IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2024, No. 3, pp. 177–199. DOI:10.46586/tosc.v2024.i3.177-199

Perfect Monomial Prediction for Modular Addition

Kai Hu[1,3(✉)] and Trevor Yap[2,4]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
kai.hu@sdu.edu.cn

[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
trevor.yap@ntu.edu.sg

[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China

[4] Temasek Laboratories, Nanyang Technological University, Singapore, Singapore

Abstract. Modular addition is often the most complex component of typical Addition-Rotation-XOR (ARX) ciphers, and the division property is the most effective tool for detecting integral distinguishers. Thus, having a precise division property model for modular addition is crucial in the search for integral distinguishers in ARX ciphers. Current division property models for modular addition either (a) express the operation as a Boolean circuit and apply standard propagation rules for basic operations (COPY, XOR, AND), or (b) treat it as a sequence of smaller functions with carry bits, modeling each function individually. Both approaches were originally proposed for the two-subset bit-based division property (2BDP), which is theoretically imprecise and may overlook some balanced bits.

Recently, more precise versions of the division property, such as parity sets, three-subset bit-based division property without unknown subsets (3BDPwoU) or monomial prediction (MP), and algebraic transition matrices have been proposed. However, little attention has been given to modular addition within these precise models.

The propagation rule for the precise division property of a vectorial Boolean function $\boldsymbol{f}$ requires that $\boldsymbol{u}$ can propagate to $\boldsymbol{v}$ if and only if the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ appears in $\pi_{\boldsymbol{v}}(\boldsymbol{f})$. Braeken and Semaev (FSE 2005) studied the algebraic structure of modular addition and showed that for $\boldsymbol{x} \boxplus \boldsymbol{y} = \boldsymbol{z}$, the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{y})$ appears in $\pi_{\boldsymbol{w}}(\boldsymbol{z})$ if and only if $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$. Their theorem directly leads to a precise division property model for modular addition. Surprisingly, this model has not been applied in division property searches, to the best of our knowledge.

In this paper, we apply Braeken and Semaev's theorem to search for integral distinguishers in ARX ciphers, leading to several new results. First, we improve the state-of-the-art integral distinguishers for all variants of the Speck family, significantly enhancing search efficiency for Speck-32/48/64/96 and detecting new integral distinguishers for Speck-48/64/96/128. Second, we determine the exact degrees of output bits for 7-round Speck-32 and all/16/2 output bits for 2/3/4-round Alzette for the first time. Third, we revisit the choice of rotation parameters in Speck instances, providing a criterion that enhances resistance against integral distinguishers. Additionally, we offer a simpler proof for Braeken and Semaev's theorem using monomial prediction, demonstrating the potential of division property methods in the study of Boolean functions.

We hope that the proposed methods will be valuable in the future design of ARX ciphers.

Keywords: Modular addition · Division property · Monomial prediction · Speck · Alzette

Received: 2024-06-01 Accepted: 2024-08-01 Published: 2024-09-06

1 Introduction

Automatic search tools play a crucial role in the cryptanalysis of many symmetric-key ciphers in this day and age. Developing precise and compact models is a fundamental task in this area. The division property [Tod15, TM16] and its automatic search method [XZBL16, HWW20, Udo21] have been the dominant technique in the search for integral distinguishers [DKR97, KW02] for block ciphers, including the Addition-Rotation-XOR (ARX) ciphers. Although dozens of papers have studied the security of ARX ciphers against the division properties [SWW17, HW19, WHG+19, DL22], most of them only focused on the two-subset bit-based division property (2BDP)[TM16], an efficient but lossy variant of division properties. In this context, a lossy model is imprecise and potentially missing some integral distinguishers.

Several papers have studied the precise version of the division property. At CRYPTO 2016, Boura and Canteaut proposed the notion of *parity sets* as another view of the division property [BC16], where precise propagation rules are studied (the precise propagation rules were reflected by [BC16, Table 2], but they did not count the number of trails). Similar results, called *the three-subset bit-based division property without unknown subsets* (3BDPwoU), were implied by Wang et al. in [WHG+19] and Hao et al. in [HLM+20]. However, they focused more on the automatic search aspect and did not provide rigorous proof. Later, at ASIACRYPT 2020, Hebborn used the parity sets to prove that 3BDPwoU can precisely indicate the existence of a specific plaintext monomial in the output polynomial [HLLT20]. At ASIACRYPT 2020, Hu et al. proposed the monomial prediction (MP) and developed their propagation rules [HSWW20]. They also proved that the monomial prediction is the same as the 3BDPwoU in the search for integral distinguishers. Recently, at FSE 2024, Beyne and Verbauwhede introduced the algebraic transition matrices and unified the monomial trails and parity sets by formalizing the exact nature of their duality. [BV23]. In the following, we mainly use the language of the monomial prediction to describe the propagation rules as it is easier to integrate with Braeken and Semaev's theorem [BS05] for the modular addition operation, as discussed in detail later.

It has been shown that the 2BDP is a *no-false-alarm* approximation of the MP; that is, if an integral property is detectable by the 2BDP, it must also be detectable by the MP. However, the converse is not necessarily true [HSWW20]. Therefore, whenever possible, an MP model should be applied to a block cipher first. The difficulty in applying an MP model lies in the need to count the number of allowed MP trails. However, the modular addition operation found in ARX ciphers is inherently complex, making it believed to be impossible (or at least very difficult) to count the exact number of MP trails for ARX ciphers. As a result, most papers on the division properties of ARX ciphers, such as [SWW17, BBdS+20, DL22], focus primarily on the 2BDP, where the most technically challenging aspect is modeling the 2BDP propagation for modular addition.

There are two major 2BDP models for the modular addition operation. The first model, proposed by Sun, Wang, and Wang [SWW17, SWLW18], decomposes the modular addition into three kinds of basic operations, i.e., COPY, AND and XOR, and model these basic operations by their standard 2BDP propagation rules. The model is heavy and complex since they decompose the addition modulo 2^n into $2n-1$ XORs, $3n-1$ COPYs, and $2n-1$ ANDs, leading to a search model containing $10n-4$ linear constraints and $12n-19$ intermediate variables. The second method was proposed by Beierle et al. in [BBdS+20]. This is a much simpler model of the modular addition requiring only 2 inequalities per bit. Consider an addition of two n-bit words $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_2^n$ and let $\boldsymbol{y} = \boldsymbol{a} \boxplus \boldsymbol{b} \bmod 2^n$. It can be computed from recursively apply the function $f(a_i, b_i, c_i) = (a_i \oplus b_i \oplus c_i, \mathrm{Maj}(a_i, b_i, c_i))$ where Maj is the majority function and c_i the i-th carry. Specifically, each iteration yields $(y_i, c_{i-1}) = f(a_i, b_i, c_i)$. The 2BDP table of the function f can be modeled with only two inequalities. However, because the 2BDP is a lossy model, this method still has the risk of overlooking some balanced bits although it is a lightweight model.

To describe the MP propagation for modular addition, a trivial transformation can be performed by replacing the 2BDP propagations for COPY, AND, XOR with their counterparts of MP for the first method and by replacing the 2BDP table with the MP table for the second method. The trivial substitution of the first method was performed in [HW19]. The "propagation of $\mathbb{L}$" in that paper is the MP version of the first model. However, this trivial transformation leads to an MP model for modular addition with redundant trails caused by the COPY operations. More importantly, although the propagation of MP trails was modeled, the exact number of MP trails was not computed due to the large number of redundant trails. Consequently, the model in [HW19] is still a lossy one as a no-false-alarm approximation of the MP. Hence, there is still a need for a perfect MP model for the modular addition.

According to the propagation rule of the MP [HSWW20], for a vectorial Boolean function $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the propagation $(\boldsymbol{u}) \to (\boldsymbol{v})$ is valid if and only if $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ appears in the polynomial of $\pi_{\boldsymbol{v}}(\boldsymbol{f}(\boldsymbol{x}))$. At FSE 2005, Braeken and Semaev studied the algebraic structure of the modular addition [BS05]. Their work proved that for $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$, an input monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{y})$ appears in the output monomial $\pi_{\boldsymbol{w}}(\boldsymbol{z})$ if and only if $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$, note that here "+" is the addition in the integer ring. Such a theorem directly leads to a perfect MP model for the modular addition. Surprisingly, this fact has gone unnoticed by researchers in the division property community, despite the division property being a widely used tool for nearly a decade.

Our contributions. In this paper, we apply Braeken and Semaev's theorem to the search for integral distinguishers for ARX ciphers, which is the main contribution of this paper. To make it clearer, for $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$, the propagation of their exponents is described as $(\boldsymbol{u}, \boldsymbol{v}) \to (\boldsymbol{v})$ if and only if $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$. Additionally, we find their proof [BS05] is somewhat involved, so we offer a new and simpler proof in this paper according to the monomial trails.

Braeken and Semaev's theorem can be implemented directly with an SMT model where the "+" operation is built-in. In this model, no auxiliary variables are required at all. Considering that in many cases where a MILP model is desirable, we as well introduce a MILP model with auxiliary variables but prove that these auxiliary variables do not cause any redundant MP trails.

With the precise propagation models, new results are obtained for some ARX ciphers. In [WHG+20], Wang et al. used a variant of the 3BDP where a new propagation rule for "XOR with secret round key" was introduced to bypass the effect of the round key and found new integral distinguishers that are not detectable by previous tools [SWW17, WHG+19, HW19] for Speck-32, -48, -64, and -96. By counting the number of monomial trails, our proposed method can easily re-identify these previously found integral distinguishers. Furthermore, the time for detecting these integral distinguishers is reduced from hours or days to seconds or minutes. Moreover, our technique also finds new integral distinguishers for Speck-48, -64, -96 and -128 that are not feasible with Wang et al.'s method. New integral distinguishers for all members of the ARX permutation Sparkle (the underlying permutation of the NIST LWC competition finalist Schwaemm and Esch [BBdS+19]) are also found.

The precise version of division property is not only a tool for detecting the integral properties accurately, but also useful to calculate the exact algebraic degree[1] or its *lower* bound of a symmetric-key cipher. In [HSWW20], the MP was used to compute the exact algebraic degrees of Trivium up to 834 rounds. In [HLLT20], the 3BDPwoU was used to give lower bounds for block ciphers such as Skinny-64 [BJK+16], Present [BKL+07],

[1]In this paper, the *algebraic degree* means the greatest algebraic degree over all possible key parameters. See Section 2.2 for a detailed explanation.

