EasyChair Preprint

№ 14972

Complexity of Monomial Prediction in Cryptography and Machine Learning

Pranjal Dutta, Mahesh Rajasree and Santanu Sarkar

September 21, 2024

Complexity of Monomial Prediction in Cryptography and Machine Learning*

Pranjal Dutta
School of Computing
National University of Singapore (NUS)
Singapore
duttpranjal@gmail.com

Mahesh Sreekumar Rajasree
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
Delhi, India
srmahesh1994@gmail.com

Santanu Sarkar
Department of Mathematics
Indian Institute of Technology Madras
Tamil Nadu, India
santanu@iitm.ac.in

***Abstract*—In this paper, we focus on the *monomial prediction* problem in two settings: (1) Decide whether a particular monomial m is present in a composite function $f := f_r \circ f_{r-1} \circ \dots f_0$, where f_i are quadratic boolean functions, (2) Decide whether a particular monomial m is present in a composite function $f := f_r \circ f_{r-1} \circ \dots f_0$, where polynomials f_i are efficiently computable by *Probabilistic Generating circuits* over rationals. Probabilistic generating circuits (PGCs) are economical representations of multivariate probability generating polynomials (PGPs), which capture many tractable probabilistic models in machine learning.**

The first problem has a strong connection with the security of symmetric-key primitives. Dinur and Shamir proposed the *cube attack* for distinguishing a cryptographic primitive from a random function, which can be thought of as an efficient monomial prediction. In a general setting, over any large finite field or integers, monomial prediction is known to be NP-hard. Here, we show that in the quadratic setting, the problem is $\oplus$P-complete. $\oplus$P is an interesting complexity class that is not known to contain NP, however, it is believed to contain computationally hard problems. On the other hand, we also present several new *zero-sum distinguishers* for 5-round Ascon, which is one of the ten finalists for NIST light weight cryptography standardization competition.

We show that the second problem is #P-complete. It is known that PGCs have efficient inference, i.e. given a monomial, one can efficiently output (which signifies the probability) its coefficient in the polynomial computed by the circuit. However, a composition of such functions makes the inference *hard*. Composition of probabilistic models and their efficient inference play a crucial role in the semantic contextualization and framework of uncertainty theories in graphical modelling.

***Index Terms*—boolean function, monomial prediction, probabilistic generating circuits, $\oplus$P-complete, #P-complete, cube testers, Ascon, zero-sum distinguisher**

I. Introduction

Dinur and Shamir [1] proposed the *cube attack* against symmetric-key primitives with a secret key and a public input. It has since evolved into a universal tool for assessing the security of cryptographic primitives, and it has been successfully applied to a variety of symmetric primitives. Roughly speaking, the output bit of a cipher can be seen as an *unknown* Boolean polynomial $f(\boldsymbol{x}, \boldsymbol{v})$, over $\mathbb{F}_2$, where $\boldsymbol{x} = (x_0, \dots, x_{n-1})$ is a vector of secret input variables, and $\boldsymbol{v} = (v_0, \dots, v_{m-1})$ is a vector of public input variables, and $\mathbb{F}_2 = \{0, 1\}$ is the field containing 2 elements. Given a Boolean function $f(\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{F}_2[\boldsymbol{x}, \boldsymbol{v}]$ and a monomial $t \in \mathbb{F}_2[\boldsymbol{v}]$, one can express $f(\boldsymbol{x}, \boldsymbol{v})$ as

$$f(\boldsymbol{x}, \boldsymbol{v}) \;=\; t \cdot p_t(\boldsymbol{x}, \boldsymbol{v}) \;+\; q_t(\boldsymbol{x}, \boldsymbol{v}) \;,$$

such that none of the monomials in $q_t(\boldsymbol{x}, \boldsymbol{v})$ is divisible by t. The function p_t is called the *superpoly* of t in f. Let $I = (i_1, i_2, \dots, i_k)$ be the index subset such that $t = \prod_{i \in I} v_i$. Then, it can be easily verified that for any constants $c_j, \forall j \notin I$,

$$\sum_{\substack{(v_{i_1}, v_{i_2}, \dots, v_{i_k}) \in \mathbb{F}_2^k \\ v_j = c_j, \forall j \notin I}} f(\boldsymbol{x}, \boldsymbol{v}) = p_t(\boldsymbol{x}, \boldsymbol{c}) \;.$$

A *cube tester* basically computes the above summation (called *cube sum*) for a carefully chosen monomial t such that its superpoly p_t is equal to constant zero. This serves as a distinguishing attack between f and a random polynomial.

