Where are the constants? New Insights On The Role of Round Constant Addition in The SymSum Distinguisher

Sahiba Suryawanshi, Dhiman Saha

de.ci.phe.red Lab
Department of Electical Engineering and Computer Science
Indian Institute of Technology Bhilai, India
(sahibas,dhiman)@iitbhilai.ac.in

Abstract. The current work makes a systematic attempt to describe the effect of *the relative order of round constant (*RCon*) addition in the round function of an SPN cipher* on its algebraic structure. The observations are applied to the SymSum distinguisher, introduced by Saha *et al.* in FSE 2017 which is one of the best distinguishers on the SHA3 hash function reported in literature. Results show that certain ordering (referred to as Type-LCN) of RCon makes the distinguisher less effective but it still works with some limitations. Results in the form of new SymSum distinguishers are reported on concrete Type-LCN constructions - NIST LWC competition finalist Xoodyak-Hash and its internal permutation Xoodoo. New linear structures are also reported on Xoodoo that augment the distinguisher to penetrate more rounds. Final results include SymSum distinguishers on 7 rounds of Xoodoo and 5 rounds of Xoodyak-Hash with complexity 2^{128} and 2^{32}, respectively. All practical distinguishers have been verified. The characterization encompassing the algebraic structure and effect of RCon provided by the current work improves the understanding of SymSum in general and constitutes one of the first such result on Xoodyak-Hash and Xoodoo.

Keywords: Higher Order Derivative · SPN cipher · SymSum Distinguisher · ZeroSum Distinguisher · Xoodoo · Xoodyak-Hash.

1 Introduction

Substitution-Permutation Networks (SPN) have emerged as one of the most popular cipher design strategies for modern Symmetric-key cryptography. Since Rijndael [14], which is an SPN design, was announced as the winner of the AES [3] competition, SPN based crypto primitives have gained a lot of attention. Security evaluation of symmetric-key crypto has widely benefited from public cryptanalysis, which forms the cornerstone of trust on such constructions since they are not *provably* secure as their asymmetric counterparts. Public competitions like eSTREAM [5], CAESAR [1] and the National Institute of Standards and Technology Lightweight Cryptography Competition (NIST-LWC) [2] have largely

contributed into the evolution of SPN strategy both from design and cryptanalysis perspectives. Among various cryptanalysis strategies employed, devising distinguishers targets the most fundamental requirement of a crypto primitive - *non-randomness*. A very popular technique to make such distinguishers is based on higher-order differential cryptanalysis [4] and relies on computing what is known as the ZeroSum. It is based on the higher-order derivatives principle, stating that the $(d+1)^{th}$ order derivative of a $d-$degree function leads to a zero function. This is evidenced by obtaining a zero XOR-Sum for 2^{d+1} computations of function on a $(d+1)-$dimensional subspace.

A very interesting demonstration of the ZeroSum idea was by Aumasson *et al.* on the internal permutation (Keccak-f) of the hash function Keccak [9] which went on to be the winner of the NIST SHA3 [8] competition. The idea constituted what is referred to as the inside-out technique that allows to devise the ZeroSum property from the middle round of the permutations and extending in either direction. This work spawned a rich body of results [10,11,15,17] including full-round ZeroSum distinguishers. However, the reliance on the inside-out strategy implied that the results were inapplicable on the Keccak/SHA3 hash function. In FSE 2017, Saha *et al.* came up with the idea of the SymSum distinguisher [22] which was more efficient than ZeroSum by a factor of 4 and constituted the most efficient distinguishing attack on SHA3 at that time. SymSum exploited the fact that RCon were added after the non-linear operation in the SHA3 round function. Augmenting this with symmetry preserving property of the round sub-operations, $(d-1)^{th}-$fold vectorial derivatives (Refer Definition 1) over symmetric input subspaces led to what the authors called as the *Symmetric-Sum* or SymSum. Suryawanshi *et al.* extended the SymSum distinguisher using linearization to reach higher number of rounds [23].

The current work uncovers new insights on effect of RCon addition on algebraic structure in the light of SymSum. This systematic attempt tries to formalize the SPN structure that leads to SymSum-like properties. In doing so, we classify SPN designs in three classes: Type-LNC,Type-LCN and Type-CLN based on the relative order of RCon addition with regards to substitution and permutation layers. Our research reveals that while Type-LNC is captured by results reported on SHA3 by Saha *et al.*, linearization used by Suryawanshi *et al.* actually maps to Type-CLN. However, analysis of a Type-LCN SPN construction is furnished for the first time in the current work. The findings of work are finally verified in the form of new SymSum distinguishers on a concrete Type-LCN SPN design and NIST-LWC finalist - Xoodyak-Hash [13] and its internal permutation Xoodoo.

Related Work Despite being a relatively new design by the same team who designed Keccak, both Xoodoo and Xoodyak have had a fair share of distinguishing attacks. In 2020, Liu *et al.* proposed a full-round ZeroSum distinguishing attack on Xoodoo [19]. Since then, other researchers have introduced new distinguishing attacks on round-reduced Xoodoo, such as using rotational cryptanalysis reported by Liu *et al.* in [20], a functional distinguisher introduced by Bellini and Minematsu in [6], and a higher-order differential-linear distinguisher presented by Hu

et al. in [18]. Moreover, Dunkelman *et al.* introduced a distinguishing attack using differential-linear cryptanalysis on Xoodyak in [16].

Along with the theoretical analysis of the three types of SPN constructions state above, the current work makes an in-depth study of the round-function of Xoodoo to mount SymSum on both Xoodoo and Xoodyak-Hash. We report that the Xoodoo state is symmetric in multiple dimensions (Refer Definition 3) leading to distinguishers in two different axes. This is a stark difference with Keccak-f, where symmetry is only in the $z-$axis. We also report linear structures in the Xoodoo round-function that allow the SymSum property with lesser complexity. Overall, using Xoodoo and Xoodyak-Hash, we successfully verify our theoretical result on Type-LCN SPN primitives which states that for Type-LCN ordering of RCon, SymSum outperforms ZeroSum by a factor of 2. Final results constitute distinguishers on 5 rounds of Xoodyak-Hash and 7 rounds of Xoodoo with complexities of 2^{16} and 2^{128}, respectively. Table 1 summarizes our results.

Table 1: Summary of the results, here DoF is degree of freedom

	Xoodoo			Xoodyak-Hash		
#Rounds	ZeroSum	SymSum	Remark	ZeroSum	SymSum	Remark
1	2^1	2^0	Only 1 input required	2^3	2^0	Only 1 input required
2	2^1	2^0	Only 1 input required	2^5	2^4	SymSum
3	2^5	2^4	SymSum+ 1R Linearization	2^9	2^8	SymSum
4	2^9	2^8	SymSum+ 1R Linearization + Insideout	2^{17}	2^{16}	SymSum
5	2^{17}	2^{16}	SymSum+ 1R Linearization	2^{33}	2^{32}	SymSum
6	2^{33}	2^{32}	SymSum+ 1R Linearization + Insideout	2^{65}	-	Exceed DoF
7	2^{129}	2^{128}	SymSum	2^{129}	-	Exceed DoF
8	2^{257}	-	Exceed DoF	-	-	Exceed

Organization: Here is the structure of the paper: Section 2 provides an overview of the m-fold vectorial derivative. 3 explores the impact of reordering the RCon on the algebraic structure of SPN cipher. Finally, in 4, we apply our study practically to Xoodoo/Xoodyak-Hash and discuss the linearization technique for Xoodoo, including their complexity and DoF. 5 presents experimental evidence supporting our claims and in 6, we conclude the paper. The Appendix includes brief details of Xoodoo and Xoodyak-Hash.

