# Integral Resistance and Degree Bounds for Complex Linear Layers: Application to PRINCE and Lower-Latency Alternatives

Simon Gerhalter and Maria Eichlseder

Graz University of Technology, Graz, Austria
simon.gerhalter@tugraz.at, maria.eichlseder@tugraz.at

**Abstract.** The integral-resistance property provides strong arguments against integral distinguishers. Recently, Zeng and Tian proposed a new method to show this property for AES. In this paper, we provide a generalized framework and tool called `intres` to extend and apply this method to other ciphers with complex linear layers. We derive properties that a cipher must fulfill in order for the method to be applicable. Furthermore, we introduce a degree propagation model which helps us determine the valid key masks for the integral-resistance matrix. The degree model can also be used to upper-bound the algebraic degree of cipher constructions. This allows us to provide tighter upper bounds for the degree of Rijndael-256. We propose algorithmic improvements to substantially decrease the runtime of the offline phase with the `intres` framework. As a result, we are able to show the integral-resistance property for 7 rounds of PRINCE and 6 rounds of Beanie. Finally, we develop a heuristic MILP-based approach to search for lower-latency alternatives to the MixColumns matrices of PRINCE while maintaining integral resistance. After showing that using this new matrix we still achieve 7-round integral resistance, we validate our method with SAT-based trail counting. While using a MixColumns matrix only optimized for integral resistance might affect security against other types of attacks, we believe these lower-latency matrices have their place in constructions similar to ZIP-ciphers, where integral resistance is particularly critical.

**Keywords:** Integral cryptanalysis · Integral resistance property · Degree bound · Rijndael-256 · PRINCE · MDS · MILP

## 1 Introduction

Symmetric primitives are an important part of many cryptographic protocols that ensure secure communication over the internet and elsewhere. In recent years, there has been a big push towards more low-latency symmetric primitives like PRINCEv2 [11] and QARMAv2 [2]. New constructions like ZIP-ciphers [18] aim to reduce the latency even further. However, careful cryptanalysis is necessary to provide confidence in their resistance to attacks. Among the most

Published at SAC 2026

common techniques used for cryptanalysis are differential [8] and linear cryptanalysis [29]. These techniques have been thoroughly studied for many years and it is possible to use tool-assistance to provide bounds on their effectiveness. While these bounds often do not take the key schedule into account and disregard clustering, they are good indicators for the security of the primitive against these types of attacks.

Integral cryptanalysis is another important technique to evaluate the security of ciphers. Knudsen and Wagner [26] first coined the term integral cryptanalysis. The authors use structural properties of the cipher to find integral distinguishers. However, the first practical attack of this type was conducted on the AES-like cipher Square [12]. Before this, the theoretical foundation of integral attacks had been laid by Lai [27] and Knudsen [25]. Lai establishes a link between the algebraic properties of a cipher and its susceptibility to integral attacks. Using this, the division property [34], and equivalently monomial prediction [23], accurately model the propagation of monomials through the operations of a cipher. It is common to implement this technique as Boolean satisfiability problems (SAT) [32,17,24,35] or via mixed integer linear programs (MILP) [36,31,28,16,15,20]. However, for ciphers with large linear layers, like MDS matrices, using these constraint programs quickly becomes computationally infeasible when aiming for accurate models.

Hebborn et al. [21] show how to use monomial prediction to prove lower bounds on the algebraic degree of a cipher. Furthermore, they also establish how many rounds it takes until all maximum-degree monomials are present in the algebraic normal form (ANF) of the ciphers SKINNY [6], PRESENT [9], and GIFT [4]. This property ensures the absence of integral distinguishers that form a cube over the input space. However, as demonstrated in previous work, it is possible to conduct integral attacks where the used plaintext set is not a linear subspace [28] or is linear, but not of cube structure [15]. To also ensure the absence of these types of integral distinguishers and under the assumption of independent round keys, Hebborn et al. [22] introduced the integral-resistance property. They were able to show this property for SKINNY, CRAFT, PRESENT, GIFT, but were not able to show this property for ciphers with complex linear layers like AES [13]. Beierle et al. [5] show how to apply the integral-resistance property to primitives that add the key via modular addition. Furthermore, Zeng and Tian [37] demonstrate how to prove the integral-resistance property when no whitening key is added. Recently, Zeng and Tian [38] introduced a new approach that makes it possible to show the integral-resistance property for AES. They achieve this by imposing additional constraints on the key variables present in the monomial prediction. This drastically reduces the number of monomial transitions possible and therefore the complexity of the monomial prediction.

*Contributions.* In this paper, we improve various aspects of the technique presented by Zeng and Tian [38] and introduce an easy-to-use way of estimating the degree of substitution–permutation network (SPN) ciphers. This makes it possible to efficiently show the integral-resistance property for a broader range of strongly aligned SPN ciphers. In more detail, we contribute the following.

- We expand the theoretical basis needed to apply the method from Zeng and Tian (minimal-transitions method). First, we revisit and update the necessary properties of the S-Box for a successful application of this method. Next, we argue that maximizing the degree of the key monomials is equivalent to minimizing the degree of the state monomials. This helps us to precisely define the required properties of the key monomials for the valid application of the minimal-transitions method. Furthermore, we demonstrate what happens when different input/output configurations lead to different Hamming weights of the key monomials.
- We use a new cell-based degree-propagation MILP model to more efficiently find the constraints on the key monomials needed for the minimal-transitions method to be applicable. This model can also be used to find upper bounds on the degree of various SPN ciphers. For example, we find improved upper bounds for Rijndael-256 (see Table 2).
- We introduce the `intres` framework, a dedicated tool written in `rust` that implements the minimal-transitions method for strongly aligned SPN ciphers with suitable S-Boxes. This framework implements algorithmic improvements for calculating the MixColumns transitions in the offline phase. Instead of days, the offline phase now takes minutes. Furthermore, our framework is easily extendable. Using different S-Boxes or MixColumns matrices can be done by a change of parameters. Changing the cipher size requires only slight code changes. We use `intres` to show the integral-resistance property of 7-round PRINCE and 6-round Beanie.
- Lastly, we use a heuristic approach to find a lower-latency alternative to the PRINCE MixColumns matrices. To this end, we show that the invertibility approach first described by Zhang and Rijmen [39] can be used in MILP, contrary to the claims of Hu et al. [24]. Next, we demonstrate that a PRINCE-like cipher using the improved matrices still fulfills the 7-round integral-resistance property. Due to the sparse matrix, we are able to validate the correctness of the minimal-transitions method with SAT-based monomial-trail counting for the first time.

We provide the source code of `intres` and our degree bounding model. Furthermore, we supply the MILP model needed for the minimization of the MixColumns matrix Hamming weight. Finally, we deliver the integral-resistance matrices and a tool to check the validity with monomial trail counting. The accompanying code can be found at:

`https://extgit.isec.tugraz.at/castle/tool/intres`

*Outline.* After some background information in Section 2, we revisit the requirements needed for the S-Box in Section 3 and explain our degree bounding model in Section 4. Next, we introduce our `intres` framework in Section 5 and end with Section 6, where we optimize the latency of the linear layer without sacrificing security against integral attacks.

## 2 Background

Integral cryptanalysis is closely linked to the algebraic representation of a cipher. We can represent a cipher as a vectorial Boolean function $E : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^n$; $(\boldsymbol{k}, \boldsymbol{x}) \mapsto E(\boldsymbol{k}, \boldsymbol{x})$. Often, it is beneficial to represent the cipher as a keyed permutation $E_{\boldsymbol{k}} : \mathbb{F}_2^n \to \mathbb{F}_2^n$; $\boldsymbol{x} \mapsto E_{\boldsymbol{k}}(\boldsymbol{x})$. Given $E_k(\boldsymbol{x}) = (E_k^{(1)}(\boldsymbol{x}), \cdots, E_k^{(n)}(\boldsymbol{x}))$, we can describe the ANF of each coordinate function by

$$E_k^{(i)}(\boldsymbol{x}) = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n, \boldsymbol{v}\in\mathbb{F}_2^m} a_{\boldsymbol{u},\boldsymbol{v}}^{(i)} \cdot \boldsymbol{k}^{\boldsymbol{v}} \boldsymbol{x}^{\boldsymbol{u}} = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} \left( \sum_{\boldsymbol{v}\in\mathbb{F}_2^m} a_{\boldsymbol{u},\boldsymbol{v}}^{(i)} \cdot \boldsymbol{k}^{\boldsymbol{v}} \right) \cdot \boldsymbol{x}^{\boldsymbol{u}} = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}^{(i)}(\boldsymbol{k}) \cdot \boldsymbol{x}^{\boldsymbol{u}},$$

where the coefficients $a \in \mathbb{F}_2$ and the monomial $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^n x_i^{u_i}$, with

$$x_i^{u_i} = \begin{cases} x_i & \text{if } u_i = 1, \\ 1 & \text{if } u_i = 0. \end{cases}$$

We refer to $\boldsymbol{u}$ as a monomial mask. Analogous to $\boldsymbol{x}^{\boldsymbol{u}}$, given $\boldsymbol{y} = E_k(\boldsymbol{x})$, we can calculate the output monomials with $\boldsymbol{y}^{\boldsymbol{w}} = \prod_i^n E_k^{(i)}(\boldsymbol{x})^{w_i}$. In the following, we also refer to this product of coordinate functions as a component function. A monomial $\boldsymbol{x}^{\boldsymbol{u}}$ is *contained* in $\boldsymbol{y}^{\boldsymbol{w}}$ if the corresponding coefficient $a_{\boldsymbol{u}}$ is nonzero, denoted by $\boldsymbol{x}^{\boldsymbol{u}} \to \boldsymbol{y}^{\boldsymbol{w}}$. The degree $d$ of a monomial $\boldsymbol{x}^{\boldsymbol{u}}$ can be calculated by the Hamming weight $d = hw(\boldsymbol{u})$. Furthermore, the degree of a Boolean function is determined by the highest-degree monomial it contains. In case of a vectorial Boolean function, the minimum degree is equal to the degree of the coordinate functions with the lowest degree.