Simon [BSS+15] and so forth. However, it is still difficult to calculate the exact algebraic degrees for ARX ciphers so far. For example, in [BBdS+20], the authors had to do experiments to show that there are some 32-degree monomials in the output bits of Alzette.

The application of Braeken and Semaev's theorem is also helpful in terms of the computation of the algebraic degrees. We show that the smallest exact degree of the 3-round output bits of Alzette that our method can find is 42 while the smallest exact degree for the 4-round Alzette output that we can identify is 63. In [BBdS+20], only experiments were done to show the lower bounds are 32. For 7-round Speck-32, we prove that the exact algebraic degrees of all output bits are 31. As far as we know, this is the first time that we can give a theoretical calculation for ARX ciphers' algebraic degrees.

Finally, we revisit the choices of rotation constants used in the Speck family. We develop a criterion that indicates the complexity of the corresponding superpoly and use it to search for better rotation constants that make Speck stronger against integral distinguishers. The variants of Speck-32, -48, and -64 equipped with our new rotation constants can resist the integral attacks with 6 rounds, while the original ones need to be 7 rounds, up to checking all $2n-1$ active patterns of plaintexts. Our proposed methods can be useful in the future design of ARX ciphers.

Outline of the remaining paper. The following paper is organized as follows. Section 2 introduces the notations, definitions, and background knowledge used in this paper. In Section 3, we use a result published almost 20 years ago to develop the perfect MP propagation rule for the modular addition and build the automatic search model. We apply our perfect model to Speck in Section 4 and to Alzette and Sparkle in Section5, respectively. In Section 6, we revisit the choice of rotation constants of Speck family. Section 7 concludes the paper and discusses possible future works.

2 Preliminaries

2.1 Notations and Definitions.

In this paper, $\mathbb{F}_2$ is the binary field, $\mathbb{F}_2^n$ is the space for bit vectors with the length of n, and $\mathbb{Z}$ is the integer ring. The XOR, modular addition, and addition in $\mathbb{Z}$ are respectively denoted by $\oplus$, $\boxplus$ and $+$, where the length of the operands of these operations should be clear from the context. We use bold italic lowercase letters to represent bit vectors, for example, $\boldsymbol{x} \in \mathbb{F}_2^n$. $\boldsymbol{x}$ is also written as $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ with x_i being its i-th coordinate and x_{n-1} is the least significant bit (LSB). The Hamming weight of $\boldsymbol{x}$ is defined as $\mathrm{wt}(\boldsymbol{x}) = \sum_i x_i$. A partial order is defined for two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ where $\boldsymbol{x} \preceq \boldsymbol{y}$ if and only if $x_i \leq y_i$ for $0 \leq i < n$.

Given $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{F}_2^n$, $\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{u_i=1} x_i$ is called a monomial of $\boldsymbol{x}$ with respect to $\boldsymbol{u}$. Suppose $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function, the algebraic normal form (ANF) of a Boolean function is defined as

$$f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \prod_{i=0}^{n-1} x_i^{u_i}$$

where $a_{\boldsymbol{u}} \in \mathbb{F}_2$. If $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ appears in the ANF of a Boolean function f, we denote it by $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to f$, otherwise, $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \nrightarrow f$.

We define $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ to be the vectorial Boolean function as $\boldsymbol{f}(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$. For $\boldsymbol{v} \in \mathbb{F}_2^m$, the products of some coordinates in $\boldsymbol{f}(\boldsymbol{x})$ be

$$\pi_{\boldsymbol{v}}(\boldsymbol{f}(\boldsymbol{x})) = \prod_{i=0}^{m-1} f_i^{v_i}(\boldsymbol{x}).$$

2.2 Integral Attack, Division Property and Algebraic Degree

The integral attack was first introduced in cryptanalysis of Square [DKR97], which was called SQUARE attack. Later, it was formalized as an integral attack by Knudsen and Wagner in [KW02]. The possible status of a word or bit of the intermediate states can be either saturation ($\mathcal{A}$), zero-sum ($\mathcal{S}$), constant ($\mathcal{C}$) or unknown ($\mathcal{U}$), and their changes can be traced. At EUROCRYPT 2015, Todo introduced the division properties into integral properties by studying the intermediate status between the saturation status and zero-sum status [Tod15]. For a multiset $\mathbb{X} \subset \mathbb{F}_2^n$, if we say its (word-based) division property is $\mathcal{D}_k^n$, then $\sum_{\boldsymbol{x} \in \mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = 0$ for $\boldsymbol{u}$ with $\mathrm{wt}(\boldsymbol{u}) < k$ while it cannot be determined for $\boldsymbol{u}$ with $\mathrm{wt}(\boldsymbol{u}) \geq k$ (note that $\mathbb{X}$ is a multiset produced by applying a *keyed* cipher to a plaintext multiset). In this generalization, the zero-sum and saturation properties are represented by $\mathcal{D}_2^n$ and $\mathcal{D}_n^n$ (for a *set*) respectively. Nowadays, Todo's division property is regarded as an effective method to detect the zero-sum properties (some versions of division properties also applicable to detect the one-sum case). It is worth mentioning that very recently, another generalization of integral attacks, called the *ultrametric integral attacks*, were presented in [BV24], where the zero-sum property is interpreted as the case that the frequency of the appearing of "1" is a multiple of 2, while the saturation means the frequency is a multiple of 2^{n-1} (suppose the input is of length n). The ultrametric integral attack studies all cases where the frequency is a multiple of $2^t, 1 \leq t \leq n-1$.

As we mentioned in the introduction, the precise versions of the division properties have been extensively studied, such as parity sets [BC16], 3BDPwoU [HLM+20] and most recently algebraic transition matrices [BV23]. One purely algebraic interpretation of the division property is offered by the monomial prediction proposed in [HSWW20]. In this paper, we mainly use the language of the monomial prediction to describe our work, for it is easier to integrate with Braeken and Semaev's theorem [BS05].

MP trails and propagation rules. Suppose $\boldsymbol{f}$ is a composition of r vectorial Boolean functions $\boldsymbol{f}^{(i)} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ for $0 \leq i \leq r-1$ (we assume each round function has the same length) defined as

$$\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \dots \boldsymbol{f}^{(0)}.$$

We denote $\boldsymbol{x}^{(i)} \in \mathbb{F}_2^n$ and $\boldsymbol{x}^{(i+1)} \in \mathbb{F}_2^n$ to be the input and output of the vectorial Boolean function $\boldsymbol{f}^{(i)}$. We also define $\boldsymbol{u}^{(i)} \in \mathbb{F}_2^n$ to be the exponent of $\boldsymbol{x}^{(i)}$. We say that the sequence of the exponents $(\boldsymbol{u}^{(0)}, \boldsymbol{u}^{(1)}, \dots, \boldsymbol{u}^{(r)})$ is an r-round monomial trail connecting $\boldsymbol{u}^{(0)}$ and $\boldsymbol{u}^{(r)}$ with respect to the composite function $\boldsymbol{f}$ if

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

If there exists at least one such monomial trail from $\boldsymbol{u}^{(0)}$ to $\boldsymbol{u}^{(r)}$, we denote this as $\boldsymbol{u}^{(0)} \rightsquigarrow \boldsymbol{u}^{(r)}$. Otherwise, we have $\boldsymbol{u}^{(0)} \not\rightsquigarrow \boldsymbol{u}^{(r)}$.

We let the set of monomial trails from $\boldsymbol{u}^{(0)}$ to $\boldsymbol{u}^{(r)}$ be called the monomial hull which is denoted as $\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(r)}$. The goal of the monomial prediction is to determine whether $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. The following theorem presents the relationship between the number of monomial trails within the monomial hull and the existence of a monomial.

Theorem 1 ([HSWW20])**.** *Let the notation be defined as above.* $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ *if and only if*

$$|\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(r)}| \equiv 1 \bmod 2.$$

Let $\boldsymbol{x}^{(0)} = (\boldsymbol{k}, \boldsymbol{p})$ and $\boldsymbol{x}^{(r)} = \boldsymbol{c}$. Obviously, if for any $\boldsymbol{w} \in \mathbb{F}_2^m$, $|(\boldsymbol{w}, \boldsymbol{u}) \bowtie \boldsymbol{v}| \equiv 0 \bmod 2$, $\pi_{\boldsymbol{v}}(\boldsymbol{c})$ will have an integral property as $\bigoplus_{\boldsymbol{x} \preceq \boldsymbol{u}} \boldsymbol{f}(\boldsymbol{k}, \boldsymbol{p}) = 0$ for any $\boldsymbol{k}$. Thus, by calculating the size of the monomial hull, the integral properties can be detected exactly.

(Greatest) Algebraic degree. The algebraic degree of a keyed Boolean function $f_{\boldsymbol{k}}(\boldsymbol{x})$ where $\boldsymbol{k}$ is the fixed parameter is defined as

$$\deg(f_{\boldsymbol{k}}) = \max_{\boldsymbol{u}:\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to f_{\boldsymbol{k}}} \mathrm{wt}(\boldsymbol{u}) = \max_{\boldsymbol{u}:\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to f_{\boldsymbol{k}}} \sum_i u_i$$

If $\boldsymbol{k}$ is not assumed fixed, $f_{\boldsymbol{k}}(\boldsymbol{x})$ becomes a Boolean function with both $\boldsymbol{k}$ and $\boldsymbol{x}$ as input, i.e., $f(\boldsymbol{k}, \boldsymbol{x})$. The *greatest algebraic degree of* f over all possible keys is defined as

$$\overline{\deg}(f) = \max_{\boldsymbol{k}} \deg(f_{\boldsymbol{k}}) = \max_{\boldsymbol{u}:\pi_{\boldsymbol{w}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to f} \mathrm{wt}(\boldsymbol{u}) = \max_{\boldsymbol{u}:\pi_{\boldsymbol{w}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to f} \sum_i u_i$$

Thus, if we prove that there is a monomial $\pi_{\boldsymbol{w}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x})$ with $\mathrm{wt}(\boldsymbol{u}) = d$ existing in f, the lower bound of $\overline{\deg}(f)$ is d. If we further prove there is no $\pi_{\boldsymbol{w}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x})$ with $\mathrm{wt}(u) > d$ contained by f, $\overline{\deg}(f) = d$.