2 Preliminaries

This work relies on the idea $m-$fold Boolean vectorial derivatives, which allow differentiation with respect to a specific subspace. While simple Boolean derivatives capture change in a function w.r.t a change in value of a single variable, vectorial derivatives capture simultaneous change in set of variables [21]. Higher order vectorial derivatives use multiple such disjoint partitions. Saha *et al.* used this operator in [22] and is restated below.

Definition 1 (m-Fold Vectorial Derivative [21,22]). *Let* $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}, \mathbf{x_{m+1}}\}$ *be* $(m+1)$ *partitions of Boolean variables* $(x_1, x_2, \cdots, x_n)$ *and* $f(\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}, \mathbf{x_{m+1}}) =$

$f(x_1, x_2, \cdots, x_n) = f(\mathbf{x})$ a Boolean function of n variables, then

$$\left.\frac{\partial^m f}{\partial \mathbf{x_m} \cdots \partial \mathbf{x_2} \partial \mathbf{x_1}}\right|_{\substack{(\mathbf{x_1},\mathbf{x_2},\cdots,\mathbf{x_m}) \\ =(\mathbf{c_1},\mathbf{c_2},\cdots,\mathbf{c_m})}} = \frac{\partial}{\partial \mathbf{x_m}} \left(\cdots \left(\frac{\partial}{\partial \mathbf{x_2}} \left(\left.\frac{\partial f}{\partial \mathbf{x_1}}\right|_{\mathbf{x_1}=\mathbf{c_1}} \right) \right|_{\mathbf{x_2}=\mathbf{c_2}} \right) \cdots \right) \Bigg|_{\mathbf{x_m}=\mathbf{c_m}}$$

is the $\mathbf{m}-$***fold vectorial derivative*** *of the Boolean function* $f(\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}, \mathbf{x_{m+1}})$ *with regards to the* m *partitions* $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}\}$.

$$\left.\frac{\partial^m f}{\partial \mathbf{x_m} \cdots \partial \mathbf{x_2} \partial \mathbf{x_1}}\right|_{\substack{(\mathbf{x_1},\mathbf{x_2},\cdots,\mathbf{x_m}) \\ =(\mathbf{c_1},\mathbf{c_2},\cdots,\mathbf{c_m})}} = \bigoplus_{\substack{\{\mathbf{x_1},\mathbf{x_2},\cdots,\mathbf{x_m}\}\in\mathbf{C} \\ \mathbf{x_{m+1}}=\mathbf{c_{m+1}}}} f(\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}, \mathbf{x_{m+1}}) \qquad (1)$$

$$\text{where, } \mathbf{C} = \begin{bmatrix} c_1, c_2, \cdots, c_{m-1}, c_m \\ c_1, c_2, \cdots, c_{m-1}, \overline{c_m} \\ c_1, c_2, \cdots, \overline{c_{m-1}}, c_m \\ c_1, c_2, \cdots, \overline{c_{m-1}}, \overline{c_m} \\ \vdots \ \vdots \ \ddots \ \vdots \ \vdots \\ \overline{c_1}, \overline{c_2}, \cdots, \overline{c_{m-1}}, \overline{c_m} \end{bmatrix}_{2^m \times m} \qquad \mathbf{c_i} \in \mathbb{F}_2^{|\mathbf{x_i}|}$$

3 Investigating Commutativity of Round-Constant Addition with the Linear and Non-Linear Operation

SPN is a round-based iterative function that in a generic form consists of combination of linear ($\mathcal{L}$) and non-linear operations ($\mathcal{N}$) along-with RCON addition ($\mathcal{C}$) which is aimed to reduce any symmetry which might eventually develop in the internal state. Though RCON addition is essentially a linear operation, we look at it in isolation for reasons that will be apparent soon. Our aim is to study the algebraic structure of SPN ciphers considering the position of RCON addition relative to the ordered pair $(\mathcal{L}, \mathcal{N})$ implying 3 possibilities: $(\mathcal{L}, \mathcal{N}, \texttt{RCON})$, $(\mathcal{L}, \texttt{RCON}, \mathcal{N})$ and $(\texttt{RCON}, \mathcal{L}, \mathcal{N})$. We respectively classify SPN ciphers into 3 categories: Type-LNC, Type-LCN and Type-CLN. Our investigation introduces the algebraic structure of these ciphers which is based on the nature of the monomials that appear in their Algebraic Normal Form (ANF) and classify[1] them into 3 types: Type-I monomials are free from any RCON, Type-II monomials involve both RCON and state variables and Type-III monomials consist only of constant terms. To illustrate Type-I, Type-II and Type-III monomials, we use following example.

Example 1. Let us consider an arbitrary Boolean function f with the ANF: $f = x_1x_2x_3x_4 + x_1x_3x_4c_2 + x_1x_4x_5 + x_2x_4c_1c_4 + c_1c_2c_3 + c_2c_4$, where c_i is a constant.

$$\begin{aligned} f &= x_1x_2x_3x_4 + x_1x_3x_4c_2 + x_1x_4x_5 + x_2x_4c_1c_4 + c_1c_2c_3 + c_2c_4 \\ &= \overbrace{(x_1x_2x_3x_4 + x_1x_4x_5)}^{f_{\text{Type-I}}} + \overbrace{(x_1x_3x_4c_2 + x_2x_4c_1c_4)}^{f_{\text{Type-II}}} + \overbrace{(c_1c_2c_3 + c_2c_4)}^{f_{\text{Type-III}}} \end{aligned}$$

[1] Note that this classification was introduced in [22].

Throughout the section, we use $X = \{x_1, x_2, \ldots x_n\}$ to denote the state variables for the initial state of the cipher while $C = \{c_1, c_2, \ldots c_m\}$ denote RCon added at various rounds. λ denotes algebraic degree of non-linear component $\mathcal{N}$. In the following subsections, we analyze the algebraic structure of Type-LNC, Type-LCN and Type-CLN SPN cipher.