To broaden the integral and higher-order differential distinguishers, Todo [2] introduced the *division property*. Soon division property based cube attack became a hot topic in the community. In [3], a new technique termed *monomial prediction* was proposed that captures the algebraic basics of various attempts to improve the detection of division property. The goal was simple: detect a monomial $\boldsymbol{x}^{\boldsymbol{u}_1}$ in the product $\boldsymbol{y}^{\boldsymbol{u}_2}$ of any output bits of a vectorial Boolean function $\boldsymbol{y} = f(\boldsymbol{x})$. Further, this monomial prediction approach was shown to be equivalent to the proposed three-subset bit-based division property without unknown subset [4].

Monomial prediction: What and why? From an algorithmic perspective, let us ask the following (decision) question:

Some of the results were presented at the Workshop on Coding and Cryptography (WCC 2022), but it was not published in any proceedings. This is an extended version along with the new results in the Machine Learning setting.

Given a monomial $\mathbf{m}$, and a given (black-box) polynomial f, over a certain domain R, can we efficiently decide whether the coefficient of $\mathbf{m}$ in f is zero or not?

In [5], Kayal considered this computational problem, known as 'monomial prediction', from a complexity theoretic (hardness) point of view, which can be broadly (re)stated as follows: Given a black-box access to an n-variate degree-d polynomial $f(\boldsymbol{x})$, over a finite field $\mathbb{F}$ and a monomial $\boldsymbol{x}^{\boldsymbol{e}} = x_1^{e_1} \cdots x_n^{e_n}$, determine the coefficient of $\boldsymbol{x}^{\boldsymbol{e}}$ in $f(\boldsymbol{x})$. Before Kayal, a similar search problem was also studied by Malod [6] in his PhD thesis, from an algebraic complexity theoretic lens.

Kayal termed this problem as CoeffSLP and showed that it is $\#\mathbf{P}$-complete over integers (for a self-contained proof, see [5, Appendix A]). We recall that $\#\mathbf{P}$ essentially captures the number of solutions of a given instance, and thus obviously a $\#\mathbf{P}$ problem must be *at least as hard* as the corresponding $\mathbf{NP}$ problem. However, roughly speaking, for practical purposes, f is a composition of *easy* functions.

(1) In stream ciphers f can be thought as a *composition* of a bunch of *linear* and *quadratic* Boolean functions, and not *arbitrary* compositions,

(2) In machine learning, many probabilistic models are compositions of *tractable probabilistic model* where *probabilistic inference* (i.e. coefficient-extraction) is easy.

Motivated thus, one could ask the following:

Given a monomial $\mathbf{m}$, and a given (black-box) polynomial f, over a certain domain R, as a composition of "easy" functions g_i, i.e. $f = g_r \circ g_{r-1} \circ \dots g_0$, can we efficiently decide whether the coefficient of $\mathbf{m}$ in f is zero or not?

Since the inherent assumption that g_i as individual functions are easy functions, it is *unclear* if [5] would still imply that the monomial prediction is hard (say over $\mathbb{Z}$). This makes the monomial prediction paradigm, given as compositions, both theoretically and practically interesting! So, we restrict ourselves to two particular cases and ask the complexity of the following problems. The first problem is motivated as the theoretical understanding of stream ciphers, viewed as a composition of quadratic polynomials over $\mathbb{F}_2$. Trivially, quadratic polynomials are easy polynomials.

Problem 1. *Given a composition $f := (f_1, \dots, f_n) := g_r \circ g_{r-1} \circ \dots g_0$, and a monomial m, where each $g_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, is a quadratic function, decide whether m is a monomial in f_1.*

The second problem is motivated from the machine learning perspective. To define the setup, we first define *Probabilistic Generating Circuits* (PGCs). In [7], Zhang et al. proposed this model. These models represent probability distributions by probability generating functions. Probability generating functions represent the joint distribution of a set of binary random variables as the coefficients of a multi-linear polynomial.

Definition 1. *Let* Pr *be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as*

$$g(x_1, \dots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \mathsf{Pr}[X^S] \cdot x^S$$

where $\mathsf{Pr}[X^S] = \mathsf{Pr}[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

This new model is efficiently tractable as it supports an extremely efficient marginal inference. Furthermore, it is very efficient, it subsumes all the probabilistic models, e.g. decomposable probabilistic circuits (PCs), determinantal point processes (DPPs).

Definition 2 (PGC)**.** *A probabilistic generating circuit (PGC) is a directed acyclic graph consisting of three types of nodes:*

1) *Sum nodes $+$ with weighted edges to children;*
2) *Product nodes $\times$ with unweighted edges to children;*
3) *Leaf nodes, which are variables x_i or constants.*

A PGC has one node of out-degree 0 (edges are directed from children to parents), and we refer to it as the root of the PGC. The size of a PGC is the number of edges in it.

