On Extending Integral Distinguishers

Dachao Wang[1], Hosein Hadipour(✉)[2], and Simon Gerhalter[3]

[1] Lund University, Lund, Sweden
dachao.wang@eit.lth.se
[2] Ruhr University Bochum, Bochum, Germany
hossein.hadipour@rub.de
[3] TU Graz, Graz, Austria
simon.gerhalter@tugraz.at

Abstract. Integral cryptanalysis analyzes block ciphers using input structures for which the sum of a chosen function of the output bits becomes key-independent. However, most methods still test one output expression at a time, so they can miss distinguishers that emerge when several outputs are combined, either linearly or nonlinearly. They are also not designed to capture key-dependent integral combinations, which may hold deterministically on part of the key space. In this work, we develop Split-and-Cancel, a method that combines exact expansion in a short final part with an oracle on the preceding rounds to determine which suffix monomials can survive from the chosen structure and records them in a binary matrix. Key-independent combinations are then extracted from the left kernel of this matrix. We first apply the method in a reduced model with omitted boundary key additions, where linear dependencies in this matrix yield certified sum combinations independent of keys among output bits and higher-degree output products. When the omitted round key in the combination is restored, the same combinations yield deterministic weak-key distinguishers. In connection with the weak-key distinguishers, we further investigate their probabilistic behaviour and show that the probability-estimation formula used in Zhang et al.'s EUROCRYPT 2026 work is invalid. The found flaw is supported by our experiments. We apply the method to SIMON, SIMECK, SPECK, PRESENT, and GIFT. Our strongest deterministic results improve the best integral distinguishers for SIMON, SIMECK, and SPECK by one round at every standard block size. For PRESENT and GIFT, we improve deterministic weak-key integral distinguishers by one round. In each case, the exact weak-key class covers at least a quarter of the key space: 2^{78} of 2^{80} keys for PRESENT-80, 2^{126} of 2^{128} keys for PRESENT-128, GIFT-64 and GIFT-128. These results show that exact modeling of a short final part can reveal key-independent and weak-key integral behavior missed by analysis using single observables.

Keywords: Symmetric-key cryptanalysis, Boolean functions, Integral cryptanalysis, Linear algebra, SAT

1 Introduction

Integral cryptanalysis is one of the standard tools for analyzing symmetric-key primitives. Traditionally, it examines sums of outputs over carefully chosen input structures and identifies deterministic, key-independent properties such zero-sum combinations. The technique originated in the Square attack [1] and was later formulated as a general cryptanalytic framework by Knudsen and Wagner [2].

Over time, the field evolved from relying on handcrafted arguments to employing automatic cryptanalysis. A major advance in integral cryptanalysis was the introduction of the division property by Todo [3]. Todo and Morii refined this approach to the bit level and applied it successfully to SIMON [4]. Xiang et al. then cast division-trail search as a mixed-integer linear programming (MILP) problem and improved the best integral distinguishers for several ciphers, including SIMON, SIMECK, and PRESENT [5]. Hao et al. further extended this line through the three-subset division property without unknown subset [6]. Hu et al. connected these developments to monomial prediction by interpreting the search in terms of monomials in the algebraic normal form (ANF) [7]. Together, these works made searches of long integral distinguishers far more systematic than earlier handcrafted analyzes.

Despite this progress, most automatic methods still inspect one output expression at a time. That viewpoint is effective when a distinguisher is visible on a single bit, but it can miss distinguishers that emerge only after several output expressions are combined. This limitation has already surfaced in previous work. Hebborn et al. took linear combinations into account in their proof of GIFT-64's integral resistance [8]. Derbez and Fouque used a restricted family of combinations to sharpen division-property propagation [9]. Zhang et al. searched for useful output combinations with neural techniques [10]. Hwang et al. recovered spaces of combinations from sampled data by kernel computations [11]. Hadipour and Eichlseder discovered that certain highly nonlinear combinations of output bits can improve integral distinguishers of WARP by one round [12], and refer to these as generalized integral properties. In general, integral cryptanalysis seeks deterministic, key-independent properties. However, some integral combinations, survive only under additional conditions, for example on a subset of keys or input structures. These give rise to weak-key or conditional distinguishers. An exact method that systematically recovers such combination-based distinguishers directly from the cipher model, while also separating their deterministic weak-key core from their probabilistic interpretation, is still missing.

In this paper, we address that gap by turning the idea of combination-based cancellation into an explicit algorithmic method. Our starting point is that, once a short final part of the cipher is expanded exactly in algebraic normal form (ANF), different output observables often share unresolved monomials. Even when no observable is individually balanced, these shared monomials can cancel in suitable XOR-combinations, producing simpler Boolean functions and thereby revealing additional integral properties. In this setting, the relevant search object is no longer a single output bit, but a space of combined observables. To recover this space systematically, we split the target round count as $r = p + q$. We first

analyze the last q rounds by exact ANF expansion and enumerate the suffix monomials contributed by the chosen observable family. We then use monomial prediction on the first p rounds to determine which of these suffix monomials can survive from the chosen input structure. The surviving monomials are recorded in a binary incidence matrix, and kernel vectors of this matrix yield exact zero-sum combinations. Because the observable family may also include higher-degree output products, the method is not limited to linear combinations of output bits. This changes the search target from asking which single output is zero-sum to asking which combinations of observables are zero-sum. To analyze key-dependent behavior, we first derive such exact combinations in a model with omitted boundary key additions. We then ask for which key values these combinations remain exact after the key additions are reinstated, which yields deterministic weak-key distinguishers on explicit key slices.

1.1 Our Contributions

Our three main contributions are as follows.

- We introduce Split-and-Cancel, an exact split search strategy that turns combination-based cancellation into an explicit algorithm. It recovers spaces of key-independent combinations from an incidence matrix of unresolved suffix monomials and naturally handles both linear combinations of output bits and higher-degree output products.
- We propose a constant-sum pullback criterion to extend results found on shorter rounds to longer ones. While monomial prediction on the p side or exact ANF expansion on the q side of Split-and-Cancel may be the bottleneck in time complexity, this theorem allows us to derive key-independent combinations efficiently from combinations already found on a smaller number of rounds. It also helps derive deterministic weak-key distinguishers from exact combinations obtained in a model with omitted boundary key additions.
- We discuss about the probability estimation in probabilistic distinguishers. While the previous work on Eurocrypt 2026 [10] found some examples, we point out a flaw in the estimation, invalidating the previous formula. Our discussion consists of a theoretical view and a large number of experiments to support the our finding.
- We apply the method to SIMON, SIMECK, SPECK, PRESENT, and GIFT. Our strongest deterministic results improve the best integral distinguishers for SIMON, SIMECK, and SPECK by one round at every standard block size. For PRESENT and GIFT, we improve deterministic weak-key integral distinguishers by one round; in each case, the exact weak-key class covers at least a quarter of the key space. Detailed results are summarized in Table 1.

The *Data* column in Table 1 gives the size of the input set used by the distinguisher; for the standard active-bit structures used in most rows, a label 2^d means d active input bits. For Beyne and Verbauwhede [17], we report explicit same-dataset subspaces exhibited in their paper. At data 2^{60}, their Equation (20)

Table 1: Summary of results. For SPECK, SIMON, SIMECK, and GIFT rows written as r (r+1), r is the core round count obtained directly from the search, and (r+1) is the standard one-round input-side extension. Rows marked with † are empirical SPECK-32 distinguishers from [11].

Cipher	Rounds	Data	Basis dim.	Portion	Time (min.)	Reference
SIMON-32	14 (15)	2^{31}	3	1	—	[4,13,11]
	14 (15)	2^{31}	16	1	0.07	Subsection 5.3
	15 (16)	2^{31}	3	1	0.07	Subsection 5.3
SIMON-48	15 (16)	2^{47}	24	1	—	[5,13,14]
	16 (17)	2^{47}	24	1	1.36	Subsection 5.3
SIMON-64	17 (18)	2^{63}	22	1	—	[5,14]
	17 (18)	2^{63}	23	1	—	[13]
	18 (19)	2^{63}	22	1	4.60	Subsection 5.3
SIMON-96	21 (22)	2^{95}	5	1	—	[5,14]
	22 (23)	2^{95}	5	1	5.72	Subsection 5.3
SIMON-128	25 (26)	2^{127}	3	1	—	[5,13,14]
	26 (27)	2^{127}	3	1	11.70	Subsection 5.3
SIMECK-32	14 (15)	2^{31}	7	1	—	[4,5,14,13,11]
	15 (16)	2^{31}	7	1	0.05	Subsection 5.4
SIMECK-48	17 (18)	2^{47}	5	1	—	[5,14,13]
	18 (19)	2^{47}	5	1	0.17	Subsection 5.4
SIMECK-64	20 (21)	2^{63}	5	1	—	[5,14,13]
	21 (22)	2^{63}	5	1	0.34	Subsection 5.4
SPECK-32	6 (7)†	2^{30}	9	—	—	[11]
	6 (7)	2^{29}	1	1	—	[**?**]
	6 (7)	2^{29}	11	1	2.69	Subsection 5.5
	6 (7)	2^{30}	20	1	2.77	Subsection 5.5
	7 (8)†	2^{30}	1	—	—	[11]
	7 (8)	2^{29}	1	1	2.69	Subsection 5.5
	7 (8)	2^{30}	1	1	2.77	Subsection 5.5
SPECK-48	6 (7)	2^{45}	1	1	—	[15,14,16]
	6 (7)	2^{42}	1	1	2.63	Subsection 5.5
	7 (8)	2^{43}	1	1	2.63	Subsection 5.5
SPECK-64	6 (7)	2^{61}	1	1	—	[15,14,16]
	6 (7)	2^{50}	6	1	3.06	Subsection 5.5
	7 (8)	2^{50}	1	1	3.06	Subsection 5.5
SPECK-96	6 (7)	2^{93}	1	1	—	[15,14,16]
	6 (7)	2^{66}	45	1	2.80	Subsection 5.5
	7 (8)	2^{66}	1	1	2.80	Subsection 5.5
SPECK-128	6 (7)	2^{125}	1	1	—	[15,14]
	6 (7)	2^{82}	45	1	3.24	Subsection 5.5
	7 (8)	2^{82}	1	1	3.24	Subsection 5.5
PRESENT	9	2^{60}	1	1	—	[5,14]
	9	2^{60}	4	1	—	[13,11]
	9	2^{60}	2	1	—	[17]
	9	2^{63}	7	1	—	[17]
	9	2^{63}	28	1	—	[13]
	9	2^{60}	3	1	16.37	Subsection 5.1
	9	2^{60}	4	$\geq 2^{-2}$	16.37	Subsection 5.1
	10	2^{60}	1	$\geq 2^{-2}$	8.34	Subsection 5.1
	10	2^{63}	1	$\geq 2^{-2}$	8.17	Subsection 5.1
GIFT-64	9 (10)	2^{61}	5	1	—	[14]
	9 (10)	2^{63}	32	1	—	[18]
	9 (10)	2^{61}	8	1	1.04	Subsection 5.2
	9 (10)	2^{61}	9	$\geq 2^{-2}$	1.04	Subsection 5.2
	10 (11)	2^{62}	1	$\geq 2^{-2}$	1.08	Subsection 5.2
	10 (11)	2^{63}	1	$\geq 2^{-2}$	1.04	Subsection 5.2
GIFT-128	11 (12)	2^{127}	32	1	—	[14]
	11 (12)	2^{127}	32	1	78.85	Subsection 5.2
	12 (13)	2^{127}	1	$\geq 2^{-2}$	24.01	Subsection 5.2

gives a second exact combination on the same input set as the property of [5], so these two distinct combinations span a 2-dimensional exact subspace. At data 2^{63}, their Appendix C [17], which they state gives a basis of their dimension-22 quotient space, contains seven exact combinations on the affine set $x_5 + x_9 = 0$; these seven combinations are therefore linearly independent. The *Basis dimension* column gives the dimension of the reported space of key-independent combinations for the stated input set; for Beyne and Verbauwhede, the values 2 and 7 refer to the explicit same-dataset subspaces just described. For previous distinguishers on single output bits, this is simply the number of balanced output bits. If the basis dimension is d, then the corresponding space contains $2^d - 1$ nontrivial key-independent combinations. The *Portion* column gives the fraction of the key space for which the distinguisher is proved to hold.An entry of the form $\geq 2^{-2}$ indicates that the reported distinguisher holds for at least a 2^{-2} fraction of the key space. An em dash indicates that this fraction is unknown or not claimed.

All runs reported in this paper were performed on a laptop with an Apple M4 processor, 24 GB RAM, and 10 CPU cores. The *Time* column in Table 1 reports wall-clock running time in minutes only for our rows. The code and experiment scripts used in this paper are publicly available.[4]

1.2 Paper Organization

The paper follows the logic of the method. Section 2 positions the method with respect to integral cryptanalysis, monomial prediction, and recent linear-algebraic approaches. Section 3 fixes notation and the prefix-suffix viewpoint. Section 4 presents the Split-and-Cancel method, the matrix formulation, the pullback step, and toy examples. Section 5 applies the method to PRESENT, GIFT, SIMON, SIMECK, and SPECK. Section 6 gives a conclusion and outlines future directions. The appendix explains modeling, gives short specifications of the ciphers, collects comparison tables, and a proof of the pullback criterion.

2 Related Work

Two themes matter most for the present paper: increasingly precise algebraic models of integral behavior and increasingly scalable ways to search them. From an algebraic viewpoint, monomial prediction clarified what the division property is really detecting. Hu et al. showed that monomial prediction gives a clean way to reason about the presence or absence of monomials in composed Boolean functions and proved its equivalence to the most precise three-subset framework without unknown subset [7]. That perspective was later used in concrete integral distinguisher discovery, including the integral analysis of WARP [12], where the authors first obtained monomial-prediction-based integral distinguishers and then observed that highly nonlinear combinations of output bits extend them

[4] `https://github.com/hadipourh/splitandcancel`

by one additional round. The integral resistance property proposed by Hebborn et al. [8] gives strong evidence against integral distinguishers. However, they consider only linear combinations of output bits.

Three recent lines of work are close to ours. Zhang et al. [10] used neural feature discovery to suggest promising output combinations for integral cryptanalysis and then verified those combinations with a split search that backward-expands a short suffix and checks the remaining monomials with a prefix model. At a high level, their verification stage and ours share the same $p + q$ intuition. Both isolate a short suffix and ask which monomials remain compatible with the prefix side. The difference is that their pipeline is candidate-driven, relying on data-derived feature hints and per-candidate verification. Our method builds a symbolic matrix for an entire row family and extracts all canceling combinations at once. The neural-guided route can explore large feature spaces without requiring a symbolic row family in advance, but it inherits training cost, interpretability work, and no completeness guarantee on the discovered candidates.

Beyne and Verbauwhede [17] introduced algebraic transition matrices, a change-of-basis representation of transition matrices for integral cryptanalysis that unifies parity sets and monomial trails and leads to a search algorithm for generalized integral properties. Their work is methodologically closer to ours than the neural-guided line because both approaches keep the search symbolic and model-based rather than empirical. There is also a direct link to monomial prediction tables [12] for a small Boolean component, the monomial prediction table records the same monomial-incidence pattern as the corresponding algebraic transition matrix, up to row/column transposition. The main difference is therefore not whether a matrix is used, but how and where it enters the analysis. In their work, the matrix is attached to the primitive or its components through a change of basis, which gives a global algebraic description and supports several search strategies for generalized integral properties. Their practical algorithm can then be instantiated with existing automated subroutines, including division-property- or monomial-trail-based models. In our work, the matrix is built only after a $p + q$ split: its rows are candidate output observables and its columns are unresolved suffix monomials that remain compatible with the prefix filter. We then compute the left kernel of that reduced matrix to recover all canceling combinations in the chosen observable family at once. This makes Split-and-Cancel more specialized, but also more direct and lightweight when the last rounds are sparse enough that exact suffix expansion remains practical.

Hwang et al. [11] then recast balance discovery as a kernel problem over empirical parity data collected under many independent keys. This is the closest data-driven analogue to our linear-algebra step. In both works, kernel vectors describe output combinations that are balanced. The crucial difference is the matrix semantics. Their rows are sampled parity observations, so the method is data-driven, feature-dependent, and requires a separate validation stage to eliminate spurious masks. Our rows are symbolic observables and our columns are unresolved suffix monomials that survive a forward algebraic filter. Hence our kernel computation produces exact deterministic certificates rather than

empirical candidates. In return, our method requires an exact suffix model and remains most effective when the last rounds are sparse enough that symbolic expansion is still feasible. Our position in this landscape is therefore specific: we keep the exact, model-based flavor of algebraic analysis, but we use linear algebra to search an entire observable family at once.

3 Preliminaries and Background

3.1 Boolean Functions, Monomials, and Naming

In this paper, all computations take place over $\mathbb{F}_2$, and every sum is an $\mathbb{F}_2$-sum. We use $+$ for addition in $\mathbb{F}_2$ and $\mathbb{F}_2^n$ and $\sum$ for indexed sums. To represent bit vectors, we use bold italic lowercase letters. For example, $\boldsymbol{x} \in \mathbb{F}_2^n$ denotes the vector $(x_0, \ldots, x_{n-1})$. We write $\mathbf{0}$ for the all-zero vector and $\boldsymbol{e}_i$ for the unit vector whose only nonzero coordinate is the ith one. For $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{F}_2^n$, we write $\boldsymbol{x} \preceq \boldsymbol{u}$ if $x_i \leq u_i$ for all i, with respect to the usual bit order $0 \leq 1$; this defines a partial order on $\mathbb{F}_2^n$. A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is uniquely represented by its algebraic normal form (ANF)

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}\, \boldsymbol{x}^{\boldsymbol{u}},$$

where $a_{\boldsymbol{u}} \in \mathbb{F}_2$ and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$. We also use the equivalent shorthand $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ for the monomial $\boldsymbol{x}^{\boldsymbol{u}}$. The coefficients satisfy

$$a_{\boldsymbol{u}} = \sum_{\boldsymbol{x} \preceq \boldsymbol{u}} f(\boldsymbol{x}).$$

Thus, ANF coefficients and integral sums over subcubes are two views of the same quantity.