3.1 Algebraic Structure of Type-LNC SPN Cipher

Type-LNC function is obtained by iterating $\mathcal{C} \circ \mathcal{N} \circ \mathcal{L}$ sequence. After 1 round, resulting polynomial takes form $\sum_k \prod_{x_i \in \mathbf{x}_k \subset X} x_i + \sum_{c_j \in \mathbf{c}_r \subset C} c_j$ where $|\mathbf{x}_k| \leq \lambda, \forall k$. Thus, algebraic degree (d°) of monomials is upper bounded by λ. This also implies that for a Type-LNC cipher, no Type-II monomials are generated after the first round. It is only after second round that Type-II monomials may be generated. Also after the second round (d°_{max}Type-I $- d^\circ_{max}$Type-II $\leq \lambda$). Thus difference in the highest degrees *always* persists even at higher rounds. In [22], Saha *et al.* utilized this property to develop SymSum distinguisher. The basic idea was to use $m-$fold vectorial derivatives to eliminate Type-II monomials thereby arriving at a RCon independent function. When derivatives were computed over specially selected symmetric subspaces, the output sum was deterministically symmetric i.e. SymSum. However, the analysis furnished in [22] was only limited to Type-LNC design SHA3. In the current work we give it a more generalized treatment and study SymSum property for other variants Type-LCN and Type-CLN.

3.2 Algebraic Structure of Type-LCN SPN Cipher

We can obtain the Type-LCN function by iterating the $\mathcal{N} \circ \mathcal{C} \circ \mathcal{L}$ sequence. The algebraic form of the resulting function after one iteration is given below.

$$\sum_k \prod_{x_i \in \mathbf{x}_k \subset X} x_i + \sum \prod_{\substack{x_m \in \mathbf{x}_m \subset X \\ c_l \in \mathbf{c}_l \subset C}} x_m c_l + \sum_r \prod_{c_j \in \mathbf{c}_r \subset C} c_j \tag{2}$$

Analyzing this polynomial easily reveals that the highest degrees of Type-I, Type-II and Type-III monomials are λ, $\lambda - 1$ and 0, respectively. Here we can see that the highest degree of RCon independent monomials is greater than RCon dependent monomials. Our work studies the case when RCon addition precedes non-linear (or both linear and non-linear) operations. We argue that SymSum remains more effective than ZeroSum distinguisher by a factor of 2, even after switching the operations. In addition, we offer a theoretical validation for our argument using similar approach as in [22], making it more comprehensible. To support our claim, we rely on Theorem 1 that builds upon the following Lemma.

Lemma 1 *Let $\mathcal{F}$ be a* SPN *round function with $\mathcal{N} \circ \mathcal{C} \circ \mathcal{L}$ components, where $\mathcal{C}$, $\mathcal{N}$ and $\mathcal{L}$ represent the non-linear, round-constant addition and linear components, respectively. Then, we can express the function $\mathcal{F}$ as:*

$$\mathcal{F} = \mathcal{G} + C \times \mathcal{H} + C,$$

where $d^\circ\mathcal{F} = d^\circ\mathcal{G} > d^\circ\mathcal{H}$, $\mathcal{G}, \mathcal{H} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *and* C *is a constant*

Proof. Function $\mathcal{F}^{n_r}$, which consists of n_r rounds, can be expressed as follows:

$$\begin{aligned}\mathcal{F} &= (\mathcal{N} \circ \mathcal{C}_{n_r} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{N} \circ \mathcal{C}_2 \circ \mathcal{L}) \circ (\mathcal{N} \circ \mathcal{C}_1 \circ \mathcal{L}) \\ &= \Big[\big(\mathcal{N} \circ \mathcal{C}_{n_r} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{N} \circ \mathcal{C}_2 \circ \mathcal{L})\big) \circ (\mathcal{N} \circ \mathcal{C}_1)\Big] \circ \mathcal{L} \end{aligned} \tag{3}$$

The monomials that contain RCon are unaffected[2] by the linear function $\mathcal{L}$ in the first round due to the order of operations illustrated in Equation (3). To distinguish monomials, we need to segregate them. The $\mathcal{F}^{n_r}$ function in n_r round SPN can be expressed as: $\mathcal{F}^{n_r} = \mathcal{F}^{n_r}_{\textsf{Type-I}} + \mathcal{F}^{n_r}_{\textsf{Type-II}} + \mathcal{F}^{n_r}_{\textsf{Type-III}}$

Let us examine degree of monomials $d^\circ\mathcal{F}^{n_r} = max(d^\circ\mathcal{F}^{n_r}_{\textsf{Type-I}}, d^\circ\mathcal{F}^{n_r}_{\textsf{Type-II}}, d^\circ\mathcal{F}^{n_r}_{\textsf{Type-III}})$ Now, let us pursue the inductive proof. Note that $d^\circ\mathcal{F}_{\textsf{Type-III}} = 0$ by definition.

Base case: For $n_r = 1$, the degrees of monomials are:

$$\begin{aligned} d^\circ\mathcal{F}_{\textsf{Type-I}} &\leq \lambda \text{ (degree of non-linear layer)} \\ d^\circ\mathcal{F}_{\textsf{Type-II}} &\leq \lambda - 1 \text{ (Due to } Exp \text{ (2))} \end{aligned}$$

Thus, highest degree $d^\circ\mathcal{F}_{\textsf{Type-I}} > d^\circ\mathcal{F}_{\textsf{Type-II}}$. Thus, statement holds for $n_r = 1$.

Inductive hypothesis: Assume that Lemma is true for $n_r = k$, then $d^\circ\mathcal{F}^k = d^\circ\mathcal{F}^k_{\textsf{Type-I}}$ (maximum degree of SPN function $\mathcal{F}$ is $k\lambda$) and $d^\circ\mathcal{F}^k_{\textsf{Type-I}} > d^\circ\mathcal{F}^k_{\textsf{Type-II}}$

Inductive step: Let $n_r = k + 1$ then $\mathcal{F}^{k+1} = \mathcal{N} \circ \mathcal{C}^{k+1} \circ \mathcal{L} \circ \mathcal{F}^k$

$$\begin{aligned} d^\circ\mathcal{F}^{k+1}_{\textsf{Type-I}} &= d^\circ(\mathcal{N} \circ \mathcal{C}^{k+1} \circ \mathcal{L}) + d^\circ\mathcal{F}^k_{\textsf{Type-I}} \\ &> d^\circ(\mathcal{N} \circ \mathcal{C}^{k+1} \circ \mathcal{L}) + d^\circ\mathcal{F}^k_{\textsf{Type-II}} \ (\because d^\circ\mathcal{F}^k_{\textsf{Type-I}} > d^\circ\mathcal{F}^k_{\textsf{Type-II}}) \\ &> d^\circ\mathcal{F}^{k+1}_{\textsf{Type-II}} \end{aligned}$$

Hence, by induction, the Lemma holds $\forall n_r \in \mathbb{N}$. □

As a result, we obtain $d^\circ\mathcal{F}^{n_r} = d^\circ\mathcal{F}^{n_r}_{\textsf{Type-I}} > d^\circ\mathcal{F}^{n_r}_{\textsf{Type-II}}$, which indicates that even after swapping the non-linear operation with RCon addition, the highest degree monomial of $\mathcal{F}$ is of Type-I. Obtaining an upper bound on the maximum degree of the Type-II and understanding how it relates to the highest degree of the Type-I from Lemma 1 establishes the following:

Theorem 1. *The upper-bound on the degree of Type-II monomials is given by the following expression:* $d^\circ\mathcal{F}^{n_r}_{\textsf{Type-II}} \leq d^\circ\mathcal{F}^{n_r} - 1$

Proof. The proof is very similar to the proof of Lemma 1.