### 2.1 Integral Resistance

Integral cryptanalysis uses distinguishers that take a set of plaintext inputs. The corresponding ciphertext set then sums to a constant value, regardless of the used key. Given the degree $d$ of the plaintext variables, it is possible to construct an integral distinguisher with data complexity $2^{d+1}$ [27]. However, this alone is not enough to prove the absence of integral distinguishers with data complexity lower than $2^{d+1}$ [21]. There may be plaintext monomials of lower degree not contained in the ANF of a component function of the output. In case of a keyed permutation, this means $\exists \boldsymbol{u} \forall \boldsymbol{v} : hw(\boldsymbol{u}) < d+1, \boldsymbol{x}^{\boldsymbol{u}}\boldsymbol{k}^{\boldsymbol{v}} \not\to \boldsymbol{y}^{\boldsymbol{w}}$. For a cipher to be resistant to integral cryptanalysis, it is therefore desirable to show that all plaintext monomials $\boldsymbol{x}^{\boldsymbol{u}}$ are key dependent. Since ciphers are balanced Boolean functions, this means all plaintext monomials up to the maximum degree $n-1$. Hebborn et al. [22] formalize this with the *integral-resistance property* to provide strong arguments against integral distinguisher. We indicate the mask of the max degree monomials by $\bar{\boldsymbol{e}}_i$, where all coordinates except $i$ are nonzero. The integral-resistance property enforces that no plaintext set under any linear combination of coordinate functions leads to an integral attack. Thus, for $s \geq n^2$ key masks $\boldsymbol{v} \in \mathbb{F}_2^m$, we need to find the coefficients of all max-degree monomials $x^{\bar{\boldsymbol{e}}_i}$ for

each coordinate function and show their linear independence. This is done by using these coefficients to construct the integral resistance matrix, defined by

$$\mathcal{I}(E) = \begin{pmatrix} a^{(1)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_1} & a^{(2)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_1} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_1} & a^{(1)}_{\bar{\boldsymbol{e}}_2,\boldsymbol{v}_1} & \cdots & a^{(j)}_{\bar{\boldsymbol{e}}_i,\boldsymbol{v}_1} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_n,\boldsymbol{v}_1} \\ a^{(1)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_2} & a^{(2)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_2} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_2} & a^{(1)}_{\bar{\boldsymbol{e}}_2,\boldsymbol{v}_2} & \cdots & a^{(j)}_{\bar{\boldsymbol{e}}_i,\boldsymbol{v}_2} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_n,\boldsymbol{v}_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a^{(1)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_s} & a^{(2)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_s} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_s} & a^{(1)}_{\bar{\boldsymbol{e}}_2,\boldsymbol{v}_s} & \cdots & a^{(j)}_{\bar{\boldsymbol{e}}_i,\boldsymbol{v}_s} & \cdots & a^{(n)}_{\bar{\boldsymbol{e}}_n,\boldsymbol{v}_s} \end{pmatrix}.$$

**Definition 1 (Integral-Resistance Property [22]).** *Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a block cipher with independent round keys and $\mathcal{I}(E)$ be a corresponding integral-resistance matrix. If $\mathcal{I}(E)$ has full rank and $\boldsymbol{k_0}$ is an independent whitening key, $E_k(\boldsymbol{x} + \boldsymbol{k_0})$ fulfills the integral-resistance property, i.e., for every proper subset $M \subset \mathbb{F}_2^n$ and output mask $\boldsymbol{\beta} \in \mathbb{F}_2^n$, the sum*

$$\sum_{\boldsymbol{x} \in M} \langle \boldsymbol{\beta}, E_k(\boldsymbol{x} \oplus \boldsymbol{k_0}) \rangle$$

*is key-dependent.*

### 2.2 Monomial Prediction

To be able to determine the presence of a monomial in a vectorial Boolean function of high dimension, Todo [34] introduced the division property. A more intuitive way of modeling the propagation of monomials through a cipher is called monomial prediction [23]. This technique is equivalent to the most accurate version of the division property and allows us to find the coefficients $a^{(i)}_{\bar{\boldsymbol{e}}_i,\boldsymbol{v}}$ needed for the integral-resistance property.

The general idea is that a cipher $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is composed of various functions $f_i$. Ciphers typically sequentially apply linear and non-linear functions. However, these linear and non-linear functions usually consist of the parallel application of various smaller functions like S-Boxes and MixColumns operations. Due to this smaller size, it is feasible to calculate the ANF of these functions. Given $\boldsymbol{y}_i = f_i(\boldsymbol{x}_i)$ for one of these functions $f_i$, in the context of monomial prediction we call $\boldsymbol{x}_i^{\boldsymbol{u}_i} \to \boldsymbol{y}_i^{\boldsymbol{w}_i}$ a valid *monomial transition* from $\boldsymbol{x}_i^{\boldsymbol{u}_i}$ to $\boldsymbol{y}_i^{\boldsymbol{w}_i}$ and can visualize all valid monomial transitions in the monomial transition table (MPT, see Table 1). We chain together these valid monomial transitions through the various functions $f_i$ of $f$ in order to get a *monomial trail* connecting $\boldsymbol{x}$ and $\boldsymbol{y}$, denoted by $\boldsymbol{x}^{\boldsymbol{u}} \rightsquigarrow \boldsymbol{y}^{\boldsymbol{w}}$. This gives rise to Lemma 1.

**Lemma 1 ([23]).** *If $\boldsymbol{x}^{\boldsymbol{u}} \to \boldsymbol{y}^{\boldsymbol{w}}$ then $\boldsymbol{x}^{\boldsymbol{u}} \rightsquigarrow \boldsymbol{y}^{\boldsymbol{w}}$ and therefore $\boldsymbol{x}^{\boldsymbol{u}} \not\rightsquigarrow \boldsymbol{y}^{\boldsymbol{w}}$ implies $\boldsymbol{x}^{\boldsymbol{u}} \not\to \boldsymbol{y}^{\boldsymbol{w}}$.*

However, this is not enough to show the presence of a monomial in the ANF of a cipher. The reason for this is that there could be multiple monomial trails connecting $\boldsymbol{x}^{\boldsymbol{u}}$ and $\boldsymbol{y}^{\boldsymbol{w}}$ via different intermediate monomials. This leads to the same monomial being present in the ANF multiple times.

**Definition 2 (Monomial hull [23]).** *We indicate by* $\boldsymbol{x}^{\boldsymbol{u}} \bowtie \boldsymbol{y}^{\boldsymbol{w}}$ *the monomial hull of a specific composite function* $f$*. The monomial hull is the set of all monomial trails connecting* $\boldsymbol{x}^{\boldsymbol{u}}$ *and* $\boldsymbol{y}^{\boldsymbol{w}}$*. The number of trails corresponds to the size of the set, written as* $|\boldsymbol{x}^{\boldsymbol{u}} \bowtie \boldsymbol{y}^{\boldsymbol{w}}|$*.*

Since we are in $\mathbb{F}_2$, an even number of identical monomials cancels out.

**Theorem 1 (Existence of a monomial [23]).** $\boldsymbol{x}^{\boldsymbol{u}} \to \boldsymbol{y}^{\boldsymbol{w}}$ *if and only if* $|\boldsymbol{x}^{\boldsymbol{u}} \bowtie \boldsymbol{y}^{\boldsymbol{w}}|$ *is odd.*

The round keys of a cipher act as an additional input, which we can incorporate into our monomial prediction model. Thus, in order to determine $a^{(i)}_{\bar{\boldsymbol{e}}_j,\boldsymbol{v}}$ required for the integral resistance matrix, we need to count the number of trails $|\boldsymbol{x}^{\bar{\boldsymbol{e}}_j}\boldsymbol{k}^{\boldsymbol{v}} \bowtie \boldsymbol{y}^{\boldsymbol{e}_i}|$ for a given $\boldsymbol{v}$.

Multiple works model the division property or monomial prediction in MILP [36,31,28,16,15,20] and SAT [32,17,24,35] to find integral distinguisher by proving the absence of monomials in the ANF. This is also the common way to show the presence/absence of the coefficients in the integral resistance matrix [22,5,37]. In the original division property paper [34], Todo uses a dedicated algorithm. Furthermore, Derbez and Fouque [14] precompute valid monomial transitions of Super S-Boxes to discard invalid monomial trails preemptively.

*Monomial transitions of large linear functions.* Calculating up to $2^n$ monomials contained in the ANF of $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ becomes infeasible once $n$ gets large. We get a similar exponential growth in the number of constraints needed to express valid monomial transitions. To still be able to construct constraint-programs for finding monomial trails in ciphers with big linear layers, the $\mathcal{S}$ [33] and $\mathcal{ZR}$ [39,24] methods have been proposed. The $\mathcal{S}$ method decomposes the linear layer into 2-bit functions by introducing additional variables for each nonzero entry in the primitive (=binary matrix representation) matrix of the linear transformation. We can then use the basic operations COPY and XOR to model the relation between the input, output, and intermediate variables. However, this method introduces invalid monomial transitions. To overcome this, Zhang and Rijmen [39] link the invertibility of a matrix to valid monomial transitions with the $\mathcal{ZR}$ method. Hu et al. [24] provide a generalized version in Theorem 2. Note that in the remainder of the paper we refer to this refined version as $\mathcal{ZR}$ method.

**Theorem 2 ($\mathcal{ZR}$ method [24]).** *Let* $M$ *be the* $p \times q$ *primitive matrix of a linear transformation. For* $\boldsymbol{u} \in \mathbb{F}_2^q$ *and* $\boldsymbol{v} \in \mathbb{F}_2^p$*,* $\boldsymbol{u} \xrightarrow{M} \boldsymbol{v}$ *is a valid monomial transition of the linear layer* $M$ *if and only if* $M_{\boldsymbol{v},\boldsymbol{u}}$ *is invertible.*

This method perfectly models the monomial transitions. By building a SAT model that dynamically checks the invertibility of a submatrix of $M$ [24], it is possible to check all valid monomial trails. The downside of this method is that checking the invertibility with a SAT solver is a computationally complex task. This makes counting the monomial trails, needed to show the presence of monomials in the ANF of a cipher, infeasible. Due to this limitation, as far as we know, this method has not been used to prove the integral resistance property of ciphers.

### 2.3 Restricting Possible Monomial Transitions

To restrict the number of monomial transitions for ciphers with large linear functions, Zeng and Tian [38] take advantage of carefully selecting specific key masks. Given these key masks, they proceed with a dedicated program that propagates the monomials through a cipher in order to determine the coefficients needed for the integral-resistance matrix. In this section we explain the details of the minimal-transitions method based on the method used by Zeng and Tian [38].

*Restricting possible state degrees.* It is possible to model the round-key addition to the state by a bitwise application of the XOR rule (see Definition 7). This rule states that either the key or the state variable is present in the monomial, but never both. Thus, by choosing these key variables cleverly, we can limit the degree that the monomial of each state can take. While this fact has been used by Hebborn et al. [22], they do not systematically select the key mask in such a way that at each state the monomials can only have a single degree. By using this method we limit the used key masks to a fraction of all possible key masks. Since we can freely select the key masks used for each row of the integral-resistance matrix, this does not invalidate the integral-resistance property.

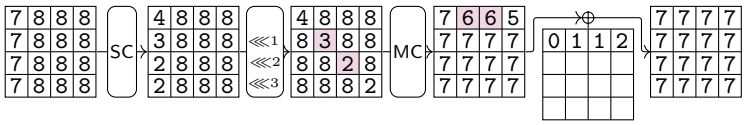
In the following we describe the minimal transition method that is closely tied to the approach that Zeng and Tian [38] use to show the integral-resistance of AES. To obtain the restrained degree of each state we use MILP. For 5 rounds of AES we restrict the degree of the output to 1 and the input to the max degree 127. Next, we maximize the degree of the key variables present in the monomial. Note that Zeng and Tian [38] equivalently minimize the state degree. The equivalence can be derived from the XOR rule, i.e., when adding a variable to the key monomial, the corresponding variable gets removed from the state monomial. The key maximization affects the possible degree propagation through non-linear and linear cipher components. Intuitively, we start with a valid monomial trail from the input to the output without adding key variables. The linear operation can reduce the degree from the output to the input, while the non-linear layer can increase and decrease the degree. By adding key variables to the trail, we decrease the state degree and therefore gradually restrict the possible transitions of the operations. In linear layers, the degree reduction from the output to the input decreases. To compensate, in the non-linear layers the degree must increase more from output to input. We add as many keys as possible while still obtaining valid monomial trails. This leads us to the following properties. From Proposition 2 and 3 by Zeng and Tian [38] we derive Property 1 and from Proposition 5 and 6 by Zeng and Tian [38] we derive Property 2.