Throughout this paper, only the greatest algebraic degree is considered. For the sake of convenience, we will use algebraic degree in the meaning of the greatest algebraic degree. Also, when we say $\deg(f)$, it is actually $\overline{\deg}(f)$.

2.3 Automatic Search Model for Monomial Prediction

According to the definition of monomial trails for $\boldsymbol{f}(\boldsymbol{x})$, $(\boldsymbol{u}) \to (\boldsymbol{v})$ is a valid propagation if and only if $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to \pi_{\boldsymbol{v}}(\boldsymbol{f}(\boldsymbol{x}))$. Thus, when the size of $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is not too large, we can construct a 2-dimensional MP table defined as

$$\mathrm{MPT}[\boldsymbol{u}][\boldsymbol{v}] = \begin{cases} 1, & \pi_{\boldsymbol{u}}(\boldsymbol{x}) \to \pi_{\boldsymbol{v}}(\boldsymbol{f}(\boldsymbol{x})) \\ 0, & \pi_{\boldsymbol{u}}(\boldsymbol{x}) \nrightarrow \pi_{\boldsymbol{v}}(\boldsymbol{f}(\boldsymbol{x})) \end{cases}$$

for all $\boldsymbol{u} \in \mathbb{F}_2^n$ and $\boldsymbol{v} \in \mathbb{F}_2^m$. Since COPY, AND, and XOR are all vectorial Boolean functions with small sizes, their MP table can be constructed easily.

COPY. Let the input and output MP variable, i.e., the exponents of monomials, are u and (v_0, v_1) for the COPY operation. The MPT of the COPY operation is then

$(u)\backslash(v_0, v_1)$	(0, 0)	(0, 1)	(1, 0)	(1, 1)
(0)	1	0	0	0
(1)	0	1	1	1

AND. Let the input and output MP variable, i.e., the exponents of monomials, are (u_0, u_1) and (v) for the AND operation. The MPT of the AND operation is then

$(u_0, u_1)\backslash(v)$	(0)	(1)
(0, 0)	1	0
(0, 1)	0	0
(1, 0)	0	0
(1, 1)	0	1

XOR. Let the input and output MP variable, i.e., the exponents of monomials, are (u_0, u_1) and (v) for the XOR operation. The MPT of the XOR operation is then

$(u_0, u_1)\backslash(v)$	(0)	(1)
(0, 0)	1	0
(0, 1)	0	1
(1, 0)	0	1
(1, 1)	0	0

A description of the above properties using inequalities or CNFs can be found in [SHW+14, SWW17, HW19]. In Section 3, a perfect search model for the modular addition will be introduced by a direct application of [BS05]. With the perfect search model, the monomial prediction is applied to Speck, Alzette and Sparkle, to search for integral distinguishers or calculating the algebraic degrees.

3 Monomial Prediction for Modular Addition

The modular addition is the core operation of ARX ciphers. In Section 3.1, we introduce a perfect model for the MP propagation of the modular addition, which is a direct application of the theorem proposed almost 20 years ago in [BS05] but has never been used in the MP/division property cryptanalysis of ARX ciphers. Considering that the original proof is quite involved, we provide a new and simpler proof based on monomial trails. This shows the potential of MP (as well as division properties, parity sets and algebraic transition matrices) in proving complex mathematical theorems. We construct its SMT model which does not require any auxiliary variables. We also propose a bit-based MILP model with auxiliary variables but we have proven that these auxiliary variables do not introduce any redundant MP trails.

3.1 MP for Modular Addition

To develop the MP model of the modular addition, we need to know its algebraic structure. As early as 2005, Braeken and Semaev proved the following theorem [BS05, Theorem 1],

Theorem 2 (adapted from [BS05])**.** *For $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$ where $\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_2^n$,*

$$\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{y}) \rightarrow \pi_{\boldsymbol{w}}(\boldsymbol{z}) \quad \text{if and only if} \quad \boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}.$$

According to Theorem 2, a monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{y})$ appears in $\pi_{\boldsymbol{w}}(\boldsymbol{z})$ if and only if $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$ where "+" lies in the integer. This can be directly used in the MP of the modular addition. However, we find that the original proof in [BS05] is somehow involved, as they first proved two recursive complicated formulas through induction (see Lemmas 1 & 2 of [BS05]). Theorem 2 is then proven through these two formulas. Hence, we give a new and simpler proof for Theorem 2, which is based on the MP trails directly. Our proof also gives a compact bit-based MP model for the modular addition.

Proof. $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$ can be written as iteratively bitwise operations:

$$\begin{cases} z_i = x_i \oplus y_i \oplus c_{i+1} \\ c_i = \mathrm{Maj}(x_i, y_i, c_{i+1}) = x_i y_i \oplus x_i c_{i+1} \oplus y_i c_{i+1} \end{cases} \tag{1}$$

where $0 \le i < n$, $\boldsymbol{c} = (c_0, c_1, \ldots, c_n)$ are the carry variables with $c_n = 0$. It can be represented by successive applications of the function S

$$(z_i, c_i) = S(x_i, y_i, c_{i+1}) = (x_i \oplus y_i \oplus c_{i+1}, \mathrm{Maj}(x_i, y_i, c_{i+1}))$$

as illustrated in Figure 1.

Denote the exponent for $\boldsymbol{c}$ by $\boldsymbol{\ell} = (\ell_0, \ell_1, \ldots, \ell_n)$. According to Theorem 1, the fact that $\pi_{\boldsymbol{u}}(\boldsymbol{x})\pi_{\boldsymbol{v}}(\boldsymbol{y}) \rightarrow \pi_{\boldsymbol{w}}(\boldsymbol{z})$ is equivalent to that there are odd-number monomial trails connecting $(\boldsymbol{u}, \boldsymbol{v})$ and $(\boldsymbol{w})$. In other words, the odd number of possible $\boldsymbol{\ell}$ that satisfy $(u_i, v_i, \ell_{i+1}) \rightarrow (w_i, \ell_i), 0 \le i \le n$. Table 1 is the MPT of S, we can find that when (u_i, v_i, ℓ_{i+1}) is determined, (w_i, ℓ_i) will be uniquely determined accordingly. Since ℓ_0 and ℓ_n are not the real input and output of a modular addition, they are set as 0. Thus, as

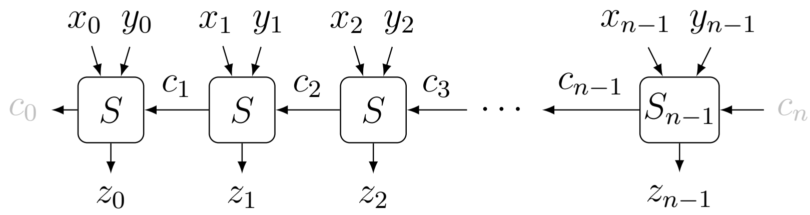


Figure 1: Iteratively bitwise representation of the modular addition operation.

Table 1: Monomial prediction table of $S : (x_i, y_i, c_{i+1}) \to (z_i, c_i)$.

$(u_i, v_i, \ell_{i+1}) \backslash (w_i, \ell_i)$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(0,0,0)$	1	0	0	0
$(0,0,1)$	0	0	1	0
$(0,1,0)$	0	0	1	0
$(0,1,1)$	0	1	0	0
$(1,0,0)$	0	0	1	0
$(1,0,1)$	0	1	0	0
$(1,1,0)$	0	1	0	0
$(1,1,1)$	0	0	0	1

long as $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ are given, all $\ell_i, 1 \le i < n$ will be uniquely determined, i.e., there is only one possible choice of $\boldsymbol{\ell}$.

Furthermore, since w_i and ℓ_i are uniquely determined by (u_i, v_i, ℓ_{i+1}), we can write their ANFs according to Table 1 as follows:

$$\begin{cases} w_i = u_i \oplus v_i \oplus \ell_{i+1} \\ \ell_i = \mathrm{Maj}(u_i, v_i, \ell_{i+1}) = u_i v_i \oplus u_i \ell_{i+1} \oplus v_i \ell_{i+1}. \end{cases} \tag{2}$$

Consider $\ell_0 = 0$ and $\ell_n = 0$, Equation 2 is just the iteratively bitwise representation of $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$. □

SMT model for the modular addition. The SMT tool supports the built-in "+" operation, thus the SMT model for MP of the modular addition is straightforward. Given the input and output exponents $\boldsymbol{u}, \boldsymbol{v}$ and $\boldsymbol{w}$, we can directly develop the SMT model by modeling $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w}$. This paper uses STP as the SMT solver whose input language can be the CVC language, so the constraint for MP of the addition modulo 2^n is

$$\mathrm{ASSERT}(\boldsymbol{w} = \mathrm{BVPLUS}(n, \boldsymbol{u}, \boldsymbol{v}));\ \mathrm{ASSERT}(\mathrm{BVLE}(\boldsymbol{u}, \boldsymbol{w}));\ \mathrm{ASSERT}(\mathrm{BVLE}(\boldsymbol{v}, \boldsymbol{w})).$$

In this constraint, $\boldsymbol{u}, \boldsymbol{v}$ and $\boldsymbol{w}$ are n-bit vectors, and the last two statements are used to protect against the possible overflow. For the remaining operations in an ARX cipher such as the XOR and COPY, the propagation rules are usually bitwise. Since CVC language allows access to the i-th bit of a vector by the index, such as $u[i]$, we do not need to transform $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ to bit variables.

MILP model for the modular addition. MILP is another popular tool for searching MP. The well-known MILP solver, Gurobi, supports the integer-type variables. However, as discussed above, propagation rules for the remaining operations of an ARX cipher are still bitwise, and Gurobi does not allow flexible access to the i-th bit of an integer. Thus, the integer-type variables are not convenient.

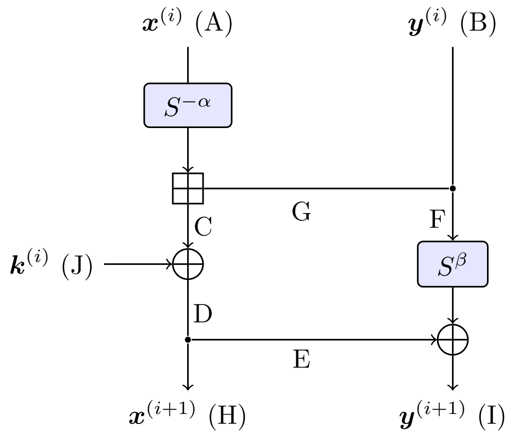


Figure 2: Structure of Speck family of ciphers.