Each node in a PGC represents a polynomial, (i) each leaf in a PGC represents the polynomial x_i or a constant, (ii) each sum node represents the weighted sum over the polynomials represented by its children, and (iii) each product node represents the unweighted product over the polynomials represented by its children. The polynomial represented by a PGC is the polynomial represented by its root. We know that inference is extremely efficient for PGCs [8].

Since, we have defined what a PGC is, now we are ready to state the second problem.

Problem 2. *Given a composition $f := (f_1, \dots, f_n) := g_r \circ g_{r-1} \circ \dots g_0$, and a monomial m, where each $g_i : \mathbb{Q}^n \longrightarrow \mathbb{Q}^n$, is computed by a $\mathsf{poly}(n)$-size PGCs, output the coefficient of m in f_1.*

The composition of functions introduces complexity into probabilistic inference and makes it perhaps challenging. In the realm of graphical modelling, combining probabilistic models and efficiently inferring their outcomes is fundamental for semantic contextualization and establishing robust frameworks for uncertainty theories! This synergy lays the groundwork for understanding many complex systems with nuanced uncertainties, enriching our ability to model real-world phenomena accurately.

A. Our contributions

Our paper revolves around the above problems, and contributes in two directions – theoretical and practical.

Theoretical contribution. Though it may sound obvious to be 'theoretically' hard, the hardness proof is far from the obvious. We consider more general parameters and indeed show that Problem 1 is $\oplus\mathbf{P}$-complete (see definition 3); for details see Theorem 1 and its proof in Section III. On the other hand, using a similar proof strategy, we show that Problem 2

is $\#\mathbf{P}$-complete; for details see Theorem 3 and its proof in Section III.

Practical contribution. From a practical point of view, we exploit superpoly and monomial prediction to give better cube attack for Ascon. Most importantly, these attacks are *not restricted* to Ascon, and can be used in other cryptosystems as well, since most cryptosystems, e.g., SHA3 [9], TinyJambu [10], etc. can be thought as a composition of quadratic functions. Ascon [11] is one of the elegant designs of authenticated encryption with associated data (AEAD) that was selected as the first choice for lightweight applications in the CAESAR competition, which also has been submitted to NIST lightweight cryptography standardization. On March 29, 2021, NIST announced ten finalists and Ascon is still on the race. It has been in the literature for a while, however, there has been no successful AEAD which is secure and at the same time lighter than Ascon.

In Section IV, we present a new zero-sum distinguisher for 5-round Ascon with complexity 2^{14} which *improves* the best known cube distinguishers [12] by a factor of 2^2. We emphasis that the blackbox and the non-blackbox distinguishers proposed by Gerault *et al.* [13] possess a generic complexity greater than 2^{75}.

Brief Comparison with the previous methods: In the conventional cube attack papers, one of the obvious ways is to find an upper bound d on the degree of $f(\boldsymbol{x}, \boldsymbol{v})$. Once the upper bound is found, it is not hard to show that a $d+1$-sized cube tester works. Moreover, typically the previous methods wanted p_t, the superpoly to be constant zero.

In [3], the authors used MILP to find the monomial with the *largest* hamming weight and odd number of monomial trails, to find the algebraic degree of $f(\boldsymbol{x}, \boldsymbol{v})$. This is where our method *differs* from them. We observe that, as long as each monomial in the superpoly p_t contains a public variable or is a constant, the indices corresponding to t can be used as a cube tester! Note that, this is directly related to Problem 1. In Section IV, we give a procedure that computes an approximate polynomial $f'(\boldsymbol{v})$ for $f(\boldsymbol{x}, \boldsymbol{v})$ such that if a monomial $m \in \mathbb{F}_2[\boldsymbol{v}]$ is not present in $f'(\boldsymbol{v})$, then there does not exist any monomial $p \cdot m$ where $p \in \mathbb{F}_2[\boldsymbol{x}]$ in $f(\boldsymbol{x}, \boldsymbol{v})$. This helps us in building new zero-sum distinguishers for Ascon. For details, see Section IV.

II. Preliminaries and Notations

A. Complexity Classes

Definition 3 (The class $\oplus\mathbf{P}$)**.** *In computational complexity theory, the complexity class $\oplus\mathbf{P}$ (pronounced 'parity $\mathbf{P}$') is the class of decision problems solvable by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that the number of accepting computation paths is* odd*.*

The class $\#\mathbf{P}$ is the class of function problems of the form "compute $f(x)$", where f is the number of accepting paths of a nondeterministic Turing machine running in polynomial time. One can think of $\oplus$ as $\#\mathbf{P}$ problems $\pmod 2$.