*Property 1 (Restricted S-Box transitions).* When maximizing the key variables present in the monomial, the valid monomial transitions $\boldsymbol{x}^{\boldsymbol{u}} \to \boldsymbol{y}^{\boldsymbol{w}}$ of a balanced $k$-bit S-Box are constrained by

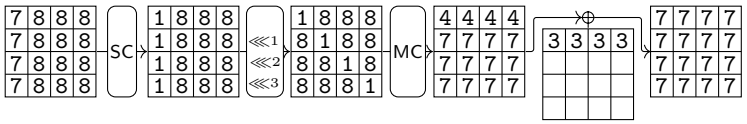
$$\begin{cases} hw(\boldsymbol{w}) = k & \text{iff } hw(\boldsymbol{u}) = k, \\ hw(\boldsymbol{w}) = 0 & \text{iff } hw(\boldsymbol{u}) = 0, \\ hw(\boldsymbol{w}) = 1 & \text{iff } hw(\boldsymbol{u}) = k - 1. \end{cases}$$

*Property 2 (Restricted MixColumns transitions).* When maximizing the key variables present in the monomial, the valid monomial transitions $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow \boldsymbol{y}^{\boldsymbol{w}}$ for linear functions are constrained by $hw(\boldsymbol{u}) = hw(\boldsymbol{w})$.

Applying the key maximization restricts each state of the cipher to a specific degree. The degree of each cell in a column is stored as weight patterns (see Section 4.4 for more details). An example weight pattern is visualized in Figure 1b.




(a) Degree propagation of one round AES without maximized degree of the key.




(b) Constrained degree propagation of one round AES.

Fig. 1: With a fixed input and output mask $\boldsymbol{u}, \boldsymbol{w}$ of the specified weights (degrees) $hw(\boldsymbol{u}), hw(\boldsymbol{w})$ per cell, the intermediate states can contain monomials with various degrees. An example is shown in Figure 1a. However, when maximizing the degree of the key variables, there is only a unique degree assignment possible at each state (Figure 1b.)

*Offline phase.* Given the weight patterns at the input and output of each MixColumns operation, we drastically reduce the number of possible monomial transitions at the MixColumns operation. Zeng and Tian [38] use a branch-and-bound method implemented in CUDA to calculate all valid monomial transitions through the MixColumns operation that follow the given weight patterns. They store the valid transitions in a table, which is then used in the online phase.

*Online phase.* To determine the presence of monomials in the ANF of AES, Zeng and Tian [38] use a dedicated tool, similar to the one proposed by Derbez and Fouque [14]. The idea is to iteratively propagate a set of monomials backward through the rounds of the cipher using key masks that follow the weight patterns. In contrast to monomial-trail counting, we can remove parasitic monomials immediately. For each degree 1 monomial at the output, we then determine the max-degree monomials that are reachable at the input. These monomials correspond to the monomials contained in the ANF of the coordinate function represented by the degree 1 monomial.

As sketched in Figure 1, starting from the back, we first apply the key addition. Due to the previously mentioned XOR rule, for each key variable present in

the monomial, we remove the corresponding state variable from the monomial. In case the monomial does not contain all key variables that get added to the state, we remove this monomial from our set. Next, we use the table computed in the offline phase to propagate the monomials through the MixColumns operation. The ShiftRows operation simply shuffles the indices of the variables of the monomials. Lastly, we again use a look-up table to propagate the monomial through the S-Box according to Property 1. After each round it is possible to filter out all monomials that do not match the weight patterns.

In contrast to previous works, this limits the transitions at the MixColumns and S-Box operations and filters monomials that are not in accordance with the weight patterns. Furthermore, Zeng and Tian [38] choose the key variables in such a way that a minimal number of transitions are possible at the S-Box. This makes it possible to determine the coefficients needed to construct a full-rank integral-resistance matrix for AES.

## 3 Revisiting the Requirements for the S-Box

Until now, we did not consider the properties of the S-Box. However, in order to use the minimal-transitions method, the S-Box must have certain properties. For this, Zeng and Tian [38] introduce the notion of a complete S-Box.

**Definition 3 (Complete S-Box [38]).** *Let $S$ be a $k$-bit S-Box such that $\boldsymbol{y} = S(\boldsymbol{x})$, where $\boldsymbol{y} = (y_1, y_2, \ldots, y_k)$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in \mathbb{F}_2^k$. If the monomial $\boldsymbol{x}^{\boldsymbol{u}}$ appears in the ANF of some output bit $y_i$ for every vector $\boldsymbol{u}$ with $\mathrm{hw}(\boldsymbol{u}) = k-1$, then $S$ is called a complete S-Box.*

This notion is necessary because, due to our key selection, we only allow S-Box transitions from monomials of degree $k-1$ to 1 (besides the 0 and $k$ degree transitions). Especially in the first round (see Figure 2), it is necessary that each $k-1$ degree monomial at the S-Box input can propagate to a degree 1 output.

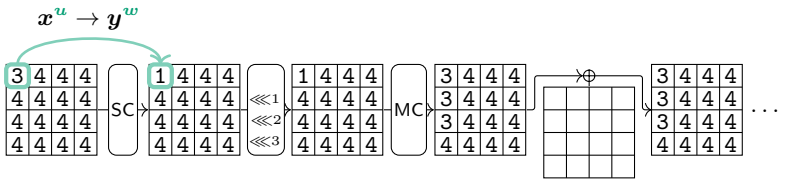



Fig. 2: Example degree propagation for a SPN cipher with 4-bit S-Boxes.

However, this is not a sufficient criterion. Even when we use a complete S-Box, it might not be possible to construct a full-rank integral-resistance matrix. Consider a 64-bit SPN cipher with the ORTHROS S-Box. Using the minimal-transitions method of Section 2.3, we potentially can reach all 63-degree monomials after 7 rounds from any output coordinate $i$. We focus on the values pos-

Table 1: Monomial Prediction Table of the ORTHROS S-Box. This table shows the valid transitions from the input monomial $\boldsymbol{x}^{\boldsymbol{u}}$ to the output monomial $\boldsymbol{y}^{\boldsymbol{w}}$.

| $\boldsymbol{u}$ \ $\boldsymbol{w}$ | 0 | 1 2 4 8 | 3 5 6 9 a c | 7 b d e | f |
|---|---|---|---|---|---|
| 0 | × | × . . . | . . . . . . | . . . . | . |
| 1 | . | × . . . | . . . . . . | . . . . | . |
| 2 | . | × × . . | . . . . . . | . . . . | . |
| 4 | . | . × . . | × . . . . . | . . . . | . |
| 8 | . | . . . × | . . . × . . | . . . . | . |
| 3 | . | × × × . | . . . . . . | . . . . | . |
| 5 | . | . × . × | × . . . . . | . . . . | . |
| 6 | . | . × × . | × . × . . . | . . . . | . |
| 9 | . | . × . . | . . . × × . | . . . . | . |
| a | . | × . . . | × . . . × . | . × . . | . |
| c | . | . . × . | . × × . × × | × × × × | × |
| 7 | . | × × × . | × × × × . × | . . × . | . |
| b | . | × . . . | × . × . × × | . × . × | . |
| d | . | . × . × | . × × . . . | × × × × | × |
| e | . | . . × × | . . × × . × | . . × × | × |
| f | . | . . . . | . . . . . . | . . . . | × |

sible for the coefficients $a^{(i)}_{\bar{\boldsymbol{e}}_1,\boldsymbol{v}_j}, a^{(i)}_{\bar{\boldsymbol{e}}_2,\boldsymbol{v}_j}, a^{(i)}_{\bar{\boldsymbol{e}}_3,\boldsymbol{v}_j}, a^{(i)}_{\bar{\boldsymbol{e}}_4,\boldsymbol{v}_j}$ in 4 columns of the integral-resistance matrix. Due to Property 1 and as shown in Figure 2, the input monomials $\boldsymbol{x}^{\bar{\boldsymbol{e}}_1}, \boldsymbol{x}^{\bar{\boldsymbol{e}}_2}, \boldsymbol{x}^{\bar{\boldsymbol{e}}_3}, \boldsymbol{x}^{\bar{\boldsymbol{e}}_4}$ can only be reached when in the first round the output monomials of the first S-Box have degree 1 and the output of the rest of the S-Boxes has degree 4. Assume that for a disjoint set of key monomials $\boldsymbol{v}_j$ (corresponding to a row in the integral-resistance matrix), we are able to reach all possible degree 1 monomials $\boldsymbol{y}^{\boldsymbol{w}}$ contained in the output of the first S-Box, where $\boldsymbol{w} \in \{1, 2, 4, 8\}$. For each key monomial, we therefore get one or more of these degree 1 monomials at the output of the first S-Box. Using the MPT shown in Table 1, we can now propagate these monomials backward through the S-Box. Each degree 3 input monomial can be reached from the degree 1 output monomials for the ORTHROS S-Box. Definition 3 is therefore satisfied, and all coefficients in the 4 columns of our integral-resistance matrix can be 1. However, the values of our coefficients are determined by a linear combination of the degree 3 to degree 1 transitions in the MPT. Since in the case of the ORTHROS S-Box these transitions are not full rank, it is also not possible to reach full rank in the integral-resistance matrix. This leads to the following S-Box requirement.

**Definition 4 (Max-degree full-rank S-Box).** *A $k$-bit S-Box is max-degree full-rank iff the submatrix of the MPT encompassing all transitions $\boldsymbol{x}^{\boldsymbol{u}}$ to $\boldsymbol{y}^{\boldsymbol{w}}$, with $hw(\boldsymbol{u}) = k - 1$ and $hw(\boldsymbol{w}) = 1$, is full rank.*

We have to ensure the S-Box is max-degree full-rank when showing the integral-resistance of ciphers using the minimal-transitions method. Since the AES S-Box has this property, the results provided by Zeng and Tian [38] are still valid. Due to the structure of a cipher, it can happen that the only way to show the integral-resistance is via degree $k - 1$ to 1 transitions (besides degree

0 and k transitions) in the first S-Box layer. In this case, we can make a general statement about the integral resistance of a cipher (see Property 3).