Lemma 2 *The* $d^\circ\mathcal{F}-$*fold vectorial derivative of* $\mathcal{F}^{n_r}$ *is a function which is unaffected by the* RCon .

[2] In terms of the change in algebraic degree.

This follows logically from Theorem 1 that $d^{\circ}\mathcal{F}-$fold vectorial derivative of $\mathcal{F}^{n_r}$ will give function without Type-II or Type-III monomials. Lemma 2, thus, leads us toward obtaining a RCon independent function. Later in this work, we demonstrate practical application of this Lemma in form of new SymSum distinguishers on real-world Type-LCN primitives namely Xoodoo and Xoodyak-Hash.

3.3 Algebraic Structure of Type-CLN SPN Cipher

Type-CLN is generated by iteratively applying the $\mathcal{N} \circ \mathcal{L} \circ \mathcal{C}$ sequence. When we apply $\mathcal{N} \circ \mathcal{L} \circ \mathcal{C}$ once, we get a polynomial of the following form.

$$\sum \prod_{\substack{x_u \in \mathbf{x}_w \subset X \\ c_v \in \mathbf{c}_s \subset C}} (x_u + c_v) = \sum_k \prod_{x_i \in \mathbf{x}_k \subset X} x_i + \sum \prod_{\substack{x_m \in \mathbf{x}_m \subset X \\ c_l \in \mathbf{c}_l \subset C}} x_m c_l + \sum_r \prod_{c_j \in \mathbf{c}_r \subset C} c_j \quad (4)$$

It easy to see that Exp 4 is same as Exp 2. Thus $\mathcal{N} \circ \mathcal{L} \circ \mathcal{C} \equiv \mathcal{N} \circ \mathcal{C} \circ \mathcal{L}$ in terms of the algebraic structure implying that $\mathcal{L} \circ \mathcal{C} \equiv \mathcal{C} \circ \mathcal{L}$ or alternatively $\mathcal{C}$ and $\mathcal{L}$ satisfy the commutative property. As a result, similar to the Type-LCN scenario, we can deduce that for Type-CLN, the highest degrees of Type-I, Type-II and Type-III monomials are λ, $\lambda - 1$ and 0, respectively. Thus Type-CLN follows all the properties of Type-LCN.

4 Concrete Applications of Type-LNC Xoodoo/Xoodyak-Hash

This section will explore how the concepts discussed in the preceding sections can be applied practically, specifically focusing on Xoodoo/ Xoodyak (brief description of Xoodoo/Xoodyak is given in the Appendix A). To accomplish this, we will investigate the behaviour of Xoodoo/ Xoodyak under symmetric-inputs. We will also analyze the benefits of utilizing SymSum over ZeroSum distinguisher after deploying it on Xoodoo/Xoodyak-Hash.

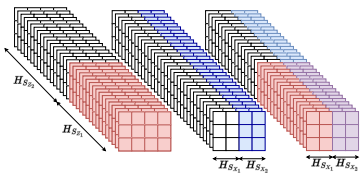
4.1 Multi-Dimensional-Symmetric State

Xoodoo is a permutation that operates on a 3-D array of 384 bits ($4 \times 3 \times 32$) and applies a round function to the input state for a specified number of rounds (n_r), denoted as $X^{n_r} = \texttt{Xoodoo}[384, n_r]$. S defines the internal state of X^{n_r}. In order to capture the notion of symmetry in the internal state of Xoodoo/Xoodyak-Hash we will use the following definitions.

Definition 2. *Symmetric-Half-State* (SHS)*: A state that can be split into two identical halves is SHS. A SHS in* Xoodoo *has a size of* 192 *bits and can be split in two directions: H_{S_z} along the z-axis with size* $4 \times 3 \times 16$ *and H_{S_x} along the x-axis with size* $2 \times 3 \times 32$.

Definition 3. *Multi-Dimensional-Symmetric State* (MDSS): *Each member of* $S^{\#}$ *is referred to as Multi-Dimensional-Symmetry if* $S^{\#}$ *is the set of all states in which both the conditions satisfy:*

$$H_{S_{x_1}} = H_{S_{x_2}} \text{ and } H_{S_{z_1}} = H_{S_{z_2}} \tag{5}$$



(a) Symmetry along z, x and both axes

8BED	8BED	EC14	EC14	8BED	8BED	EC14	EC14
5453	5453	D705	D705	5453	5453	D705	D705
51FF	51FF	25D6	25D6	51FF	51FF	25D6	25D6

(b) Self-Symmetric State with $H_{S_{z_1}}$ is shown in black and $H_{S_{z_2}}$ is shown in blue and $H_{S_{x_2}}$ is highlighted in yellow

Fig. 1: Exhibiting Self-Symmetric State

Fig. 1a depicts three symmetric states, each with unique symmetric characteristics. The leftmost state shows symmetry along the $z-$axis, where $H_{S_{z_1}}$ (red) is identical to $H_{S_{z_2}}$ (white). Similarly, state in the center has symmetry in $x-$axis in which first two sheets (white) are same as the other two (blue). The right-most state has symmetry in both x and z axes, with first 16 slices being identical to the last 16 slices and the first two columns in each slice being the same as the next two. This state is an example of MDSS in two directions. When symmetry is in one of the directions ($x-$axis or $z-$axis), it is clear from Definitions 2 that $|S^x| = |S^z| = 2^{192}$, where S^x and S^z have all states that have symmetry in the $x-$axis and $z-$axis, respectively, one of the examples is illustrated Table 2. When symmetry is in both the $x-$axis and $z-$axis, then $|S^{\#}| = 2^{96}$ (by Definition 3); one of the examples is given in Table 1b.

Table 2: Depicting the Self-Symmetric State in the $x-$axis with $H_{S_{x_1}}$ (black) and $H_{S_{x_2}}$ (blue) and the $z-$axis $H_{S_{z_1}}$ (black) and $H_{S_{z_2}}$ (highlighted in yellow).

Symmetry in z-axis				Symmetry in x-axis							
FFFA6482	DEEE4E3B	FFFA6482	DEEE4E3B	8BED	8BED	EC14	EC14	3B68	3B68	EF3F	EF3F
C2F49C55	F04F94D1	C2F49C55	F04F94D1	5453	5453	D705	D705	0C7F	0C7F	970A	970A
571AAB4A	335CD3F0	571AAB4A	335CD3F0	51FF	51FF	25D6	25D6	48A0	48A0	B154	B154

4.2 Distinguishing attack using Symmetric property in Xoodoo

This section explores the behaviour of symmetry in Xoodoo. When a symmetric state is fed into Xoodoo, the output after one round displays near-symmetry because the ι function introduces asymmetry in some bits. As a result, we get

an output symmetry of over 70% for up to two rounds. However, as the number of rounds increases, the symmetry diminishes from the third round. This property enables us to identify round-reduced Xoodoo. Therefore, with just one message, we can distinguish round-reduced Xoodoo up to two rounds.