Definition 4 ($\oplus\mathbf{P}$-complete)**.** *A problem L is said to be $\oplus\mathbf{P}$-hard if every problem in $\oplus\mathbf{P}$ can be reduced to L in polynomial-time. It is said to be $\oplus\mathbf{P}$-complete if it is in $\oplus\mathbf{P}$ and also $\oplus\mathbf{P}$-hard.*

Similarly, one can define $\#\mathbf{P}$-complete problems. There are interesting problems known to be $\oplus\mathbf{P}$-complete (simiarly $\#\mathbf{P}$-complete) [14]. We will also use one of them in our hardness result, for details see section III.

a) How hard are $\oplus\mathbf{P}$-complete problems?: It is not hard to show that $\oplus\mathbf{P}$ contains the graph isomorphism problem. On the other hand, $\mathbf{P}^{\oplus\mathbf{P}}$ (oracle power to a machine computing $\oplus\mathbf{P}$ function) is *not known* to contain $\mathbf{NP}$. However, it is *not known* (or believed) to be as strong as $\#\mathbf{P}$. This distinction is important in our context since, monomial prediction in the *most general setting* over $\mathbb{Z}$ or $\mathbb{Q}$, is known to be $\#\mathbf{P}$-complete [5], while the scenario changes when one works over the field $\mathbb{F}_2$, or when the structure of the given function is *restricted*.

B. Description of Ascon

Dobraunig et al. designed Ascon. It is a permutation-based family of authenticated encryption with associated data algorithms (AEAD). The Ascon AEAD algorithm takes as inputs a secret key K, a nonce N, a block header AD (a.k.a associated data) and a message M. It then outputs a ciphertext C of same length as M, and an authentication tag T which authenticates the associated data AD and the message M. There are two variants of Ascon, namely Ascon-128 and Ascon-128a.

C. The Ascon Permutation

The core permutation p of Ascon is based on substitution permutation network (SPN) design paradigm. It operates on a 320-bit state arranged into five 64-bit words and is defined as $p : p_L \circ p_S \circ p_C$. The state at the input of r-th round is denoted by $X_0^r \| X_1^r \| X_2^r \| X_3^r \| X_4^r$ while $Y_0^r \| Y_1^r \| Y_2^r \| Y_3^r \| Y_4^r$ represents the state after the p_S layer. We use $X_i^r[j]$ (resp. $Y_i^r[j]$) to denote the j-th bit (starting from left) of X_i^r (resp. Y_i^r). We now describe the three steps p_C, p_S, and p_L in detail (superscripts are removed for simplicity).

a) Addition of constants (p_C).: We add an 8-bit constant to the bits $56, \cdots, 63$ of word X_2 at each round.

b) Substitution layer (p_S).: We apply a 5-bit S-box on each of the 64 columns. Let $(x_0, x_1, x_2, x_3, x_4)$ and $(y_0, y_1, y_2, y_3, y_4)$ denote the input and output of the S-box, respectively. Then the algebraic normal form (ANF) of the S-box is given in Equation (1). Note that here x_i and y_i are the bits of word X_i and Y_i, respectively.

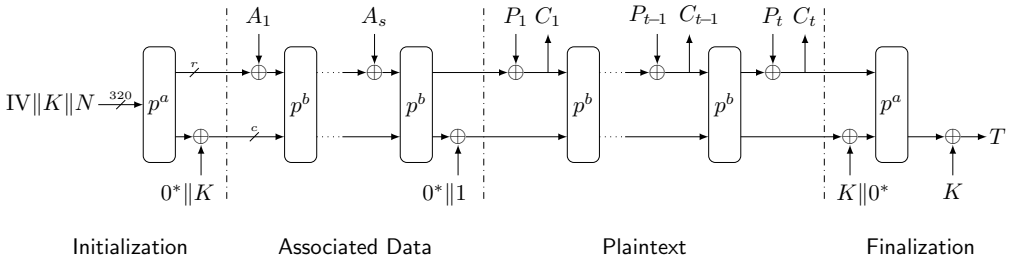

Fig. 1. Ascon's mode of operation (encryption phase)

TABLE I
ASCON VARIANTS AND THEIR RECOMMENDED PARAMETERS

Name	State size	Rate r	Size of			Rounds		IV
			Key	Nonce	Tag	p^a	p^b	
Ascon-128	320	64	128	128	128	12	6	80400c0600000000
Ascon-128a	320	128	128	128	128	12	8	80800c0800000000

$$\begin{cases} y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\ y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \\ y_2 = x_4x_3 + x_4 + x_2 + x_1 + 1 \\ y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \\ y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \end{cases} \tag{1}$$

c) Linear diffusion layer (p_L).: Each 64-bit word is updated by a linear operation Σ_i which is defined in Equation (2). Here $\ggg$ is the right cyclic shift operation over a 64-bit word.