Therefore, we provide a bit-based model for MP of the modular addition based on auxiliary variables. The auxiliary variables are just $\boldsymbol{\ell}$ in the above proof which serves as the exponents of the carry bits. Note that $\boldsymbol{\ell}$ *is unique according to our proof, so it will not bring redundant MP trails and it is still perfect.*

By transforming the "+" into bit operations, we obtain a set of constraints that represents a perfect MP model of modular addition. The constraints are as follows:

$$u_i + v_i + \ell_{i+1} - w_i - 2\ell_i = 0, \quad \ell_0 = \ell_n = 0$$

4 Application to Speck Families

In this section, we apply the perfect MP model to Speck family, speeding up the search and finding new integral distinguishers that cannot be found by any previous tools. For Speck-32, we also prove that all possible 31-degree terms appear in every 7-round output bit. The experiments were conducted on a Linux platform with `Intel(R) Xeon(R) Gold 6338N CPU @ 2.20GHz` with 128 processors and 2 TB RAM.

4.1 Specification of Speck and Previous Results

Speck [BSS+15] is a family of lightweight block ciphers published by the National Security Agency (NSA). It adopts the ARX structure which uses modular addition as its nonlinear operation. According to the block size, Speck family of ciphers can be represented as Speck-$2n$, where $n \in \{16, 24, 32, 48, 64\}$. The round structure of Speck is shown in Figure 2, where $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \in \mathbb{F}_2^{n\times n}$ and $(\boldsymbol{x}^{(i+1)}, \boldsymbol{y}^{(i+1)})$ are the input and output of the i-th round function, $\boldsymbol{k}^{(i)} \in \mathbb{F}_2^n$ is the i-th round key, $S^{-\alpha}$ denotes right circular shift by α bits and S^{β} denotes left circular shift by β bits. The parameters α and β are 8 and 3, respectively, except in the case of Speck-32, where they are 7 and 2 respectively.

In [SWLW18], Sun et al. applied their 2-subset division property model to all Speck family of block ciphers and obtained one 6-round integral distinguisher for each cipher (see lines labeled by ■ in Table 2). The data complexities are 2^{31} for Speck-32, 2^{45} for Speck-48, 2^{61} for Speck-64, 2^{93} for Speck-96 and 2^{125} for Speck-128. In [HW19], Hu and Wang used a variant of the three-subset division property to detect a new 6-round integral distinguisher for Speck-32 with 2^{31} data complexity (see the line labeled by ■ in Table 2). In [WHG+20], Wang et al. introduced a new model for division properties based on a special treatment of the key-XOR operation and identified additional integral distinguishers for Speck-32,

-48, -64, and -96 than [HW19, SWW17] (see lines labeled by ▲ in Table 2). In this work, we show that our model can re-identify these distinguishers in less time and detect more integral distinguishers than all the previous tools (see the lines labeled by ◆).

4.2 Search for New Integral Distinguishers for Speck Family

Since our model is a precise MP model for modular addition, we explicitly represent the whole MP model for Speck-$2n$ and try to count all the MP trails. At first glance, it seems rather difficult to enumerate all the MP trails for an ARX cipher due to the complicated modular addition operation. However, thanks to the compactness of our model, with some optimization search strategy, we succeed in enumerating all the trails under specific input and output conditions.

Modeling MP propagation. To model the propagation of the monomial trails, we introduce some n-bit auxiliary variables for the intermediate states of Speck-$2n$, as $A, B, C, D, E, F, G, H, I$ shown in Figure 2, and use the propagation rules introduced in Section 2.2 and Section 3 to connect all these variables in a standard way. This basic model for R-round Speck-$2n$, denoted by $\mathcal{M}_R$, is provided in Section A in the appendix.

For r-round Speck-$2n$, the r n-bit round keys are assumed as independent variables, i.e., we do not consider the influence of the key schedule. This is a popular simplification to make the search easy to mount, for example, in [WHG+20, HLLT20], the same assumptions were also used.

In [WHG+20], the authors applied their model to 6-round Speck-32, -48, -64 and -96 and found one more integral distinguisher than previous tools. The newly-found integral distinguisher for each Speck instance is related to one bit of key. The authors of [WHG+20] managed to find these distinguisher by introducing a new propagation rule for the key addition. From the perspective of MP, it is very likely that the superpoly related to this key bit is very simple. We first use the same input and output exponents as [WHG+20] and add an objective function for $\mathcal{M}_6$, where $\boldsymbol{u}^{(0)}, \boldsymbol{u}^{(6)}$ are the input and output exponents that lead to integral distinguishers detected by [WHG+20] (these $\boldsymbol{u}^{(0)}$ and $\boldsymbol{u}^{(6)}$ are listed in lines labeled by ▲ in Table 2). We set the objective function to maximize $\mathcal{K} = \sum_{r=0}^{R-1} \sum_{i=0}^{2n-1} w_i^{(r)}$, where $w_i^{(r)}$ denotes the exponent corresponding to the key bit $k_i^{(r)}$. The value of the max $\mathcal{K}$ can indicate how many key bits are involved into the superpoly. For the sake of completeness, we state the following optimization problem below.

Model 1: Together with the constraints of model $\mathcal{M}_R$, the optimization problem solves

$$\textbf{Max:} \quad \sum_{r=0}^{R-1} \sum_{i=0}^{2n-1} w_i^{(r)}$$

subjected to the fix values of

$$\boldsymbol{u}^{(0)} \text{ and } \boldsymbol{u}^{(R)}. \tag{3}$$

Optimization for Model* 1. The `PoolSearchMode` of Gurobi enables the MILP model to enumerate all possible solutions, thus, allowing us to count the number of monomial trials. Now, let **Model*** 1 be the model when enabling the `PoolSearchMode` of Gurobi option for **Model 1**. However, using **Model*** 1 could take a significant amount of time to count the number of monomial trails. To accelerate the search, we apply two optimization strategies based on the divide-and-conquer ideas to the search. The first strategy considers the MP

Table 2: Integral distinguishers for the families of Speck ciphers. The results labeled by 🟦 and 🟧 are detectable with [SWW17] and [HW19], respectively. Lines labeled by ▲ are found by [WHG+20] and lines labeled by ◆ are only detectable by our methods in this paper. **Time1** is the time reported in [WHG+20] while **Time2** is the time of our methods to find those distinguishers. We denote p to be the superpoly obtained.

Cipher	Integral Distinguishers	#Trails	$\mathcal{K}_{\max}$	p	Time1	Time2
🟦 Speck-32	$\overline{\{26\}} \xrightarrow{6R} \{15\}$	Infeasible	–	0	0 sec	0 sec
🟧 Speck-32	$\overline{\{25\}} \xrightarrow{6R} \{15\}$	Infeasible	–	0	0 sec	0 sec
▲ Speck-32	$\overline{\{25, 26\}} \xrightarrow{6R} \{15\}$	2570	1	0	1 hrs	0 sec
◆ Speck-32	$\overline{\{24, 26\}} \xrightarrow{6R} \{15\}$	237 660	3	p_0	–	13 sec
🟦 Speck-48	$\overline{\{40, 41, 42\}} \xrightarrow{6R} \{23\}$	Infeasible	–	0	0 sec	0 sec
▲ Speck-48	$\overline{\{39, 41, 42\}} \xrightarrow{6R} \{23\}$	11 464	1	0	13 hrs	0 sec
◆ Speck-48	$\overline{\{39, 40, 42\}} \xrightarrow{6R} \{23\}$	92 356	3	0	–	2 sec
◆ Speck-48	$\overline{\{39, 40, 41\}} \xrightarrow{6R} \{23\}$	288 742	3	0	–	5 sec
🟦 Speck-64	$\overline{\{56, 57, 58\}} \xrightarrow{6R} \{31\}$	Infeasible	–	0	0 sec	0 sec
▲ Speck-64	$\overline{\{55, 57, 58\}} \xrightarrow{6R} \{31\}$	2214	1	0	17.6 hrs	0 sec
◆ Speck-64	$\overline{\{55, 56, 58\}} \xrightarrow{6R} \{31\}$	3642	1	0	–	0 sec
◆ Speck-64	$\overline{\{55, 56, 57\}} \xrightarrow{6R} \{31\}$	27 590	1	0	–	1 sec
◆ Speck-64	$\overline{\{54, 55, 56\}} \xrightarrow{6R} \{31\}$	25 891 421	3	p_1	–	1.3 hrs
◆ Speck-64	$\overline{\{54, 55, 57\}} \xrightarrow{6R} \{31\}$	29 195 923	3	p_2	–	2 hrs
◆ Speck-64	$\overline{\{54, 55, 58\}} \xrightarrow{6R} \{31\}$	11 173 350	3	p_3	–	762 sec
◆ Speck-64	$\overline{\{54, 56, 57\}} \xrightarrow{6R} \{31\}$	11 002 854	3	0	–	207 sec
◆ Speck-64	$\overline{\{54, 56, 58\}} \xrightarrow{6R} \{31\}$	3 716 639	3	p_4	–	47 sec
🟦 Speck-96	$\overline{\{88, 89, 90\}} \xrightarrow{6R} \{47\}$	Infeasible	–	0	0 sec	0 sec
▲ Speck-96	$\overline{\{87, 89, 90\}} \xrightarrow{6R} \{47\}$	199 540	1	0	7.4 days	5 sec
◆ Speck-96	$\overline{\{87, 88, 90\}} \xrightarrow{6R} \{47\}$	638 800	1	0	–	15 sec
◆ Speck-96	$\overline{\{87, 88, 89\}} \xrightarrow{6R} \{47\}$	4 271 106	1	0	–	18.6 min
🟦 Speck-128	$\overline{\{120, 121, 122\}} \xrightarrow{6R} \{63\}$	Infeasible	–	–	0 sec	0 sec
◆ Speck-128	$\overline{\{119, 121, 122\}} \xrightarrow{6R} \{63\}$	34 739 144	1	0	–	16 hrs
◆ Speck-128	$\overline{\{119, 120, 122\}} \xrightarrow{6R} \{63\}$	98 135 968	1	0	–	7.33 days
◆ Speck-128	$\overline{\{119, 120, 121\}} \xrightarrow{6R} \{63\}$	878 711 604	1	0	–	8.75 days

where the superpolies p_0, p_1, p_2 and p_3 are listed as follows:

$$
\begin{aligned}
p_0 = {} & k_{13}^{(4)} + k_{12}^{(4)}k_{13}^{(3)} + k_6^{(4)} + k_6^{(4)}k_{12}^{(3)} + k_5^{(4)}k_{13}^{(3)} + k_5^{(4)}k_6^{(3)} + k_8^{(3)} + k_8^{(3)}k_{12}^{(3)} + k_8^{(3)}k_5^{(3)} + k_7^{(3)}k_{13}^{(3)} \\
& + k_7^{(3)}k_6^{(3)} + k_7^{(3)}k_8^{(2)} \\
p_1 = {} & k_{28}^{(2)}k_0^{(4)}k_1^{(4)}k_2^{(4)}k_4^{(4)}k_5^{(4)}k_6^{(4)}k_7^{(4)}k_8^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{19}^{(4)} \\
& + k_4^{(2)}k_0^{(4)}k_1^{(4)}k_2^{(4)}k_5^{(4)}k_6^{(4)}k_7^{(4)}k_8^{(4)}k_{12}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{16}^{(4)}k_{17}^{(4)}k_{18}^{(4)} \\
& + k_{31}^{(1)}k_0^{(4)}k_7^{(4)}k_8^{(4)}k_{10}^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{18}^{(4)}k_{20}^{(4)} \\
p_2 = {} & k_{28}^{(2)}k_0^{(4)}k_1^{(4)}k_2^{(4)}k_4^{(4)}k_5^{(4)}k_6^{(4)}k_7^{(4)}k_8^{(4)}k_9^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{12}^{(4)}k_{14}^{(4)}k_{19}^{(4)} + k_4^{(2)}k_0^{(4)}k_5^{(4)}k_6^{(4)}k_{10}^{(4)}k_{13}^{(4)}k_{19}^{(4)} \\
& + k_{31}^{(1)}k_0^{(4)}k_2^{(4)}k_3^{(4)}k_4^{(4)}k_7^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{12}^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{16}^{(4)}k_{17}^{(4)}k_{20}^{(4)} \\
p_3 = {} & k_0^{(4)}k_1^{(4)}k_3^{(4)}k_7^{(4)}k_9^{(4)}k_{12}^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{16}^{(4)}k_{18}^{(4)}k_{21}^{(4)} + k_{23}^{(2)}k_0^{(4)}k_1^{(4)}k_9^{(4)}k_{11}^{(4)}k_{12}^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{17}^{(4)} \\
& + k_{12}^{(2)}k_0^{(4)}k_1^{(4)}k_3^{(4)}k_4^{(4)}k_5^{(4)}k_6^{(4)}k_7^{(4)}k_{14}^{(4)}k_{15}^{(4)} + k_9^{(2)}k_0^{(4)}k_2^{(4)}k_5^{(4)}k_8^{(4)}k_{12}^{(4)}k_{15}^{(4)}k_{16}^{(4)} \\
& + k_4^{(2)}k_0^{(4)}k_2^{(4)}k_4^{(4)}k_7^{(4)}k_9^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{18}^{(4)} + k_1^{(2)}k_0^{(4)}k_1^{(4)}k_3^{(4)}k_4^{(4)}k_5^{(4)}k_6^{(4)}k_7^{(4)}k_{14}^{(4)}k_{15}^{(4)} \\
& + k_{31}^{(1)}k_0^{(4)}k_3^{(4)}k_6^{(4)}k_{11}^{(4)}k_{13}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{16}^{(4)}k_{17}^{(4)}k_{18}^{(4)} + k_{28}^{(1)}k_0^{(4)}k_1^{(4)}k_2^{(4)}k_5^{(4)}k_6^{(4)}k_{10}^{(4)}k_{14}^{(4)}k_{15}^{(4)} \\
p_4 = {} & k_{31}^{(2)}k_0^{(4)}k_2^{(4)}k_4^{(4)}k_7^{(4)}k_8^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{13}^{(4)}k_{17}^{(4)} + k_{27}^{(2)}k_0^{(4)}k_5^{(4)}k_7^{(4)}k_8^{(4)}k_9^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{12}^{(4)}k_{13}^{(4)}k_{14}^{(4)} \\
& + k_{23}^{(2)}k_0^{(4)}k_2^{(4)}k_5^{(4)}k_{10}^{(4)}k_{11}^{(4)}k_{13}^{(4)}k_{16}^{(4)} + k_{15}^{(2)}k_0^{(4)}k_1^{(4)}k_3^{(4)}k_5^{(4)}k_8^{(4)}k_9^{(4)}k_{13}^{(4)}k_{16}^{(4)} \\
& + k_7^{(2)}k_0^{(4)}k_1^{(4)}k_3^{(4)}k_4^{(4)}k_7^{(4)}k_8^{(4)}k_{10}^{(4)}k_{14}^{(4)}k_{15}^{(4)}k_{16}^{(4)} + k_3^{(2)}k_0^{(4)}k_4^{(4)}k_5^{(4)}k_6^{(4)}k_9^{(4)}k_{11}^{(4)}k_{13}^{(4)}k_{14}^{(4)} \\
& + k_{30}^{(1)}k_0^{(4)}k_1^{(4)}k_2^{(4)}k_4^{(4)}k_6^{(4)}k_7^{(4)}k_9^{(4)}k_{10}^{(4)}k_{12}^{(4)}k_{13}^{(4)}k_{15}^{(4)}
\end{aligned}
$$

nature of the modular addition. Let $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ be the exponents of the two inputs and one output of a modular addition. According to Theorem 2, if $\boldsymbol{w} = (0, 0, \ldots, 0, 1)$, $\{\boldsymbol{u}, \boldsymbol{v}\}$ must be $\{(0, 0, \ldots, 0, 0), (0, 0, \ldots, 0, 1)\}$ or $\{(0, 0, \ldots, 0, 1), (0, 0, \ldots, 0, 0)\}$; if $\boldsymbol{w} = (0, 0, \ldots, 0, 0)$, $(\boldsymbol{u}, \boldsymbol{v})$ has to be $\{(0, 0, \ldots, 0, 0), (0, 0, \ldots, 0, 0)\}$. Consequently, we can divide the search for r-round Speck-$2n$ to four cases consisting of one $(r-1)$-round search and three $(r-2)$-round search, as shown in Figure 2. Note that all cases depicted in Figure 2 correspond to scenarios where the exponents of the last two rounds of key bits are all-zero. This is because according to the XOR propagation rules, the non-zero exponents of the last two rounds of key bits will only lead to infeasible propagations. Thus, all feasible cases arise when the exponents of these key bits are all-zero. The number of monomial trails for r-round Speck-$2n$ will be the summation of trails from all these four cases. The second

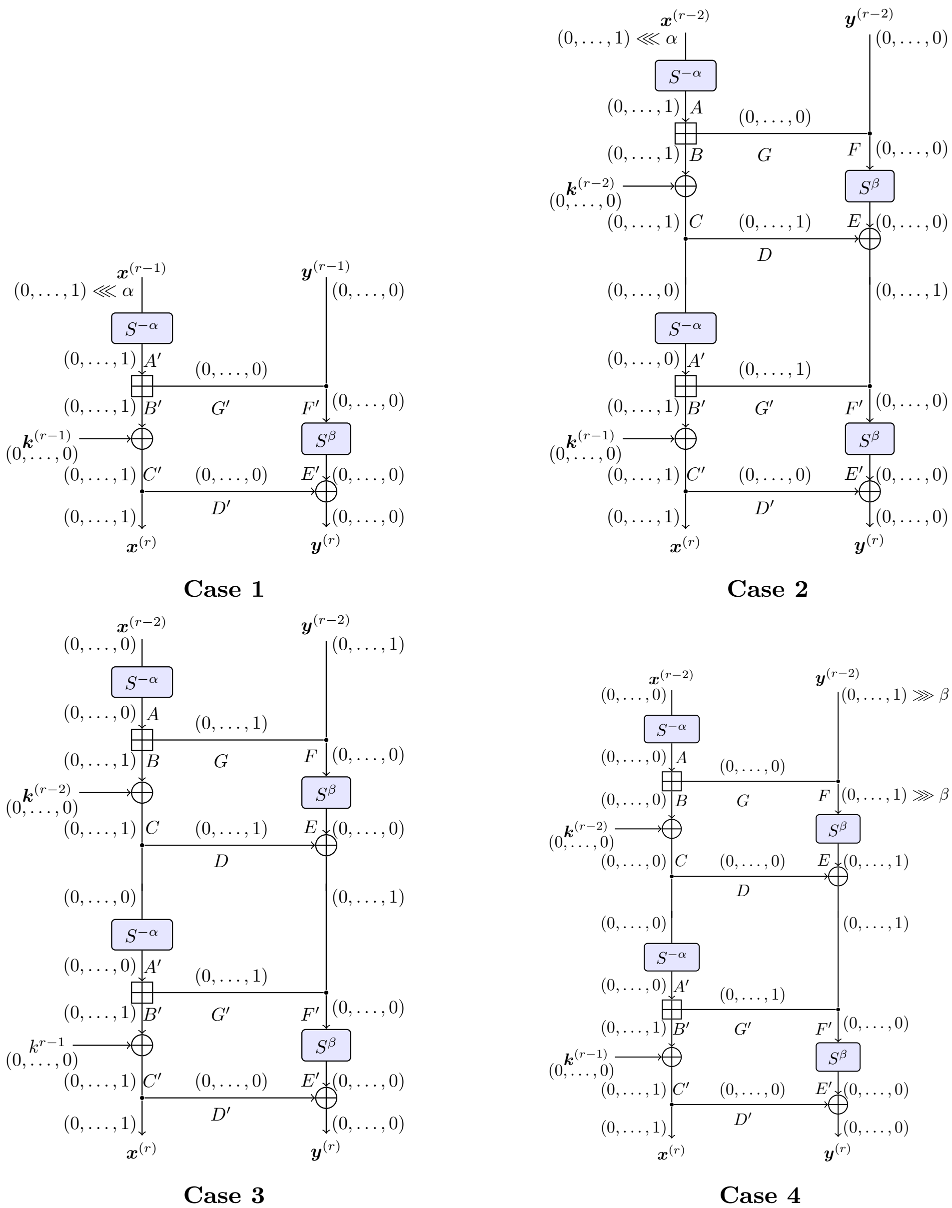


Figure 3: A r-round search is divided into 4 cases.