Corollary 1. *The highest degree of a monomial including round-constant for n_r rounds of the* Xoodoo *permutation is $d^{\circ}\mathcal{K}^{n_r} - 1$.*

Proof. The Xoodoo design $\mathcal{F}$ expresses as $\mathcal{F} = \mathcal{L}_2 \circ \mathcal{N} \circ \mathcal{C} \circ \mathcal{L}_1$, with linear components $\mathcal{L}_1$ and $\mathcal{L}_2$, non-linear component $\mathcal{N}$ and constant addition component $\mathcal{C}$. $\mathcal{F}^{n_r}$, the Xoodoo-permutation in n_r rounds, unwraps as follows:

$$\begin{aligned}\mathcal{F} &= (\mathcal{L}_2 \circ \mathcal{N} \circ \mathcal{C}_{n_r} \circ \mathcal{L}_1) \circ \cdots \circ (\mathcal{L}_2 \circ \mathcal{N} \circ \mathcal{C}_2 \circ \mathcal{L}_1) \circ (\mathcal{L}_2 \circ \mathcal{N} \circ \mathcal{C}_1 \circ \mathcal{L}_1) \\ &= (\mathcal{L}_2 \circ \mathcal{N}) \circ \mathcal{C}_{n_r} \circ (\mathcal{L}_1 \circ \mathcal{L}_2) \circ \cdots \circ (\mathcal{L}_1 \circ \mathcal{L}_2) \circ \mathcal{N} \circ \mathcal{C}_2 \circ (\mathcal{L}_1 \circ \mathcal{L}_2) \circ \mathcal{N} \circ \mathcal{C}_1 \circ \mathcal{L}_1 \\ &= (\mathcal{N}^{'} \circ \mathcal{C}_{n_r} \circ \mathcal{L}^{'}) \circ (\mathcal{N} \circ \mathcal{C}_{n_r-1} \circ \mathcal{L}^{'}) \circ \cdots \circ (\mathcal{N} \circ \mathcal{C}_2 \circ \mathcal{L}^{'}) \circ (\mathcal{N} \circ \mathcal{C}_1 \circ \mathcal{L}_1)\end{aligned}$$

Here, the composition of two linear functions always is a linear function denoted by $\mathcal{L}^{'}$(where $\mathcal{L}^{'} = \mathcal{L}_1 \circ \mathcal{L}_2$). Also, a nonlinear function followed by a linear function is a nonlinear function $\mathcal{N}^{'}$(where $\mathcal{N}^{'} = \mathcal{L}_2 \circ \mathcal{N}$). Therefore, one round of the Xoodoo permutation can represent $\mathcal{N} \circ \mathcal{C} \circ \mathcal{L}$. Thus by Theorem 1, $d^{\circ}\mathcal{K}^{n_r} - 1$ is the highest degree of RCon dependent monomial for n_r rounds. □

Proposition 1 *While computing the d°*Xoodoo*$-$fold vectorial derivative of the* Xoodoo*'s self-symmetric input states, the symmetric property will be maintained.*

As RCon addition disturbs symmetry, by Proposition 1 the symmetry will be preserved while computing (d°Xoodoo)$-$fold vectorial derivative of Xoodoo using self-symmetric inputs. However, this property is not assured for $m < (d^{\circ}$Xoodoo$)$. The experimental result for the theoretical claim is given in Section 5.

Degree of freedom: There are 2^{384} ways to generate Xoodoo states, but it must meet at least one of the two conditions from Equations (5) to produce a symmetric state. Thus, the maximum number of symmetric states is 2^{192} due to 192 conditions. By Corollary 1, (d°Xoodoo)$-$fold vectorial derivatives are required for the symmetric state of Xoodoo. Therefore, this distinguisher can be used on round-reduced Xoodoo up to 7 rounds with complexity 2^{128}.

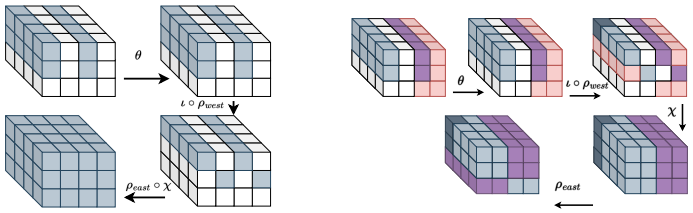
4.3 Extending the Distinguisher on Xoodoo using Linearization

This section formalizes the idea of linearizing Xoodoo for one round, inspired by the linear structure of Keccak, proposed by Guo *et al.* in [17]. Xoodoo's non-linear function χ acts on the column as $x_i \oplus (x_{i+1} \oplus 1)x_{i+2}$, where $i, i{+}1, i{+}2 \in \{0, 1, 2\}$. Thus to linearize χ, we must take at most one variable in a column. Moreover, to handle θ diffusion, we need to maintain state parity which can be achieved by following constraints.

$$A_{(0,0,*)} \oplus A_{(1,0,*)} = C_0 \tag{6}$$

$$A_{(0,2,*)} \oplus A_{(1,2,*)} = C_2 \tag{7}$$

Here, C_0 and C_1 are constants, lane $A_{(x,y,*)}$ is located at coordinate (x, y) and $*$ represents the entire lane of 32 bits. Equation (6) and (7) ensure a constant parity for the first and third sheet, respectively. Fig. 2a illustrates the idea of linearizing Xoodoo. The size of each cell in the figure is 1 byte for a clearer demonstration. The lanes with white cells are constant, while those with grey cells have an algebraic degree of 1, which meet the requirements of Equation (6) and (7) to deal with θ. Due to those conditions after θ, the number of elements



(a) Forward round linearization with 2^{64} DoF.

(b) One round linearization of symmetric state with 2^{32} DoF.

Fig. 2: Showing one round linearization of Xoodoo which is represented in bytes rather than in bits for brevity.

with algebraic degree 1 remain the same. ι function does not affect degrees. However, for the subsequent operation, ρ_{west}, the bit positions change, aiding in maintaining the input to χ. As we can see in the Fig. 2a, the input to χ should have only linear terms in each column to maintain linearity of the state after χ. ρ_{east} does not impact the degree, only alters the positions.

Degree of freedom: To linearize the state, we can take at most one bit in each column to handle χ. Thus total variable we can take is 128. These variables should satisfy 64 constraints to handle θ. As a result, we can have $2^{128-64} = 2^{64}$ such states that maintain the linearity for 1 round. Therefore, the degrees of freedom of such states is 2^{64}.

Linearization in Symmetric States: Both x and z axes can provide symmetry to the Xoodoo input. To linearize the symmetric input state, we need to apply the following equations.

$$A_{(x,*,*)} \oplus A_{(x+2,*,*)} = 0 \text{ where } x \in \{0, 1\} \tag{8}$$

$$A_{(*,*,z)} \oplus A_{(*,*,z+16)} = 0 \text{ where } z \in \{0, 1, \ldots, 15\} \tag{9}$$

$$A_{(0,0,*)} \oplus A_{(1,0,*)} = C_0 \tag{10}$$

Here, $*$ in above equations represents all; more specifically, $*$ represents $x \in \{0, 1, 2, 3\}; y \in \{0, 1, 2\}$; and $z \in \{0, 1, \ldots, 31\}$ on the x, y, and z axis, respectively. Here either Equation (8) or Equation (9) is used to provide symmetry in the direction of x or z axis and Equation (10) handles θ for linearization.