$$\begin{cases} X_0 \leftarrow \Sigma_0(Y_0) = Y_0 + (Y_0 \ggg 19) + (Y_0 \ggg 28) \\ X_1 \leftarrow \Sigma_1(Y_1) = Y_1 + (Y_1 \ggg 61) + (Y_1 \ggg 39) \\ X_2 \leftarrow \Sigma_2(Y_2) = Y_2 + (Y_2 \ggg 1) + (Y_2 \ggg 6) \\ X_3 \leftarrow \Sigma_3(Y_3) = Y_3 + (Y_3 \ggg 10) + (Y_3 \ggg 17) \\ X_4 \leftarrow \Sigma_4(Y_4) = Y_4 + (Y_4 \ggg 7) + (Y_4 \ggg 41) \end{cases} \tag{2}$$

III. On the hardness of monomial prediction: Answering Problem 1 and Problem 2

To show the hardness of problem 1, formally, we define the following language.

$$\begin{aligned} L := \{(f,m) \mid & \mathrm{coef}_m(f_1) = 1, \text{ where } f = (f_1, \ldots, f_{n_{r+1}}) \\ & = g_r \circ g_{r-1} \circ \ldots g_0 \text{ and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}, \\ & n_i \in \mathbb{N},\ \forall\ i \in [r+1], \text{with } n_0 = n, \\ & \text{monomial } m \in \mathbb{F}_2[x_1, \ldots, x_n], \text{and} \\ & \deg((g_i)_j) \leq 2 \}\ . \end{aligned}$$

In general, we will be working with $r, n_i = \mathsf{poly}(n)$ (so that one can think of the input complexity with respect to parameter n). Given (f, m) as an input where f is described by providing a compact representation of g_i's, we want to decide whether $(f, m) \in L$. Note that this maybe 'easier' than solving a large system of multivariate polynomial equations over $\mathbb{F}_2$, and thus one cannot rigorously argue the hardness. However, we show that deciding $(f, m) \in L$ is $\oplus\mathbf{P}$-complete.

Theorem 1 ($\oplus\mathbf{P}$-completeness)**.** *Given a composition of quadratic functions f and a monomial m, deciding whether $(f, m) \in L$ is $\oplus\mathbf{P}$-complete.*

Remark 1. *The hardness proof can be adapted to work over a finite field $\mathbb{F}$ or integer ring $\mathbb{Z}$. In fact, over integers, one can show $\mathbf{NP}$-hardness of the above problem, by adapting the proof strategy of [5]. Of course, a hardness proof over $\mathbb{Z}$ usually* does not *translate to very small characteristic fields like $\mathbb{F}_2$.*

Remark 2. *We are* unable *to show $\mathbf{NP}$-hardness of L. We do not know whether $\oplus\mathbf{P}$ contains $\mathbf{NP}$ or not. In fact, $\mathbf{P}^{\oplus\mathbf{P}}$ is not known to contain $\mathbf{NP}$. Therefore, it will be interesting to show $\mathbf{NP}$-hardness of the above problem.*

Proof. The proof is motivated from algebraic complexity theory and uses the *Hamiltonian Cycle polynomial*, HC_n, defined below, which is a well-known $\mathbf{VNP}$-complete[1] polynomial over $\mathbb{F}_2$ [6], [15]. Remarkably the motivation of studying the hardness of HC_n is quite different from ours and concerns arithmetic circuit complexity while in this paper, we are interested in *Boolean hardness* results!

Recall the definition of *Hamiltonian cycle*: it is a closed loop on a graph where every node (vertex) is visited *exactly* once. It is known that the problem `Odd Hamiltonian Cycle` – deciding whether a given graph $G = (V, E)$ has

[1]The class $\mathbf{VNP}$, Valiant's $\mathbf{NP}$, is known as the *algebraic* $\mathbf{NP}$ class in the algebraic complexity theory.

an odd number of Hamiltonian cycles, is $\oplus\mathbf{P}$-complete [14], i.e., it is in $\oplus\mathbf{P}$ and also is $\oplus\mathbf{P}$-hard. We will use `Odd Hamiltonian Cycle` to show the completeness of L. In particular, we will show the proof in two parts –

1) **Part A** – a reduction from `Odd Hamiltonian cycle` $\leq_{\mathbf{P}} L$ this implying that our problem is $\oplus\mathbf{P}$-hard, and
2) **Part B** – L is in $\oplus\mathbf{P}$.

Part A: Proof of $\oplus\mathbf{P}$-hardness

Define the Hamiltonian Cycle polynomial (HC_n) for a graph with n nodes, with the adjacency matrix $(x_{i,j})_{1\leq i,j\leq n}$ (they are just elements from $\{0,1\}$), as follows:

$$\mathsf{HC}_n\left(x_{1,1},\ldots,x_{n,n}\right) \;=\; \sum_{\sigma\in S_n}\prod_{i=1}^{n} x_{i,\sigma(i)}\;,$$

where S_n is the symmetric group on a set of size n and the sum is taken over all n-cycles of S_n (i.e. , every monomial in HC_n corresponds to a Hamiltonian cycle in the complete directed graph on n vertices). Here is the crucial lemma.

