080007c4001000001
20(8–1–11	118 tot 54 diff. 32 lin. (×2)	0x00004001200080000000880030000440 0x00000000000020000000000000000000 00000000000000100000000000000000?????10000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000 0x02000006864000008000000000000000 0x000004602004022020000020000c8001

Table 12: Differential-linear – Ballet-256/256. Each row gives the top input/output differences, the middle deterministic output, and the bottom linear input/output masks.

Rounds	Weight	Trail
9 (4–1–4)	14 tot 6 diff. 4 lin. (×2)	0x0200000000000000000000000000000000000000000000000000000001000000000000 0x0000000000000002000000000080400000000000100800000000000000004000 0000000000000000000000000000000000000001000000001000000000000000???100000000000 ??1000000000000000000000000000000000000000000000000000000000000000000000001000000001000000000000000000 0x0000000000000000000000001800000000000001800000000000000080000 0x0000000000000000000000800000000000008000000000300000000000
10(4-1-5)	18 tot 8 diff. 5 lin. (×2)	0x0000000080400000000000010000000000000000800000000000008040000 0x0000000000000000000001000000000002000000000000000000000000 0000000000000000000000100000000000000000000000000000000?????????????1000000000000000000000000000000000000000000000000000000000000000000000 ??????????10000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000 0x0000000000000000010000000000000010000000000000000000000000 0x0000000000000018080000000000018080000000000010000000004000
11 (5–1–5)	22 tot 10 diff. 6 lin. (×2)	0x0000201000000000000040000000000000002000000000000020100000000 0x0040000000000008000000000010000000000002010800000000000000 1000000000000000000000000000000000000000000000000010000000??100000000 ??1000000000000000000000000000000000000000000000000000000000000000000010000000010000 0x0000000000000000000000004000000000000040000000000000002000 0x0000000800000000000000660000000000800660000000001000000000
12 (5-1-6)	30 tot 10 diff. 10 lin. (×2)	0x1008000000000000200000000000000010000000000001018000000000 0x0000000000000200000000080400000000001008000000000000004000 0000000000000000000000000000000000000100000000100000000000000000000000000000000000000000000000000100000000 ??1000000000000000000000000000000000000000000000000000000000000000000000001000000001000000000000000000 0x0000000000000000000000001000000000000000100000000000000000000000 0x0000004030000000000003a1900000000400221900000000804000000000
13 (6-1-6)	39 tot 21 diff. 9 lin. (×2)	0x1300000000000002000000000000002000000000000011000800000000 0x0000000010800000000040120100000000802000000000000000210100000 0000000000000000000010000000010010000000100000000000000000000?????????????????????????????????000000000000000000000000000 ????????????????????????????1000000000000000000000000000000000000000000000000000000000000000000000000100000000100000000000000000000000000000000000 0x0000000000000000000030000000000000030000000000000000000000000 0x00c0600000000000007433000000008004433000000000c0000000000001
14 (7-1-6)	tot diff. lin. (×2)	0x0110080000000012200000000000000220010000000001010000000000 0x0000000021000000000802402000000010040000000000000000420200000 0000000000000000000100000000100100000001000000000000000000000?????????????????????????????????000000000000000000000000000 ????????????????????????????1000000000000000000000000000000000000000000000000000000000000000000000001000000010000000000000000000000000000000000 0x0000000000000000000060000000000000060000000000000000000000000 0x0180c0000000000000e864000000000088640000000101000000000002
15 (7-1-7)	60 tot 28 diff. 16 lin. (×2)	0x0010110080000000022000000000000022001000000000101000000000 0x0000000000210000000000802402000000001004000000000000000420200 0000000000000000000000000000000100000000100100000001000000000?????????????????????????????????????000000000000000000 ??1000000000000000000000000000000000000000000000000000000000000000000000001000000001000000000000000000000000000000000 0x0000000000000000000000001000000000000001000000000000000080000 0x0180880c000000000048646640000000084026640000010010100000000001
16 (8-1-7)	70 tot 44 diff. 13 lin. (×2)	0x0000000000000002400000000000000000000000000080020100000000004 0x0000000880000000001101000000000022000000000000000000000000000 0000000000000000001000100000001000000000000000000000000000?????????????????????????1000000000000000000000000000000000 ??????????????????1000000000000000000000000000000000000000000000000000000000000000000000001000100000000000000000000000000000000000000000000 0x00000000c000000000000001008000000000001008000000000000060000 0x00c0600000000000007422000000000c0044220000000000c0000000000001
17 (9-1-7)	83 tot 59 diff. 12 lin. (×2)	0x0000000000040001000000000008002200000000000002110080000004000 0x0000000800000000001000000000000200000000000000000000000000 0000000000000000001000000000000000000000000000000000000000000?????????????????????1000000000000000000000000000000000000 ?????1000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000 0x0000010000000000000002010000000000002010000000000000c000000 0x0080000000000010a846000000000c846000000001000000000000201
18 (10-1-7)	91 tot 65 diff. 13 lin. (×2)	0x0088000000000000010000000000000010000000000000980040000000 0x0000000840000000000090080000000001000000000000000800000000 0000000000000000000000100100000001000000000000000000000000?????????????????????????????000000000000000000000000000000 ??????????????????????1000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000 0x0000000100000000000000002010000000000002010000000000000080000 0x0100c0000000000000ac64000000000cc6400000000101000000000002
19 (7-1-11)	94 tot 28 diff. 33 lin. (×2)	0x8088040000000001100000000000001100080000000080800000000000 0x0000000010800000000040120100000000802000000000000000210100000 0000000000000000000010000000010010000000100000000000000000000?????????????????????????????????000000000000000000000000000 ??????????????????????????????1000000000000000000000000000000000000000000000000000000000000000000000001000000001000000000000000000000000000000000 0x0000000000310000000000040240000000000000040000000000000001800 0x0000000000000030100000000000001100000000003001000000000006000
20 (7-1-12)	107 tot 29 diff. 39 lin. (×2)	0x0008088040000000001100000000000001100080000000080800000000 0x0000000000108000000000401201000000008020000000000000210100 0000000000000000000000000000000010000000100100000001000000000?????????????????????????????????????000000000000000 ??????????????????????????????????????1000000000000000000000000000000000000000000000000000000000000000000000001000000001000000000000000000000000 0x000000000000c0600000000000010c000000000000100000000000000000 0x11010000000003018c8ccc0000000000004ccc0000030010200000000000002