Fig. 2b depicts an overview of one round of linearization for symmetric input to Xoodoo, with the symmetry shown along $x-$axis. Here, purple and grey bytes are equal. Similarly, white and pink bytes are equal. White and pink cells are constants, and grey and purple cells have an algebraic degree 1. The initial state satisfies conditions stated in Equation (8) and (10), ensuring that the plane's parity will be constant. As a result, the symmetry of the state is preserved after θ. The symmetry is destroyed due to ι at lane $(0,0)$ and the asymmetry induced due to ι is further propagated by χ as depicted in dark gray. Consequently, there is only one variable in each column. Thus, the highest degree of the state remains 1 after one round.

Lemma 3 *Lemma 1 holds under linearization.*

If the SPN round function $\mathcal{F}$ were to be linearized for l_r rounds, revised $\mathcal{F}^{'}$ could be represented as:

$$\begin{aligned}\mathcal{F}^{'} &= (\mathcal{N} \circ \mathcal{C}_{n_r} \circ \mathcal{L}) \circ (\mathcal{N} \circ \mathcal{C}_{n_r-1} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{N} \circ \mathcal{C}_{n_r-l_r} \circ \mathcal{L}) \\ &\quad \circ (\mathcal{L}^{'} \circ \mathcal{C}_{l_r} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{L}^{'} \circ \mathcal{C}_1 \circ \mathcal{L}) \\ &= \Big[(\mathcal{N} \circ \mathcal{C}_{n_r} \circ \mathcal{L}) \circ (\mathcal{N} \circ \mathcal{C}_{n_r-1} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{N} \circ \mathcal{C}_{n_r-l_r} \circ \mathcal{L}) \\ &\quad \circ (\mathcal{L}^{'} \circ \mathcal{C}_{l_r} \circ \mathcal{L}) \circ \cdots \circ (\mathcal{L}^{'} \circ \mathcal{C}_1)\Big] \circ \mathcal{L} \end{aligned} \tag{11}$$

Here $\mathcal{L}^{'}$ is a linearized version of $\mathcal{N}$. The lemma mentioned above can be trivially proved, by observing Eq (11)

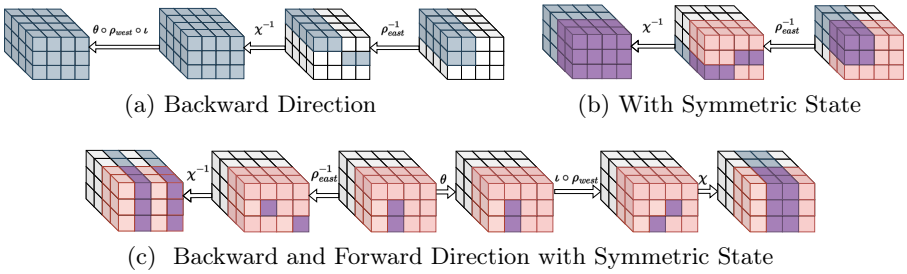
Theorem 2. *For an iterated SPN round function $\mathcal{F}$, the relationship between the upper bound on degree of Type-I and Type-II monomials will remain same after linearization such that: $d^{\circ}\mathcal{F}^{n_r}_{\text{Type-II}} \leq d^{\circ}\mathcal{F}^{n_r}_{\text{Type-I}} - 1$*

Lemma 1, Lemma 3, and Theorem 2 can be used to easily prove this Theorem.

Corollary 2. *With l_r linearized rounds $d^{\circ}\mathcal{F}-$fold vectorial derivative of $\mathcal{F}$ is a function which is independent of round constants.*

The symmetry will be retained by Corollary 2 when computing the d°Xoodoo$-$fold vectorial derivative of linearized Xoodoo with self-symmetric inputs. Section 5 provides the experimental outcome for theoretical claim.

Degree of freedom: There are 2^{384} Xoodoo states, that can be generated, out of which 2^{192} symmetric states are possible. Nevertheless, due to the 32 conditions given in Equation (10) that must be fulfilled to achieve 1-round linearization, it drops to $2^{192-32} = 2^{160}$. Thus, 2^{160} is the DoF for these states. Since there are four lanes of variables, each with a size of 32, we have 128 variables while fixing the constants. Due to symmetry, half of the variables should be same as the others because of Equation (8) and (9) (one of them). However, for 1 round of linearization of Xoodoo, 32 conditions are required. As a result, we can have $2^{128-64-32} = 2^{32}$ states that maintain 1-round linearization while the input to Xoodoo is symmetric. Therefore, DoF of such states is 2^{32}.



(a) Backward Direction (b) With Symmetric State

(c) Backward and Forward Direction with Symmetric State

Fig. 3: Linearization of Xoodoo

Linearization in Backward Direction The inside-out technique is widely recognized in the most well-known classical distinguishing attack ZeroSum to attack any permutation since it can start from the middle and move in both directions. We have also explored linearization in the backward direction, as shown in Fig. 3a. Instead of four lanes, we can use a single plane, as shown in the figure. Furthermore, this linearization method can be applied to symmetric states, as illustrated in Fig.3b.

Extending SymSum Distinguisher in Xoodoo Using the linearization technique symmetry property described in the preceding section can be extended to 1 additional round. We explain our approach by assuming symmetry along the $x-$axis. However, other directions can also be applied. To linearize 1-round along with fulfilling the essential condition for self-symmetry of Xoodoo, the input set must satisfy the following criteria (for details, see section 4.3):

1. To maintain the input symmetry state should satisfy: $H_{S_{x_1}} = H_{S_{x_2}}$.
2. The constraint for linearization is $A_{(2,1,*)} \oplus A_{(1,1,*)} = C$ where $*$ define the whole lane, and C is a 32 bit constant.

Given the circumstances mentioned above, this state has a dimension of 32. As a result, this strategy can be used up to 6-round with a complexity of 2^{32}. Furthermore, there are 160 constant bits, each with a value of either 0 or 1. So, we can construct 2^{160} of such sets using various fixed values. Using linearization and inside-out technique, we have 16 degree of freedom. As a result, we can attack up to 10 ($4-$round backward + $1-$round backward linearization + $1-$round forward linearization + $4-$round forward) rounds with complexity 2^{16}. To simultaneously visualize the linearization in a backward and forward direction, refer to Fig. 3c.