Lemma 2 (Composition lemma)**.** *Let $G=(V,E)$ be a given graph with the adjacency matrix $\boldsymbol{x}=(x_{i,j})_{i,j\in[n]}$. Let $\boldsymbol{y}=(y_1,\ldots,y_n)$ and $\boldsymbol{z}=(z_1,\ldots,z_n)$ be $2n$ variables. Then, there exist $g_0,\ldots,g_n$, polynomial maps such that*

(i) *$g_0:\mathbb{F}_2^{n^2+2n}\longrightarrow\mathbb{F}_2^{2n^2}$, and $g_i:\mathbb{F}_2^{2n^2}\longrightarrow\mathbb{F}_2^{2n^2}$, for $i\in[n]$, with $\deg((g_i)_j)\leq 2$, and*

(ii) *$\mathrm{coef}_{y_1\cdots y_n\cdot z_1\cdots z_n}(f_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})) = \mathsf{HC}_n(\boldsymbol{x})$, where $(f_1,\ldots,f_{2n^2})=g_n\circ\ldots\circ g_0$.*

The above lemma directly implies that for a given graph $G=(V,E)$ with adjacency matrix $(x_{i,j})_{i,j}$, ($f:=g_n\circ\ldots\circ g_0, m:=y_1\cdots y_n z_1\cdots z_n)\in L \iff G$ has odd number of Hamiltonian cycles, which would finish the proof.

Proof of Lemma 2. In the proof, we will often interchange k-th coordinate with (i,j)-th position, for $k\in[n^2]$, where $k-1=(i-1)+n(j-1)$, and $i,j\in[n]$. Since, $k-1\in[0,n^2-1]$ can be *uniquely* written as $(i-1)+n(j-1)$, for some $i,j\in[n]$, there is a *one-to-one* correspondence. We divide the proof into two:

Part 1 : Construction of g_i's. Define the polynomial map $g_0:\mathbb{F}_2^{n^2+2n}\longrightarrow\mathbb{F}_2^{2n^2}$, by defining each coordinate of g_0, namely $(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k$, for $k\in[2n^2]$ by:

$$\left(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\right)_k := \begin{cases} x_{i,j}, & \text{when } k\leq n^2, \text{where}\\ & k-1=(i-1)+n(j-1),\\ y_i\cdot z_j, & \text{when } n^2<k\leq 2n^2, \text{where}\\ & k-1-n^2=(i-1)+n(j-1).\end{cases}$$

In the above, we used the fact that $k-1-n^2\in[0,n^2-1]$ and hence the one-to-one correspondence exists. Trivially any coordinate $(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k$ is at most a quadratic polynomial.

Now define $g_1:\mathbb{F}_2^{2n^2}\longrightarrow\mathbb{F}_2^{2n^2}$, on $2n^2$ variables $\boldsymbol{w}:=(w_{i,j})_{i,j\in[n]}$ and $\boldsymbol{s}:=(s_{i,j})_{i,j\in[n]}$, as follows:

$$\left(g_1(\boldsymbol{w},\boldsymbol{s})\right)_k := \begin{cases} w_{i,j}\cdot s_{i,j}, & \text{when } k\leq n^2, \text{where}\\ & k-1=(i-1)+n(j-1),\\ \left(g_1(\boldsymbol{w},\boldsymbol{s})\right)_{k-n^2}, & \text{when } n^2<k\leq 2n^2.\end{cases}$$

Basically, g_1 repeats the first n^2 coordinates. Again, by definition, each ordinate is a quadratic polynomial. Now, we can define $g_\ell:\mathbb{F}_2^{2n^2}\longrightarrow\mathbb{F}_2^{2n^2}$, again on $2n^2$ variables $(\boldsymbol{w},\boldsymbol{s})$, for $\ell>1$, as follows:

$$\left(g_\ell(\boldsymbol{w},\boldsymbol{s})\right)_k := \begin{cases} \sum_{r=1}^{n} w_{i,r}\cdot s_{r,j}, & \text{when } k\leq n^2, \text{where}\\ & k-1=(i-1)+n(j-1),\\ s_{i,j}, & \text{when } n^2<k\leq 2n^2, \text{where}\\ & k-1-n^2=(i-1)+n(j-1).\end{cases}$$

It is easy to see that, by definition, g_ℓ, restricted to the last n^2 coordinates, is an *identity* map. Also, trivially, each coordinate is a quadratic polynomial.

Part 2 : Getting HC_n as a coefficient of $g_n\circ\ldots\circ g_0$. We will prove two claims about the structure of the compositions. Here is the first claim.