A vectorial Boolean function is a function $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with coordinate functions in n variables. We write $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to \pi_{\boldsymbol{v}}(\boldsymbol{y})$ when the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ appears in the ANF of the product $\pi_{\boldsymbol{v}}(\boldsymbol{y})$, and we write $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \nrightarrow \pi_{\boldsymbol{v}}(\boldsymbol{y})$ otherwise. For a block cipher with block size n, we write its ciphertext bits as $c_0, \ldots, c_{n-1}$. The indices increase from the least significant bit of the full ciphertext block to the most significant bit. Thus, c_0 is the rightmost and c_{n-1} the leftmost ciphertext bit. An *output-bit observable*, or *degree-1 observable*, is a single ciphertext bit c_i. A *quadratic observable* is a product $c_i c_j$ with $i < j$.

3.2 Integral Cryptanalysis

In the bit-based integral cryptanalysis, one studies a set of inputs in which some coordinates vary over all possibilities while the remaining coordinates are fixed. For $\boldsymbol{u} \in \mathbb{F}_2^n$, we write

$$\mathbb{C}_{\boldsymbol{u}} = \{\boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{x} \preceq \boldsymbol{u}\}$$

for the canonical subcube where the support of $\boldsymbol{u}$ determines the active coordinates. An affine chosen set is obtained by fixing the inactive coordinates to arbitrary constants rather than all-zero constants. The correct algebraic object for such a structure is the corresponding derived function, obtained by substituting those fixed constants before analyzing ANF coefficients. In particular, bits fixed by the chosen structure remain constants throughout the model rather than additional free variables.

For a finite structure $\mathcal{S} \subseteq \mathbb{F}_2^n$ and a Boolean function f, we define the xor-sum of f over $\mathcal{S}$ by

$$\mathcal{I}_{\mathcal{S}}(f) = \sum_{\boldsymbol{x} \in \mathcal{S}} f(\boldsymbol{x}) \in \mathbb{F}_2.$$

Here the summation is taken in $\mathbb{F}_2$. The symbol $\mathcal{I}$ serves only as shorthand for this quantity. For a Boolean function f, the sum of f over $\mathbb{C}_{\boldsymbol{u}}$ is exactly the ANF coefficient $a_{\boldsymbol{u}}$. For an affine chosen set, the same statement holds after passing to the derived function: the sum over the affine set equals the coefficient of the monomial on the active coordinates in the derived ANF. Hence a standard way to prove that an output bit is balanced is to show that the relevant monomial is absent from the ANF of the corresponding coordinate function. This is the bridge between integral cryptanalysis and algebraic reasoning.

In a keyed primitive, the relevant output observable is a Boolean function in both public and secret variables. For an affine chosen set, the zero-sum or key-independent criterion must be applied to the derived function after the fixed public constants have been substituted. An exact zero-sum distinguisher follows when every monomial that contains all active public variables is absent in that derived function, regardless of the values of the secret variables. By contrast, a key-independent distinguisher only requires the absence of the same family of monomials with a nonconstant secret factor. This is the viewpoint that connects integral cryptanalysis to monomial prediction and the division property.

3.3 Monomial Prediction and Modeling

Monomial prediction was introduced by Hu et al. [7]. It determines the presence or absence of selected monomials in a vectorial Boolean function without expanding the full ANF. Assume the primitive is written as a composition of vectorial Boolean functions

$$\boldsymbol{f} = \boldsymbol{f}_r \circ \cdots \circ \boldsymbol{f}_1,$$

where $\boldsymbol{x}^{(i)} = \boldsymbol{f}_i(\boldsymbol{x}^{(i-1)})$ for $1 \leq i \leq r$. Then, monomial prediction works by checking monomial trails defined in Definition 1.

Definition 1 (Monomial trail, adapted from [7]). *The sequence*

$$\left(\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}), \ldots, \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})\right)$$

is an r-round monomial trail if

$$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}).$$

If at least one such trail exists, we write $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$*. Otherwise, we write* $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$*.*

The following criterion is the no-false-alarm direction of monomial prediction, adapted from [7].

Lemma 1 (Monomial-absence criterion). *If* $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$*, then* $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \nrightarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$*. In other words, the absence of a monomial trail guarantees the absence of the corresponding ANF monomial.*

Hu et al. further showed that the parity of the number of monomial trails gives exact monomial presence, but computing that parity is often more expensive than a simple existence test [7]. For integral cryptanalysis, the no-false-alarm direction above is already powerful because it certifies that selected ANF coefficients vanish.

For affine chosen structures, we specialize the derived-function viewpoint to the keyed setting of this paper. Let $\boldsymbol{\Gamma}^0, \boldsymbol{\Gamma}^1, \boldsymbol{\Gamma}^p \in \mathbb{F}_2^n$ be pairwise disjoint masks on the public-input coordinates whose supports partition those coordinates. They indicate the coordinates fixed to 0, fixed to 1, and active, respectively. The secret symbolic variables are the key bits, which we continue to denote by $\boldsymbol{k} \in \mathbb{F}_2^\kappa$. The chosen affine input structure is

$$X_0 = \{\boldsymbol{x} + \boldsymbol{\Gamma}^1 : \boldsymbol{x} \in \mathbb{C}_{\boldsymbol{\Gamma}^p}\}.$$

Equivalently, we obtain a derived observable g^d by substituting the fixed public coordinates into the original observable g. We may write

$$g^d(\boldsymbol{k}, \boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} \sum_{\boldsymbol{v} \in \mathbb{F}_2^\kappa} a_{\boldsymbol{u},\boldsymbol{v}}\, \pi_{\boldsymbol{u}}(\boldsymbol{x}) \pi_{\boldsymbol{v}}(\boldsymbol{k}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k})\, \pi_{\boldsymbol{u}}(\boldsymbol{x}),$$

where

$$a_{\boldsymbol{u}}(\boldsymbol{k}) = \sum_{\boldsymbol{v} \in \mathbb{F}_2^\kappa} a_{\boldsymbol{u},\boldsymbol{v}}\, \pi_{\boldsymbol{v}}(\boldsymbol{k}).$$

By the ANF-sum correspondence, $\mathcal{I}_{X_0}(g) = \mathcal{I}_{\mathbb{C}_{\boldsymbol{\Gamma}^p}}(g^d) = a_{\boldsymbol{\Gamma}^p}(\boldsymbol{k})$. Hence, we obtain a zero-sum property on X_0 when $a_{\boldsymbol{\Gamma}^p,\boldsymbol{v}} = 0$ for every $\boldsymbol{v}$. We get a key-independent integral property on X_0 when $a_{\boldsymbol{\Gamma}^p,\boldsymbol{v}} = 0$ for every nonzero $\boldsymbol{v}$. In the second case, the sum over X_0 is constant in $\mathbb{F}_2$, but it may be either 0 or 1.

Throughout this paper, we use *key-independent* in this standard integral sense: the sum over the chosen structure is constant in $\mathbb{F}_2$ and may equal either 0 or 1. Thus zero-sum is the special case in which that constant equals 0. The direct SAT/CP monomial-absence criterion above can certify either notion, provided that the boundary conditions for $\boldsymbol{\Gamma}^0$ and $\boldsymbol{\Gamma}^1$ are built into the model. However, the unresolved-support matrix, the left-kernel criterion, and the current implementation reported in this paper certify only zero-sum combinations. Accordingly, every kernel-based combination in the sequel should be read as zero-sum unless stated otherwise.

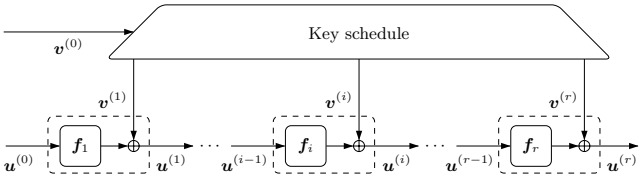

Fig. 1: Main variables in a SAT/CP monomial prediction model with key schedule, adapted from [12].

To model this in SAT or CP, we enforce the affine boundary conditions on the public input and then keep a state monomial and a key monomial at every round. At round i, the state side is described by $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)})$ and the round-key side by $\pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{k}^{(i)})$. The master-key monomial is $\pi_{\boldsymbol{v}^{(0)}}(\boldsymbol{k}^{(0)})$. The round-key monomials are linked through the key schedule and injected into the data path.

This model is useful because satisfiability has a direct cryptanalytic meaning. To test whether output bit c_i is zero-sum on $\mathbb{C}_{\boldsymbol{u}}$, we fix $\boldsymbol{u}^{(0)} = \boldsymbol{u}$ and $\boldsymbol{u}^{(r)} = \boldsymbol{e}_i$, while leaving $\boldsymbol{v}^{(0)}$ free. If the SAT/CP model is unsatisfiable, then no monomial $\pi_{\boldsymbol{v}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x})$ reaches c_i for any $\boldsymbol{v}$. By the monomial-absence criterion, $\mathcal{I}_{\mathbb{C}_{\boldsymbol{u}}}(c_i) = 0$ for every key. To test whether the same sum is key-independent but not necessarily zero, we add the constraint $\boldsymbol{v}^{(0)} \neq \boldsymbol{0}$. If the model is still unsatisfiable, then every coefficient with a nonconstant key monomial vanishes. Therefore the sum over $\mathbb{C}_{\boldsymbol{u}}$ is independent of the key, although it may equal 1 instead of 0. The same logic applies to higher degree observables. For a small nonlinear component $\boldsymbol{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ with output $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, its monomial prediction table (MPT) is the $2^m \times 2^n$ table whose entry $(\boldsymbol{u}, \boldsymbol{v})$ is 1 exactly when $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to \pi_{\boldsymbol{v}}(\boldsymbol{y})$. This representation is convenient for SAT and CP models because it turns local nonlinear propagation into a finite Boolean combination. For the analyzes in this paper, we generated the CP/SAT encodings for S-boxes and other small nonlinear Boolean functions with S-box Analyzer [19][5] also used in [20,21,22]. For a concrete PRESENT example of this modeling step, see Section A. This kept the models aligned with the underlying Boolean structure.

4 Methodology

In this section, we propose Split-and-Cancel, a method for finding integral distinguishers whose zero-sum property appears in combinations of ciphertext observables. The overview is shown in Figure 2: we write an r-round cipher as $E = E_q \circ E_p$, with a p-round prefix E_p and a q-round suffix E_q. We begin with a chosen plaintext structure $\mathcal{S}$ and an observable family $\mathcal{G} = \{g_0, \ldots, g_{t-1}\}$ on the ciphertext bits. Exact suffix expansions through E_q and prefix oracle queries through E_p identify unresolved middle monomials; canceling them yields certified zero-sum combinations of observables. We discuss two ways to extend these distinguishers to more rounds.

[5] `https://github.com/hadipourh/sboxanalyzer`

4.1 The Split-and-Cancel Method

We split the iterative block cipher of r rounds into a p-round prefix E_p and a q-round suffix E_q. The intermediate state after E_p is denoted by $\boldsymbol{x}$. Given the chosen structure $\mathcal{S}$, we build a prefix oracle for E_p, denoted by $\mathrm{POracle}_{E_p,\mathcal{S}}$. On input a boundary monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$, it uses monomial prediction to decide whether $\mathcal{I}_{\mathcal{S}}\big(\pi_{\boldsymbol{u}}(\boldsymbol{x})\big) = 0$ or must be left unresolved. On the E_q side, we derive the Boolean expression of each observable over the boundary state variables $\boldsymbol{x}$. Let $g_0, \ldots, g_{t-1}$ be the chosen output observables. After an exact backward expansion through the last q rounds, each observable has the form

$$g_i(\boldsymbol{x}, \boldsymbol{k}_q) = \sum_{m \in S_i} m(\boldsymbol{x}, \boldsymbol{k}_q),$$

where S_i is the set of monomials that actually appear in the suffix ANF of g_i and $\boldsymbol{k}_q$ denotes the suffix key variables involved in E_q. In the notation of Subsection 3.1, each such monomial can be written as $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{k}_q)$ for suitable exponent vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

For each appearing monomial $m = \pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{k}_q)$, we query the prefix oracle on its boundary-state part $\pi_{\boldsymbol{u}}(\boldsymbol{x})$. If the oracle certifies that $\mathcal{I}_{\mathcal{S}}\big(\pi_{\boldsymbol{u}}(\boldsymbol{x})\big) = 0$, then m is resolved and cannot affect the sum of g_i under the chosen structure, because the factor $\pi_{\boldsymbol{v}}(\boldsymbol{k}_q)$ is constant over $\mathcal{S}$. If the oracle returns '?', we keep m as unresolved. Operationally, this prefix check is an existence-only monomial prediction query. If no monomial trail reaches the boundary monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$, then the suffix monomial is discarded immediately. If at least one compatible trail survives, we do not try to determine its exact coefficient parity at this stage and instead keep the monomial in the unresolved pool.

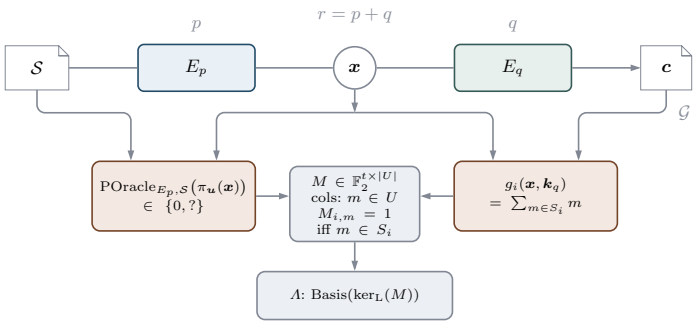

Fig. 2: Overview of the Split-and-Cancel method.

The results from the oracle queries of all monomials are arranged in a matrix over $\mathbb{F}_2$ called *incidence matrix*, defined as follows.

Definition 2 (Unresolved support and incidence matrix). *The unresolved support is*

$$U = \big\{\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{k}_q) \in \bigcup_{i=0}^{t-1} S_i : \mathcal{I}_{\mathcal{S}}\big(\pi_{\boldsymbol{u}}(\boldsymbol{x})\big) =?\big\}.$$

The incidence matrix of the chosen observables is the binary matrix $M \in \mathbb{F}_2^{t\times|U|}$ *whose rows are indexed by observables and whose columns are indexed by unresolved monomials, with*

$$M_{i,m} = \begin{cases} 1, & m \in S_i \cap U, \\ 0, & m \notin S_i \cap U. \end{cases}$$

The columns are the unresolved monomials that appear in at least one of the suffix ANFs. Each row shows the surviving monomials in the corresponding observable. Then we search linear combinations of rows that cancel all unresolved entries. The resulting combinations are zero-sum on the chosen structure. To extract the complete space of desired combinations, we compute the left kernel of the incidence matrix. Any left-kernel vector yields a certified zero-sum combination, as Lemma 2 explains.

Lemma 2 (Kernel criterion). *Let* M *be the incidence matrix defined above. For* $\lambda = (\lambda_0, \ldots, \lambda_{t-1}) \in \mathbb{F}_2^t$*, define the combined observable*

$$G_\lambda = \sum_{i=0}^{t-1} \lambda_i g_i.$$

If $\lambda M = 0$*, then* G_λ *is zero-sum on the chosen structure.*

Proof. Expand G_λ in ANF. Every unresolved monomial $m \in U$ appears with coefficient equal to the corresponding entry of λM. If $\lambda M = 0$, all unresolved monomials cancel. Any surviving monomial has the form $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{k}_q)$ with the prefix oracle certifying $\mathcal{I}_{\mathcal{S}}\big(\pi_{\boldsymbol{u}}(\boldsymbol{x})\big) = 0$. Since $\pi_{\boldsymbol{v}}(\boldsymbol{k}_q)$ is constant over $\mathcal{S}$, the full monomial also has zero sum on $\mathcal{S}$. Therefore $\mathcal{I}_{\mathcal{S}}(G_\lambda) = 0$.

Algorithm 1 briefly describes the Split-and-Cancel searching method. The bottleneck of this approach is the size of the unresolved support U. Choosing q too large makes exact suffix expansions expensive and greatly increases the size of U. Using observables with large degrees also enlarges U. Thus, this approach is most effective when q is small and the suffix ANF is sparse. Throughout the paper, we set $q = 1$. In that regime, the method remains exact on the suffix side, deterministic on the prefix side, and inexpensive enough to enumerate a large row family. In our applications to PRESENT and SIMON, moving from degree 1 to degree 2 is what unlocks the strongest new combinations. Using degree 3 for these ciphers does not give us better results. For GIFT-64, we use a limited number of degree-3 combinations spanned by the outputs of a particular S-box.

These parameter sets already suffice to expose new nonlinear combinations for the chosen ciphers. More generally, the method is well suited to binary-field block ciphers whose last round can be expressed by low-degree Boolean formulas.