optimization strategy is to divide the search according to the intermediate states, which is similar to the method used in [HLM+20]. We first extract all possible solutions for intermediate states after the second round, denoted by

$$S_2 = \{\boldsymbol{u}^{(2)} : \boldsymbol{u}^{(0)} \rightsquigarrow \boldsymbol{u}^{(2)} \rightsquigarrow \boldsymbol{u}^{(R)}\},$$

Then, for each value in S_2, we count the solutions in $\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(2)}$ and $\boldsymbol{u}^{(2)} \bowtie \boldsymbol{u}^{(R)}$. The number of all solutions in $\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(R)}$ is calculated by

$$|\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(R)}| = \sum_{\boldsymbol{u}^{(2)} \in S_2} |\boldsymbol{u}^{(0)} \bowtie \boldsymbol{u}^{(2)}| \times |\boldsymbol{u}^{(2)} \bowtie \boldsymbol{u}^{(R)}|.$$

With these two optimization strategies, the search is accelerated significantly. We recommend that readers read our code to better understand these optimizations.

https://github.com/hukaisdu/MP_of_ModularAddition.git

Re-detect the integral distinguishers in [WHG+20]. The results of **Model 1** with $R = 6$ for Speck-32, -48, -64, -96 show that the maximum of $\mathcal{K} = \sum_{r=0}^{5} \sum_{i=0}^{2n-1} w^{(r)_i}$ is only 1 with the input and output settings found by [WHG+20] (denoted as ▲ in Table 2). That means the superpoly contains only one key variable. We then turn on the `PoolSearchMode` of Gurobi option for **Model 1** (denoted by **Model*** 1 in the following context) to count the number of monomial trails. The number of trails is given in Table 2. For each key monomial that appears in the trails, the number of corresponding trails is even. Therefore, by counting the number of trails, we easily re-detect the same integral distinguishers as [WHG+20] according to Theorem 1. Furthermore, our method obtains the results faster compared to the approach presented in [WHG+20].

Finding new integral distinguishers for all Speck instances. Inspired by the above observation, the max value of $\sum_{r=0}^{R-1} \sum_{i=0}^{2n-1} w_i^{(r)}$ under a certain pair of input and output exponents $\boldsymbol{u}^{(0)}$ and $\boldsymbol{u}^{(R)}$, (i.e., $\max \mathcal{K}$ for R-round Speck-$2n$ under the fixed values of $\boldsymbol{u}^{(0)}$ and $\boldsymbol{u}^{(R)}$) indeed indicates how complex the corresponding superpoly is. When the value of $\max \mathcal{K}$ is relatively small, the superpoly is expected to be simple, and then there is a possibility of recovering the superpoly by counting all monomial trails. If the number of monomial trails is even for all the key monomials, we obtain integral distinguishers. When the value of $\max \mathcal{K}$ is large, we expect that the number of monomial trails will be too huge for the solver to count within a reasonable amount of time.

Since all integral distinguishers found in [WHG+20] have $2n - 2$ (Speck-32) or $2n - 3$ (for other Speck instances) active bits in the plaintext and one balanced ciphertext bit in the rightmost of the left branch, we use **Model 1** to traverse all possible $\boldsymbol{u}^{(0)}$ with a hamming weight $2n - 2$ (for Speck-32) or $2n - 3$ (for others Speck variants) and $\boldsymbol{e}_{n-1}$ to test $\max \mathcal{K}$. For Speck-32, -48, -64, we try all such cases with $\max \mathcal{K} \leq 3$; for Speck-96 and -128, we try all such cases with $\max \mathcal{K} = 1$[2]. We expect that the corresponding superpolies of these cases are simple so we can count the monomial trails. The experimental results meet our expectations. When we apply **Model*** **1** to all the above cases, many of them lead to integral distinguishers. As a result of our proposed model, we detect new integral distinguishers, as listed in Table 2. Some are not integral distinguishers, but their superpolies are indeed simple; we then list their superpolies below the table.

[2]Since the superpolies of Speck-96 and 128 are too heavy, cases where $\max \mathcal{K} > 1$ becomes impractical with our method.

4.3 Calculate Lower Bounds on Speck-32's Algebraic Degree

In [HLLT20], Hebborn et al. introduced how to determine lower bounds on the algebraic degrees for output bits of a block cipher and applied this method to Sbox-based block ciphers Gift-64, Skinny-64, AES and Present. Their method has not been explored for ARX ciphers until now. With our perfect model for the modular addition, we managed to apply the method to Speck-32 and determined the exact lower bounds for all the output bits. Further, we prove that all 31-degree terms of the plaintext do appear in all output bits.

The idea in [HLLT20] is to assume that all round keys are independent and find a proper monomial of them denoted by $\boldsymbol{\pi}_{(\boldsymbol{t}^{(0)},\boldsymbol{t}^{(1)},\ldots,\boldsymbol{t}^{(R-1)})}(\boldsymbol{k}^{(0)},\boldsymbol{k}^{(1)},\ldots,\boldsymbol{k}^{(R-1)})$ that satisfies

$$\left|(\boldsymbol{u},\boldsymbol{t}^{(0)},\boldsymbol{t}^{(1)},\ldots,\boldsymbol{t}^{(R-1)}) \bowtie \boldsymbol{v})\right| = 1 \bmod 2 \tag{4}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the exponents of the plaintext and ciphertext, respectively. The approach of finding the monomial of a key in [HLLT20] is a heuristic one. Let $\boldsymbol{u}^{(0)},\boldsymbol{u}^{(1)},\ldots,\boldsymbol{u}^{(R)}$ be the exponents of the intermediate states. Firstly, they search for a pair of proper $\boldsymbol{u}^{(R-1)}$ and $\boldsymbol{t}^{(R-1)}$ whose Hamming weights are as high as possible and satisfy

$$\left|(\boldsymbol{u}^{(R-1)},\boldsymbol{t}^{(R-1)}) \bowtie \boldsymbol{u}^{(R)}\right| = 1 \bmod 2.$$

Then, they iteratively find a pair of $\boldsymbol{u}^{(R-2)}$ and $\boldsymbol{t}^{(R-2)}$ whose Hamming weights are as high as possible and satisfy

$$\left|(\boldsymbol{u}^{(R-2)},\boldsymbol{t}^{(R-2)}) \bowtie \boldsymbol{u}^{(R-1)}\right| = 1 \bmod 2$$
$$\left|(\boldsymbol{u}^{(R-2)},\boldsymbol{t}^{(R-2)},\boldsymbol{t}^{(R-1)}) \bowtie \boldsymbol{u}^{(R)}\right| = 1 \bmod 2.$$

Then they repeat this process to R_{mid} and obtain $(\boldsymbol{t}^{(mid+1)},\ldots,\boldsymbol{t}^{(R-1)})$ and $(\boldsymbol{u}^{(mid+1)},\ldots,\boldsymbol{u}^{(R-1)})$. Finally, $(\boldsymbol{t}^{(0)},\ldots,\boldsymbol{t}^{(mid-1)})$ are searched to satisfy Equation 4.

We apply the same idea to 7-round Speck-32 as there are already integral distinguishers for 6 rounds. For each output bit and each 31-degree monomial of the plaintext, we find a monomial of the round key under which the number of monomial trails connecting the plaintext and ciphertext monomials is an odd number. Thus, for each ciphertext bit, all 31-degree terms indeed appear in its polynomial.

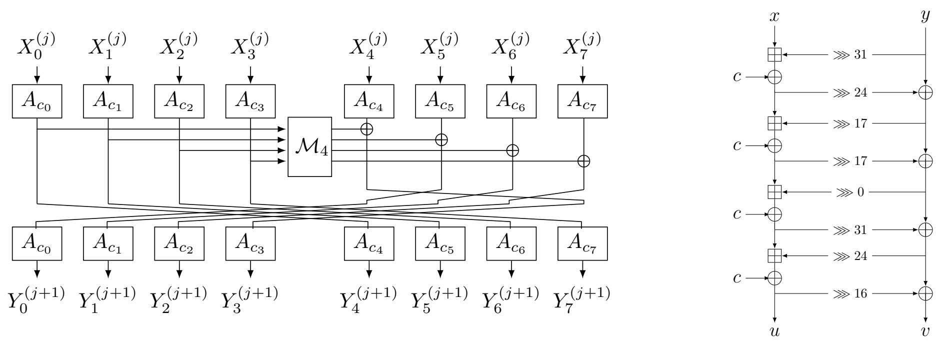
Remark. In [HLLT20], the authors went further to prove that for all linear combinations of the ciphertext bits, all $(2n-1)$-degree (the block size of Speck-$2n$ is $2n$) plaintext monomials appear. To achieve this, we need to calculate the number of trails connecting all $(2n-1)$-degree plaintext monomials and single ciphertext bit monomials for some key monomials and make a matrix of full rank correspondingly. Once it is done, if there is a pre-whitening key, the same authors provided a strong and tight security guarantee against all integral distinguishers for a block cipher in [HLLT21]. In our experiments, after finding the key monomials for each 31-degree plaintext monomial and ciphertext bit, we failed to prove the full rank property for the corresponding matrix. This is because, for some plaintext/ciphertext monomial pairs, the number of solutions is too large to finish the search. We also applied the method to Speck-48. We want to prove that all 47-degree monomials of plaintext appear in the ciphertext bit. Unfortunately, for some plaintext and ciphertext monomials, we cannot find a key monomial that yields odd-number trails (all key monomials used during our search process yield even-number trails). As a result, we could not complete the first part of the methodology stated above. In Section 6, we show that if Speck uses a different rotation parameter, the proof can be easily achieved even for 6 rounds.

5 Application to Alzette and Sparkle

In this section, we apply our compact MP models of the modular addition to an ARX Sbox Alzette and Sparkle permutation. For Alzette, we give the exact algebraic degrees of their weakest bits, i.e., the rightmost bit of the left branch, which provides a strong argument for the security of Alzette. We also present integral distinguishers for Alzette after using our compact MP model in Appendix B. For Sparkle, we obtain improved integral distinguishers upon the previous best integral distinguishers found.