# 4 Upper-Bounding the Degree by a Monomial Prediction Abstraction

Zeng and Tian [38] use the MILP model described in Section 2.3 strictly to derive the weight pattern of the keys and the states. While this bit-based model is able to find the weight pattern for AES, we argue that using a faster cell-based model is sufficient for this task. Furthermore, we believe that this abstraction of the monomial prediction technique has some further application in upper bounding the degree. While for 64-bit ciphers with sparse linear layers we can use monomial prediction to provide an upper bound of the degree [21], this becomes infeasible for ciphers like Rijndael-256. Furthermore, we can gauge the degree-propagation potential of certain cipher constructions without taking into account the internals of the S-Box and linear layer. For designing a cipher, this is a valuable tool to settle on the overall structure of the cipher. In this section, we introduce the cell-based degree propagation MILP model. Next, we refine the degree propagation for bit-sliced applied MixColumns operations and bit permutations. At the end we present the upper bounds on the minimal degree for various ciphers and compare these bounds with the bounds from other works.

## 4.1 Degree Propagation Abstraction

We provide a cell-based abstraction of the bit-based MILP model used by Zeng and Tian [38] to find the weight patterns of SPN ciphers like AES. For easy generalizability, we define a round of an SPN cipher in terms of AES operations by $R = \text{AddRoundKey} \circ \text{MixColumns} \circ \text{SubBytes} \circ \text{ShiftRows}$. Due to the commutative property of the SubBytes and ShiftRows as well as the ShiftRows and AddRoundKey most SPN ciphers can be represented this way. This is not the case when the SubBytes and MixColumns steps are swapped (e.g., QARMAv2 [2]). Although it is still possible to apply the techniques presented in this paper, it would require more tweaking. In the rest of the paper, we use the AES ShiftRows operation (see Figure 3), since it is also used in PRINCE. However, for all the methods presented in this paper, any cell permutation can be used. To group different SPN ciphers into classes, we use a ($m$, $l$, $k$)-SPN notion, similar to the one introduced by Todo [34]. For such an SPN, the SubBytes operation consists of $m$ $l$-bit S-Boxes applied in parallel. Furthermore, the MixColumns operation consists of a $k$-bit linear function.

The main idea is to assign to each cell a variable representing the degree $d_c$ that it contains. For the bits $b$ contained in a cell, we can calculate $d_c$ from the state mask $\boldsymbol{u}$ with $hw(\boldsymbol{u}[b])$. Thus, each variable can take a number between 0 and the size of the cell $l$. Next, we model the degree propagation through each operation of the cipher. The non-linear S-Box operation is applied to each cell. Since the S-Box is a balanced Boolean function, each $l$-degree monomial

propagates to an $l$-degree monomial. Furthermore, a cell without a monomial retains this property after the S-Box application. Given a complete S-Box, the third case encompasses a degree between 1 and $l-1$, and due to the non-linear function, can lead to a degree between 1 and $l-1$. Using a `Minizinc`-like syntax, we can model this with the $\mathit{SBox}$ constraint

$$\mathit{SBox}(in, out) \coloneqq \begin{cases} \mathtt{if}\ in = l\ \mathtt{then}\ out = l \\ \mathtt{elseif}\ in = 0\ \mathtt{then}\ out = 0 \\ \mathtt{else}\ 0 < out < l\ \mathtt{endif}. \end{cases}$$

The $\mathit{ShiftRows}$ permutes the indices of the cells. Furthermore, the MixColumns operation also preserves the degree but can mix it between cells. For AES, we apply the $\mathit{MixColumns}$ to each column containing 4 cells as

$$\mathit{MixColumns}(in, out) \coloneqq \mathit{sum}(in) = \mathit{sum}(out). \tag{1}$$

The key addition consists of $l$ parallel XOR applications (see Definition 7). Thus, we count how many key variables get added to each cell and subtract that number from the output cell with

$$\mathit{KeyAddition}(in, out, key) \coloneqq in = out - key.$$

With these few constraints, we can model the degree propagation of AES, as illustrated in Figure 3. By maximizing the degree at the input for each degree 1 output, we get an upper bound on the possible degree for each coordinate function of the cipher. To show the integral-resistance property, we need at least as many rounds as it takes to reach a full degree for all coordinate functions. In a second step, we maximize the number of key monomials present in the ANF. This has the effect that each output of the S-Box always gets assigned the minimum degree possible (1 in the third case of $\mathit{SBox}$, see Property 1). Thus, when propagating the monomials through the cipher, we restrict each state to a certain degree. For a $k$-bit linear function, this drastically reduces the number of monomial transitions from $2^k$ to $\binom{k}{i}$.

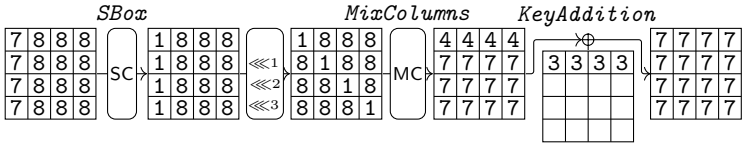



Fig. 3: Possible degree propagation of AES. We first maximize the input degree given a fixed output degree. In a second step, we maximize the degree of the key monomials added to the state. This has the effect that before the MixColumns operation, each cell can only contain monomials of degree 0, 1, or 8.

### 4.2 Improved Modeling

For ciphers with a complex linear layer like AES, PRINCE, and LED we assume that the total degree at the input and at the output of the MixColumns operation are the same and do not impose any further constraints. However, for ciphers like SKINNY or PRESENT, which either feature bit-sliced applications of a MixColumns matrix or a bit-permutation over the whole state, we can further refine our degree bounding model.

To do this, we first need to adapt the basic bitwise operations [33] and extend them to cell-based operations. Since this means that for an $l$-bit cell we apply $l$ bit-based operations in parallel, we can simply extend the domain of the involved variables from binary to 0 to $l$. This leads us to the adapted, cell-based definitions of the COPY, XOR, and AND operations.

**Definition 5 (Copy, cell-based).** *The degree propagation through* $\texttt{Copy} : \mathbb{F}_l \to \mathbb{F}_{2l}$; $(a) \mapsto (b_0, b_1)$ *can be modeled by the equality*

$$\texttt{Copy}(a, b_0, b_1) \coloneqq a - b_0 - b_1 = 0.$$

**Definition 6 (AND, cell-based).** *The degree propagation through* $\texttt{AND} : \mathbb{F}_{2l} \to \mathbb{F}_l$; $(a_0, a_1) \mapsto (b)$ *can be modeled by the inequalities*

$$\texttt{AND}(a_0, a_1, b) \coloneqq \begin{cases} b - a_0 \geq 0, \\ b - a_1 \geq 0, \\ b - a_0 - a_1 \leq 0. \end{cases}$$

**Definition 7 (XOR, cell-based).** *The degree propagation through* $\texttt{XOR} : \mathbb{F}_{2l} \to \mathbb{F}_l$; $(a_0, a_1) \mapsto (b)$ *can be modeled by the equality*

$$\texttt{XOR}(a_0, a_1, b) \coloneqq a_0 + a_1 - b = 0.$$

**Bit-Sliced Linear Layers** With these operations, it is possible to model linear layers applied in a bit-sliced fashion. For example, we model the application of the MixColumns matrix $\mathbf{M}$ of SKINNY by using intermediate variables $t \in \mathbb{F}_2$

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \qquad \texttt{MCSKINNY}(a, b) \coloneqq \begin{cases} \texttt{Copy}_3(a_0, t_0, b_1, t_{30}), \\ \texttt{Copy}_3(a_2, t_{02}, t_{22}, t_{32}), \\ \texttt{XOR}_3(t_{00}, t_{02}, a_3, b_0), \\ \texttt{XOR}(a_1, t_{22}, b_2), \\ \texttt{XOR}(t_{30}, t_{32}, b_3). \end{cases}$$

Since the number corresponding to the degree contained in each cell does not encode the actual bit mask of the monomial, we allow degree propagations that are invalid when doing bit-based monomial prediction.

**Example 1 (Invalid degree propagation)** *Consider two input cells that we bitwise* XOR*. The monomial in each cell can be represented by* $\boldsymbol{x}^{\boldsymbol{u}}, \boldsymbol{u} = \texttt{1100}$*. At each coordinate, only one input can be active when applying the bitwise* XOR *rule. Thus, this is an invalid monomial transition. However, when using our degree propagation model, we lose the information on which coordinates of* $\boldsymbol{u}$ *the* `1`*s are located. Each input cell gets assigned the degree* $a = hw(\boldsymbol{u}) = 2$*. If we apply Definition 7, we get a valid output assignment* $b = 0 + 2 + 2 = 4$*.*

Nevertheless, given a valid monomial propagation, the corresponding degree propagation is always correct. Thus, our model can be used to upper-bound the degree of a cipher.

**Bit Permutations** Ciphers like PRESENT or GIFT use a bit permutation as the linear layer. Thus, we cannot apply the cell-based COPY, AND, or XOR operations. However, as can be seen in Figure 4, each of the 4 input bits of an S-Box comes from 4 different S-Boxes in the previous round. Thus, for each input S-Box that has a degree of 0, the maximal degree of the output S-Box decreases by one. Given the output degrees $a[0], \ldots, a[3]$ of the four S-Boxes of the first layer, we can impose the following constraint on each degree $b$ of one of the S-Boxes of the second layer:

$$\texttt{BitPermutation}(\boldsymbol{a}, b) \coloneqq b \leq 4 - \#\{i = 0..3 \mid \boldsymbol{a}[i] = 0\}.$$

Since the above property also holds for the inverse round function, we can also apply these constraints in the inverse direction.

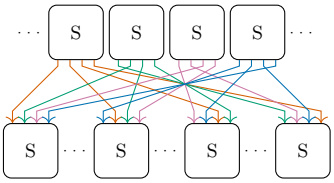



Fig. 4: Part of the GIFT round function.

### 4.3 Results

Using the abstraction model, we implemented various SPN ciphers and bound their minimal degree. We choose a selection of ciphers to demonstrate the different modeling approaches of the linear layer described in Section 4.1. For ciphers with strong mixing layers (AES [13], Beanie [19], and PRINCE [11]) we use the general `MixColumns` (see Equation 1) model. Furthermore, QARMAv2 [2], SKINNY [6], and CRAFT [7] can be represented using bit-sliced mixing and are thus implemented using the respective bit-sliced mixing operations as described

in Section 4.2. Lastly, we use the technique described in Section 4.2 to implement the bit-permutation linear layer of PRESENT [9] and GIFT [4]. We present our results in Table 2 and compare them with results from other works [10,21]. The method of Boura and Canteaut [10] is quite general and does not take into account the full internals of the S-Box and the linear layer. Hebborn et al. [21] on the other hand, use bit-based monomial prediction models to exactly model each part of the cipher. Our approach makes some assumptions about the linear layer and can therefore be seen as a middle ground. This is also reflected in the tightness of our bounds compared to the other methods, as shown in Table 2.