Algorithm 1: Split-and-Cancel workflow

Input: A split $E = E_q \circ E_p$ with $r = p + q$, a chosen structure $\mathcal{S}$, and an observable family $\mathcal{G} = \{g_0, \ldots, g_{t-1}\}$

Output: A basis of certified zero-sum combinations in $\mathcal{G}$

$U \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $t - 1$ **do**
  $S_i \leftarrow \mathrm{SuffixSupport}_{E_q}(g_i)$;
  foreach $m = \pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{k}_q) \in S_i$ **do**
    if $\mathrm{POracle}_{E_p,\mathcal{S}}\big(\pi_{\boldsymbol{u}}(\boldsymbol{x})\big) =?$ **then**
      $U \leftarrow U \cup \{m\}$;
    end
  end
end
Construct the zero matrix $M \leftarrow 0 \in \mathbb{F}_2^{t \times |U|}$ with columns indexed by U;
for $i \leftarrow 0$ **to** $t - 1$ **do**
  foreach $m \in S_i \cap U$ **do**
    $M_{i,m} \leftarrow 1$;
  end
end
$\Lambda \leftarrow \mathrm{LeftKernelBasis}(M)$;
foreach $\lambda = (\lambda_0, \ldots, \lambda_{t-1}) \in \Lambda$ **do**
  $G_\lambda \leftarrow \sum_{i=0}^{t-1} \lambda_i g_i$;
end
return $\{G_\lambda : \lambda \in \Lambda\}$;

Remark 1. The method is sound but not complete. It only detects explicit cancellation of identical unresolved monomials. It does not detect deeper algebraic combinations among different unknown monomials. For sparse last rounds, this explicit cancellation already recovers combinations that bit-by-bit search misses.

4.2 From Candidate Verification to Kernel Extraction

Earlier split-search methods, including the neural-inspired follow-up search of Zhang et al. [10], typically start from one proposed output combination, expand it backward through the last q rounds, and then test each surviving suffix monomial with a prefix model. If every suffix monomial is ruled out, the candidate combination is zero-sum.

Our method contains this candidate-by-candidate verification as a special case. If the row set $\mathcal{G}$ contains only one observable, the matrix M has one row and Lemma 2 reduces to the usual direct verification condition. The row family $\mathcal{G}$ can contain many observables at once. Instead of testing one externally suggested combination, we keep all unresolved suffix monomials across the entire row family and solve one kernel problem to recover every provable cancellation inside that family. First, it removes the need for a heuristic or learned candidate-generation stage before verification begins. Second, it returns a basis of the full combination

space within the chosen observables, so it can strengthen a known distinguisher even when the round count itself does not increase.

4.3 Extend the Split-and-Cancel Method Further

The Split-and-Cancel method in previous subsections is precise but has some limitations in practice. To capture more complex combinations, one might take degrees higher than 3. However, the total numbers of higher-degree combinations increase exponentially. Furthermore, to handle the large number of monomial predictions, the precise monomial prediction by counting monomial trails cannot finish in a reasonable time. The practical approach of only checking the existence of monomial trails may overlook certain balanced monomials. Hence, the current method is limited by the actual computation capability.

Next, we propose a generic pullback criterion in Theorem 1 to push our distinguishers further. With a limited computation capability, one can only find distinguishers for a small number of rounds. Our theorem shows that these shorter-round combinations can still be pushed through the remaining rounds to obtain longer-round distinguishers. The proof of Theorem 1 is given in Section F.

Theorem 1 (Constant-sum pullback criterion). *Let the keyed permutation E be split as $E = E_2 \circ E_1$, where $E_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ denotes the first r_1 rounds and $E_2 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ the last r_2 rounds. Let $\mathcal{P} \subseteq \mathbb{F}_2^n$ be a chosen structure. For $\boldsymbol{p} \in \mathcal{P}$, write $\boldsymbol{x} = E_1(\boldsymbol{p})$ and $\boldsymbol{y} = E(\boldsymbol{p}) = E_2(\boldsymbol{x})$. For each fixed key, E_2^{-1} denotes the inverse of the keyed map E_2. Assume that separate analysis of E_1 yields Boolean functions $g_0(\boldsymbol{x}), g_1(\boldsymbol{x}), \ldots, g_{t-1}(\boldsymbol{x})$ and constants $\beta_0, \beta_1, \ldots, \beta_{t-1} \in \mathbb{F}_2$ such that, for every key,*

$$\sum_{\boldsymbol{p} \in \mathcal{P}} g_i(E_1(\boldsymbol{p})) = \beta_i, \qquad 0 \le i < t.$$

Rewrite each g_i through E_2^{-1}, so $g_i(E_2^{-1}(\boldsymbol{y}))$ becomes a Boolean function of the output variables $\boldsymbol{y}$ and the key variables $\boldsymbol{k}_2$ appearing in E_2. Suppose that among these rewritten expressions one can find an identity

$$\sum_{i=0}^{t-1} \lambda_i(\boldsymbol{k}_2) g_i(E_2^{-1}(\boldsymbol{y})) = h(\boldsymbol{y}) + \rho(\boldsymbol{k}_2)$$

where each $\lambda_i(\boldsymbol{k}_2)$ and $\rho(\boldsymbol{k}_2)$ depends only on $\boldsymbol{k}_2$, and $h(\boldsymbol{y})$ depends only on $\boldsymbol{y}$. Then, for every key,

$$\sum_{\boldsymbol{p} \in \mathcal{P}} h(E(\boldsymbol{p})) = (|\mathcal{P}| \bmod 2)\, \rho(\boldsymbol{k}_2) + \sum_{i=0}^{t-1} \lambda_i(\boldsymbol{k}_2)\, \beta_i.$$

In particular, if $|\mathcal{P}|$ is even, the term $\rho(\boldsymbol{k}_2)$ disappears from the xor-sum. Hence, if $|\mathcal{P}|$ is even and every $\beta_i = 0$, then h is zero-sum on $\mathcal{P}$. More generally, if the expression $(|\mathcal{P}| \bmod 2)\, \rho(\boldsymbol{k}_2) + \sum_{i=0}^{t-1} \lambda_i(\boldsymbol{k}_2)\, \beta_i$ is independent of $\boldsymbol{k}_2$, then h is key-independent on $\mathcal{P}$.

Phrased differently, the attacker first analyzes E_1 and obtains known r_1-round constant-sum combinations g_i. The extension step does not search an $r_1 + r_2$-round observable from scratch. Instead, it rewrites the already known r_1-round functions g_i through E_2^{-1}. It then searches among these rewritten expressions for an identity whose right-hand side has the form $h(\boldsymbol{y}) + \rho(\boldsymbol{k}_2)$. When such an identity exists, the theorem gives the xor-sum of $h(E(\boldsymbol{p}))$ over $\boldsymbol{p} \in \mathcal{P}$. Theorem 1 of Hadipour and Eichlseder [12] can be viewed as a cipher-specific special case of this criterion. This extension is cheap in computation. In the zero-sum setting produced by the kernel step in this paper, every β_i equals 0. Then it suffices to rewrite the already known zero-sum functions through E_2^{-1} and search for such a relation. In practice, this reduces to canceling all mixed monomials that involve both output variables and key variables, which can be done by computing the basis of the kernel, similar to Subsection 4.2. Any non-trivial observable in the resulting basis then implies a derived integral distinguisher. We want to emphasize that such an extension *does not replace* our previous Split-and-Cancel method, although it takes a similar approach. Instead, Theorem 1 can derive further observables *only when* the Split-and-Cancel method finds new results. By contrast, if some mixed output/key monomials cannot be cancelled, then the pulled-back expression remains key-dependent. This can be explained as weak-key distinguishers. For example, assume that the pulled-back expression contains a mixed monomial $y_0 k_0$. Then, setting k_0 to 0 and 1 gives two output observables that can be applied to the target cipher. By checking which observable gives the expected constant-sum, we know whether k_0 is 0 or 1, which identifies weak-key classes. Such results can also be viewed as probabilistic integral distinguishers, if we consider the behaviour under a randomly chosen key from the whole key space.

An analogous idea can extend distinguishers by one round from the input side, given in Theorem 2. For SIMON, Wang et al. [23] already used this idea by specifying the structure after one round and mapping it back to plaintexts by inverting the first round. A similar step also applies to the reduced-round SPECK analysis of [16]. We do not claim novelty, but isolate the common mechanism as a general theorem with explicit assumptions.

Theorem 2 (Generic one-round input extension under key xor). *Let E be a keyed round function on n state bits, having the form*

$$E_k(\boldsymbol{x}) = F(\boldsymbol{x}) + L(\boldsymbol{k}),$$

where $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a permutation independent of $\boldsymbol{k}$, and $L(\boldsymbol{k}) \in \mathbb{F}_2^n$ depends only on the round key. Let $\mathcal{S}$ be an affine chosen structure at the input of the next round, described by a set of active positions and arbitrary constants on the remaining positions. Assume that a given integral distinguisher depends only on that active/fixed pattern, in the sense that it remains valid for every choice of those fixed constants. Define

$$\mathcal{S}' = F^{-1}(\mathcal{S}).$$

Then for every round key $\boldsymbol{k}$, the image $E_k(\mathcal{S}')$ has the same active/fixed pattern as $\mathcal{S}$. Consequently, the same distinguisher extends one round backward through E_k.

Proof. By definition of $\mathcal{S}'$, we have $F(\mathcal{S}') = \mathcal{S}$. Hence for every round key k,

$$E_k(\mathcal{S}') = F(\mathcal{S}') + L(\boldsymbol{k}) = \mathcal{S} + L(\boldsymbol{k}).$$

Xor with the fixed vector $L(\boldsymbol{k})$ only changes the constant values on the inactive positions. It does not change which coordinates are active and which are fixed. Therefore $\mathcal{S} + L(\boldsymbol{k})$ has exactly the same active/fixed pattern as $\mathcal{S}$. Since the distinguisher is assumed to hold for every choice of the fixed constants compatible with that pattern, it also holds on $E_k(\mathcal{S}')$. Thus the distinguisher extends one round backward.

The assumption that the distinguisher tolerates arbitrary constants is essential. Without it, the xor offset $L(\boldsymbol{k})$ may move the structure to a different fixed-constant instance on which the distinguisher no longer holds. So Theorem 2 is a statement about active/fixed patterns, not about one specific choice of constants.

4.4 Toy Example

Example 1 (Combinations). Consider three observables having suffix ANFs

$$\begin{aligned} y_0 &= x_0 + k_0 + x_1x_2 + x_2x_3, \\ y_1 &= x_1 + k_1 + x_1x_2 + x_1x_3, \\ y_2 &= x_2 + k_2 + x_2x_3 + x_1x_3. \end{aligned}$$

Assume the forward prefix oracle certifies that x_0, x_1, x_2, k_0, k_1, and k_2 are zero-sum. It leaves an unresolved set $U = \{x_1x_2, x_2x_3, x_1x_3\}$. Using the column order $(x_0, x_1, x_2, k_0, k_1, k_2, x_1x_2, x_2x_3, x_1x_3)$, the full support matrix is

$$\widetilde{M} = \begin{array}{c} \\ y_0 \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccccccccc} x_0 & x_1 & x_2 & k_0 & k_1 & k_2 & x_1x_2 & x_2x_3 & x_1x_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

The prefix oracle removes the first six columns, and the incidence matrix becomes

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

The vector $\lambda = (1, 1, 1)$ satisfies $\lambda M = 0$. By Lemma 2, the combination

$$y_0 + y_1 + y_2 = x_0 + x_1 + x_2 + k_0 + k_1 + k_2$$

is zero-sum even though none of y_0, y_1, and y_2 is individually certified. This is exactly the phenomenon the matrix step is meant to recover.

Example 2 (Higher-degree observables and weak-key distinguishers). Let $m_1 = x_1x_2$ and $m_2 = x_3x_4$. Consider three output bits influenced by a post-whitening key, whose suffix expansions are

$$\begin{aligned} y_0 &= x_0 + m_1 + k_0, \\ y_1 &= x_5 + m_2 + k_1, \\ y_2 &= x_6 + x_0x_5 + x_0m_2 + x_5m_1 + m_1m_2 + k_2. \end{aligned}$$

Assume that the forward prefix oracle certifies that x_0, x_5, x_6, x_0x_5, x_0m_2, x_5m_1, k_0, k_1, k_2, and k_0k_1 are zero-sum. It leaves m_1, m_2, and m_1m_2 unresolved. At degree 1, the row family is $\{y_0, y_1, y_2\}$. After removing the certified zero-sum terms, its unresolved support is $U = \{m_1, m_2, m_1m_2\}$. The incidence matrix is

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Its left kernel is trivial, so no nonzero combination of y_0, y_1, and y_2 is certified at degree 1. At degree 2, we enlarge the row family with

$$y_0y_1 = x_0x_5 + x_0m_2 + x_0k_1 + x_5m_1 + m_1m_2 + m_1k_1 + x_5k_0 + m_2k_0 + k_0k_1,$$

The monomials x_0k_1 and x_5k_0 are zero-sum, because the state parts x_0 and x_5 are already certified. Then, the unresolved row of y_0y_1 is $(0, 0, 1, 1, 1)$. With column order $(m_1, m_2, m_1m_2, m_1k_1, m_2k_0)$, the augmented matrix becomes

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The combination $y_2 + y_0y_1$ reduces to $m_1k_1 + m_2k_0$ and the degree-2 combination is no longer key-independent. It reduces to a zero-sum only on the weak-key class $k_0 = k_1 = 0$.

Example 2 shows both why that richer row space can matter and how post-whitening can turn an exact combination into a weak-key one when using ciphertext products. The degree-2 search used later applies the same matrix test to a richer row space. This is how the nonlinear PRESENT combinations emerge.

Example 3 (Pulling back an old cancellation). Assume that Example 1 has already produced the following zero-sum combination for r rounds.

$$G_\lambda = x_0 + x_1 + x_2$$

where x_0, x_1, and x_2 denote the three relevant boundary bits after those r rounds. Let E_2 denote the added last round, and write $\boldsymbol{y} = E_2(\boldsymbol{x})$ for the final output

variables. Consider two output observables $h_0(\boldsymbol{y})$ and $h_1(\boldsymbol{y})$ whose expressions in the boundary variables are

$$h_0(E_2(\boldsymbol{x})) = x_0 + k_3 x_0, \qquad h_1(E_2(\boldsymbol{x})) = x_1 + x_2 + k_3 x_1 + k_3 x_2.$$

Suppose we now try to find a $(r+1)$-round zero-sum combination by the direct split search. With column order $(x_0, x_1, x_2, k_3 x_0, k_3 x_1, k_3 x_2)$, the resulting incidence matrix is

$$M_{\text{dir}} = \begin{pmatrix} 1\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,1\,1 \end{pmatrix}.$$

Its left kernel is trivial, so this direct $r+1$ search returns no combination. Nevertheless,

$$h_0(E_2(\boldsymbol{x})) + h_1(E_2(\boldsymbol{x})) = (1 + k_3)(x_0 + x_1 + x_2) = (1 + k_3) G_\lambda.$$

So the expression of $h_0 + h_1$ in the boundary variables is a key-dependent multiple of the already known r-round combination. This matches Theorem 1 with E_1 equal to the original r rounds, E_2 equal to the added round, $g_0(\boldsymbol{x}) = G_\lambda$, $\lambda_0(k_3) = 1 + k_3$, and $\rho(k_3) = 0$. Since G_λ is zero-sum, the theorem shows that $h_0 + h_1$ is also zero-sum for every key.

4.5 Discussion on Probability Estimates

In Subsection 4.3, we mentioned potential integral distinguishers that are non-deterministic, while [10] found some probabilistic integral distinguishers. In this subsection, we discuss the problem of estimating the probabilities in this type of distinguishers. The relevant probability space is the joint choice of the key and the fixed part of the plaintext structure.

Without loss of generality, we assume that the last round key is added at the end of the reduced-round cipher. Denote the state variables before the key addition by $\boldsymbol{x}$ and the round key by $\boldsymbol{k}$. Then, the ciphertext bits are $\boldsymbol{c} = \boldsymbol{x} + \boldsymbol{k}$. Suppose we have found a key-independent combination $f(\boldsymbol{x})$ when ignoring $\boldsymbol{k}$ and a set $G = \{g_0, g_1, \ldots, g_{t-1}\}$ of previously known key-independent combinations. Write $g_i(\boldsymbol{x})$ for the corresponding expression in the state variables before the last key addition. After substituting $\boldsymbol{x}$ by $\boldsymbol{x} + \boldsymbol{k}$ in $f(\boldsymbol{x})$, we have $f(\boldsymbol{x} + \boldsymbol{k}) = f(\boldsymbol{x}) + r(\boldsymbol{x}, \boldsymbol{k}) + f(\boldsymbol{k})$ where $r(\boldsymbol{x}, \boldsymbol{k})$ contains all monomials mixing $\boldsymbol{x}$ and $\boldsymbol{k}$. $r(\boldsymbol{x}, \boldsymbol{k})$ can be further reduced by taking residual over the ideal $\langle f(\boldsymbol{x}), g_0(\boldsymbol{x}), g_1(\boldsymbol{x}), \ldots, g_{t-1}(\boldsymbol{x}) \rangle$. To simplify explanation, we assume that $r(\boldsymbol{x}, \boldsymbol{k})$ is the final reduced expression having the form

$$r(\boldsymbol{x}, \boldsymbol{k}) = \sum_{i=0}^{m} \boldsymbol{x}^{\boldsymbol{u}_i} \boldsymbol{k}^{\boldsymbol{v}_i}$$

where the total number of monomials m, the exponents $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are determined by the above residual computation.