4.4 Adapting the Distinguisher on Xoodyak-Hash

Xoodyak combines the Cyclist mode of operation with the Xoodoo permutation, which is responsible for preparing the data before inputting it into Xoodoo and

Table 3: Input to XOODYAK-HASHand intermediate state after cyclist operation

Input to Cyclist				input to Xoodoo			
0E2E0E2E	0AAE0AAE	0C440C44	018001	0E2E0E2E	0AAE0AAE	0C440C44	01800180
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000080

computing output according to required operation. For example, when calculating the digest in hash and keyed hash modes, domain separator value is set to 0x01 and 0x03, respectively. Similarly, in other modes, the domain separator values differ. However, this study focuses on XOODYAK-HASH mode, which absorbs at most 16 bytes at a time, and 0x01 is added as a padding bit to indicate the end of input. Thus, to ensure input state is symmetric, we need to input 15 bytes with a fixed value of 0x01. However, since we cannot control the capacity bytes and the Cyclist mode adds domain separator in the capacity portion of the state, our state *cannot* become fully symmetric and there will always be an asymmetric byte in the state. Table 3 shows the input state before and after the Cyclist mode. XOODYAK-HASH exhibits near-symmetry in its output when input has a near-symmetric state, but the symmetry is lost as the number of rounds increase. This property allows us to distinguish round-reduced XOODYAK-HASH with just one input message. The operations that disturb symmetry are RCON addition and domain separator. However, by corollary 1, d°XOODOO$-$vectorial derivative of XOODOO is independent of round-constant addition, allowing us to observe near-symmetry in the (d°XOODOO)$-$fold vectorial derivative of XOODYAK-HASH when using near-symmetric input states. Experimental results conforming to the theoretical justifications are presented in Section 5.

Degree of freedom: There are 2^{128} ways to generate XOODYAK-HASH states, but only those satisfying at least one condition from Equation (3) result in a symmetric state. There are a maximum of 2^{64} symmetric states due to 64 conditions for symmetry generation. However, one additional condition is needed to handle the padding byte, limiting the number of possible states to 2^{56}. Therefore, DoF for XOODYAK-HASH symmetric states is 2^{56}, and corollary 1 requires (d°XOODOO) vectorial derivatives. This distinguisher can be used on round-reduced XOODOO up to 5 rounds with a complexity of 2^{32}.

5 Experimental Verification

This section provides experimental proof for supporting the previous claims by demonstrating 1-round linearization of 4-rounds XOODOO. The degree for 4-rounds is reduced to $2^{4-1} = 8$ due to 1-round linearization, and in line with Corollary 2, vectorial derivative of the 8^{th} order will possess the SYMSUM property. Fig. 1 shows input state for XOODOO.

Table 4 shows the input state subspace of XOODYAK-HASH. Here the subspace is generated by setting all possible values to * while maintaining the symmetry in the $z-$axis. The text in black is dependent on the text in blue, which means that the first 16 slices are identical to the next 16 slices. This can be visualized

Table 4: Representing Xoodoo/Xoodyak-Hash state and output Sum

Xoodyak-Hash State															
Input State								Output Sum							
∗E35	∗E35	∗041	∗041	∗9B6	∗9B6	∗B80	∗B80	B68E	B68E	B68E	B68E	B68E	B68E	B68E	B68E
0000	0000	0000	0000	0000	0000	0000	0000	8C51	8C51	8C51	8C51	8C51	8C51	8C51	8C51
0000	0000	0000	0000	0000	0000	0000	0080	CA4F	CA4F	CA4F	CA4F	CA4F	CA4F	CA4F	CA4F
Xoodoo State															
Input State								Output Sum							
∗∗∗∗∗∗∗∗	7E5B9440	∗∗∗∗∗∗∗∗	7E5B9440					B9D83814	96F1AF94	B9D83814	96F1AF94				
††††††††	24A2A799	††††††††	24A2A799					A33C141F	AB79F93C	A33C141F	AB79F93C				
14DA894E	4642ACED	14DA894E	4642ACED					B7ECCBF7	5A5C712F	B7ECCBF7	5A5C712F				

by left most figure in Fig. 1a. While for the linearized Xoodoo state, the subspace is generated by setting all possible values to $*$ and † to maintain the parity of columns while maintaining the symmetry in the $x-$axis, which means that the first two sheets are identical to the next two sheets. In Table 4, text in blue is identical to blacktext, which can be visualized by the middle figure in Fig. 1a.

6 Conclusion

The current work thoroughly investigated the algebraic structure of SPN cipher with varying RCon ordering relative to the linear and non-linear layers of an SPN round-function. We also showed how our findings can be used in practice by mounting the SymSum distinguisher on Xoodoo/Xoodyak-Hash while achieving one of the most efficient distinguishers reported on Xoodyak-Hash so far - 5 rounds with 2^{32}. We demonstrated how symmetry propagation in Xoodoo/Xoodyak-Hash allows us to identify attacks with only one symmetric state for up to two rounds. Furthermore, regardless of the degree of nonlinear operation, we provided theoretical proof that the SymSum distinguisher outperforms the ZeroSum distinguisher by a factor of two. Finally, we applied linearization on Xoodoo, which resulted in a $10-$round SymSum distinguisher with a complexity of 2^{16} leveraging the inside-out technique. Our research expands the understanding of SPN ciphers with regards to the relation of their algebraic structure and the influence of RCon on the highest degree monomials which we believe provides valuable insights for designing future cryptographic algorithms.

Acknowledgment

The first author receives financial assistance through the TCS Research Scholarship Programme (TCS RSP) thanks to the support of Tata Consultancy Services (TCS). We also thank Darunjeet Bag for his support with this research.

References

1. Caesar: Competition for authenticated encryption: Security, applicability, and robustness. http://competitions.cr.yp.to/caesar.html
2. NIST LWC: National institute of standards and technology lightweight cryptographic https://csrc.nist.gov/Projects/lightweight-cryptography/finalists