5.1 Brief Introduction to Sparkle and Alzette

Sparkle is a family of ARX-based permutations [BBdS+19] consisting of three members, i.e., Sparkle256, Sparkle384 and Sparkle512 according to their sizes. Sparkle is the underlying permutation of the NIST LWC finalist Sparkle suite. Figure 4a illustrates the structure of the 1.5-step Sparkle512 permutation as an example, where $A_{c_i} : \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32} \times \mathbb{F}_2^{32}$ is an ARX box parameterized with a constant c_i named as Alzette (see Figure 4b). For convenience, the input and output of the j-th step of the Sparkle permutation are denoted by $X^{(j)} = (X_0^{(j)}, \dots, X_{z-1}^{(j)})$ and $Y^{(j)} = (Y_0^{(j)}, \dots, Y_{z-1}^{(j)})$, where $z = 4, 6, 8$ for Sparkle256, Sparkle384 and Sparkle512, respectively.



(a) The structure of 1.5-step of Sparkle512 permutation. In this instance, there are eight 64-bit branches.

(b) Alzette parameterized by c.

Figure 4: Illustration of Sparkle and Alzette.

5.2 Exact Algebraic Degree of Alzette

In this section, we show that it is possible to calculate the exact algebraic degree for specific output bits of round-reduced Alzette.

Recall that the algebraic degree of a Boolean function f is defined as

$$\deg(f) = \max_{\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to f} \mathrm{wt}(\boldsymbol{u}^{(0)}). \tag{5}$$

In [HSWW20], the authors used the MP to determine the exact algebraic for Trivium up to 834 rounds. In this paper, we apply this method to Alzette output bits.

1. For each output bit denoted by f, we find the exponent $\boldsymbol{u}$ with the largest Hamming weight such that $\pi_{\boldsymbol{u}}(\boldsymbol{x}^{(0)}) \rightsquigarrow f$.

2. Compute the size of the monomial hull $|\pi_{\boldsymbol{u}}(\boldsymbol{x}^{(0)}) \bowtie f|$. If the value of $|\pi_{\boldsymbol{u}}(\boldsymbol{x}^{(0)}) \bowtie f|$ is odd, then the exact degree is $wt(\boldsymbol{u})$. Otherwise, $\pi_{\boldsymbol{u}}(\boldsymbol{x}^{(0)}) \not\to f$ then we have to exclude this vector from the search model and repeat the first step until we find the desired exponent $\boldsymbol{u}$.

In [BBdS+20], the designers used experimental methods to show that for each bit of Alzette, there is a monomial of degree 32. With our method, we can find monomials with higher degrees that appear in the ANF of output bits.

Table 3: Exact algebraic degrees of 2-round Alzette output bits.

Bit Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Degree	62	61	60	59	58	57	56	55	54	52	50	48	46	44	42	40
Bit Index	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Degree	38	36	34	32	30	28	26	24	22	20	18	16	14	12	11	10
Bit Index	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Degree	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	11
Bit Index	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Degree	10	62	61	60	59	58	57	56	55	54	52	50	48	46	44	42

Since the results are the same for all the constants c_0 to c_7 in Alzette, we shall present the result for c_0. For 2-round Alzette, we find the exact algebraic degrees of all output bits. These are listed in Table 3. For round 3 onwards, the number of trails becomes too large to calculate for many output bits. However, from Table 3, we can see that several rightmost bits of the left branch as well as the corresponding bits of the right branch up to the last rotation constant have smaller algebraic degrees (for 2 rounds, the last rotation constant is 17, as shown in Figure 4b, so the 48-th output bit is also of low degree). This indicates that the rightmost bits of the left branch of (round-reduced) Alzette have lower algebraic degrees. By calculating the upper bounds of algebraic degrees for the output of 3-round Alzette, we indeed find that those bits (listed in Table 4) have smaller upper bounds. We manage to confirm their exact degrees and find their algebraic degrees are at least 42. The other bits are expected to have higher degrees than these bits (as their degree's upper bound are 63). For 4-round Alzette, we calculate the exact degrees of the rightmost bit of the left branch and the corresponding 47-th bit of the output. Both of their exact degrees are 63.

Table 4: Exact algebraic degrees of 3-round Alzette output bits.

Bit Index	24	25	26	27	28	29	30	31	55	56	57	58	59	60	61	62
Degree	60	58	55	52	49	46	44	42	60	58	55	52	50	48	46	44

5.3 Improved Integral Distinguishers for Sparkle

In [BBdS+19], the designers analyzed the Sparkle families using the 2BDP for the integral distinguishers. Using our compact MP model for modular addition, we managed to find new integral distinguishers. Compared to those found in [BBdS+19] using 2BDP, the newly found integral distinguishers have one less active bit, i.e., the data complexities are reduced to half of the previous ones. We show the integral distinguisher in Table 5. Note that since Sparkle is a family of permutations, the integral distinguishers here are a bit different from those for block ciphers. It is more like a high-order differential distinguisher [Xue94], i.e., when the input bits are those active bits of our integral distinguisher, the algebraic degree is strictly smaller than the number of active bits. In our search, we let the inactive bits be free variables; thus, our integral distinguishers work even if any key is XORed with the input data.

Check other ciphers. Other than Speck, Alzette and Sparkle, we have tried our compact MP model on other ciphers like LEA [HLK+13], HIGHT [HSH+06] and SHACAL-2 [HD01].

Table 5: Integral distinguishers of Sparkle families

Method	Cipher	Integral Distinguishers	No. of Solution	Time
[BBdS+19]	Sparkle256	$\overline{\{0-62\}} \xrightarrow{4.5R} \{128\text{-}255\}$	–	–
	Sparkle384	$\overline{\{0-126\}} \xrightarrow{4.5R} \{192\text{-}383\}$	–	–
	Sparkle512	$\overline{\{0-190\}} \xrightarrow{4.5R} \{256\text{-}511\}$	–	–
Ours	Sparkle256	$\overline{\{0-63\}} \xrightarrow{4.5R} \{128\text{-}255\}$	Infeasible	220.883 sec
	Sparkle384	$\overline{\{0-127\}} \xrightarrow{4.5R} \{192\text{-}383\}$	Infeasible	333.903 sec
	Sparkle512	$\overline{\{0-191\}} \xrightarrow{4.5R} \{256\text{-}511\}$	Infeasible	532.853 sec

We have managed to re-detect integral distinguishers identified by [SWW17] and verified that even using our method, there are no more integral distinguishers for these ciphers. A detailed discussion is provided in Appendix C.

6 Revisiting the Rotation Parameters of Speck Family

The Speck family (as well as Simon family) of ciphers were designed by NSA, and they have been criticized for not giving a design rationale [BSS+15]. To respond these criticism and support their ISO/IEC standardization attempt, the designers provided design rational for Speck (and Simon) [BSS+17]. However, the confusion over the choice of various parameters in Speck has left an impression on the community and attracted lots of controversy [AL21].

Although a general opinion is that the differential and linear attacks are the two main threats to Speck and the integral attack is not, an understanding of how the parameters, especially the rotation constants, affect the security against the integral distinguishers is still important. Indeed, integral attacks help to check if a cipher has some algebraic vulnerabilities.

In this section, we show a heuristic method to scrutinize the strength of a Speck instance with a rotation constant pair (α, β) against the integral distinguishing attacks. We manage to find many (α, β) pairs that make the Speck instance resistance against integral distinguishers for 6 rounds and above (up to checking all $2n-1$ active bits for plaintext).

For convenience, we denote a Speck instance with $2n$ block size and (α, β) rotation constant pair by $\mathsf{Speck}_{\alpha,\beta}$-$2n$. According to Section 4, $\max \mathcal{K} = \max \sum_{r=0}^{R-1} \sum_{i=0}^{2n-1} w_i^{(r)}$ can be a criterion to predict how complex the corresponding superpoly is. When $\max \mathcal{K}$ is larger, the superpoly is more likely not to be zero. This suggests that finding an integral distinguisher is unlikely, or at the very least, counting all trails is extremely challenging, rendering it infeasible to identify integral distinguishers within the limited resources. These two cases suggest that the corresponding output bit tested consists of a complicated superpoly, indicating strong algebraic properties.

Therefore, our method of checking different (α, β) pairs is easy. Firstly, we traverse the (α, β) combinations in a space, then try all possible cases in which the number of active plaintext bits is $2n-1$. Finally, we calculate the $\max \mathcal{K}$ for the $(n-1)$-th ciphertext bit as this bit is the weakest. Considering that the families of Speck are designed as lightweight ciphers, $\alpha \leq 8$ and $\beta \leq 8$ will make it efficient in a low-end 8-bit processor, so we set a limit on α and β such that $1 \leq \alpha \leq 8$ and $1 \leq \beta \leq 8$ and $\alpha \neq \beta$. In table 6, we list the 10 (α, β) pairs with their minimum of $\max \mathcal{K}$ for Speck-32, -48, -64, -96 and -128 among all $\boldsymbol{u}$ with $2n-1$ active bits (i.e., for each $\boldsymbol{u}$ with $2n-1$ active bits, we calculate $\max \mathcal{K}$ for this $\boldsymbol{u}$, the values in Table 6 are the minimum of $\max \mathcal{K}$ for all $\boldsymbol{u}$).

To show that the (α, β) parameters truly strengthen Speck's resistance against the

Table 6: Ten first 10 (α, β) pairs with smallest $\max \mathcal{K}$.

Speck-32										
(α, β)	(15, 14)	(15, 13)	(14, 13)	(15, 12)	(14, 12)	(13,12)	(15, 11)	(14, 11)	(13, 11)	(12, 11)
min. of $\max \mathcal{K}$	15	14	14	13	13	13	12	12	12	12
Speck-48										
(α, β)	(23, 22)	(23, 21)	(22, 21)	(23, 20)	(22, 20)	(21,20)	(23, 19)	(22, 19)	(21, 19)	(20, 19)
min. of $\max \mathcal{K}$	23	22	22	21	21	21	20	20	20	20
Speck-64										
(α, β)	(31, 30)	(31, 29)	(30, 29)	(31, 28)	(30, 28)	(29,28)	(31, 27)	(30, 27)	(29, 27)	(28, 27)
min. of $\max \mathcal{K}$	31	30	30	29	29	29	28	28	28	28
Speck-96										
(α, β)	(47, 46)	(47, 45)	(46, 45)	(47, 44)	(46, 44)	(45,44)	(47, 43)	(46, 43)	(45, 43)	(44, 43)
min. of $\max \mathcal{K}$	47	46	46	45	45	45	44	44	44	44
Speck-128										
(α, β)	(63, 62)	(63, 61)	(62, 61)	(63, 60)	(62, 60)	(61, 60)	(63, 59)	(62, 59)	(61, 59)	(60, 59)
min. of $\max \mathcal{K}$	63	62	62	61	61	61	61	60	60	60

integral distinguishers, we find that all 31-degree monomials have appeared in all output bits of 6-round $\text{Speck}_{15,14}$-32. Note that for the original Speck-32, i.e., $\text{Speck}_{7,2}$-32, 7 rounds are necessary for this property. We also find that all 47-degree and 63-degree monomials have appeared in every output bit of 6-round $\text{Speck}_{23,22}$-48 and $\text{Speck}_{31,30}$-64. That is to say, with these constraints, it becomes much easier to prove a stronger resistance for Speck against integral distinguishers with even fewer rounds.