Consider combinations of degrees no larger than 2 for PRESENT. All monomials in the expression $r(\boldsymbol{x}, \boldsymbol{k})$ then have the form $x_i k_j$. When Algorithm 4 in

[10] is applied to such a simplified residual, its estimate depends only on the involved key-index set K, and

$$\widehat{p}_{\mathrm{Alg4}}(|K|) = 2^{-|K|} + \left(1 - 2^{-|K|}\right) \cdot \frac{1}{2} = \frac{1}{2} + 2^{-(|K|+1)}.$$

This rule uses two assumptions. First, the involved key bits are treated as independent uniform bits. Second, whenever the key does not force the residual to vanish identically, the remaining expression is treated as balanced and therefore zero with probability $1/2$. Because the rule keeps only $|K|$, it does not capture algebraic structure in the residual. For the example $x_0k_0 + x_1k_1$, Algorithm 4 in [10] gives $0.25 + 0.75 \times 0.5 = 0.625$. However, for $x_0k_0 + x_0k_1 = x_0(k_0 + k_1)$, the same rule still gives 0.625, while the grouped form shows that the natural heuristic value is 0.75.

We conduct experiments to assess the validity of these assumptions. For PRESENT and GIFT, we run probabilistic integral attacks on small plaintext structures. We then compare the empirical probabilities with the estimates from [10]. The complete results are given in Section D and Section E. Two observations are particularly relevant here. First, for combinations with nontrivial key-side structure, there is usually a significant difference between the empirical result and the estimate from [10]. Second, the remaining discrepancy is most visible when the prefix is short. For small p, the balanced-residual assumption can fail because the surviving state-side expressions still have structural dependencies inherited from the shallow prefix. In that regime, the event $r(\boldsymbol{x}, \boldsymbol{k}) = 0$ need not occur with probability exactly $1/2$ once some coefficient $h_i(\boldsymbol{k})$ is nonzero. For example, the empirical probability of the combination $c_0 + c_{32} + c_{16}c_{48}$ in PRESENT changes from 0.70 at $(p, q) = (5, 1)$ to 0.66 at $(6, 1)$ and 0.63 at $(7, 1)$, approaching the estimated value $5/8$. For GIFT, the empirical probabilities in Section E show the same trend as p increases. These observations suggest that the independence model becomes more accurate for longer prefixes. For short prefixes, however, part of the heuristic model does not hold exactly. Developing a more accurate probability estimate that captures these short-prefix dependencies and remains reliable for more complex combinations is left for future work.

5 Applications

We apply the proposed method to PRESENT, GIFT, SIMON, SIMECK, and SPECK. These well-known ciphers cover most common designs such as SPN, Feistel, and ARX.

5.1 PRESENT

PRESENT is the clearest place to see the main message of the paper. The best published integral distinguisher reaches 9 rounds [5,14,13,11]. The chosen observable family contains all 64 ciphertext bits and all pairwise products. The specification used in the paper is summarized in Subsection B.1.

First, we search distinguishers by using $p = 8$ and $q = 1$, giving results in Table 4. The chosen structure with 60 active bits enlarge the known space of integral distinguishers, where we find a nonlinear combination $(c_{32} + c_0c_{16} + c_0c_{48})$. Then, we remove the final key addition and discover more combinations. By restoring key addition in these additional combinations, we obtain several weak-key distinguishers. Most of them work on a 2^{-2} fraction of the final round key. The required conditions on this final round key are given in Table 4.

Next, we take $p = 9$ and $q = 1$, and consider the model without the final key addition. This parameter set gives us many new results, listed in Table 5. The results reveal a 10-round quadratic combination. By restoring the final key addition, we obtain a 10-round weak-key integral distinguisher with data complexity 2^{60}. The plaintext structure `fffffffffffffe` yields another set of weak-key combinations with data complexity 2^{63}. The longest runtime of these searches was at $(p, q) = (8, 1)$, which took 16.4 minutes.

5.2 GIFT

For GIFT-64, the best published integral distinguisher reaches 9 rounds and is typically reported as 10 rounds after the trivial one-round input-side extension [14]. The specification of GIFT is summarized in Subsection B.2. We study nonlinear output combinations for GIFT-64 at $(p, q) = (8, 1)$ and $(9, 1)$ and for GIFT-128 at $(p, q) = (10, 1)$ and $(11, 1)$. Our search uses observables of degree up to 3, but the included degree-3 observables are restricted to those whose factors come from the same last-round S-box output group. This keeps the search manageable. Hence, we do not claim to recover every kernel basis vector.

For GIFT-64, we obtain 8 zero-sum ciphertext bits, shown as the first row of Table 6. By searching under the key-omitting model, we find more nonlinear combinations. Two examples are listed in the table. The first nonlinear example has degree 2 and, after restoring the last key addition, is a weak-key distinguisher for 2^{-1} portion of the final round key. We demonstrate the effectiveness of higher degree in the second example. Its degree is 3 and gives a zero-sum for 2^{-2} portion of the final round key. When moving to $(p, q) = (9, 1)$, we can only find zero-sum combinations of degree 3 in the key-omitting model. We search under 3 different plaintext structures, and each of them gives degree-3 combinations for 2^{-2} portion of the final round key. Table 7 shows three example combinations. Finally, our best result yields a 11-round weak-key distinguisher with data complexity 2^{62} for 2^{-2} portion of the final round key.

The same degree-3 search is also applied to GIFT-128. Using $(p, q) = (10, 1)$, we obtain 30 zero-sum ciphertext bit and 32 nonlinear combinations that are unconditionally zero-sum. Table 8 lists the results, where we provide one example of the nonlinear combinations. By using the key-omitting model and restoring the round key later, we obtain two more families of nonlinear combinations for weak-key distinguishers. Same as Table 6, we show an example for degree 2 and another for degree 3. The degree-2 combination is zero-sum for 2^{-1} portion of the final round key, while the degree-3 one works for 2^{-2} portion. For $(p, q) = (11, 1)$, we find 30 and 31 degree-3 combinations for two different plaintext structures.

Both of them are zero-sum for 2^{-2} portion of the final round key and have the same data complexity. Finally, by using the extension on the input side, our best result is a 13-round weak-key distinguisher. The longest runtime was the GIFT-128 computation at $(p, q) = (10, 1)$ and degree 3, which took 78.9 minutes.

5.3 SIMON

SIMON is the family where the split view scales most cleanly with the block size. A single backward round has sparse ANF. Each new left-half bit contains one copied right-half bit, one rotated linear term, one quadratic AND term, and one key bit. This is the ideal setting for the unresolved-support matrix. The specification is recalled in Subsection B.3.

Due to the one-round extension on the input side [23], the total number of rounds of our distinguisher is always $1 + p + q$. All experiments on SIMON take $q = 1$ and degree 2. For SIMON-32, we use the corresponding independent-round-key test and require a nonzero prefix-side round-key mask in the SAT monomial-prediction model. This excludes the all-zero prefix-side key assignment and lets us search for key-independent combinations that may sum to 1 rather than 0. In Table 10, we report the structure mask, data complexity, and core rounds; each row also admits the standard one-round input-side extension.

Against the published works, our SIMON results improve the best integral distinguishers by one round at every standard block size. For SIMON-32, we recover the known 14 (15)-round zero-sum space and, with the one-sum search above, obtain a 15 (16)-round key-independent space of dimension 3. For SIMON-48, SIMON-64, SIMON-96, and SIMON-128, we obtain zero-sum nonlinear combinations. The longest runtime was the SIMON-128 at $(p, q) = (25, 1)$, which took 11.7 minutes.

The regularity of the basis vectors is also notable. For the smaller variants, the combinations form long arithmetic families that mirror the Feistel geometry. This makes the kernel output easy to inspect and suggests that the extra round is structural rather than accidental.

5.4 SIMECK

SIMECK is a good target to see whether the same explanation transfers automatically from SIMON. It uses a SIMON-like round function with different rotation constants. The specification is summarized in Subsection B.4.

Using $q = 1$, we obtain 16, 19, and 22-round distinguishers for SIMECK-32, SIMECK-48, and SIMECK-64 with data 2^{31}, 2^{47}, and 2^{63}, respectively. These extend by one round after the standard input-side extension the best published bit-based integral distinguishers reported by Xiang et al. [5] for all three standard block sizes. In the experiments, no linear or quadratic observable is individually zero-sum. The improved distinguishers appear only after combining observables, giving basis dimensions 7, 5, and 5, respectively. The longest runtime was the SIMECK-64 at $(p, q) = (20, 1)$, which took 0.34 minutes.

Compared with SIMON, the SIMECK kernels are smaller, but the mechanism is the same: repeated unresolved support across different rows creates exact cancellations that no individual row exposes. A broader SIMECK search over other q values, higher observable degrees, and alternative key models remains an important direction for future work.

5.5 SPECK

SPECK is a typical target in ARX ciphers. Its round function contains modular addition, so even a one-round suffix can grow quickly when the chosen ciphertext observables reach deep carry chains. For this reason, we carefully choose a family of observables over part of the output bits rather than an exhaustive one.

In the SPECK experiments, we fix $p = 5$, $q = 1$, and degree to 2. On the E_q part, we keep only the lowest 14 output bits of each ciphertext word. Our results are listed in Table 12. By Theorem 2, every 6-round result in Table 12 immediately yields a 7-round input-extended distinguisher. Our discoveries improve the best published SPECK integral distinguishers by one round at every standard block size. Additionally, we significantly reduce the data complexity of the best found integral distinguishers of SPECK.

When a 6-round combination admits an exact one-round inverse rewrite, we list the resulting 7-round combination in Table 13. In that table, c'_i denotes a 6-round output bit and c_i denotes a 7-round output bit. Rows labeled "deterministic" give key-independent 7-round ciphertext observables. Rows labeled "weak-key" contain a mixed term multiplied by k_0; imposing $k_0 = 0$ removes that term and yields a 7-round ciphertext observable on half of the key space. We do not count one-round pullback combinations with nonlinear key dependence in the reported 7-round dimensions. The absolute low-data SPECK-48 row `fffffffff11f` has no basis vector that admits such a pullback to 7 rounds, but the nearby mask `fffffffff31f` does. Additionally, each displayed 7-round result can be extended by one round on the input side, giving 8 rounds in Table 1.

5.6 Discussion on Applications to Key Recovery

In this subsection, we discuss about future applications to key recovery attacks, though this work focuses on distinguishers. By showing the benefits from our non-linear combinations, we want to provide some potential ideas for future works on this direction.

Our SPECK results provide concrete examples of how extending integral distinguishers can benefit key recovery. The selected distinguishers have lower data complexity than previous results while covering one additional round. Their balanced output expressions also include linear combinations of bits, such as $c_i + c_j$. Here, c_i and c_j denote bits at the output of the distinguisher, which the attacker recovers by partially decrypting the ciphertexts. The attacker needs to check $\mathcal{I}_{\mathcal{S}}(c_i + c_j) = 0$, which is equivalent to checking $\mathcal{I}_{\mathcal{S}}(c_i) = \mathcal{I}_{\mathcal{S}}(c_j)$. Using the meet-in-the-middle technique, the attacker can compute two summations separately, guessing only the key information required by each computation, and

then match their values. The matching candidates must agree on shared key information and satisfy the relevant key schedule constraints. Thus, the attacker does not need to evaluate $c_i + c_j$ for every combined key guess during partial decryption. If each separate computation requires an amount of key information comparable to that needed for the previous single-bit test, the expensive computations can remain comparable in cost. The overall benefit must be established through a complete attack analysis in future works.

Another benefit is that quadratic balanced expressions can strengthen filtering during key recovery. Suppose that a search restricted to individual output bits identifies two balanced bits, c_i and c_j, which are outputs of the same S-box. The attacker then checks $\mathcal{I}_{\mathcal{S}}(c_i) = 0$ and $\mathcal{I}_{\mathcal{S}}(c_j) = 0$. Suppose that our extension method additionally establishes that $c_i c_j$ is balanced. An attacker who already evaluates the relevant S-box and obtains both bits for each text under the same key guess can also compute their product without guessing any additional key information. The attacker can therefore check three conditions instead of two: $\mathcal{I}_{\mathcal{S}}(c_i) = 0$, $\mathcal{I}_{\mathcal{S}}(c_j) = 0$, and $\mathcal{I}_{\mathcal{S}}(c_i c_j) = 0$. The product must be accumulated before the relevant information is discarded, since its sum cannot generally be reconstructed from the two separate sums. This additional condition gives a stronger filter if it provides information beyond the existing tests. Under independent, uniform wrong-key test results, the survival probability decreases from 2^{-2} to 2^{-3} per structure. If the original attack requires several independent structures, the stronger filter may achieve the same wrong-key rejection with fewer structures and hence less total data. It does not reduce the size of each individual structure. The benefit is therefore additional filtering without additional key guesses under the stated conditions, while accounting for any extra computation or storage.

6 Conclusion and Future Works

In this work, we extended integral distinguishers by considering combinations of ciphertext bits. We proposed the Split-and-Cancel method, which combines forward monomial filtering on the prefix with exact suffix expansion. By canceling unresolved monomials through a kernel computation, we obtained new integral distinguishers that are visible only through ciphertext-bit combinations. For SIMON, SIMECK, and SPECK, our best distinguishers improve the previous integral results by one round at every standard block size. For PRESENT, we obtained 10-round weak-key integral distinguishers with data complexities 2^{60} and 2^{63}. For GIFT-64 and GIFT-128, we found new distinguishers in the ordinary and weak-key settings. After the standard input-side extension, these results yield 10- and 11-round distinguishers for GIFT-64 and 12- and 13-round distinguishers for GIFT-128. These findings suggest that combination-based observables provide a useful extension of the classical integral toolkit.

The Split-and-Cancel method is not limited to the ciphers studied in this paper, and applying it to other symmetric primitives is a natural direction for future work. It would be interesting to study whether higher-degree observables

lead to further improvements. In our implementations, the p-side oracle checks only the existence of monomial trails. This choice keeps the oracle cheaper, but the trail counting may reveal additional balanced properties.

Acknowledgments

This work began at ASK 2026, and we thank the organizers for making that collaboration possible. Hosein Hadipour is supported by ERC project 101097056 (SYMTRUST). This research was funded in part by the Austrian Science Fund (FWF) SFB project SPyCoDe (10.55776/F85). This research was also supported by the ELLIIT program.

References

1. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In *FSE 1997*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.
2. Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In *FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.
3. Yosuke Todo. Structural evaluation by generalized integral property. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.
4. Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016.
5. Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 648–678, 2016.
6. Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset – improved cube attacks against Trivium and Grain-128AEAD. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 466–495. Springer, 2020.
7. Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 446–476. Springer, 2020.
8. Phil Hebborn, Baptiste Lambin, Gregor Leander, and Yosuke Todo. Strong and tight security guarantees against integral distinguishers. In *ASIACRYPT 2021*, volume 13090 of *LNCS*, pages 362–391. Springer, 2021.
9. Patrick Derbez and Pierre-Alain Fouque. Increasing precision of division property. *IACR Trans. Symmetric Cryptol.*, 2020(4):173–194, 2020.
10. Liu Zhang, Yiran Yao, Danping Shi, Dongchen Chai, Jian Guo, and Zilong Wang. Neural-inspired advances in integral cryptanalysis. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026*, volume 16546 of *LNCS*, pages 451–481. Springer, 2026.
11. Yunjae Hwang, Insung Kim, Sunyeop Kim, Myung-Kyu Lee, Hanbeom Shin, Deukjo Hong, Seokhie Hong, Dongjae Lee, Jaechul Sung, and Byoungjin Seok. Improving neural-inspired integral distinguishers via a linear-algebraic approach. *IACR Cryptol. ePrint Arch.*, 2026:340, 2026.

12. Hosein Hadipour and Maria Eichlseder. Integral cryptanalysis of WARP based on monomial prediction. *IACR Trans. Symmetric Cryptol.*, 2022(2):92–112, 2022.
13. Senpeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. Milp-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 398–427. Springer, 2019.
14. Zahra Eskandari, Andreas Brasen Kidmose, Stefan Kölbl, and Tyge Tiessen. Finding integral distinguishers with ease. In *SAC 2018*, volume 11349 of *LNCS*, pages 115–138. Springer, 2018.
15. Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 128–157. Springer, 2017.
16. Senpeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. Exploring secret keys in searching integral distinguishers based on division property. *IACR Trans. Symmetric Cryptol.*, 2020(3):288–304, 2020.
17. Tim Beyne and Michiel Verbauwhede. Integral cryptanalysis using algebraic transition matrices. *IACR Trans. Symmetric Cryptol.*, 2023(4):244–269, 2023.
18. Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, 2017.
19. Hosein Hadipour, Marcel Nageler, and Maria Eichlseder. Throwing boomerangs into feistel structures application to CLEFIA, WARP, LBlock, LBlock-s and TWINE. *IACR Trans. Symmetric Cryptol.*, 2022(3):271–302, 2022.
20. Hosein Hadipour, Patrick Derbez, and Maria Eichlseder. Revisiting differential-linear attacks via a boomerang perspective with application to aes, ascon, clefia, skinny, present, knot, twine, warp, lblock, simeck, and SERPENT. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024*, LNCS, pages 38–72. Springer, 2024.
21. Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14007 of *LNCS*, pages 128–157. Springer, 2023.
22. Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible differential and zero-correlation attacks application to ascon, forkskinny, skinny, mantis, PRESENT and qarmav2. *IACR Trans. Symmetric Cryptol.*, 2024(1):234–325, 2024.
23. Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of reduced-round SIMON32 and SIMON48. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 143–160. Springer, 2014.
24. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.
25. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *DAC 2015*, pages 175:1–175:6. ACM, 2015.

26. Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 307–329. Springer, 2015.

A Modeling S-boxes and Nonlinear Components with S-box Analyzer

S-box Analyzer [19] is a Python package for generating compact local models of S-boxes and other small nonlinear Boolean components; it can be installed with `pip install sboxanalyzer` and has also been used in [20,21,22]. The PRESENT S-box gives a concrete example of this workflow. Listing 1.1 shows the S-box Analyzer calls that return the minimized CNF, MILP, and CP encodings of the PRESENT monomial prediction table. In the implementation, the CNF clauses are then translated into the local variable numbering of each PRESENT nibble and appended to the SAT model.

```
from sboxanalyzer import SboxAnalyzer
from sage.crypto.sboxes import PRESENT as PRESENT_SBOX

sa = SboxAnalyzer(PRESENT_SBOX)
mpt = sa.monomial_prediction_table()
cnf, milp, cp = sa.minimized_integral_constraints(mode=6)

print(mpt)
print(cnf)    # SAT/CNF encoding
print(milp)   # MILP inequalities
print(cp)     # CP constraints
```

Listing 1.1: Extracting minimized PRESENT MPT constraints with S-box Analyzer.

B Cipher Specifications

B.1 PRESENT

PRESENT is a 64-bit substitution-permutation network with a round-key addition, a parallel 4-bit S-box layer (S-box specification in Table 2), and a bit permutation [24]. We write PRESENT without a key-size suffix because the distinguisher model depends only on the round function. The round structure is shown in Figure 3.

Table 2: PRESENT S-box.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	c	5	6	b	9	0	a	d	3	e	f	8	4	7	1	2

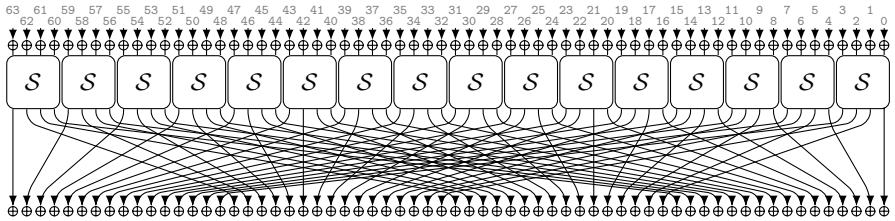

Fig. 3: One PRESENT round in the bit ordering used for the hexadecimal mask notation.

B.2 GIFT

GIFT is a substitution-permutation network specified with a 64-bit and 128-bit bloksize version. We draw the 64-bit round layout below because it fixes the nibble ordering used by the hexadecimal mask notation. The same local S-box layer and partial round-key pattern also apply to GIFT-128. One round consits of a parallel 4-bit S-box layer (S-box specification in Table 3), a bit permutation and a partial round-key addition. We show this structure in Figure 4.

Table 3: GIFT S-box.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

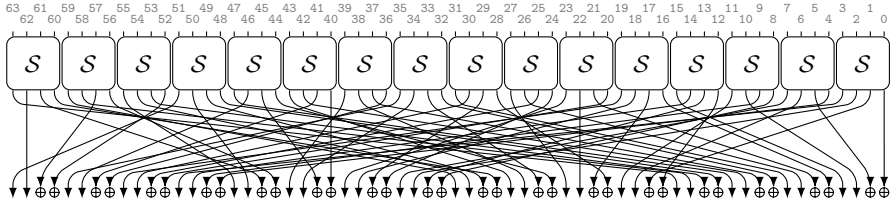

Fig. 4: One GIFT round in the bit ordering used for the hexadecimal mask notation. The XOR symbols indicate the round-key additions.

B.3 SIMON

SIMON is a balanced Feistel network whose round function updates

$$(L, R) \mapsto (R + f(L) + k,\ L)$$

with

$$f(L) = (L \lll 1 \,\&L \lll 8) + (L \lll 2)$$

for the standard variants studied in this paper [25]. Throughout the paper, SIMON-n denotes the family of SIMON variants with block size n. The key size is suppressed because the reported distinguishers depend only on the round function. We use the convention that $L[0]$ (resp. $R[0]$) is the least significant bit (LSB) of L (resp. R). Thus for an output (L, R) with w-bit words, we treat R as the low word and L as the high word, so the flattened order is

$$c_0 = R[0],\ c_1 = R[1],\ \ldots,\ c_{w-1} = R[w-1],\ c_w = L[0],\ \ldots,\ c_{2w-1} = L[w-1].$$

Hence in SIMON tables and combinations, c_0 is the least significant bit of the whole output state and c_{n-1} its most significant bit. The round structure is shown in Figure 5.

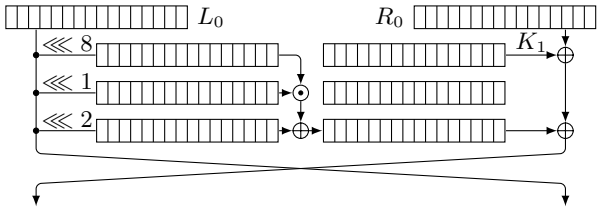

Fig. 5: The SIMON-32 round function.

B.4 SIMECK

SIMECK is a SIMON-like Feistel cipher with the update

$$(L, R) \mapsto (R + f(L) + k,\ L)$$

and

$$f(L) = (L \,\&\, L \lll 5) + (L \lll 1)$$

under the parameter sets considered in [26]. Throughout the paper, SIMECK-n denotes the family of SIMECK variants with block size n. The key size is suppressed since our distinguishers are not affected by the key size. The round structure is shown in Figure 6. The flattened ciphertext indexing is the same as for SIMON.

B.5 SPECK

SPECK is a family of ARX ciphers operating on two words (X_1, X_2) [25]. The state gets updated by

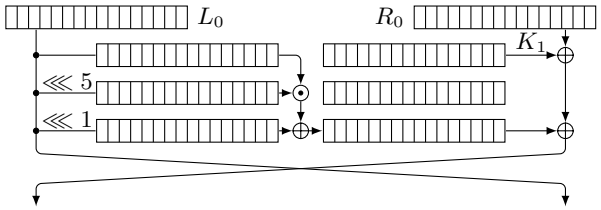

Fig. 6: The SIMECK-32 round function.

$$X_1^{(i+1)} = \left((X_1^{(i)} \ggg \alpha) \boxplus X_2^{(i)}\right) + k^{(i)}, \qquad X_2^{(i+1)} = (X_2^{(i)} \lll \beta) + X_1^{(i+1)}.$$

Throughout the paper, SPECK-n denotes the family of SPECK variants with block size n and word size $n/2$. The rotation constants are $\alpha = 7$ and $\beta = 2$ for SPECK-32 and $\alpha = 8$ and $\beta = 3$ for the other variants. Its flattened ciphertext bits follow the same whole-state convention as SIMON and SIMECK. The round structure is shown in Figure 7.

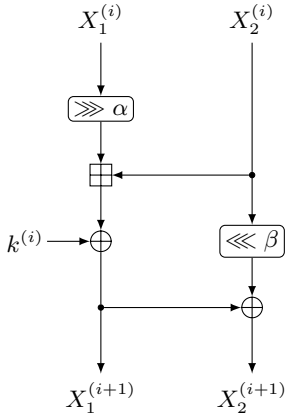

Fig. 7: The SPECK round function.

C Complete Results of Integral Distinguishers

In this section, we present our integral distinguishing properties for the ciphers PRESENT, GIFT, SIMON, SIMECK, and SPECK.

C.1 PRESENT

Table 4 show the $p = 8$, $q = 1$ results of PRESENT and Table 5 show the $p = 9$, $q = 1$ results without final key addition.

Table 4: Results of PRESENT. $p = 8$, $q = 1$, and input structure `ffff...fff0`.

Representative combination	Key condition	Portion
c_0	none	1
$c_{16} + c_{48}$	none	1
$c_0c_{16} + c_{16}c_{32}$	$k_{16} = 0,\ k_0 = k_{32}$	2^{-2}
$c_{32} + c_{16}c_{48}$	$k_{16} = k_{48} = 0$	2^{-2}
$c_{32} + c_0c_{16} + c_0c_{48}$	none	1
$1 \le i < 16$: $c_i + c_{i+32} + c_{i+16}c_{i+48}$	$k_{i+16} = k_{i+48} = 0$	2^{-2}
$i \in \{4, 8, 12\}$: $c_i + c_{i+32} + c_ic_{i+16} + c_ic_{i+48}$	$k_i = 0$, $k_{i+16} = k_{i+48}$	2^{-2}
$c_{16} + c_{32} + c_0c_{16} + c_{32}c_{48}$	$k_0 = k_{32} = k_{48} = 0$	2^{-3}
$i \in \{20, 24, 28\}$: $c_i + c_{i+16} + c_ic_{i+16} + c_{i+16}c_{i+32}$	$k_{i+16} = 0$, $k_i = k_{i+32}$	2^{-2}
$i \in \{1, 2, 3, 6, 10, 14\}$: $c_i + c_{i+16} + c_ic_{i+16} + c_ic_{i+48} + c_{i+16}c_{i+32} + c_{i+32}c_{i+48}$	$k_i = k_{i+32}$, $k_{i+16} = k_{i+48}$	2^{-2}

Table 5: Results of PRESENT. $p = 9$, $q = 1$, and no final key addition.

Structure mask	Representative combination	Key condition	Portion
`fffffffffffffff0`	$c_0 + c_{32} + c_{16}c_{48}$	$k_{16} = k_{48} = 0$	2^{-2}
`fffffffffffffffe`	$0 \le i < 16$: $c_i + c_{i+32} + c_{i+16}c_{i+48}$	$k_{i+16} = k_{i+48} = 0$	2^{-2}
`fffffffffffffffe`	$i \in \{0, 4, 8, 12\}$: $c_i + c_{i+16} + c_ic_{i+16} + c_ic_{i+48} + c_{i+16}c_{i+32} + c_{i+32}c_{i+48}$	$k_i = k_{i+32}$ $k_{i+16} = k_{i+48}$	2^{-2}

C.2 GIFT

Table 6 show the $p = 8$, $q = 1$ results of GIFT-64 and Table 7 show the $p = 9$, $q = 1$ results without final key addition. We also apply our method to $p = 10$, $q = 1$ (see Table 8) and $p = 10$, $q = 1$ (see Table 9) of GIFT-128.

Table 6: Results of GIFT-64. $p = 8$, $q = 1$, and input structure `ffff...fff8`.

Representative combination	Portion
c_i for all $i \in \{4, 20, 24, 28, 32, 36, 48, 56\}$	1
$c_{18} + c_1c_{35}$	2^{-1}
$c_{35} + c_1c_{18} + c_1c_{48} + c_{18}c_{35} + c_{18}c_{48} + c_1c_{18}c_{48}$	2^{-2}

Table 7: Results of GIFT-64. $p = 9$, $q = 1$, and no final key addition.

Structure mask	Representative combination	Portion
`fffffffffffffff6`	$c_6 + c_{23} + c_{36} + c_{53} + c_6c_{36} + c_6c_{53} + c_{23}c_{36}c_{53}$	2^{-2}
`fffffffffffffffe`	$c_6 + c_{23} + c_{36} + c_{53} + c_6c_{36} + c_6c_{53} + c_{23}c_{36}c_{53}$	2^{-2}
`fffffffffffffffb`	$c_3 + c_{16} + c_{33} + c_{50} + c_{16}c_{50} + c_{33}c_{50} + c_3c_{16}c_{33}$	2^{-2}

Table 8: Results of GIFT-128. $p = 10$, $q = 1$, and input structure `ffff...fffb`.

Representative combination	Portion
c_i for all $i \in \{i \in 4\mathbb{Z} \mid 0 \leq i \leq 124,\ i \notin \{28, 40\}\}$	1
$c_{33} + c_{66} + c_0c_{99}$	1
$c_{34} + c_1c_{67}$	2^{-1}
$c_{99} + c_0c_{33} + c_0c_{66} + c_{33}c_{66} + c_{66}c_{99} + c_0c_{33}c_{66}$	2^{-2}

C.3 SIMON

Table 10 show our results on various variants of SIMON.

C.4 SIMECK

Table 11 show our results on various variants of SIMECK.

Table 9: Results of GIFT-128. $p = 11$, $q = 1$, and no final key addition.

Structure mask	Representative combination	Portion
`ffff...fffb`	$c_{11} + c_{40} + c_{73} + c_{106} + c_{40}c_{106} + c_{73}c_{106} + c_{11}c_{40}c_{73}$	2^{-2}
`ffff...fff7`	$c_7+c_{36}+c_{69}+c_{102}+c_{36}c_{69}+c_{36}c_{102}+c_{69}c_{102}+c_7c_{36}c_{69}$	2^{-2}

Table 10: SIMON degree-2 kernel results. The Rounds column lists the core $p+q$ rounds; each row also admits the standard one-round input-side extension.

Variant	Structure mask	Data	Kernel dim.	Single-bit	Rounds	Representative combination
SIMON-32	`7fffffff`	2^{31}	16	none	14	$c_{16} + c_{14} + c_8c_{15}$
	`7fffffff`	2^{31}	3	none	15	$c_{16} + c_{14} + c_8c_{15}$
SIMON-48	`7fffffffffff`	2^{47}	24	none	16	$c_{24} + c_{22} + c_{16}c_{23}$
SIMON-64	`7fffffffffffffff`	2^{63}	22	none	18	$c_{32} + c_{30} + c_{24}c_{31}$
SIMON-96	`fff7ffffffffffffffffffff`	2^{95}	5	none	22	$c_{90} + c_{40} + c_{34}c_{41}$
SIMON-128	`fff7ffffffffffffffffffffffffffff`	2^{127}	3	none	26	$c_{117} + c_{51} + c_{45}c_{52}$

Table 11: SIMECK degree-2 kernel results. The Rounds column lists the core $p+q$ rounds; each row also admits the standard one-round input-side extension.

Variant	Structure mask	Data	Kernel dim.	Single-bit	Rounds	Representative combination
SIMECK-32	`7fffffff`	2^{31}	7	none	15	$c_{31} + c_{14} + c_{10}c_{15}$
SIMECK-48	`7fffffffffff`	2^{47}	5	none	18	$c_{47} + c_{22} + c_{18}c_{23}$
SIMECK-64	`7fffffffffffffff`	2^{63}	5	none	21	$c_{63} + c_{30} + c_{26}c_{31}$

C.5 SPECK

Table 12 show our results on various variants of SPECK. We extend these distinguisher by one round in Table 13.

Table 12: SPECK degree-2 kernel results for $p = 5$, $q = 1$.

Variant	Structure mask	Data	Kernel dim.	Single-bit	Rounds	Representative 6-round combination
SPECK-32	`ffffff1f`	2^{29}	11	none	6	$c_2 + c_{18}$
	`ffffff9f`	2^{30}	20	none	6	$c_2 + c_{18}$
SPECK-48	`fffffffff1f`	2^{45}	27	c_{24}	6	$c_3 + c_{27}$
	`ffffffff11f`	2^{42}	1	none	6	$c_6 + c_{30}$
	`ffffffff31f`	2^{43}	4	none	6	$c_8 + c_{32}$
SPECK-64	`fffffffffffff1f`	2^{61}	49	c_{32}	6	$c_3 + c_{35}$
	`fffffffffff1001f`	2^{50}	6	none	6	$c_6 + c_{38}$
SPECK-96	`ffffffffffffffffffffe9f`	2^{93}	45	none	6	$c_6 + c_{54}$
	`ffffffffffffff10000001f`	2^{66}	45	none	6	$c_6 + c_{54}$
SPECK-128	`ffffffffffffffffffffffffffffff1f`	2^{125}	48	c_{64}	6	$c_3 + c_{67}$
	`ffffffffffffffffff100000000001f`	2^{82}	45	none	6	$c_6 + c_{70}$

Table 13: Representative 7-round SPECK combinations obtained by one-round pullback of the 6-round combinations in Table 12. Here c'_i denotes a 6-round output bit and c_i denotes a 7-round output bit. In deterministic rows, the linear key terms k_i cancel in the structure sum. In weak-key rows, imposing $k_0 = 0$ removes the mixed term.