3. Advanced encryption standard (AES). Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti (2001), //https://doi.org/10.6028/NIST.FIPS.197
4. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced keccak-f and for the core functions of luffa and hamsi. rump session of Cryptographic Hardware and Embedded Systems-CHES **2009**, 67 (2009)
5. Babbage, S., Canniere, C., Canteaut, A., Cid, C., Gilbert, H., Johansson, T., Parker, M., Preneel, B., Rijmen, V., Robshaw, M.: The estream portfolio. Citeseer (2008), https://www.ecrypt.eu.org/stream/
6. Bellini, E., Makarim, R.H.: Functional cryptanalysis: Application to reduced-round xoodoo. IACR Cryptol. ePrint Arch. p. 134 (2022)
7. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F., Todo, Y., Viguier, B.: Gimli : A cross-platform permutation. In: CHES. Lecture Notes in Computer Science, vol. 10529, pp. 299–320. Springer (2017)
8. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak SHA-3 submission. Submission to NIST (Round 3) (2011), http://keccak.noekeon.org/Keccak-submission-3.pdf
9. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 313–314. Springer (2013)
10. Boura, C., Canteaut, A.: A zero-sum property for the keccak-f permutation with 18 rounds. In: ISIT. pp. 2488–2492. IEEE (2010)
11. Boura, C., Canteaut, A., Cannière, C.D.: Higher-order differential properties of keccak and *Luffa*. In: FSE. Lecture Notes in Computer Science, vol. 6733, pp. 252–269. Springer (2011)
12. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of xoodoo and xoofff. IACR Trans. Symmetric Cryptol. **2018**(4), 1–38 (2018)
13. Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: Xoodyak, a lightweight cryptographic scheme. IACR Trans. Symmetric Cryptol. **2020**(S1), 60–87 (2020)
14. Daemen, J., Rijmen, V.: The block cipher rijndael. In: Quisquater, J., Schneier, B. (eds.) Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1820, pp. 277–284. Springer (1998). https://doi.org/10.1007/10721064_26, https://doi.org/10.1007/10721064_26
15. Duan, M., Lai, X.: Improved zero-sum distinguisher for full round keccak-f permutation. IACR Cryptol. ePrint Arch. p. 23 (2011)
16. Dunkelman, O., Weizman, A.: Differential-linear cryptanalysis on xoodyak. In: NIST Lightweight Cryptography Workshop (2022)
17. Guo, J., Liu, M., Song, L.: Linear structures: Applications to cryptanalysis of round-reduced keccak. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10031, pp. 249–274 (2016)
18. Hu, K., Peyrin, T.: Revisiting higher-order differential(-linear) attacks from an algebraic perspective - applications to ascon, grain v1, xoodoo, and chacha. IACR Cryptol. ePrint Arch. p. 1335 (2022)
19. Liu, F., Isobe, T., Meier, W., Yang, Z.: Algebraic attacks on round-reduced keccak/xoodoo. IACR Cryptol. ePrint Arch. p. 346 (2020)
20. Liu, Y., Sun, S., Li, C.: Rotational cryptanalysis from a differential-linear perspective - practical distinguishers for round-reduced friet, xoodoo, and alzette. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 741–770. Springer (2021)

21. Posthoff, C., Steinbach, B.: Logic functions and equations. Binary models for computer science. Springer Publishing Company, Incorporated. (2004)

22. Saha, D., Kuila, S., Chowdhury, D.R.: Symsum: Symmetric-sum distinguishers against round reduced SHA3. IACR Trans. Symmetric Cryptol. pp. 240–258 (2017)

23. Suryawanshi, S., Saha, D., Sachan, S.: New results on the symsum distinguisher on round-reduced SHA3. In: AFRICACRYPT. Lecture Notes in Computer Science, vol. 12174, pp. 132–151. Springer (2020)

A Xoodoo Permutation [12]

Daemen *et al.* presented Xoodoo, a 48-byte cryptographic permutation at ToSC 2018, inspired by Keccak [9] and Gimli [7]. It operates on a $3D$ array of size $4 \times 3 \times 32$, where row, column, and lane refer to $1D$ arrays in the x, y, and z directions respectively. Slices, sheets, and planes illustrate the $2D$ arrays in (x, y), (y, z), and (x, z) planes. These arrays are 12−bits, 96−bits, and 128−bits in size. Fig. 4 depicts all the terms described above.

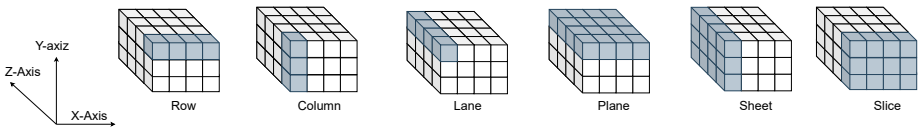


Fig. 4: The state displayed is of size 3×4×4 bytes, achieved by combining 8 cells into 1 for brevity, resulting in each cell of the state being of size 1 byte.

The Xoodoo permutation is a sequence of iterations on a $3D$ Xoodoo state using five different mappings: $X = \rho_{east} \circ \chi \circ \iota \circ \rho_{west} \circ \theta$. The θ mapping is responsible for diffusing the state linearly, while ρ_{west} and ρ_{east} rotate the bits of the planes in the x and z directions by specific values for each plane. Depending on the round number, the ι mapping adds a unique RCon to the first lane of plane A_0. The only non-linear function that operates on the plane is χ.

All the sub-functions of the Xoodoo round-function are listed below.

$$\theta : \begin{cases} A_y & = A_y \oplus E \text{ where } y \in \{0,1,2\} \\ E & = P \lll (1,5) \oplus P \lll (1,14) \\ P & = A_0 \oplus A_1 \oplus A_2 \end{cases}$$

$$\rho_{west} : \begin{cases} A_1 & = A_1 \lll (1,0) \\ A_2 & = A_2 \lll (0,11) \end{cases}$$

$$\iota : \begin{cases} A_0 = A_0 \oplus RC_i \end{cases}$$

$$\chi : \begin{cases} A_y & = A_y \oplus B_y \text{ where } y \in \{0,1,2\} \\ B_y & = \sim A_{y+1 \bmod 3} \cdot A_{y+2 \bmod 3} \end{cases}$$

$$\rho_{east} : \begin{cases} A_1 & = A_1 \lll (0,1) \\ A_2 & = A_2 \lll (2,8) \end{cases}$$

B Xoodyak-Hash [13]

Xoodyak-Hash, one of the ten NIST-LWC finalists, is a versatile cryptographic primitive combining sponge structure and Xoodoo permutation based on an operational mode termed Cyclist. The number of rounds in Xoodyak-Hash is 12, which provides the designed primitive with a sufficient safety margin against all potential attacks. Both hash and keyed modes are available in Xoodyak-Hash. The block sizes for the hash, the input, and the output in keyed modes are set, respectively, by the R_{hash}, R_{kin}, and R_{kout} block sizes to the mode of operation Cyclist, which depends on cryptographic permutation.

The hash mode consumes input strings and squeezes digests. Depending on the input string length, the absorbing function is called more than once because it can only absorb up to 16 bytes at once, and depending on the data absorbed so far, Squeeze(l) outputs an l-byte, where $l = 128$ bits. The Xoodyak-Hash offers 128-bit security and, as a result, it generates 256-bits (32 bytes) of the digest by performing considerable squeeze operations, as seen in Fig. 5

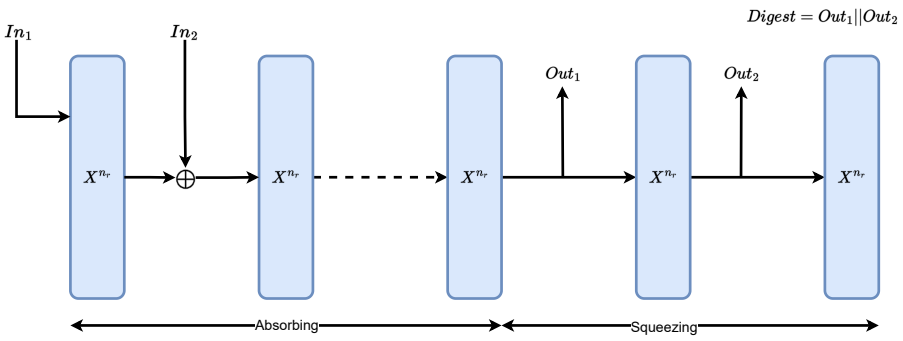


Fig. 5: This construction absorbs a variable input size and produces a 32-byte digest.