Table 2: Upper bounds on the minimal degree $d$ of various ciphers. This entails the existence of an integral distinguisher with data complexity $2^{d+1}$. Bit-based monomial prediction is indicated by Exact. Cell-based, Bit-Sliced and Bit Perm. use the degree abstraction models from Equation 1, Section 4.2, and Section 4.2, respectively. The degree bound of Boura and Canteaut [10] is also shown.

| Cipher (Type) | Model | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(8, 4, 16)$ | Cell-Based | 3 | 9 | 21 | 27 | 30 | 31 | | | | | |
| Beanie | Exact | **6-round integral resistance** | | | | | | | | | | |
| $(16, 4, 16)$ | Cell-Based | 3 | 9 | 27 | 49 | 57 | 61 | 63 | | | | |
| PRINCE | Exact | **7-round integral resistance** | | | | | | | | | | |
| $(16, 4, 64)$ | Cell-Based | 3 | 9 | 27 | 51 | 59 | 62 | 63 | | | | |
| (16,8,32) | Cell-Based | 7 | 28 | 112 | 124 | 127 | | | | | | |
| AES | Exact [38] | 5-round integral resistance | | | | | | | | | | |
| $(32, 8, 32)$ | Bound [10] | 7 | 28 | 113 | 235 | 250 | 254 | 255 | | | | |
| Rijndael-256 | Cell-Based | 7 | 28 | **112** | **232** | **249** | 254 | 255 | | | | |
| QARMA-MC | Bit-Sliced | 3 | 9 | 27 | 49 | 57 | 61 | 63 | | | | |
| SKINNY-64-MC | Bit-Sliced | 3 | 6 | 9 | 21 | 34 | 50 | 58 | 61 | 62 | 63 | 63 |
| SKINNY-64 | Exact [21] | 2 | 3 | 8 | 17 | 33 | 47 | 58 | 61 | 62 | 62 | 63 |
| SKINNY-64 | Exact [22] | 13-round integral resistance | | | | | | | | | | |
| SKINNY-128-MC | Bit-Sliced | 7 | 14 | 21 | 49 | 79 | 111 | 123 | 125 | 126 | 127 | |
| CRAFT-MC | Bit-Sliced | 3 | 6 | 12 | 22 | 36 | 51 | 58 | 60 | 62 | 63 | 63 |
| CRAFT | Exact [22] | 14-round integral resistance | | | | | | | | | | |
| PRESENT-Perm. | Bit Perm. | 3 | 9 | 27 | 48 | 58 | 62 | 63 | | | | |
| PRESENT | Exact [21] | 2 | 4 | 8 | 18 | 42 | 54 | 59 | 62 | 63 | | |
| GIFT | Exact [21] | 2 | 4 | 9 | 21 | 42 | 53 | 59 | 62 | 63 | | |
| GIFT-128-Perm | Bit Perm. | 3 | 9 | 27 | 81 | 112 | 122 | 126 | 127 | | | |

For PRINCE, Beanie, and AES our model reaches the max degree in the same round that it is possible to show the integral-resistance property. This indicates that the complex linear functions allow for a close to optimum degree

propagation. For SKINNY, our model bounds the minimal degrees similarly to the bit-based monomial prediction model [21]. Once the degree approaches its maximum, this is especially true. It takes 13 and 14 rounds to show the integral-resistance property for SKINNY and CRAFT [22], respectively. The gap to the round it takes our model to reach max degree is 3 and 4 rounds, respectively. Interestingly, using the MIDORI-like MixColumns in QARMAv2 achieves max degree in the same number of rounds as (16, 4, 16)-SPN. Note that designers of QARMAv2 only claim security up to $2^{56}$ data. This is why the longest distinguisher is covers only 5 rounds instead of 6. Applying our degree propagation model to PRESENT and GIFT leads to bounds far from bounds obtained by bit-based monomial prediction. For ciphers with bit permutations, the internals of the S-Box play an important role in the degree propagation. Since we do not take these into account, our method has only limited meaning for these ciphers.

For Rijndael-256 our method improves the bounds on the minimal degree slightly for 3, 4, and 5 rounds compared to the method from Boura and Canteaut [10]. For 3 rounds the bound of 112 is quite far away from the best integral distinguisher (log data complexity of 56 [34]). However, the 4-round bound of 232 matches the best integral distinguisher having a log data complexity of 232.

*Using properties of the S-Box.* In case our S-Box is not max-degree full-rank, there must be degree $n-1$ to degree 2 transitions possible at the S-Boxes of the first round to be able to construct a full-rank integral-resistance matrix (see Section 3 for details). Note that this has to be the case even when not using the minimal-transitions method presented in Section 2.3. The ciphers LED, Joltik-BC, and SKINNY do not have max-degree full-rank S-Boxes. We can adapt our model to find a lower bound on the number of rounds it takes to show the integral-resistance property for these ciphers. This is done by adding a constraint that disallows degree 3 to degree 1 propagations at the S-Box of the first round.

For SKINNY we need at least 11 rounds for integral-resistance, compared to the 10-round upper bound on the minimal degree in Table 2. (16, 4, 16)-SPN ciphers without max-degree full-rank S-Boxes (like LED and Joltik-BC) require at least 8 rounds for integral-resistance. This leads to the following property.

*Property 3.* For all (16, 4, 16)-SPN ciphers with AES-like ShiftRows and without max-degree full-rank S-Boxes there exists a 7-round integral distinguisher.

*Proof.* Using the abstraction model, the degree 63 to 1 propagation of 7 rounds of (16, 4, 16)-SPN with AES-like ShiftRows is satisfiable (see Figure 5). When adding the condition needed for ciphers without max-degree full-rank S-Boxes, i.e., that there must exist degree 3 to degree 2 transitions at the first S-Box layer, the model is unsatisfiable. A valid monomial transition implies a valid degree propagation model. In turn, we can conclude that no valid monomial transitions exist for the model without max-degree full-rank S-Boxes. Therefore we can only reach degree 63 monomials via a degree 3 to degree 1 transition at the first S-Box application. Since we do not have max-degree full-rank S-Boxes, this enables us to construct an input set leading to an integral distinguisher.

However, when using (16, 4, 16)-SPN with max-degree full-rank S-Boxes (like PRINCE), it is possible to reach integral-resistance after 7 rounds (see Section 5).

### 4.4 Obtaining Weight Patterns for the Minimal Transition Method

We can use our degree abstraction model to find the state and key weight patterns for the minimal-transitions method. The notion of weight patterns (WP) is introduced by Zeng and Tian [38] for describing possible degree assignments within each column of the state. Given such a assignment of a state, we define the weight pattern $\mathrm{WP}_i^{col}$ as a $k$-tuple containing the degrees of the $k$ cells contained in column $i$ of the state. We extend the usage of weight patterns to also include a weight pattern over the whole state. For an SPN cipher represented by $l$ columns, WP is an $l$-tuple containing the degrees of each column. To obtain the WP from our degree abstraction model, we constrain the input state to a max-degree state and the output to a degree-1 state. Then we maximize the key variables added to the state and save the obtained key WP. Next, we rerun the model with this key WP and all possible max degree inputs and degree 1 outputs. For the given key WP, this allows us to obtain all state WP possible. For example, Figure 5 shows all possible state WP for a given key WP of PRINCE.

*Edge cases.* What happens when maximizing the key patterns for different input/output cell configurations leads to key masks with different Hamming weight? In this case it is still possible to construct the integral-resistance matrix. We detail the procedure in case there are two input/output cell configurations, $IO_{high}$ and $IO_{low}$, leading to the key masks $K_{high}$ and $K_{low}$, respectively. Assume that the Hamming weight of $K_{high}$ is larger than the one from $K_{low}$. Using $K_{high}$ for the $IO_{low}$ input/output configuration leads to no max-degree state monomials. Using $K_{low}$ for the $IO_{high}$ input/output configuration violates Property 1 and Property 2, which invalidates the minimal-transitions method. When constructing the integral-resistance matrix $\mathcal{I}$, this leads to the following block matrix, where * is unknown:

$$\mathcal{I} = \begin{matrix} \\ k \in K_{low} \\ k \in K_{high} \end{matrix} \begin{matrix} \begin{matrix} IO_{low} & IO_{high} \end{matrix} \\ \begin{pmatrix} \mathcal{I}_1 & \mathcal{I}_2 \\ \mathcal{I}_3 & \mathcal{I}_4 \end{pmatrix} \end{matrix} = \begin{matrix} \\ k \in K_{low} \\ k \in K_{high} \end{matrix} \begin{matrix} \begin{matrix} IO_{low} & IO_{high} \end{matrix} \\ \begin{pmatrix} 0/1 & * \\ 0 & 0/1 \end{pmatrix} \end{matrix}$$

$\mathcal{I}$ is full-rank in case $\mathcal{I}_1$ and $\mathcal{I}_4$ are full-rank. This trivially extends to having more than two different key pattern Hamming weights. In our experiments, only ciphers with bit-sliced MixColumns operations need to make use of this technique of using multiple different weights for the key patterns. In principle, this property would enable applying the minimal-transitions method to ciphers like SKINNY, MIDORI and CRAFT. However, all of these ciphers feature S-Boxes without max-degree full-rank S-Boxes, hindering the application.

*Additional key weight patterns.* All key WP with the maximum Hamming weight lead to Property 1 and Property 2 from Section 2.3 being applicable. However,

there may also be key WP with lower Hamming weight that do not violate Property 1 and Property 2.

**Proposition 1.** *Assume a cipher reaches max degree after $r$ rounds with our degree propagation model. When we require key variables of a certain round to be present in the ANF of the output and the model is satisfiable, then Property 1 and Property 2 still hold as long as we maximize the degree of the key monomials.*

*Proof.* Assume $|k_i|$ refers to the degree of the key monomial added to the state in round $i$. When maximizing the degree of the key monomials present in the ANF, this leads to a sequence of key degrees potentially starting and ending in degree zero, but with nonzero degrees in the middle. We refer to the first round key with nonzero degree as $k_m$ and the last one as $k_{m+l}$, i.e.,

$$\{|k_0|, \ldots, |k_m|, \ldots, |k_{m+l}|, \ldots, |k_r|\} = \{0, \ldots, > 0, \ldots, > 0, \ldots, 0\}.$$

Enforcing $|k_j| > 0$ for $m + l < j \leq r$ (resp. $0 \leq j < m$) potentially leads to a decrease (resp. increase) in the degree of the state monomials in round $m+l$ (resp. $m$). This in turn leads to a decrease in the degree of $k_m, \ldots, k_{m+l}$. However, since we require a satisfiable model, we can still reach our max degree monomials. We now need to show that Property 1 and Property 2 are still applicable.

- Property 1. For an $n$-bit non-linear function, $hw(\boldsymbol{u}) = hw(\boldsymbol{w}) = 0)$ and $hw(\boldsymbol{u}) = hw(\boldsymbol{w}) = n)$. In case $hw(\boldsymbol{w}) = 1$ we have $hw(\boldsymbol{u}) \in \{1, .., n-1\}$ for a key mask $\boldsymbol{v}$. However, if $hw(\boldsymbol{u}) \neq n-1$ then there exists a $\boldsymbol{v'}$ with $hw(\boldsymbol{v'}) > hw(\boldsymbol{v})$ such that $hw(\boldsymbol{u'}) = n-1$.
- Property 2. Due to the linear function, we have $hw(\boldsymbol{u}) \leq hw(\boldsymbol{w})$. In case $hw(\boldsymbol{u}) < hw(\boldsymbol{w})$ for a key mask $\boldsymbol{v}$, then there exists a $\boldsymbol{v'}$ with $hw(\boldsymbol{v'}) > hw(\boldsymbol{v})$ such that $hw(\boldsymbol{u'}) = hw(\boldsymbol{w})$.