Claim 1. *For any $\ell\geq 1$, we have $(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_k = x_{i,j}\cdot y_i\cdot z_j$, for $k\in[n^2+1,2n^2]$, where $k-1-n^2=(i-1)+n(j-1)$.*

Proof. First let us prove this for $\ell=1$. Since, there is a one-to-one correspondence between the k-th coordinate and the pair (i,j), by definition,

$$g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k \;=\; g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_{k-n^2} \;=\; x_{i,j}\cdot y_i\cdot z_j\;.$$

Since, g_ℓ is an identity map in the last n^2 coordinates, for $\ell>1$, the conclusion follows immediately. □

We remark that, in fact, in the above, it can be easily seen that $g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k = x_{i,j}\cdot y_i\cdot z_j$, for $k\in[n^2]$, where $k-1=(i-1)+n(j-1)$. However, since ℓ grows, $g_\ell\circ\ldots g_0$ looks *complicated*. Here is the main claim, about the structure of the composition, for the first n^2 coordinates.

Claim 2 (Main claim)**.** *For any $\ell\geq 2$, and $k\in[n^2]$, such that $(k-1)=(i-1)+n(j-1)$, the following holds:*

$$\begin{aligned}&\left(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)\right)_k\\ &= y_i z_j\cdot\sum_{1\leq m_1,\ldots,m_{\ell-1}\leq n} x_{i,m_1}x_{m_1,m_2}\cdots x_{m_{\ell-2},m_{\ell-1}}x_{m_{\ell-1},j}\\ &\qquad\cdot\left(\prod_{s=1}^{\ell-1} y_{m_s}z_{m_s}\right)\;.\end{aligned}$$

Proof of the Claim. We will prove this by induction on ℓ.

Base case: $\ell=2$**.** For $\ell=2$, by definition, we have

$$\begin{aligned}\left(g_2(g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))\right)_k &= \sum_{r=1}^{n}(x_{i,r}y_i z_r)\cdot(x_{r,j}y_r z_j)\\ &= y_i z_j\cdot\sum_{1\leq r\leq n} x_{i,r}x_{r,j}y_r z_r\;,\end{aligned}$$

as desired. In the above, we implicitly used the (i,r)-th coordinate, by which we mean the k'-th coordinate such that $k'-1=(i-1)+n(r-1)$, the similar correspondence as we mentioned at the beginning of the proof of Lemma 2. Thus, base case is true.

Inductive step: $(\ell+1)$-th step. Let us assume that it is true for some ℓ. To show this for $\ell+1$, again, by definition (and the one-to-one correspondence between k and (i,j)), we have

$$\begin{aligned}
&(g_{\ell+1}(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_k\\
&=\sum_{r=1}^{n}(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_{i,r}\cdot(x_{r,j}y_rz_j)\\
&=\sum_{r=1}^{n}\left(\sum_{i=1}^{\ell-1}\sum_{m_i=1}^{n}x_{i,m_1}\left(\prod_{t=1}^{\ell-2}x_{m_t,m_{t+1}}\right)x_{m_{\ell-1},r}\right.\\
&\qquad\left.\left(\prod_{s=1}^{\ell-1}y_{m_s}z_{m_s}\right)y_iz_r\right)(x_{r,j}y_rz_j)\\
&=y_iz_j\cdot\sum_{1\le m_1,\ldots,m_{\ell-1},m_\ell\le n}x_{i,m_1}x_{m_1,m_2}\cdots x_{m_{\ell-1},m_\ell}x_{m_\ell,j}\cdot\\
&\qquad\left(\prod_{s=1}^{\ell}y_{m_s}z_{m_s}\right)
\end{aligned}$$

The second last equality is by induction hypothesis while in the last equality, we renamed r by m_ℓ. In the above, by (i,r)-th coordinate, again, we mean k'-th coordinate such that $k'-1=(i-1)+n(r-1)$. This finishes the induction and the conclusion as well. □

Claim 2 with $k=1$ (i.e. $i=j=1$) and $\ell=n$, gives the following identity:

$$\begin{aligned}
&(g_n(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_1\\
&=y_1z_1\cdot\sum_{1\le m_1,\ldots,m_{n-1}\le n}x_{1,m_1}x_{m_1,m_2}\cdots x_{m_{n-2},m_{n-1}}x_{m_{n-1},1}\\
&\qquad\cdot\left(\prod_{s=1}^{n-1}y_{m_s}z_{m_s}\right)
\end{aligned}$$