Variant	Mask	6-round source	Representative 7-round combination	Type	Condition
SPECK-32	`ffffff1f`	$\ell_0 \leftarrow c'_7 + c'_{23}$	$\ell_0 := c_2 + c_{18} + c_9 + c_{25} + c_{16} + k_0$	deterministic	none
	`ffffff9f`	$\ell_0 \leftarrow c'_7 + c'_{23}$	$\ell_0 := c_2 + c_{18} + c_9 + c_{25} + c_{16} + k_0$ $\ell_1 := c_2 + c_{18} + c_3 + c_{19} + c_{10} + c_{17} + c_{26}$	deterministic	none
	`ffffff9f`	$\ell_1 \leftarrow c'_8 + c'_{24}$	$+c_2c_{16} + c_{16}c_{18} + k_0(c_2 + c_{18}) + k_1$	weak-key	$k_0 = 0$
SPECK-48	`fffffffff1f`	$\ell_0 \leftarrow c'_8 + c'_{32}$	$\ell_0 := c_{24} + c_3 + c_{27} + c_{11} + c_{35} + k_0$ $\ell_1 := c_3 + c_{27} + c_4 + c_{28} + c_{12} + c_{25} + c_{36}$	deterministic	none
	`fffffffff1f`	$\ell_1 \leftarrow c'_9 + c'_{33}$	$+c_3c_{24} + c_{24}c_{27} + k_0(c_3 + c_{27}) + k_1$	weak-key	$k_0 = 0$
	`ffffffff31f`	$\ell_0 \leftarrow c'_8 + c'_{32}$	$\ell_0 := c_{24} + c_3 + c_{27} + c_{11} + c_{35} + k_0$	deterministic	none
SPECK-64	`fffffffffffff1f`	$\ell_0 \leftarrow c'_8 + c'_{40}$	$\ell_0 := c_{32} + c_3 + c_{35} + c_{11} + c_{43} + k_0$ $\ell_1 := c_3 + c_{35} + c_4 + c_{36} + c_{12} + c_{44}$	deterministic	none
	`fffffffffffff1f`	$\ell_1 \leftarrow c'_9 + c'_{41}$	$+c_{33} + c_3c_{32} + c_{32}c_{35} + k_0(c_3 + c_{35}) + k_1$	weak-key	$k_0 = 0$
	`fffffffffff1001f`	$\ell_0 \leftarrow c'_8 + c'_{40}$	$\ell_0 := c_3 + c_{11} + c_{32} + c_{35} + c_{43} + k_0$	deterministic	none
SPECK-96	`ffffffffffffffffffffe9f`	$\ell_0 \leftarrow c'_8 + c'_{56}$	$\ell_0 := c_{48} + c_3 + c_{51} + c_{11} + c_{59} + k_0$ $\ell_1 := c_{12} + c_{60} + c_3 + c_{51} + c_4 + c_{49} + c_{52}$	deterministic	none
	`ffffffffffffffffffffe9f`	$\ell_1 \leftarrow c'_9 + c'_{57}$	$+c_3c_{48} + c_{48}c_{51} + k_0(c_3 + c_{51}) + k_1$	weak-key	$k_0 = 0$
	`fffffffffffffff10000001f`	$\ell_0 \leftarrow c'_8 + c'_{56}$	$\ell_0 := c_3 + c_{11} + c_{48} + c_{51} + c_{59} + k_0$ $\ell_1 := c_3 + c_4 + c_{12} + c_{49} + c_{51} + c_{52} + c_{60} + k_1$	deterministic	none
	`fffffffffffff10000001f`	$\ell_1 \leftarrow c'_9 + c'_{57}$	$+c_3c_{48} + c_{48}c_{51} + k_0(c_3 + c_{51})$	weak-key	$k_0 = 0$
SPECK-128	`ffffffffffffffffffffffffffffffff1f`	$\ell_0 \leftarrow c'_8 + c'_{72}$	$\ell_0 := c_{64} + c_3 + c_{67} + c_{11} + c_{75} + k_0$ $\ell_1 := c_3 + c_{67} + c_4 + c_{68} + c_{12} + c_{76}$	deterministic	none
	`ffffffffffffffffffffffffffffff1f`	$\ell_1 \leftarrow c'_9 + c'_{73}$	$+c_{65} + c_3c_{64} + c_{64}c_{67} + k_0(c_3 + c_{67}) + k_1$	weak-key	$k_0 = 0$
	`ffffffffffffffffff100000000001f`	$\ell_0 \leftarrow c'_8 + c'_{72}$	$\ell_0 := c_3 + c_{11} + c_{64} + c_{67} + c_{75} + k_0$ $\ell_1 := c_3 + c_4 + c_{12} + c_{65} + c_{67} + c_{68} + c_{76} + k_1$	deterministic	none
	`ffffffffffffffffff100000000001f`	$\ell_1 \leftarrow c'_9 + c'_{73}$	$+c_3c_{64} + c_{64}c_{67} + k_0(c_3 + c_{67})$	weak-key	$k_0 = 0$

D Probabilistic Integral Distinguishers of PRESENT

In this appendix, we provide the complete results of the analysis in Subsection 4.5. We take PRESENT as the target cipher and choose plaintext structures with reasonable sizes to keep the experiments feasible.

We choose (p, q) to be $(4, 1)$, $(5, 1)$, $(6, 1)$, and $(7, 1)$ to cover different p values. The corresponding input-structure masks and data complexities are stated in the table captions. For $(p, q) = (5, 1)$, we also include an alternative run with a smaller structure. We run the experiment with each parameter set 5000 times, where each of them takes a fresh random master key and a fresh random value for the constant part of the plaintext structure.

All results are listed in Table 14, Table 15, Table 16, Table 17, and Table 18. Each table contains the combinations, the numbers of involved key bits, the estimates based on Algorithm 4 in [10], and the observed probabilities in experiments. The residual forms of the expressions are also provided in a simpler way to specifically show how the key bits are involved. In these residual expressions, the variable y_i denotes an expression of only intermediate state variables before the final key addition. All displayed floating-point values are rounded to two decimal places. The shaded rows have absolute gaps at least 0.25 between the estimate from Algorithm 4 in [10] and the empirical probability.

D.1 Split $p = 4$, $q = 1$.

Table 14: Probabilistic integral distinguishers of PRESENT for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\|K\|$	[10]	Emp. p
c_0	0	1.00	1.00
c_1	0	1.00	1.00
c_2	0	1.00	1.00
c_3	0	1.00	1.00
$c_{16} + c_{48}$	0	1.00	1.00
$c_{17} + c_{49}$	0	1.00	1.00
$c_{18} + c_{50}$	0	1.00	1.00
$c_{19} + c_{51}$	0	1.00	1.00
$c_0c_{16} + c_{16}c_{32}$ *Residual:* $y_{16}(k_0 + k_{32}) + y_{32}k_{16} = 0$	3	0.56	0.74
$c_{32} + c_{16}c_{48}$ *Residual:* $y_{16}(k_{16} + k_{48}) = 0$	2	0.63	0.82
$c_1c_{17} + c_{17}c_{33}$ *Residual:* $y_{17}(k_1 + k_{33}) + y_{33}k_{17} = 0$	3	0.56	0.77
$c_{33} + c_{17}c_{49}$ *Residual:* $y_{17}(k_{17} + k_{49}) = 0$	2	0.63	0.82
$c_2c_{18} + c_{18}c_{34}$ *Residual:* $y_{18}(k_2 + k_{34}) + y_{34}k_{18} = 0$	3	0.56	0.90
$c_{34} + c_{18}c_{50}$ *Residual:* $y_{18}(k_{18} + k_{50}) = 0$	2	0.63	0.92
$c_3c_{19} + c_{19}c_{35}$ *Residual:* $y_{19}(k_3 + k_{35}) + y_{35}k_{19} = 0$	3	0.56	0.75
$c_{35} + c_{19}c_{51}$ *Residual:* $y_{19}(k_{19} + k_{51}) = 0$	2	0.63	0.83
$c_{32} + c_0c_{16} + c_0c_{48}$ *Residual:* $k_0(y_{16} + y_{48}) + y_0(k_{16} + k_{48}) = 0$. Since $y_0 = 0$ and $y_{16} + y_{48} = 0$.	0	1.00	1.00
$c_{33} + c_1c_{17} + c_1c_{49}$ *Residual:* $k_1(y_{17} + y_{49}) + y_1(k_{17} + k_{49}) = 0$. Since $y_1 = 0$ and $y_{17} + y_{49} = 0$.	0	1.00	1.00
$c_{34} + c_2c_{18} + c_2c_{50}$ *Residual:* $k_2(y_{18} + y_{50}) + y_2(k_{18} + k_{50}) = 0$. Since $y_2 = 0$ and $y_{18} + y_{50} = 0$.	0	1.00	1.00
$c_{35} + c_3c_{19} + c_3c_{51}$ *Residual:* $k_3(y_{19} + y_{51}) + y_3(k_{19} + k_{51}) = 0$. Since $y_3 = 0$ and $y_{19} + y_{51} = 0$.	0	1.00	1.00
$c_4 + c_{36} + c_{20}c_{52}$ *Residual:* $y_{20}k_{52} + y_{52}k_{20} = 0$	2	0.63	0.64
$c_5 + c_{37} + c_{21}c_{53}$ *Residual:* $y_{21}k_{53} + y_{53}k_{21} = 0$	2	0.63	0.74
$c_6 + c_{38} + c_{22}c_{54}$ *Residual:* $y_{22}k_{54} + y_{54}k_{22} = 0$	2	0.63	0.88
$c_7 + c_{39} + c_{23}c_{55}$ *Residual:* $y_{23}k_{55} + y_{55}k_{23} = 0$	2	0.63	0.72

Continued on next page

Table 14: Probabilistic integral distinguishers of PRESENT for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\|K\|$	[10]	Emp. p
$c_8 + c_{40} + c_{24}c_{56}$ *Residual:* $y_{24}k_{56} + y_{56}k_{24} = 0$	2	0.63	0.64
$c_9 + c_{41} + c_{25}c_{57}$ *Residual:* $y_{25}k_{57} + y_{57}k_{25} = 0$	2	0.63	0.73
$c_{10} + c_{42} + c_{26}c_{58}$ *Residual:* $y_{26}k_{58} + y_{58}k_{26} = 0$	2	0.63	0.88
$c_{11} + c_{43} + c_{27}c_{59}$ *Residual:* $y_{27}k_{59} + y_{59}k_{27} = 0$	2	0.63	0.72
$c_{12} + c_{44} + c_{28}c_{60}$ *Residual:* $y_{28}k_{60} + y_{60}k_{28} = 0$	2	0.63	0.65
$c_{13} + c_{45} + c_{29}c_{61}$ *Residual:* $y_{29}k_{61} + y_{61}k_{29} = 0$	2	0.63	0.73
$c_{14} + c_{46} + c_{30}c_{62}$ *Residual:* $y_{30}k_{62} + y_{62}k_{30} = 0$	2	0.63	0.89
$c_{15} + c_{47} + c_{31}c_{63}$ *Residual:* $y_{31}k_{63} + y_{63}k_{31} = 0$	2	0.63	0.73
$c_{16} + c_{32} + c_0c_{16} + c_{32}c_{48}$ *Residual:* $y_{16}(k_0 + k_{32}) + y_{32}k_{48} = 0$	3	0.56	0.74
$c_{17} + c_{33} + c_1c_{17} + c_{33}c_{49}$ *Residual:* $y_{17}(k_1 + k_{33}) + y_{33}k_{49} = 0$	3	0.56	0.76
$c_{18} + c_{34} + c_2c_{18} + c_{34}c_{50}$ *Residual:* $y_{18}(k_2 + k_{34}) + y_{34}k_{50} = 0$	3	0.56	0.90
$c_{19} + c_{35} + c_3c_{19} + c_{35}c_{51}$ *Residual:* $y_{19}(k_3 + k_{35}) + y_{35}k_{51} = 0$	3	0.56	0.76
$c_4 + c_{20} + c_4c_{20} + c_4c_{52} + c_{20}c_{36} + c_{36}c_{52}$ *Residual:* $(y_4 + y_{36})(k_{20} + k_{52}) + (y_{20} + y_{52})(k_4 + k_{36}) = 0$	4	0.53	0.64
$c_5 + c_{21} + c_5c_{21} + c_5c_{53} + c_{21}c_{37} + c_{37}c_{53}$ *Residual:* $(y_5 + y_{37})(k_{21} + k_{53}) + (y_{21} + y_{53})(k_5 + k_{37}) = 0$	4	0.53	0.72
$c_6 + c_{22} + c_6c_{22} + c_6c_{54} + c_{22}c_{38} + c_{38}c_{54}$ *Residual:* $(y_6 + y_{38})(k_{22} + k_{54}) + (y_{22} + y_{54})(k_6 + k_{38}) = 0$	4	0.53	0.88
$c_7 + c_{23} + c_7c_{23} + c_7c_{55} + c_{23}c_{39} + c_{39}c_{55}$ *Residual:* $(y_7 + y_{39})(k_{23} + k_{55}) + (y_{23} + y_{55})(k_7 + k_{39}) = 0$	4	0.53	0.73
$c_8 + c_{24} + c_8c_{24} + c_8c_{56} + c_{24}c_{40} + c_{40}c_{56}$ *Residual:* $(y_8 + y_{40})(k_{24} + k_{56}) + (y_{24} + y_{56})(k_8 + k_{40}) = 0$	4	0.53	0.64
$c_9 + c_{25} + c_9c_{25} + c_9c_{57} + c_{25}c_{41} + c_{41}c_{57}$ *Residual:* $(y_9 + y_{41})(k_{25} + k_{57}) + (y_{25} + y_{57})(k_9 + k_{41}) = 0$	4	0.53	0.73
$c_{10} + c_{26} + c_{10}c_{26} + c_{10}c_{58} + c_{26}c_{42} + c_{42}c_{58}$ *Residual:* $(y_{10} + y_{42})(k_{26} + k_{58}) + (y_{26} + y_{58})(k_{10} + k_{42}) = 0$	4	0.53	0.88
$c_{11} + c_{27} + c_{11}c_{27} + c_{11}c_{59} + c_{27}c_{43} + c_{43}c_{59}$ *Residual:* $(y_{11} + y_{43})(k_{27} + k_{59}) + (y_{27} + y_{59})(k_{11} + k_{43}) = 0$	4	0.53	0.73
$c_{12} + c_{28} + c_{12}c_{28} + c_{12}c_{60} + c_{28}c_{44} + c_{44}c_{60}$ *Residual:* $(y_{12} + y_{44})(k_{28} + k_{60}) + (y_{28} + y_{60})(k_{12} + k_{44}) = 0$	4	0.53	0.65
$c_{13} + c_{29} + c_{13}c_{29} + c_{13}c_{61} + c_{29}c_{45} + c_{45}c_{61}$ *Residual:* $(y_{13} + y_{45})(k_{29} + k_{61}) + (y_{29} + y_{61})(k_{13} + k_{45}) = 0$	4	0.53	0.73
$c_{14} + c_{30} + c_{14}c_{30} + c_{14}c_{62} + c_{30}c_{46} + c_{46}c_{62}$	4	0.53	0.89

Continued on next page

Table 14: Probabilistic integral distinguishers of PRESENT for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\|K\|$	[10]	Emp. p
Residual: $(y_{14} + y_{46})(k_{30} + k_{62}) + (y_{30} + y_{62})(k_{14} + k_{46}) = 0$			
$c_{15} + c_{31} + c_{15}c_{31} + c_{15}c_{63} + c_{31}c_{47} + c_{47}c_{63}$ *Residual:* $(y_{15} + y_{47})(k_{31} + k_{63}) + (y_{31} + y_{63})(k_{15} + k_{47}) = 0$	4	0.53	0.73

Table 15: Probabilistic integral distinguishers of PRESENT for $p = 5$, $q = 1$, and input-structure mask `ff00000000000000` (data complexity: 2^8).

Combination	$\|K\|$	[10]	Emp. p
$c_0 + c_{32} + c_{16}c_{48}$ *Residual:* $y_{16}k_{48} + y_{48}k_{16} = 0$	2	0.63	0.70
$c_4 + c_{36} + c_{20}c_{52}$ *Residual:* $y_{20}k_{52} + y_{52}k_{20} = 0$	2	0.63	0.64
$c_8 + c_{40} + c_{24}c_{56}$ *Residual:* $y_{24}k_{56} + y_{56}k_{24} = 0$	2	0.63	0.61
$c_{12} + c_{44} + c_{28}c_{60}$ *Residual:* $y_{28}k_{60} + y_{60}k_{28} = 0$	2	0.63	0.63
$c_0 + c_{16} + c_0c_{16} + c_0c_{48} + c_{16}c_{32} + c_{32}c_{48}$ *Residual:* $(y_0 + y_{32})(k_{16} + k_{48}) + (y_{16} + y_{48})(k_0 + k_{32}) = 0$	4	0.53	0.71
$c_4 + c_{20} + c_4c_{20} + c_4c_{52} + c_{20}c_{36} + c_{36}c_{52}$ *Residual:* $(y_4 + y_{36})(k_{20} + k_{52}) + (y_{20} + y_{52})(k_4 + k_{36}) = 0$	4	0.53	0.62
$c_8 + c_{24} + c_8c_{24} + c_8c_{56} + c_{24}c_{40} + c_{40}c_{56}$ *Residual:* $(y_8 + y_{40})(k_{24} + k_{56}) + (y_{24} + y_{56})(k_8 + k_{40}) = 0$	4	0.53	0.63
$c_{12} + c_{28} + c_{12}c_{28} + c_{12}c_{60} + c_{28}c_{44} + c_{44}c_{60}$ *Residual:* $(y_{12} + y_{44})(k_{28} + k_{60}) + (y_{28} + y_{60})(k_{12} + k_{44}) = 0$	4	0.53	0.64

Table 16: Probabilistic integral distinguishers of PRESENT for $p = 5$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\|K\|$	[10]	Emp. p
$c_0 + c_{32} + c_{16}c_{48}$ *Residual:* $y_{16}k_{48} + y_{48}k_{16} = 0$	2	0.63	0.67
$c_0 + c_{16} + c_0c_{16} + c_0c_{48} + c_{16}c_{32} + c_{32}c_{48}$ *Residual:* $(y_0 + y_{32})(k_{16} + k_{48}) + (y_{16} + y_{48})(k_0 + k_{32}) = 0$	4	0.53	0.69

Table 17: Probabilistic integral distinguishers of PRESENT for $p = 6$, $q = 1$, and input-structure mask `fff0000000000000` (data complexity: 2^{12}).

Combination	$\|K\|$	[10]	Emp. p
$c_0 + c_{32} + c_{16}c_{48}$ *Residual:* $y_{16}k_{48} + y_{48}k_{16} = 0$	2	0.63	0.66
$c_1 + c_{33} + c_{17}c_{49}$	2	0.63	0.62

Continued on next page

Table 17: Probabilistic integral distinguishers of PRESENT for $p = 6$, $q = 1$, and input-structure mask `fff0000000000000` (data complexity: 2^{12}).