Thus, if Property 1 and Property 2 do not hold, then the degree of the key monomials $hw(\boldsymbol{v})$ is not being maximized. This contradicts our assumption.

## 5 Integral Resistance Property of PRINCE and Beanie

In this section we introduce the `intres` framework for showing the integral-resistance with the minimal-transitions method. We use Theorem 2 to significantly speed up the offline-phase compared to previous work. Afterward, we discuss the monomial propagation algorithm and how to select a key mask.

### 5.1 Improved Transition Table Computation

To compute the valid monomial transitions of the MixColumns operation, Zeng and Tian [38] use a branch-and-bound method implemented in CUDA. They state that for AES it takes a week to compute all the transitions needed for the online phase. We use Theorem 2 to significantly reduce the time it takes to calculate the valid monomial transitions of the MixColumns operation. In other

words, we test whether submatrices of $M$ are invertible. Besides only needing to calculate the transitions for $hw(\boldsymbol{u}) = hw(\boldsymbol{w})$ (Property 2), we can utilize some further constraints on $\boldsymbol{u}$ and $\boldsymbol{w}$. Due to Property 1, we only need to consider cells with Hamming weight $0, 1, k$ at the input. Lastly, we can reduce the number of calculations further by confining ourselves to transitions following the weight pattern described in Section 4.4. Algorithm 1 outlines how to calculate all valid $\boldsymbol{u}$ for a given output mask $\boldsymbol{w}$. The function GetValidInput() returns all input masks that are left under the aforementioned constraints.

**Algorithm 1** Offline phase: Computing valid transitions for MixColumns

```
Require: Output mask w ∈ F_2^n, MixColumns matrix M
    for all u ∈ GetValidInput(hw(w)) do
        if M[w, u].isInvertible then
            valid_transitions[w].add(u)
    return valid_transitions
```

*Super S-Boxes.* Derbez and Fouque [14] precompute the valid monomial transitions of super S-Boxes. Due to the commutative property of the S-Box and ShiftRows operations, it is possible to reorder the application of these operations for ciphers like AES. Because of our iterative algorithm used later on, we refer to super S-Boxes as an S-Box application followed by a MixColumns operation, omitting an additional S-Box application. In Algorithm 2 we extend Algorithm 1 to also include the S-Box transitions in the offline phase. The function GetSboxTransitions() is realized with a look-up table containing all max-degree inputs for each degree 1 S-Box output (see Table 1).

**Algorithm 2** Offline phase: Computing valid transitions including the S-Box

```
Require: Output mask w ∈ F_2^n, MixColumns matrix M
    for all t ∈ GetValidInput(hw(w)) do
        if M[w, t].isInvertible then
            for all u ∈ GetSboxTransitions(t) do
                if u ∈ valid_transitions[w] then
                    valid_transitions[w].remove(u)
                else
                    valid_transitions[w].add(u)
    return valid_transitions
```

Using super S-Boxes has the advantage that during the offline phase, invalid monomial transitions stemming from an even number of monomial trails do not get added to our transition table. However, each table entry has more possible transitions. When using the early abort technique explained in Section 5.3, this can lead to an overall increase in the runtime of the online phase. Whether using

super S-Boxes is beneficial must therefore be evaluated when implementing a cipher. It takes about 5 CPU-minutes to calculate the valid monomial transitions of the AES MDS matrix when implementing the above method in `sage`.

### 5.2 Selecting the Key Mask

Zeng and Tian [38] do not explicitly state how they choose the key masks for AES. In Table 3b we list the four different key masks they use per S-Box. This shows that they select the key masks in a way such that probabilistically, many state monomials are removed from the monomial propagation. In turn, a significant reduction in the runtime of the program is achieved. As an example, adding the key mask `0x0e` before an S-Box application removes 14 out of 25 possible degree 7 to degree 1 transitions. We highlight this in Table 3a, where the three rows that are invalidated by the key mask account for the $4 + 5 + 5 = 14$ transitions.

Table 3: Key addition followed by AES S-Box. We highlight the impact of the key mask on the validity of degree 7 to degree 1 monomial transitions.

(a) We highlight the invalidated transitions when adding key variables with the mask $\boldsymbol{v} =$ `0x0e` to the monomial.

| $\boldsymbol{u}$ \ $\boldsymbol{w}$ | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | $\boldsymbol{v}$ in $\boldsymbol{u}$ |
|---|---|---|---|---|---|---|---|---|---|
| 7f | . | . | . | × | . | . | . | . | × |
| bf | . | . | . | . | × | . | . | . | × |
| df | × | × | × | . | × | . | . | . | × |
| ef | . | × | × | × | . | × | . | . | × |
| f7 | × | × | . | . | . | × | × | . | . |
| fb | . | × | × | × | . | . | × | × | . |
| fd | × | . | × | × | × | . | . | × | . |
| fe | . | . | × | . | . | . | . | . | × |

(b) Degree 3 key masks used to cancel a big portion of S-Box transitions. This speeds up the process of showing the integral-resistance property of AES [38].

| Key mask | # removed transitions |
|---|---|
| `0x0e` | 14/25 |
| `0x07` | 11/25 |
| `0x25` | 10/25 |
| `0x29` | 9/25 |

When a non-zero key mask is applied in the last round of the cipher, care has to be taken to allow transitions to all coordinate functions. For example, using `0x0e` would invalidate all transitions to the coordinates `0x40` and `0x80`. Zeng and Tian [38] remedy this by using the key masks `0x25` and `0x29` in the last round. Both cases enable a transition to `0x40` and `0x80`.

It is not always preferable to choose the key masks that remove the most transitions. In the case of AES, `0x07` is inferior to the key mask `0x16` that removes 14 instead of 11 transitions. The reason for this is to get a more diverse set of intermediate state monomials and therefore increase the chance that adding a row to the integral resistance matrix also increases its rank. Furthermore, for ciphers with small S-Boxes and sparse linear layers like the one described in Section 6, using the optimal key masks leads to no state monomials being left after a few rounds. Thus, the monomial propagation yields no result.

In general, for ciphers with large S-Boxes and linear layers, selecting the key masks in a way that many transitions get invalidated makes calculating the integral-resistance matrix via the minimal transition method viable. For ciphers with small S-Boxes we typically can calculate a row of the integral-resistance matrix in comparably little time. We therefore prefer a diverse set of key masks in order to achieve a full-rank integral-resistance matrix with as few rows as possible. Since the ciphers in the paper belong to the second category, our GetKey function returns random key masks that follow the key patterns obtained in Section 4.4. However, in case we use our framework to determine the integral-resistance of ciphers with larger states, the key mask selection should be biased in a way that minimizes the possible transitions.

### 5.3 Integral-Resistance Property

To determine the coefficients of the integral-resistance matrix, we first obtain the weight patterns of the states and the keys (see Section 4.4). For one possible key pattern of PRINCE, Figure 5 lists all possible state patterns. In the offline phase, we calculate the super S-Box transitions according to these state patterns.

In the online phase we recursively propagate a set of monomials $\Upsilon$ backward through each round of the cipher, as outlined in Algorithm 3. For each coordinate function $E_i$, we initialize $\Upsilon = \{\boldsymbol{e}_i\}$. Then we obtain the degree $n-1$ monomials contained in this coordinate function with PropagateMonomials($\Upsilon$, [ ], 0). Afterward, we call the function again with the same round keys (obtained as a return value) for all other coordinate functions $E_j, i \neq j$. We do this by calling PropagateMonomials($\{\boldsymbol{e}_j\}$, key, 0). In the end, we have calculated the values of all coefficients needed for one row of the integral-resistance matrix. After adding this row to our matrix, we check if the matrix has full rank. We repeat this procedure until this condition is met.

Depending on the cipher, some additional optimizations might be beneficial. For ciphers with $k$ columns, PropagateMonomials has $k$ nested loops. With larger $k$, it is advisable to employ an early abort technique in order to avoid running unnecessary inner loops. We can implement this by checking if the new monomial conforms to the weight pattern at each level of the loop. When using this technique, it is favorable to omit the S-Box in the precomputation of the *valid_transitions* (Algorithm 1 instead of Algorithm 2). This reduces the number of possible transitions at each loop and makes the early abort strategy more effective. Furthermore, depending on the cipher, the key masks may lead to parts of the integral-resistance matrix defaulting to 0 coefficients. If we identify these in advance, we can omit calculating these coefficients.

*Integral-Resistance Property of PRINCE.* We show the integral resistance of 7 round PRINCE with `intres`. This closely matches the longest known 6-round integral distinguisher with a data complexity of $2^{62}$ [11] (can also be derived from Table 2). For PRINCE, no key variables are added to the state in the last two rounds (see Figure 5). As a result, 3 coordinate functions lead to the same set of monomials when propagating two rounds backward. Furthermore, one out

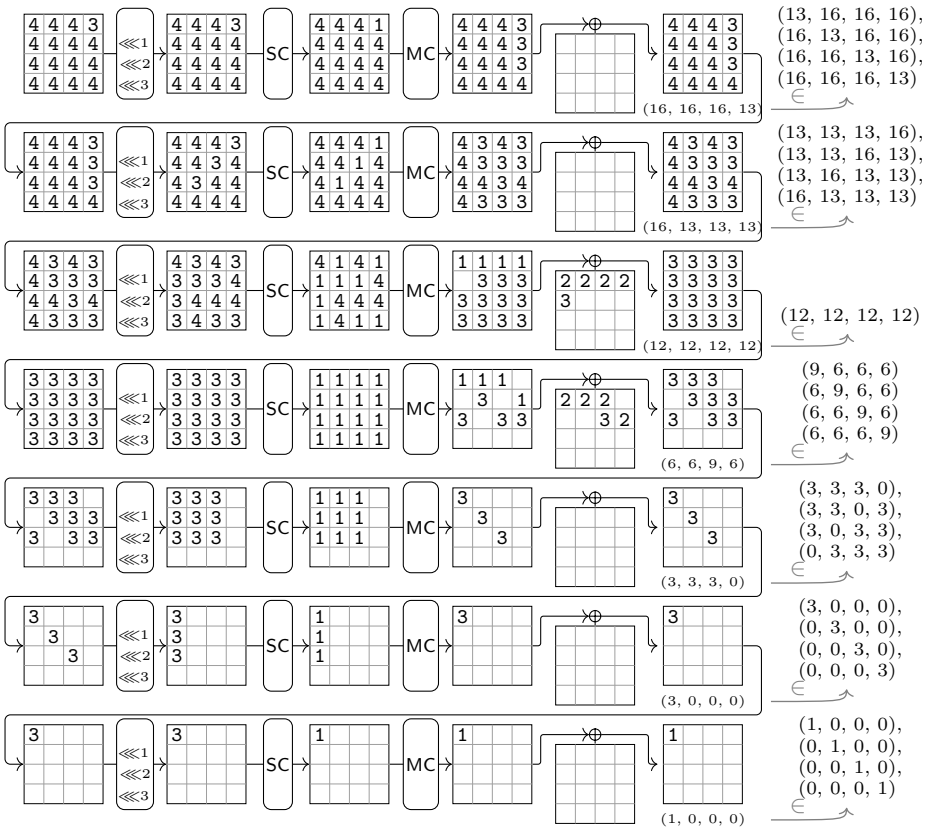



Fig. 5: Degree propagation of PRINCE when specific key variables are added to the state. We include all other possible weight patterns the state can take.

of four coordinate functions leads to no monomials being left in the set. This reduces the runtime of `intres` by a factor of 4. Finding 4096 coefficients of a single row of the integral-resistance matrix takes on average 8 minutes. Obtaining all 4096 rows therefore takes roughly 8 hours using 64 cores. However, with a high probability, this does not yield a full-rank matrix. It took roughly a day to find the right set of key masks for a full-rank integral-resistance matrix.