7 Conclusion and Open Questions

In this paper, we introduced a perfect MP model for modular addition based on a theorem that was proposed about 20 years ago. The model for modular addition is more precise and simple than all previous models. The perfect model does not introduce any redundant MP trails for the modular addition, which makes counting the number of trails easier. With the model, we re-find the integral distinguishers found in [WHG+20], and the search time is significantly reduced from hours/days to seconds/minutes. With a criterion that indicates how complex the superpoly is, we detect more integral distinguishers for Speck-48, -64, -96 and -128. For the first time, we can compute the exact degrees for Speck-32 and Alzette (for specific output bits). We also revisited the choices of rotation parameters used in Speck and provided a method to choose better parameters, making the resistance against integral distinguishers stronger. With our parameters, 6 rounds of Speck-32, -48 and -64 are sufficient to resist integral distinguishers, which is one less round than the real ciphers.

In [HLLT20, HLLT21], the authors provided a strong and tight guarantee for resisting integral attacks. With our model, there is potential to give the same guarantee for ARX ciphers, and we leave this for future work. Furthermore, in order to find more integral distinguishers, one could look into ways to speed up counting the number of trails.

Acknowledgments

We would like to thank Tim Beyne and the other anonymous reviewers for their constructive comments and suggestions to improve the quality of the paper. Kai Hu is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014, U2336207), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025), Department of Science Technology of Shandong Province (No.SYS202201), and the Program of Qilu Young Scholars of Shandong University.

References

[AL21] Tomer Ashur and Atul Luykx. An account of the ISO/IEC standardization of the simon and speck block cipher families. In Gildas Avoine and Julio Hernandez-Castro, editors, *Security of Ubiquitous Computing Systems*, pages 63–78. Springer International Publishing, 2021.

[BBdS+19] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, and Alex Biryukov. Schwaemm and esch: lightweight authenticated encryption and hashing using the sparkle permutation family. *NIST round*, 2, 2019.

[BBdS+20] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Alzette: A 64-bit arx-box - (feat. CRAX and TRAX). In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 419–448. Springer, 2020.

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 654–682. Springer, 2016.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BKL+07] 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, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BS05] An Braeken and Igor A. Semaev. The ANF of the composition of addition and multiplication mod 2^n with a Boolean function. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 112–125. Springer, 2005.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 175:1–175:6. ACM, 2015.

[BSS+17] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Notes on the design and analysis of SIMON and SPECK. *IACR Cryptol. ePrint Arch.*, page 560, 2017.

[BV23] Tim Beyne and Michiel Verbauwhede. Integral cryptanalysis using algebraic transition matrices. *IACR Trans. Symmetric Cryptol.*, 2023(4):244–269, 2023.

[BV24] Tim Beyne and Michiel Verbauwhede. Ultrametric integral cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 722, 2024.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *Fast Software Encryption - FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.

[DL22] Patrick Derbez and Baptiste Lambin. Fast MILP models for division property. *IACR Trans. Symmetric Cryptol.*, 2022(2):289–321, 2022.

[HD01] Handschuh Helena and Naccache David. SHACAL, NESSIE, 2001. *Archive available at https://www.cosic.esat.kuleuven.be/nessie/tweaks.html*, 2001.

[HLK+13] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Donggeon Lee. LEA: A 128-bit block cipher for fast encryption on common processors. In Yongdae Kim, Heejo Lee, and Adrian Perrig, editors, *Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers*, volume 8267 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2013.

[HLLT20] Phil Hebborn, Baptiste Lambin, Gregor Leander, and Yosuke Todo. Lower Bounds on the Degree of Block Ciphers. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 537–566. Springer, 2020.

[HLLT21] Phil Hebborn, Baptiste Lambin, Gregor Leander, and Yosuke Todo. Strong and tight security guarantees against integral distinguishers. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 362–391. Springer, 2021.

[HLM+20] 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 Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020.

[HSH+06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A new block cipher suitable for low-resource device. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.

[HSWW20] 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, *Advances* *in Cryptology - ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 446–476. Springer, 2020.

[HW19] Kai Hu and Meiqin Wang. Automatic search for a variant of division property using three subsets. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 412–432. Springer, 2019.

[HWW20] Kai Hu, Qingju Wang, and Meiqin Wang. Finding bit-based division property for ciphers with complex linear layers. *IACR Trans. Symmetric Cryptol.*, 2020(1):396–424, 2020.

[KW02] Lars R. Knudsen and David A. Wagner. Integral Cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[SWLW18] Ling Sun, Wei Wang, Ru Liu, and Meiqin Wang. Milp-aided bit-based division property for ARX ciphers. *Sci. China Inf. Sci.*, 61(11):118102:1–118102:3, 2018.

[SWW17] 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, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 128–157. Springer, 2017.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377. Springer, 2016.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

[Udo21] Aleksei Udovenko. Convexity of division property transitions: Theory, algorithms and compact models. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 332–361. Springer, 2021.

[WHG+19] 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, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 398–427. Springer, 2019.

[WHG+20] 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.

[Xue94] L. Xuejia. Higher Order Derivatives and Differential Cryptanalysis. In *Communications and cryptography*, pages 227–233. 1994.

[XZBL16] 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, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

A Basic MP model for R-Round Speck-2n

$\mathcal{M}_R$: Basic MP model for R-round Speck-2n.

For $0 \leq i < R$:

$$(G, F) \xleftarrow{\text{COPY}} B$$

$$C \xleftarrow{\text{MODULAR ADDITION}} (S^{-\alpha}(A), G)$$

$$D \xleftarrow{\text{XOR}} (J, C)$$

$$(H, E) \xleftarrow{\text{COPY}} D$$

$$I \xleftarrow{\text{XOR}} (E, S^{\beta}(F))$$

$A, B, C, D, E, F, G, H, I, J$ are all n-bit variables representing exponents

B Integral Distinguishers of Alzette

Table 7 illustrate all the integral distinguisher found for 6 rounds of Alzette. We did not find any integral distinguisher for 7 rounds.

C Application to Other Ciphers

We have also applied our methods on other ciphers, namely LEA [HLK+13], HIGHT [HSH+06] and SHACAL-2 [HD01]. We tried to increase the number of rounds by 1 and to increase the inactive bits by 1 from the integral distinguisher found in Table 8, but without counting the monomial trails, our model leads to the same results as [SWW17], as shown in Table 8.

Table 7: Integral distinguishers for Alzette with constant c_0

Division Property	No. of Solution	Time
$\overline{\{40\}} \xrightarrow{6R} \{28,29,30,31,45,46,47,48\}$	Infeasible	3.525 sec
$\overline{\{41\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	1.570 sec
$\overline{\{42\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	1.615 sec
$\overline{\{43\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	1.631 sec
$\overline{\{44\}} \xrightarrow{6R} \{24,25,26,27,28,29,30,31,41,42,43,44,45,46,47,48\}$	Infeasible	1.645 sec
$\overline{\{45\}} \xrightarrow{6R} \{24,25,26,27,28,29,30,31,41,42,43,44,45,46,47,48\}$	Infeasible	1.697 sec
$\overline{\{46\}} \xrightarrow{6R} \{24,25,26,27,28,29,30,31,41,42,43,44,45,46,47,48\}$	Infeasible	2.366 sec
$\overline{\{47\}} \xrightarrow{6R} \{24,25,26,27,28,29,30,31,41,42,43,44,45,46,47,48\}$	Infeasible	6.743 sec
$\overline{\{48\}} \xrightarrow{6R} \{24,25,26,27,28,29,30,31,41,42,43,44,45,46,47,48\}$	Infeasible	5.875 sec
$\overline{\{49\}} \xrightarrow{6R} \{31,48\}$	Infeasible	6.997 sec
$\overline{\{50\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	6.214 sec
$\overline{\{51\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	6.396 sec
$\overline{\{52\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	6.410 sec
$\overline{\{53\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	6.066 sec
$\overline{\{54\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	3.620 sec
$\overline{\{55\}} \xrightarrow{6R} \{25,26,27,28,29,30,31,42,43,44,45,46,47,48\}$	Infeasible	1.560 sec

We also tried to count the number of trails. However, the $\max \mathcal{K}$ value is very large for all possible $n-1$ active plaintext patterns (assume the block size is n bits). We observe that $\max \mathcal{K}$ are more than 120 for both LEA and HIGHT. For SHACAL-2, the computation of $\max \mathcal{K}$ is difficult for most bits, suggesting that $\max \mathcal{K}$ are large. Therefore, counting the number of trails is not practical for these ciphers.

Table 8: Integral distinguishers for LEA, HIGHT, SHACAL-2.

Cipher	Integral Distinguishers	No. of Solution
LEA	$\overline{\{27-31\}} \xrightarrow{8R} \{36\}$	Infeasible
HIGHT	$\overline{\{14\}} \xrightarrow{18R} \{6,7\}$	Infeasible
HIGHT	$\overline{\{14\}} \xrightarrow{18R} \{6,7\}$	Infeasible
HIGHT	$\overline{\{31\}} \xrightarrow{18R} \{7\}$	Infeasible
HIGHT	$\overline{\{46\}} \xrightarrow{18R} \{38,39\}$	Infeasible
HIGHT	$\overline{\{47\}} \xrightarrow{18R} \{38,39\}$	Infeasible
HIGHT	$\overline{\{63\}} \xrightarrow{18R} \{39\}$	Infeasible
SHACAL-2	$\overline{\{23-31,154-159\}} \xrightarrow{17R} \{249-255\}$	Infeasible
Chaskey	$\overline{\{64-127\}} \xrightarrow{3R} \{74-79\}$	Infeasible
Chaskey	$\overline{\{127\}} \xrightarrow{4R} \{78-79\}$	Infeasible