References

[1] CLAASP GitHub repository, https://github.com/Crypto-TII/claasp/

[2] Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 329–358. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_12

[3] Bellini, E., Gérault, D., Grados, J., Huang, Y.J., Makarim, R.H., Rachidi, M., Tiwari, S.K.: CLAASP: A cryptographic library for the automated analysis of symmetric primitives. In: Selected Areas in Cryptography - SAC 2023 - 30th International Conference, Fredericton, Canada, August 14-18, 2023, Revised Selected Papers. pp. 387–408. Lecture Notes in Computer Science, Springer (2023). https://doi.org/10.1007/978-3-031-53368-6_19

[4] Bellini, E., Gérault, D., Grados, J., Makarim, R.H., Peyrin, T.: Fully automated differential-linear attacks against ARX ciphers. In: Topics in Cryptology - CT-RSA 2023 - Cryptographers' Track at the RSA Conference 2023, San Francisco, CA, USA, April 24-27, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13871, pp. 252–276. Springer (2023). https://doi.org/10.1007/978-3-031-30872-7_10

[5] Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer (1999). https://doi.org/10.1007/3-540-48910-X_2

[6] Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991). https://doi.org/10.1007/BF00630563

[7] Chinese Association for Cryptologic Research (CACR): 全国密码算法设计竞赛通知 / Notice of the National Cryptographic Algorithm Design Competition. https://sfjs.cacrnet.org.cn/site/content/309.html (Jun 2018), accessed: 2025-11-22

[8] Chinese Association for Cryptologic Research (CACR): 关于全国密码算法设计竞赛算法评选结果的公示 / Announcement of the Evaluation Results of the National Cryptographic Algorithm Design Competition. https://www.cacrnet.org.cn/site/content/854.html (Jan 2020), accessed: 2026-04-17

[9] Cui, T., Chen, S., Fu, K., Wang, M., Jia, K.: New automatic tool for finding impossible differentials and zero-correlation linear approximations. Sci. China Inf. Sci. (2021). https://doi.org/10.1007/s11432-018-1506-4

[10] Cui, T., Wang, M., Fan, Y., Kai, H., Fu, Y., Huang, L.: Ballet: A software-friendly block cipher. Journal of Cryptologic Research **6**(6), 704 (2019). https://doi.org/10.13868/j.cnki.jcr.000335, http://www.jcr.cacrnet.org.cn/EN/10.13868/j.cnki.jcr.000335, also available at: https://www.proquest.com/docview/2899235146

[11] Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In: FSE 2016. LNCS, vol. 9783, pp. 268–288. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_14

[12] Gérault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9892, pp. 584–601. Springer (2016). https://doi.org/10.1007/978-3-319-44953-1_37

[13] Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer (1994). https://doi.org/10.1007/3-540-48658-5_3

[14] Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer (2001). https://doi.org/10.1007/3-540-45473-X_28

[15] Mao, Y., Wu, W., Zheng, Y., Zhang, L.: Related-Cipher Attacks: Applications to Ballet and ANT. In: Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13915, pp. 109–123. Springer (2023). https://doi.org/10.1007/978-3-031-35486-1_6

[16] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33

[17] Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: EUROCRYPT (2017). https://doi.org/10.1007/978-3-319-56617-7_7

[18] Song, L., Huang, Z., Yang, Q.: Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In: ACISP 2016. LNCS, vol. 9723, pp. 379–394. Springer (2016). https://doi.org/10.1007/978-3-319-40367-0_24

[19] Zhang, Y., Zhang, L., Zheng, Y., Wu, W.: A New Automatic Framework for Searching Rotational-XOR Differential Characteristics in ARX Ciphers. Designs, Codes and Cryptography **93**(6), 2013–2054 (Feb 2025). https://doi.org/10.1007/s10623-025-01571-6