*Integral-Resistance Property of Beanie.* Using the `intres` framework, we show the integral-resistance of 6 full rounds of the 32-bit cipher Beanie. Beanie is a low-latency cipher recently published at ToSC [19]. The cipher has 4-bit S-Boxes and applies a 16-bit MDS matrix to both columns. Adapting the framework to Beanie is straightforward. The S-Box and MixColumns matrix can be modified by a simple parameter change. For calculating the *valid_transitions* in the offline-phase this is sufficient. To obtain the state and key WP, slight tweaks to the `minizinc` MILP model are needed. In the online-phase we need to change the

**Algorithm 3** Online phase: Recursively propagating a set of monomials

```
Require: rounds, state_pattern, valid_transitions
  function PropagateMonomials(Υ, key, round)
    Υ_new = {}                                          ▷ Set of monomials
    if key[round] is empty then
      key[round] = GetKey(round)
    for all monomial ∈ Υ do
      if monomial = monomial & key then                 ▷ Key variables in monomial
        monomial = monomial & ¬key                      ▷ Remove key variables from monomial
        for all col_1 ∈ valid_transitions[monomial.getColumn(1)] do
          ⋮                                             ▷ For loop for all columns
          monomial = (col_1||col_2||col_3||col_4)
          monomial = Shift(monomial)
          if monomial.state_pattern ∈ state_pattern[round] then
            if monomial ∈ Υ_new then
              Υ_new.remove(monomial)
            else
              Υ_new.add(monomial)
    if Υ_new = {} or round + 1 == rounds then
      return (Υ_new, key)
    else
      return PropagateMonomials(Υ_new, key, round + 1)
```

data type storing the monomials to a 32-bit integer. The cipher has two columns, so we only need two for loops in Algorithm 3 and adapt the Shift operation.

Finding the 1024 coefficients of a single row of the integral-resistance matrix takes on average less than 10 seconds. When using 16 cores, obtaining all 1024 rows therefore takes roughly 10 minutes. Due to the strong mixing of the MDS matrix and many degree 3 to degree 1 transitions in the S-Box, this matrix is already almost full-rank, and only a few additional rows are required.

## 6 Towards Ciphers with Minimal Latency and Integral Resistance

The PRINCE MixColumns layer is sparse and only requires two sequentially applied XOR gates per output bit. However, using an even sparser linear layer that only needs one sequential XOR application per output would further decrease the latency of the cipher. In this section, we optimize the linear layer of a PRINCE-like cipher in such a way that it retains the 7-round integral resistance property with a more lightweight linear layer. Obviously, only optimizing the integral resistance can easily lead to a cipher vulnerable to other types of attacks. With the push towards low-latency pseudorandom functions (PRFs) like ZIP-ciphers [18], Orthros [3] and Gleeok [1], we still believe such a linear layer is useful. All these PRFs have multiple branches that get XORed at the output. Cryptographically, this has the effect that differential and linear attacks get weaker proportional

to the number of branches. However, integral attacks only needs to attack the strongest of the branches. Thus, having the lowest latency possible in a branch that retains high resistance against integral attacks is important.

*Motivation.* The reason we optimize the MixColumns layer of PRINCE are the following:

- **4-bit Sbox.** In general, a multiple of 4-bit and 8-bit S-Boxes fit a 64-bit state. Due to the lower latency, 4-bit S-Boxes are generally preferred for this setting. Even the 128-bit ciphers Orthros [3] and Gleeok [1] use 4-bit S-Boxes. Since PRINCE is a low-latency cipher, its S-Box is chosen accordingly.
- **16-bit MixColumns.** As can be seen in Table 2 ($(16, 4, 64)$-SPN), even when using a 64-bit matrix multiplication as a linear layer, it still takes 7 rounds to reach the 63-degree monomials. Since it is easier to optimize a 16-bit matrix than a 64-bit matrix, we kept the overall structure of PRINCE.

The latency of multiplying a binary matrix $M$ in hardware depends on the number $x$ of 1s in each column. We need at least $\lceil \log_2(x) \rceil$ XOR gates in series, when using XOR gates with a fan-in number of 2. Thus, we aim to minimize the number of 1s in each column of $M$. Furthermore, reducing the overall number of 1s in $M$ reduces the area and energy used by the hardware implementation.

*$\mathcal{ZR}$ method in MILP.* In order to use Theorem 2 in an SAT program, Hu et al. [24] use the following proposition.

**Proposition 2 ([24]).** *For a primitive matrix $M \in \mathbb{F}_2^{n \times n}$, a division trail $(\boldsymbol{u}, \boldsymbol{v})$ is valid if and only if $(\boldsymbol{u}, \boldsymbol{v})$ meets the following constraints:*

$$E(i,j) \cdot v_i = \sum_{k=0}^{n-1} M(i,k) \cdot v_i \cdot u_k \cdot M^{expand}_{\boldsymbol{v},\boldsymbol{u}}(k,j) = 0, \quad \text{for } 0 \leq i,j \leq n-1,$$

*with $E$ an $n \times n$ identity matrix and $M^{expand}_{\boldsymbol{v},\boldsymbol{u}} \in \mathbb{F}_2^{n \times n}$ an auxiliary matrix with $n^2$ elements.*

Hu et al. state that it is not possible to implement this in MILP due to degree-4 constraints. However, since all variables are in $\mathbb{F}_2$, the multiplication is realized as a logical AND. We can easily model AND with 4 inputs $a_i$ and output $b$ via the following constraints to minimize the matrix $M$ in MILP:

$$b \leq a_0, \quad b \leq a_1, \quad b \leq a_2, \quad b \leq a_3, \quad a_0 + a_1 + a_2 + a_3 - 3 \leq b.$$

*Global optimization.* Ideally, we want to find a binary matrix $M$ with minimal latency and a minimal number of XORs for which the corresponding cipher still exhibits the integral-resistance property after 7 rounds. For this, we would need a MILP program modeling the monomial trails from all $n-1$ degree inputs to all degree 1 outputs in parallel. Afterward, we could minimize the Hamming weight of $M$. However, such a model is computationally too expensive to run. Furthermore, there might be additional monomial trails canceling out some max-degree monomials. Due to these obstacles, we opt for a heuristic approach to find a matrix $M$ with a close to optimal Hamming weight.

*Heuristic approach.* We first determine a set of $WP_{cell}$ before and after the MixColumns operations. These weight patterns are obtained by reusing the degree propagation model described in Section 4.4. For each possible degree $n-1$ input and degree 1 output, we generate a valid degree allocation of the intermediate states. From this, we then extract the $WP_{cell}$ before and after the MixColumns operations. This leads to the weight patterns of the input and output of MixColumns displayed in Figure 6.

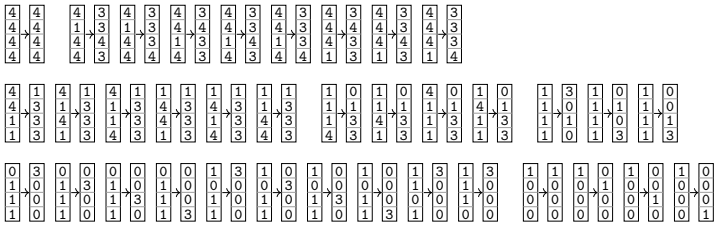


Fig. 6: A set of WP of the MixColumns operation. Using these potentially leads to each degree $n-1$ input being able to propagate to each degree 1 output.

We then construct a MILP model that ensures each weight pattern transition is possible with a MixColumns matrix $M$. For this, we enforce the `enforce_WP` constraint for all the input and output weights shown in Figure 6, defined by

$$\texttt{enforce_WP}(M, in, out) \coloneqq \exists u, w \in \mathbb{F}_2^{16} \begin{cases} \forall_{i=0}^{3} hw(u[4i, 4i+3]) = in[i], \\ \forall_{i=0}^{3} hw(w[4i, 4i+3]) = out[i], \\ \texttt{sub_matrix_invertable}(M, u, v). \end{cases}$$

We now consider vectors corresponding to a column with three cells of weight 4 and one cell of weight 1. We denote by $\boldsymbol{s}_i \in \mathbb{F}_2^{16}$, $1 \leq i \leq 16$, a vector where the $i$-th bit is 1, the other 3 bits corresponding to the same cell are 0, and all other 12 bits are 1. As pointed out in Section 3, in order to be able to construct a full-rank integral-resistance matrix with the maximize-key method, we add the constraints to obtain the MILP model in Algorithm 4:

$$\begin{aligned} \texttt{enforce_out_WP}(M, out) \coloneqq & \forall u \in \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{16}\}, \\ & \exists w \in \mathbb{F}_2^{16} \begin{cases} \forall_{i=0}^{3} hw(w[4i, 4i+3]) = out[i], \\ \texttt{sub_matrix_invertable}(M, u, v). \end{cases} \end{aligned}$$

Our implementation uses `Minizinc` with the MILP solver `or-tools` to find intermediate solutions within a few hours. While we did not find the optimal solution in reasonable time, the best solution found has a Hamming weight of 25. Since the actual monomial propagation depends heavily on the interplay between the S-Box and MixColumns monomial transitions, the MixColumns matrix $M$ found by this MILP model does not necessarily lead to a cipher having the

**Algorithm 4** Minimize Hamming weight of MixColumns matrix

**Require:** WP ▷ *input/output WP from Figure 6*
`enforce_out_WP`(M, {4,3,3,3})
**for all** $(in, out) \in WP$ **do**
  `enforce_WP`(M, int, out)
$\forall_{i=0}^{n-1}$ `sum`$(M[i, 0..15]) < 2$
$\forall_{i=0}^{n-1}$ `sum`$(M[0..15, i]) < 2$
`minimize` sum(M[0..15, 0..15])

integral-resistance property after 7 rounds. When using `intres` with this Hamming weight of 25, we did not find a full-rank integral-resistance matrix. However, after testing a few other optimized matrices with our integral-resistance framework, we find the following MixColumns matrix for which the corresponding integral-resistance matrix is full rank after 7 rounds:

$$M_{\text{min}} = \begin{pmatrix}
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}.$$

$M_{\text{min}}$ has a Hamming weight of 28 and needs 12 parallel XORs to be realized in hardware. In comparison, the PRINCE MixColumns matrices require 32 XORs in hardware and for each output bit, 2 sequential XOR applications are needed.