Combination	$\|K\|$	[10]	Emp. p
Residual: $y_{17}k_{49} + y_{49}k_{17} = 0$			
$c_2 + c_{34} + c_{18}c_{50}$ *Residual:* $y_{18}k_{50} + y_{50}k_{18} = 0$	2	0.63	0.63
$c_3 + c_{35} + c_{19}c_{51}$ *Residual:* $y_{19}k_{51} + y_{51}k_{19} = 0$	2	0.63	0.62
$c_4 + c_{36} + c_{20}c_{52}$ *Residual:* $y_{20}k_{52} + y_{52}k_{20} = 0$	2	0.63	0.62
$c_8 + c_{40} + c_{24}c_{56}$ *Residual:* $y_{24}k_{56} + y_{56}k_{24} = 0$	2	0.63	0.61
$c_{12} + c_{44} + c_{28}c_{60}$ *Residual:* $y_{28}k_{60} + y_{60}k_{28} = 0$	2	0.63	0.62
$c_0 + c_{16} + c_0c_{16} + c_0c_{48} + c_{16}c_{32} + c_{32}c_{48}$ *Residual:* $(y_0 + y_{32})(k_{16} + k_{48}) + (y_{16} + y_{48})(k_0 + k_{32}) = 0$	4	0.53	0.66

D.2 Split $p = 7$, $q = 1$.

Table 18: Probabilistic integral distinguishers of PRESENT for $p = 7$, $q = 1$, and input-structure mask `ffff000000000000` (data complexity: 2^{16}).

Combination	$\|K\|$	[10]	Emp. p
$c_0 + c_{32} + c_{16}c_{48}$ *Residual:* $y_{16}k_{48} + y_{48}k_{16} = 0$	2	0.63	0.63

E Probabilistic Integral Distinguishers of GIFT

In this appendix, we apply the analysis in Subsection 4.5 to the GIFT-64 cipher. The experiments are similar to those in Section D.

We choose (p, q) to be $(4, 1)$, $(5, 1)$, and $(6, 1)$ to cover different p values. The corresponding input-structure masks and data complexities are stated in the table captions. The considered observables include all 64 output bits, all quadratic products c_ic_j for $0 \le i < j < 64$, and the selected degree-3 observables formed from one output S-box. We again run each parameter set 5000 times, each time with a fresh random master key and a fresh random value for the constant part of the plaintext structure.

All results are listed in Table 19, Table 20, and Table 21. Each table contains the combinations, the numbers of involved key bits, the estimates based on Algorithm 4 in [10], and the observed probabilities in experiments. The residual forms of the expressions are also provided in a simpler way to show how the key bits are involved. In these residual expressions, the variables y_i and $y_{i,j}$ denote expressions involving only intermediate state variables before the final key addition. All displayed floating-point values are rounded to two decimal places.

Table 19: Probabilistic integral distinguishers of GIFT-64 for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\lvert K\rvert$	[10]	Emp. p
c_0	0	1.00	1.00
c_4	0	1.00	1.00
c_8	0	1.00	1.00
c_{12}	0	1.00	1.00
c_{16}	0	1.00	1.00
c_{20}	0	1.00	1.00
c_{24}	0	1.00	1.00
c_{28}	0	1.00	1.00
c_{32}	0	1.00	1.00
c_{36}	0	1.00	1.00
c_{40}	0	1.00	1.00
c_{44}	0	1.00	1.00
c_{48}	0	1.00	1.00
c_{52}	0	1.00	1.00
c_{56}	0	1.00	1.00
c_{60}	0	1.00	1.00
$c_{18} + c_1c_{35}$ *Residual:* $y_{35}k_1 = 0$	1	0.75	0.96
$c_{22} + c_5c_{39}$ *Residual:* $y_{39}k_5 = 0$	1	0.75	0.95
$c_{26} + c_9c_{43}$ *Residual:* $y_{43}k_9 = 0$	1	0.75	0.93
$c_{30} + c_{13}c_{47}$ *Residual:* $y_{47}k_{13} = 0$	1	0.75	0.97
$c_{34} + c_{17}c_{51}$ *Residual:* $y_{51}k_{17} = 0$	1	0.75	0.96
$c_{38} + c_{21}c_{55}$ *Residual:* $y_{55}k_{21} = 0$	1	0.75	0.95
$c_6 + c_{23}c_{53}$ *Residual:* $y_{23}k_{53} = 0$	1	0.75	0.96
$c_{42} + c_{25}c_{59}$ *Residual:* $y_{59}k_{25} = 0$	1	0.75	0.92
$c_{10} + c_{27}c_{57}$ *Residual:* $y_{27}k_{57} = 0$	1	0.75	0.94
$c_{14} + c_{31}c_{61}$ *Residual:* $y_{31}k_{61} = 0$	1	0.75	0.96
$c_{17} + c_{34} + c_0c_{51}$ *Residual:* $y_{51}k_0 = 0$	1	0.75	0.96
$c_{33} + c_{50} + c_3c_{16}$ *Residual:* $y_3k_{16} = 0$	1	0.75	0.96
$c_{33} + c_{50} + c_3c_{33}$ *Residual:* $y_3k_{33} = 0$	1	0.75	0.96

Continued on next page

Table 19: Probabilistic integral distinguishers of GIFT-64 for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\lvert K\rvert$	[10]	Emp. p
$c_{21} + c_{38} + c_4c_{55}$ *Residual:* $y_{55}k_4 = 0$	1	0.75	0.95
$c_{37} + c_{54} + c_7c_{20}$ *Residual:* $y_7k_{20} = 0$	1	0.75	0.94
$c_{37} + c_{54} + c_7c_{37}$ *Residual:* $y_7k_{37} = 0$	1	0.75	0.94
$c_{25} + c_{42} + c_8c_{59}$ *Residual:* $y_{59}k_8 = 0$	1	0.75	0.92
$c_{41} + c_{58} + c_{11}c_{24}$ *Residual:* $y_{11}k_{24} = 0$	1	0.75	0.93
$c_{41} + c_{58} + c_{11}c_{41}$ *Residual:* $y_{11}k_{41} = 0$	1	0.75	0.93
$c_{29} + c_{46} + c_{12}c_{63}$ *Residual:* $y_{63}k_{12} = 0$	1	0.75	0.96
$c_{45} + c_{62} + c_{15}c_{28}$ *Residual:* $y_{15}k_{28} = 0$	1	0.75	0.95
$c_{45} + c_{62} + c_{15}c_{45}$ *Residual:* $y_{15}k_{45} = 0$	1	0.75	0.95
$c_2 + c_{49} + c_{19}c_{32}$ *Residual:* $y_{19}k_{32} = 0$	1	0.75	0.96
$c_2 + c_{49} + c_{19}c_{49}$ *Residual:* $y_{19}k_{49} = 0$	1	0.75	0.96
$c_6 + c_{53} + c_{23}c_{36}$ *Residual:* $y_{23}k_{36} = 0$	1	0.75	0.95
$c_{10} + c_{57} + c_{27}c_{40}$ *Residual:* $y_{27}k_{40} = 0$	1	0.75	0.94
$c_{29} + c_{46} + c_{29}c_{63}$ *Residual:* $y_{63}k_{29} = 0$	1	0.75	0.96
$c_{14} + c_{61} + c_{31}c_{44}$ *Residual:* $y_{31}k_{44} = 0$	1	0.75	0.96
$c_1 + c_{18} + c_{35}c_{48}$ *Residual:* $y_{35}k_{48} = 0$	1	0.75	0.96
$c_5 + c_{22} + c_{39}c_{52}$ *Residual:* $y_{39}k_{52} = 0$	1	0.75	0.95
$c_9 + c_{26} + c_{43}c_{56}$ *Residual:* $y_{43}k_{56} = 0$	1	0.75	0.94
$c_{13} + c_{30} + c_{47}c_{60}$ *Residual:* $y_{47}k_{60} = 0$	1	0.75	0.97
$c_{51} + c_0c_{17} + c_0c_{34} + c_{17}c_{34} + c_{34}c_{51} + c_0c_{17}c_{34}$ *Residual:* $(y_{17} + y_{34} + y_{17,34})k_0$ $+ (y_{34} + y_{0,34})k_{17} + (1 + y_{34})k_0k_{17} = 0$	2	0.63	0.88
$c_{35} + c_1c_{18} + c_1c_{48} + c_{18}c_{35} + c_{18}c_{48} + c_1c_{18}c_{48}$ *Residual:* $(y_{18} + y_{18,48})k_1$ $+ (y_1 + y_{18} + y_{1,18})k_{48} + (1 + y_{18})k_1k_{48} = 0$	2	0.63	0.90
$c_2 + c_{19} + c_{49} + c_2c_{32} + c_2c_{49} + c_{19}c_{32}c_{49}$	2	0.63	0.97

Continued on next page

Table 19: Probabilistic integral distinguishers of GIFT-64 for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\lvert K\rvert$	[10]	Emp. p
Residual: $(y_2 + y_{19,49})k_{32}$ $+ (1 + y_2 + y_{19,32})k_{49} + (y_{19})k_{32}k_{49} = 0$			
$c_3 + c_{33} + c_{50} + c_{16}c_{50} + c_{33}c_{50} + c_3c_{16}c_{33}$ *Residual:* $(y_{50} + y_{3,33})k_{16}$ $+ (1 + y_{50} + y_{3,16})k_{33} + (y_3)k_{16}k_{33} = 0$	2	0.63	0.96
$c_{55} + c_4c_{21} + c_4c_{38} + c_{21}c_{38} + c_{38}c_{55} + c_4c_{21}c_{38}$ *Residual:* $(y_{21} + y_{38} + y_{21,38})k_4$ $+ (y_{38} + y_{4,38})k_{21} + (1 + y_{38})k_4k_{21} = 0$	2	0.63	0.87
$c_{39} + c_5c_{22} + c_5c_{52} + c_{22}c_{39} + c_{22}c_{52} + c_5c_{22}c_{52}$ *Residual:* $(y_{22} + y_{22,52})k_5$ $+ (y_5 + y_{22} + y_{5,22})k_{52} + (1 + y_{22})k_5k_{52} = 0$	2	0.63	0.91
$c_{23} + c_6c_{23} + c_6c_{36} + c_6c_{53} + c_{36}c_{53} + c_6c_{36}c_{53}$ *Residual:* $(y_6 + y_{53} + y_{6,53})k_{36}$ $+ (y_6 + y_{6,36})k_{53} + (1 + y_6)k_{36}k_{53} = 0$	2	0.63	0.89
$c_7 + c_{37} + c_{54} + c_{20}c_{54} + c_{37}c_{54} + c_7c_{20}c_{37}$ *Residual:* $(y_{54} + y_{7,37})k_{20}$ $+ (1 + y_{54} + y_{7,20})k_{37} + (y_7)k_{20}k_{37} = 0$	2	0.63	0.95
$c_{59} + c_8c_{25} + c_8c_{42} + c_{25}c_{42} + c_{42}c_{59} + c_8c_{25}c_{42}$ *Residual:* $(y_{25} + y_{42} + y_{25,42})k_8$ $+ (y_{42} + y_{8,42})k_{25} + (1 + y_{42})k_8k_{25} = 0$	2	0.63	0.86
$c_{43} + c_9c_{26} + c_9c_{56} + c_{26}c_{43} + c_{26}c_{56} + c_9c_{26}c_{56}$ *Residual:* $(y_{26} + y_{26,56})k_9$ $+ (y_9 + y_{26} + y_{9,26})k_{56} + (1 + y_{26})k_9k_{56} = 0$	2	0.63	0.88
$c_{27} + c_{10}c_{27} + c_{10}c_{40} + c_{10}c_{57} + c_{40}c_{57} + c_{10}c_{40}c_{57}$ *Residual:* $(y_{10} + y_{57} + y_{10,57})k_{40}$ $+ (y_{10} + y_{10,40})k_{57} + (1 + y_{10})k_{40}k_{57} = 0$	2	0.63	0.88
$c_{11} + c_{41} + c_{58} + c_{24}c_{58} + c_{41}c_{58} + c_{11}c_{24}c_{41}$ *Residual:* $(y_{58} + y_{11,41})k_{24}$ $+ (1 + y_{58} + y_{11,24})k_{41} + (y_{11})k_{24}k_{41} = 0$	2	0.63	0.96
$c_{29} + c_{46} + c_{63} + c_{12}c_{46} + c_{29}c_{46} + c_{12}c_{29}c_{63}$ *Residual:* $(y_{46} + y_{29,63})k_{12}$ $+ (1 + y_{46} + y_{12,63})k_{29} + (y_{63})k_{12}k_{29} = 0$	2	0.63	0.98
$c_{47} + c_{13}c_{30} + c_{13}c_{60} + c_{30}c_{47} + c_{30}c_{60} + c_{13}c_{30}c_{60}$ *Residual:* $(y_{30} + y_{30,60})k_{13}$ $+ (y_{13} + y_{30} + y_{13,30})k_{60} + (1 + y_{30})k_{13}k_{60} = 0$	2	0.63	0.93
$c_{31} + c_{14}c_{31} + c_{14}c_{44} + c_{14}c_{61} + c_{44}c_{61} + c_{14}c_{44}c_{61}$ *Residual:* $(y_{14} + y_{61} + y_{14,61})k_{44}$ $+ (y_{14} + y_{14,44})k_{61} + (1 + y_{14})k_{44}k_{61} = 0$	2	0.63	0.92
$c_{15} + c_{45} + c_{62} + c_{28}c_{62} + c_{45}c_{62} + c_{15}c_{28}c_{45}$ *Residual:* $(y_{62} + y_{15,45})k_{28}$ $+ (1 + y_{62} + y_{15,28})k_{45} + (y_{15})k_{28}k_{45} = 0$	2	0.63	0.97
$c_{17} + c_{34} + c_{51} + c_0c_{17} + c_0c_{34} + c_{17}c_{34} + c_0c_{17}c_{51}$ *Residual:* $(y_{17} + y_{34} + y_{17,51})k_0$ $+ (1 + y_{34} + y_{0,51})k_{17} + (1 + y_{51})k_0k_{17} = 0$	2	0.63	0.97
$c_1 + c_{18} + c_{35} + c_1c_{18} + c_1c_{48} + c_{18}c_{48} + c_1c_{35}c_{48}$	2	0.63	0.97

Continued on next page

Table 19: Probabilistic integral distinguishers of GIFT-64 for $p = 4$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\lvert K\rvert$	[10]	Emp. p
Residual: $(1 + y_{18} + y_{35,48})k_1$ $+ (y_1 + y_{18} + y_{1,35})k_{48} + (1 + y_{35})k_1k_{48} = 0$			
$c_2 + c_{19} + c_2c_{19} + c_2c_{32} + c_2c_{49} + c_{32}c_{49} + c_2c_{32}c_{49}$ *Residual:* $(y_2 + y_{49} + y_{2,49})k_{32}$ $+ (y_2 + y_{2,32})k_{49} + (1 + y_2)k_{32}k_{49} = 0$	2	0.63	0.91
$c_3 + c_{50} + c_3c_{50} + c_{16}c_{33} + c_{16}c_{50} + c_{33}c_{50} + c_{16}c_{33}c_{50}$ *Residual:* $(y_{33} + y_{50} + y_{33,50})k_{16}$ $+ (y_{50} + y_{16,50})k_{33} + (1 + y_{50})k_{16}k_{33} = 0$	2	0.63	0.88
$c_{21} + c_{38} + c_{55} + c_4c_{21} + c_4c_{38} + c_{21}c_{38} + c_4c_{21}c_{55}$ *Residual:* $(y_{21} + y_{38} + y_{21,55})k_4$ $+ (1 + y_{38} + y_{4,55})k_{21} + (1 + y_{55})k_4k_{21} = 0$	2	0.63	0.97
$c_5 + c_{22} + c_{39} + c_5c_{22} + c_5c_{52} + c_{22}c_{52} + c_5c_{39}c_{52}$ *Residual:* $(1 + y_{22} + y_{39,52})k_5$ $+ (y_5 + y_{22} + y_{5,39})k_{52} + (1 + y_{39})k_5k_{52} = 0$	2	0.63	0.96
$c_6 + c_{23} + c_{53} + c_6c_{36} + c_6c_{53} + c_{36}c_{53} + c_{23}c_{36}c_{53}$ *Residual:* $(y_6 + y_{53} + y_{23,53})k_{36}$ $+ (1 + y_6 + y_{23,36})k_{53} + (1 + y_{23})k_{36}k_{53} = 0$	2	0.63	0.96
$c_7 + c_{54} + c_7c_{54} + c_{20}c_{37} + c_{20}c_{54} + c_{37}c_{54} + c_{20}c_{37}c_{54}$ *Residual:* $(y_{37} + y_{54} + y_{37,54})k_{20}$ $+ (y_{54} + y_{20,54})k_{37} + (1 + y_{54})k_{20}k_{37} = 0$	2	0.63	0.83
$c_{25} + c_{42} + c_{59} + c_8c_{25} + c_8c_{42} + c_{25}c_{42} + c_8c_{25}c_{59}$ *Residual:* $(y_{25} + y_{42} + y_{25,59})k_8$ $+ (1 + y_{42} + y_{8,59})k_{25} + (1 + y_{59})k_8k_{25} = 0$	2	0.63	0.95
$c_9 + c_{26} + c_{43} + c_9c_{26} + c_9c_{56} + c_{26}c_{56} + c_9c_{43}c_{56}$ *Residual:* $(1 + y_{26} + y_{43,56})k_9$ $+ (y_9 + y_{26} + y_{9,43})k_{56} + (1 + y_{43})k_9k_{56} = 0$	2	0.63	0.96
$c_{10} + c_{27} + c_{57} + c_{10}c_{40} + c_{10}c_{57} + c_{40}c_{57} + c_{27}c_{40}c_{57}$ *Residual:* $(y_{10} + y_{57} + y_{27,57})k_{40}$ $+ (1 + y_{10} + y_{27,40})k_{57} + (1 + y_{27})k_{40}k_{57} = 0$	2	0.63	0.97
$c_{11} + c_{58} + c_{11}c_{58} + c_{24}c_{41} + c_{24}c_{58} + c_{41}c_{58} + c_{24}c_{41}c_{58}$ *Residual:* $(y_{41} + y_{58} + y_{41,58})k_{24}$ $+ (y_{58} + y_{24,58})k_{41} + (1 + y_{58})k_{24}k_{41} = 0$	2	0.63	0.87
$c_{46} + c_{63} + c_{12}c_{29} + c_{12}c_{46} + c_{29}c_{46} + c_{46}c_{63} + c_{12}c_{29}c_{46}$ *Residual:* $(y_{29} + y_{46} + y_{29,46})k_{12}$ $+ (y_{46} + y_{12,46})k_{29} + (1 + y_{46})k_{12}k_{29} = 0$	2	0.63	0.90
$c_{13} + c_{30} + c_{47} + c_{13}c_{30} + c_{13}c_{60} + c_{30}c_{60} + c_{13}c_{47}c_{60}$ *Residual:* $(1 + y_{30} + y_{47,60})k_{13}$ $+ (y_{13} + y_{30} + y_{13,47})k_{60} + (1 + y_{47})k_{13}k_{60} = 0$	2	0.63	0.98
$c_{14} + c_{31} + c_{61} + c_{14}c_{44} + c_{14}c_{61} + c_{44}c_{61} + c_{31}c_{44}c_{61}$ *Residual:* $(y_{14} + y_{61} + y_{31,61})k_{44}$ $+ (1 + y_{14} + y_{31,44})k_{61} + (1 + y_{31})k_{44}k_{61} = 0$	2	0.63	0.98

Table 20: Probabilistic integral distinguishers of GIFT-64 for $p = 5$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4).