*Validation.* Verifying the correctness of the `intres` tool via monomial trail counting is not feasible for the original PRINCE. However, due to the sparse $M_{\text{min}}$ matrix, there exist considerably fewer monomial trails. Indeed, checking the coefficients obtained from `intres` with a monomial trail counting model is possible. After verifying over 1000 coefficients with a value of 1 from the integral-resistance matrix, the maximum number of monomial trails is 1.8 million when using the $\mathcal{ZR}$ method to model the linear transitions. Most coefficients can be verified by far fewer trails, as indicated by the median of 1964. These verifications provide confidence in the correctness of our approach and implementation.

*Latency estimation.* In order to give an estimate about the reduction in latency when using our new MixColumns matrix compared to PRINCE MixColumns matrices, we use the NanGate 15nm latency numbers provided by Rasoolzadeh [30]. Although the PRINCE S-Box is not listed in this paper, it is reasonable to assume that the hardware implementation needs 4 sequential gates $\mathcal{G} = \{\text{INV}, \text{NAND}, \text{NOR}\}$. According to their latency measurements, at NanGate 15nm technology, the NOR gate has the highest latency of roughly 2.5 ps. Thus, we estimate the latency of

the S-Box to be $4 \cdot 2.5 = 10$ ps. While the ShiftRows operation can be realized with wires, the AddRoundKey operation is realized with an XOR, which takes about 5 ps. In the case of PRINCE, the total latency, including the 2 sequential XOR applications in the MixColumns step, is therefore about $10+5+2\cdot 5 = 25$ ps. When using $M_{\min}$ at the MixColumns operation, we reduce the latency by 20% to $10 + 5 + 5 = 20$ ps. Although this estimation can only be seen as a rough approximation, we can compare the latency numbers to a full PRINCE [2] implementation. For 12 rounds of PRINCE, the latency is 372 ps. Each round therefore takes $372/12 = 31$ ps. While this is more than our estimation, it also includes the reflector structure. Overall, it is hard to estimate the exact latency improvements, since we could, for example, fuse the XOR of the key addition with our MixColumns matrix. Still, using the MixColumns matrix $M_{\min}$ can significantly reduce the latency of PRF designs.

## 7 Conclusion

In this paper, we revisited the minimal-transitions method from Zeng and Tian [38]. We established further requirements on the S-Box and key masks needed for the method to work. Furthermore, by conceiving a degree abstraction model, we speed up the search for key masks. As a side effect, we used this model to search for upper bounds on ciphers. For Rijndael-256, this allows us to provide tighter upper bounds compared to previous works. We also highlight the potential to use such a degree propagation model to settle on a cipher structure when designing a new cipher. Next, we introduced the `intres` framework that implements the generalized minimal-transitions method. By adapting the $\mathcal{ZR}$ method, this tool has a drastically reduced runtime in the offline phase compared the program of Zeng and Tian [38]. Using `intres`, we showed the integral-resistance of 7-round PRINCE and 6-round Beanie. Furthermore, `intres` is designed in such a way that it is easy to change the S-Box and MixColumns operation for more applications. We used a heuristic approach to find new MixColumns matrices having minimal latency while still leading to integral-resistance when used in a PRINCE-like cipher. Due to the sparse nature of this new MixColumns matrix, it is possible to confirm the correctness of the minimal-transitions method via monomial-trail counting for the first time.

In future work it might be interesting to extend the degree abstraction model to the tweakey schedule. For ciphers with large states, this might enable us to find new integral distinguishers that incorporate the tweak. Moreover, it may be worth exploring whether the minimal-transitions method can be applied to ciphers without max-degree full-rank S-Boxes. However, in practice, most ciphers with such S-Boxes also have a lighter linear layer which enables us to find the integral-resistance via monomial-trail counting.

*Acknowledgments.* This research was funded in whole or in part by the Austrian Science Fund (FWF) SFB project SPyCoDe (10.55776/F85).

## References


1. Anand, R., Banik, S., Caforio, A., Ishikawa, T., Isobe, T., Liu, F., Minematsu, K., Rahman, M., Sakamoto, K.: Gleeok: A family of low-latency PRFs and its applications to authenticated encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 545–587 (2024). https://doi.org/10.46586/TCHES.V2024.I2.545-587
2. Avanzi, R., Banik, S., Dunkelman, O., Eichlseder, M., Ghosh, S., Nageler, M., Regazzoni, F.: The QARMAv2 family of tweakable block ciphers. IACR Transactions on Symmetric Cryptology **2023**(3), 25–73 (2023). https://doi.org/10.46586/TOSC.V2023.I3.25-73
3. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency PRF. IACR Transactions on Symmetric Cryptology **2021**(1), 37–77 (2021). https://doi.org/10.46586/TOSC.V2021.I1.37-77
4. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_16
5. Beierle, C., Hebborn, P., Leander, G., Perehuda, Y.: Integral resistance of block ciphers with key whitening by modular addition. In: CRYPTO 2025. LNCS, vol. 16004, pp. 497–529. Springer (2025). https://doi.org/10.1007/978-3-032-01901-1_16
6. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
7. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Transactions on Symmetric Cryptology **2019**(1), 5–45 (2019). https://doi.org/10.13154/tosc.v2019.i1.5-45
8. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer (1993). https://doi.org/10.1007/978-1-4613-9314-6
9. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer (2007). https://doi.org/10.1007/978-3-540-74735-2_31
10. Boura, C., Canteaut, A.: On the influence of the algebraic degree of $F^{-1}$ on the algebraic degree of $G \circ F$. IEEE Trans. Inf. Theory **59**(1), 691–702 (2013). https://doi.org/10.1109/TIT.2012.2214203
11. Bozilov, D., Eichlseder, M., Knezevic, M., Lambin, B., Leander, G., Moos, T., Nikov, V., Rasoolzadeh, S., Todo, Y., Wiemer, F.: PRINCEv2 – more security for (almost) no overhead. In: SAC 2020. LNCS, vol. 12804, pp. 483–511. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_19
12. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: FSE '97. LNCS, vol. 1267, pp. 149–165. Springer (1997). https://doi.org/10.1007/BFB0052343
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES – The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). https://doi.org/10.1007/978-3-662-04722-4
14. Derbez, P., Fouque, P.: Increasing precision of division property. IACR Transactions on Symmetric Cryptology **2020**(4), 173–194 (2020). https://doi.org/10.46586/TOSC.V2020.I4.173-194

15. Derbez, P., Lambin, B.: Fast MILP models for division property. IACR Transactions on Symmetric Cryptology **2022**(2), 289–321 (2022). https://doi.org/10.46586/TOSC.V2022.I2.289-321
16. ElSheikh, M., Youssef, A.M.: On MILP-based automatic search for bit-based division property for ciphers with (large) linear layers. In: ACISP 2021. LNCS, vol. 13083, pp. 111–131. Springer (2021). https://doi.org/10.1007/978-3-030-90567-5_6
17. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In: SAC 2018. LNCS, vol. 11349, pp. 115–138. Springer (2018). https://doi.org/10.1007/978-3-030-10970-7_6
18. Flórez-Gutiérrez, A., Grassi, L., Leander, G., Sibleyras, F., Todo, Y.: General practical cryptanalysis of the sum of round-reduced block ciphers and ZIP-AES. In: ASIACRYPT 2024. LNCS, vol. 15492, pp. 280–311. Springer (2024). https://doi.org/10.1007/978-981-96-0947-5_10
19. Gerhalter, S., Hodžić, S., Medwed, M., Nageler, M., Folwarczny, A., Nikov, V., Hoogerbrugge, J., Schneider, T., McConville, G., Eichlseder, M.: Beanie–a 32-bit cipher for cryptographic mitigations against software attacks. IACR Transactions on Symmetric Cryptology **2025**(4), 31–69 (2025). https://doi.org/10.46586/tosc.v2025.i4.31-69
20. Hadipour, H., Eichlseder, M.: Integral cryptanalysis of WARP based on monomial prediction. IACR Transactions on Symmetric Cryptology **2022**(2), 92–112 (2022). https://doi.org/10.46586/TOSC.V2022.I2.92-112
21. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: ASIACRYPT 2020. LNCS, vol. 12491, pp. 537–566. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_18
22. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Strong and tight security guarantees against integral distinguishers. In: ASIACRYPT 2021. LNCS, vol. 13090, pp. 362–391. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_13
23. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_15
24. Hu, K., Wang, Q., Wang, M.: Finding bit-based division property for ciphers with complex linear layers. IACR Transactions on Symmetric Cryptology **2020**(1), 396–424 (2020). https://doi.org/10.13154/TOSC.V2020.I1.396-424
25. Knudsen, L.R.: Truncated and higher order differentials. In: FSE 1994. pp. 196–211. LNCS, Springer (1994). https://doi.org/10.1007/3-540-60590-8_16
26. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: FSE 2002. pp. 112–127. LNCS, Springer (2002). https://doi.org/10.1007/3-540-45661-9_9
27. Lai, X.: Higher order derivatives and differential cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry pp. 227–233 (1994)
28. Lambin, B., Derbez, P., Fouque, P.: Linearly equivalent s-boxes and the division property. Des. Codes Cryptogr. **88**(10), 2207–2231 (2020). https://doi.org/10.1007/S10623-020-00773-4
29. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT '93. LNCS, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33
30. Rasoolzadeh, S.: Low-latency boolean functions and bijective S-boxes. IACR Transactions on Symmetric Cryptology **2022**(3), 403–447 (2022). https://doi.org/10.46586/TOSC.V2022.I3.403-447

31. Sasaki, Y., Todo, Y.: New algorithm for modeling S-box in MILP based differential and division trail search. In: SecITC 2017. LNCS, vol. 10543, pp. 150–165. Springer (2017). https://doi.org/10.1007/978-3-319-69284-5_11
32. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_5
33. Sun, L., Wang, W., Wang, M.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**(1), 12–20 (2020). https://doi.org/10.1049/IET-IFS.2018.5283
34. Todo, Y.: Structural evaluation by generalized integral property. In: EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_12
35. Udovenko, A.: Convexity of division property transitions: Theory, algorithms and compact models. In: ASIACRYPT 2021. LNCS, vol. 13090, pp. 332–361. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_12
36. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24
37. Zeng, F., Tian, T.: On the security bounds for block ciphers without whitening key addition against integral distinguishers. In: ACISP 2024. LNCS, vol. 14895, pp. 41–56. Springer (2024). https://doi.org/10.1007/978-981-97-5025-2_3
38. Zeng, F., Tian, T.: A new method for constructing integral-resistance matrix for 5-round AES. IET Inf. Secur. **2025**(1) (2025). https://doi.org/10.1049/ise2/3447652
39. Zhang, W., Rijmen, V.: Division cryptanalysis of block ciphers with a binary diffusion layer. IET Inf. Secur. **13**(2), 87–95 (2019). https://doi.org/10.1049/IET-IFS.2018.5151