Combination	$\lvert K\rvert$	[10]	Emp. p
$c_2 + c_{19} + c_{32} + c_{49} + c_2c_{32} + c_2c_{49} + c_{19}c_{32}c_{49}$ *Residual:* $(1 + y_2 + y_{19,49})k_{32}$ $+ (1 + y_2 + y_{19,32})k_{49} + (y_{19})k_{32}k_{49} = 0$	2	0.63	0.68
$c_6 + c_{23} + c_{36} + c_{53} + c_6c_{36} + c_6c_{53} + c_{23}c_{36}c_{53}$ *Residual:* $(1 + y_6 + y_{23,53})k_{36}$ $+ (1 + y_6 + y_{23,36})k_{53} + (y_{23})k_{36}k_{53} = 0$	2	0.63	0.69
$c_7 + c_{20} + c_{37} + c_{54} + c_{20}c_{54} + c_{37}c_{54} + c_7c_{20}c_{37}$ *Residual:* $(1 + y_{54} + y_{7,37})k_{20}$ $+ (1 + y_{54} + y_{7,20})k_{37} + (y_7)k_{20}k_{37} = 0$	2	0.63	0.70
$c_{11} + c_{24} + c_{41} + c_{58} + c_{24}c_{58} + c_{41}c_{58} + c_{11}c_{24}c_{41}$ *Residual:* $(1 + y_{58} + y_{11,41})k_{24}$ $+ (1 + y_{58} + y_{11,24})k_{41} + (y_{11})k_{24}k_{41} = 0$	2	0.63	0.69
$c_{12} + c_{29} + c_{46} + c_{63} + c_{12}c_{46} + c_{29}c_{46} + c_{12}c_{29}c_{63}$ *Residual:* $(1 + y_{46} + y_{29,63})k_{12}$ $+ (1 + y_{46} + y_{12,63})k_{29} + (y_{63})k_{12}k_{29} = 0$	2	0.63	0.70
$c_{15} + c_{28} + c_{45} + c_{62} + c_{28}c_{62} + c_{45}c_{62} + c_{15}c_{28}c_{45}$ *Residual:* $(1 + y_{62} + y_{15,45})k_{28}$ $+ (1 + y_{62} + y_{15,28})k_{45} + (y_{15})k_{28}k_{45} = 0$	2	0.63	0.70
$c_0 + c_{17} + c_{34} + c_{51} + c_0c_{17} + c_0c_{34} + c_{17}c_{34} + c_0c_{17}c_{51}$ *Residual:* $(1 + y_{17} + y_{34} + y_{17,51})k_0$ $+ (1 + y_0 + y_{34} + y_{0,51})k_{17} + (1 + y_{51})k_0k_{17} = 0$	2	0.63	0.68
$c_1 + c_{18} + c_{35} + c_{48} + c_1c_{18} + c_1c_{48} + c_{18}c_{48} + c_1c_{35}c_{48}$ *Residual:* $(1 + y_{18} + y_{48} + y_{35,48})k_1$ $+ (1 + y_1 + y_{18} + y_{1,35})k_{48} + (1 + y_{35})k_1k_{48} = 0$	2	0.63	0.68
$c_3 + c_{16} + c_{33} + c_{50} + c_{16}c_{33} + c_{16}c_{50} + c_{33}c_{50} + c_3c_{16}c_{33}$ *Residual:* $(1 + y_{33} + y_{50} + y_{3,33})k_{16}$ $+ (1 + y_{16} + y_{50} + y_{3,16})k_{33} + (1 + y_3)k_{16}k_{33} = 0$	2	0.63	0.68
$c_4 + c_{21} + c_{38} + c_{55} + c_4c_{21} + c_4c_{38} + c_{21}c_{38} + c_4c_{21}c_{55}$ *Residual:* $(1 + y_{21} + y_{38} + y_{21,55})k_4$ $+ (1 + y_4 + y_{38} + y_{4,55})k_{21} + (1 + y_{55})k_4k_{21} = 0$	2	0.63	0.68
$c_5 + c_{22} + c_{39} + c_{52} + c_5c_{22} + c_5c_{52} + c_{22}c_{52} + c_5c_{39}c_{52}$ *Residual:* $(1 + y_{22} + y_{52} + y_{39,52})k_5$ $+ (1 + y_5 + y_{22} + y_{5,39})k_{52} + (1 + y_{39})k_5k_{52} = 0$	2	0.63	0.68
$c_8 + c_{25} + c_{42} + c_{59} + c_8c_{25} + c_8c_{42} + c_{25}c_{42} + c_8c_{25}c_{59}$ *Residual:* $(1 + y_{25} + y_{42} + y_{25,59})k_8$ $+ (1 + y_8 + y_{42} + y_{8,59})k_{25} + (1 + y_{59})k_8k_{25} = 0$	2	0.63	0.70
$c_9 + c_{26} + c_{43} + c_{56} + c_9c_{26} + c_9c_{56} + c_{26}c_{56} + c_9c_{43}c_{56}$ *Residual:* $(1 + y_{26} + y_{56} + y_{43,56})k_9$ $+ (1 + y_9 + y_{26} + y_{9,43})k_{56} + (1 + y_{43})k_9k_{56} = 0$	2	0.63	0.68
$c_{10} + c_{27} + c_{40} + c_{57} + c_{10}c_{40} + c_{10}c_{57} + c_{40}c_{57} + c_{27}c_{40}c_{57}$ *Residual:* $(1 + y_{10} + y_{57} + y_{27,57})k_{40}$ $+ (1 + y_{10} + y_{40} + y_{27,40})k_{57} + (1 + y_{27})k_{40}k_{57} = 0$	2	0.63	0.68
$c_{13} + c_{30} + c_{47} + c_{60} + c_{13}c_{30} + c_{13}c_{60} + c_{30}c_{60} + c_{13}c_{47}c_{60}$ *Residual:* $(1 + y_{30} + y_{60} + y_{47,60})k_{13}$ $+ (1 + y_{13} + y_{30} + y_{13,47})k_{60} + (1 + y_{47})k_{13}k_{60} = 0$	2	0.63	0.68

Continued on next page

Table 20: Probabilistic integral distinguishers of GIFT-64 for $p = 5$, $q = 1$, and input-structure mask `f000000000000000` (data complexity: 2^4). [continued table]

Combination	$\|K\|$	[10]	Emp. p
$c_{14}+c_{31}+c_{44}+c_{61}+c_{14}c_{44}+c_{14}c_{61}+c_{44}c_{61}+c_{31}c_{44}c_{61}$ *Residual:* $(1+y_{14}+y_{61}+y_{31,61})k_{44}$ $+(1+y_{14}+y_{44}+y_{31,44})k_{61}+(1+y_{31})k_{44}k_{61}=0$	2	0.63	0.68

Table 21: Probabilistic integral distinguishers of GIFT-64 for $p = 6$, $q = 1$, and input-structure mask `fff0000000000000` (data complexity: 2^{12}).

Combination	$\|K\|$	[10]	Emp. p
$c_2+c_{19}+c_{32}+c_{49}+c_2c_{32}+c_2c_{49}+c_{19}c_{32}c_{49}$ *Residual:* $(1+y_2+y_{19,49})k_{32}$ $+(1+y_2+y_{19,32})k_{49}+(y_{19})k_{32}k_{49}=0$	2	0.63	0.63
$c_3+c_{16}+c_{33}+c_{50}+c_{16}c_{50}+c_{33}c_{50}+c_3c_{16}c_{33}$ *Residual:* $(1+y_{50}+y_{3,33})k_{16}$ $+(1+y_{50}+y_{3,16})k_{33}+(y_3)k_{16}k_{33}=0$	2	0.63	0.63
$c_6+c_{23}+c_{36}+c_{53}+c_6c_{36}+c_6c_{53}+c_{23}c_{36}c_{53}$ *Residual:* $(1+y_6+y_{23,53})k_{36}$ $+(1+y_6+y_{23,36})k_{53}+(y_{23})k_{36}k_{53}=0$	2	0.63	0.63
$c_{11}+c_{24}+c_{41}+c_{58}+c_{24}c_{58}+c_{41}c_{58}+c_{11}c_{24}c_{41}$ *Residual:* $(1+y_{58}+y_{11,41})k_{24}$ $+(1+y_{58}+y_{11,24})k_{41}+(y_{11})k_{24}k_{41}=0$	2	0.63	0.64
$c_{12}+c_{29}+c_{46}+c_{63}+c_{12}c_{46}+c_{29}c_{46}+c_{12}c_{29}c_{63}$ *Residual:* $(1+y_{46}+y_{29,63})k_{12}$ $+(1+y_{46}+y_{12,63})k_{29}+(y_{63})k_{12}k_{29}=0$	2	0.63	0.63
$c_{15}+c_{28}+c_{45}+c_{62}+c_{28}c_{62}+c_{45}c_{62}+c_{15}c_{28}c_{45}$ *Residual:* $(1+y_{62}+y_{15,45})k_{28}$ $+(1+y_{62}+y_{15,28})k_{45}+(y_{15})k_{28}k_{45}=0$	2	0.63	0.63
$c_{34}+c_{51}+c_0c_{17}+c_0c_{34}+c_{17}c_{34}+c_{17}c_{51}+c_{34}c_{51}+c_0c_{17}c_{34}$ *Residual:* $(y_{17}+y_{34}+y_{17,34})k_0$ $+(y_0+y_{34}+y_{51}+y_{0,34})k_{17}+(1+y_{34})k_0k_{17}=0$	2	0.63	0.63
$c_0+c_{17}+c_{34}+c_{51}+c_0c_{17}+c_0c_{34}+c_{17}c_{34}+c_0c_{17}c_{51}$ *Residual:* $(1+y_{17}+y_{34}+y_{17,51})k_0$ $+(1+y_0+y_{34}+y_{0,51})k_{17}+(1+y_{51})k_0k_{17}=0$	2	0.63	0.62
$c_1+c_{18}+c_{35}+c_{48}+c_1c_{18}+c_1c_{48}+c_{18}c_{48}+c_1c_{35}c_{48}$ *Residual:* $(1+y_{18}+y_{48}+y_{35,48})k_1$ $+(1+y_1+y_{18}+y_{1,35})k_{48}+(1+y_{35})k_1k_{48}=0$	2	0.63	0.62
$c_{38}+c_{55}+c_4c_{21}+c_4c_{38}+c_{21}c_{38}+c_{21}c_{55}+c_{38}c_{55}+c_4c_{21}c_{38}$ *Residual:* $(y_{21}+y_{38}+y_{21,38})k_4$ $+(y_4+y_{38}+y_{55}+y_{4,38})k_{21}+(1+y_{38})k_4k_{21}=0$	2	0.63	0.63
$c_4+c_{21}+c_{38}+c_{55}+c_4c_{21}+c_4c_{38}+c_{21}c_{38}+c_4c_{21}c_{55}$ *Residual:* $(1+y_{21}+y_{38}+y_{21,55})k_4$ $+(1+y_4+y_{38}+y_{4,55})k_{21}+(1+y_{55})k_4k_{21}=0$	2	0.63	0.62

Continued on next page

Table 21: Probabilistic integral distinguishers of GIFT-64 for $p = 6$, $q = 1$, and input-structure mask `fff0000000000000` (data complexity: 2^{12}). [continued table]

Combination	$\lvert K\rvert$	[10]	Emp. p
$c_5 + c_{22} + c_{39} + c_{52} + c_5c_{22} + c_5c_{52} + c_{22}c_{52} + c_5c_{39}c_{52}$	2	0.63	0.62
Residual: $(1 + y_{22} + y_{52} + y_{39,52})k_5$			
$+ (1 + y_5 + y_{22} + y_{5,39})k_{52} + (1 + y_{39})k_5k_{52} = 0$			
$c_7 + c_{20} + c_{37} + c_{54} + c_{20}c_{37} + c_{20}c_{54} + c_{37}c_{54} + c_7c_{20}c_{37}$	2	0.63	0.63
Residual: $(1 + y_{37} + y_{54} + y_{7,37})k_{20}$			
$+ (1 + y_{20} + y_{54} + y_{7,20})k_{37} + (1 + y_7)k_{20}k_{37} = 0$			
$c_8 + c_{25} + c_{42} + c_{59} + c_8c_{25} + c_8c_{42} + c_{25}c_{42} + c_8c_{25}c_{59}$	2	0.63	0.63
Residual: $(1 + y_{25} + y_{42} + y_{25,59})k_8$			
$+ (1 + y_8 + y_{42} + y_{8,59})k_{25} + (1 + y_{59})k_8k_{25} = 0$			
$c_9 + c_{26} + c_{43} + c_{56} + c_9c_{26} + c_9c_{56} + c_{26}c_{56} + c_9c_{43}c_{56}$	2	0.63	0.62
Residual: $(1 + y_{26} + y_{56} + y_{43,56})k_9$			
$+ (1 + y_9 + y_{26} + y_{9,43})k_{56} + (1 + y_{43})k_9k_{56} = 0$			
$c_{10} + c_{27} + c_{40} + c_{57} + c_{10}c_{40} + c_{10}c_{57} + c_{40}c_{57} + c_{27}c_{40}c_{57}$	2	0.63	0.62
Residual: $(1 + y_{10} + y_{57} + y_{27,57})k_{40}$			
$+ (1 + y_{10} + y_{40} + y_{27,40})k_{57} + (1 + y_{27})k_{40}k_{57} = 0$			
$c_{13} + c_{30} + c_{47} + c_{60} + c_{13}c_{30} + c_{13}c_{60} + c_{30}c_{60} + c_{13}c_{47}c_{60}$	2	0.63	0.63
Residual: $(1 + y_{30} + y_{60} + y_{47,60})k_{13}$			
$+ (1 + y_{13} + y_{30} + y_{13,47})k_{60} + (1 + y_{47})k_{13}k_{60} = 0$			
$c_{14} + c_{31} + c_{44} + c_{61} + c_{14}c_{44} + c_{14}c_{61} + c_{44}c_{61} + c_{31}c_{44}c_{61}$	2	0.63	0.63
Residual: $(1 + y_{14} + y_{61} + y_{31,61})k_{44}$			
$+ (1 + y_{14} + y_{44} + y_{31,44})k_{61} + (1 + y_{31})k_{44}k_{61} = 0$			

F Proof of Theorem 1

Proof. Fix an arbitrary key. Since $\rho(\boldsymbol{k}_2)$ and the $\lambda_i(\boldsymbol{k}_2)$ depend only on the fixed key, they are constants over $\mathcal{P}$. Using the assumed identity with $\boldsymbol{y} = E(\boldsymbol{p})$, we obtain

$$\begin{aligned}\sum_{\boldsymbol{p}\in\mathcal{P}} h(E(\boldsymbol{p})) + (|\mathcal{P}| \bmod 2)\,\rho(\boldsymbol{k}_2) &= \sum_{\boldsymbol{p}\in\mathcal{P}} \left(h(E(\boldsymbol{p})) + \rho(\boldsymbol{k}_2)\right) \\ &= \sum_{i=0}^{t-1} \lambda_i(\boldsymbol{k}_2) \sum_{\boldsymbol{p}\in\mathcal{P}} g_i(E_1(\boldsymbol{p})) \\ &= \sum_{i=0}^{t-1} \lambda_i(\boldsymbol{k}_2)\,\beta_i.\end{aligned}$$

The remaining claims follow immediately.