The coefficient of the monomial $y_1z_1\cdots y_nz_n$ is the Hamiltonian Cycle polynomial $\mathsf{HC}_n(\boldsymbol{x})$, because for any Hamiltonian cycle of length n, we must choose $m_1,\ldots,m_{n-1}$, each between 2 and n, so that such a choice generates the monomial $x_{1,m_1}x_{m_1,m_2}\cdots x_{m_{n-2},m_{n-1}}x_{m_{n-1},1}$. Since, it visits each node *exactly once*, $y_1\cdots y_nz_1\cdots z_n$ is also generated with the $\boldsymbol{x}$-monomial. This finishes the proof of the part 2. □

Since, Lemma 2 is now proved, the $\oplus\mathbf{P}$-hardness follows, as well, finishing the proof of Part A. □

Part B: Proof of $L\in\oplus\mathbf{P}$

We now show that deciding whether $(f,m)\in L$ is $\oplus\mathbf{P}$. To do this, we need to construct an $\mathbf{NP}$ machine $\mathcal{M}$ such that the number of accepting paths is *odd* due to definition 3. The input to the machine $\mathcal{M}$ is (f,m), where $m=\prod_{i\in I}x_i, I\subseteq[n]$. The output of $\mathcal{M}$ is the evaluation of f_1 by setting $x_i=0,\forall i\in[n]\setminus I$, whereas non-deterministically picking $r_j\in\{0,1\}$ and setting $x_j=r_j,\forall j\in I$.

Let $|I|=k$. As mentioned in the introduction, we can express $f_1(x)$ as

$$f_1(x)=m\cdot p_m(x)+q_m(x)$$

and we have

$$\sum_{\substack{(x_{i_1},\ldots,x_{i_k})\in F_2^k\\ x_j=0,\forall j\notin I}}f_1(x)=p_m(x)\in\{0,1\}$$

where $p_m(x)$ does not contain any variable x_i where $i\in I$ and none of the monomials in $q_m(x)$ is divisible by m. Therefore, the above sum evaluates to 1 *iff* m is a monomial in f_1. It is easy to see that each accepting path of $\mathcal{M}$ is essentially a term in the above summation being evaluated to 1. Hence, m is a monomial in f_1 *iff* the number of accepting path in $\mathcal{M}$ is *odd*. This finishes the proof of Part B. □

For problem 2, let us define the problem in a slightly more general setting, similar to problem 1. Let

$$\begin{aligned}
&f=(f_1,\ldots,f_{n_{r+1}})=g_r\circ g_{r-1}\circ\ldots g_0\,,\ g_i:\mathbb{Q}^{n_i}\longrightarrow\mathbb{Q}^{n_{i+1}},\\
&n_i\in\mathbb{N},\ \forall\ i\ \in\ [r+1],\text{with } n_0=n,\\
&g_i \text{ are computable by } \mathsf{poly}(n)-\text{size PGCs}\ .
\end{aligned}$$

Let m be a given monomial. Given (f,m), we want to compute the coefficient of m in f_1. Below, we show that this is $\#\mathbf{P}$-complete.

Theorem 3. *Given a composition of quadratic functions $f=g_r\circ g_{r-1}\circ\ldots\circ g_0$ where g_i can be computed by a* poly*-sized PGCs and a monomial m, computing the coefficient of m in f is $\#\mathbf{P}$-complete.*

Proof sketch. The proof is very similar to the proof of Theorem 1. The part A of the proof will work with a modified version of g_i denoted as $\tilde{g}_i$ such that $\tilde{g}_i$ can be computed in $\mathsf{poly}(n)$-sized PGCs. This is achieved by normalizing each output $p_j^{(i)}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ of g_i with $N_{i,j}$, the number of non-zero coefficients of $p_j^{(i)}$. To be precise, $\tilde{g}_0$ and $\tilde{g}_1$ are exactly g_0 and g_1, whereas for $\ell>1$ and $k\le n^2$, where $k-1=(i-1)+n(j-1)$,

$$(g_\ell(\boldsymbol{w},\boldsymbol{s}))_k:=\sum_{r=1}^{n}\frac{w_{i,r}\cdot s_{r,j}}{n}$$

and when $n^2<k\le 2n^2$, where $k-1-n^2=(i-1)+n(j-1)$,

$$(g_\ell(\boldsymbol{w},\boldsymbol{s}))_k:=s_{i,j}$$

Claim 1 essentially remain the same, whereas in claim 2, the output polynomial contains some additional low-degree terms. To be precise,

$$(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots))_k$$
$$= h_k(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) + \frac{y_i z_j}{n^{\ell-1}} \cdot \sum_{1\le m_1,\ldots,m_{\ell-1}\le n} x_{i,m_1} x_{m_1,m_2} \cdots$$
$$x_{m_{\ell-2},m_{\ell-1}} x_{m_{\ell-1},j} \cdot \left(\prod_{s=1}^{\ell-1} y_{m_s} z_{m_s}\right) .$$

where $h_k(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \in \mathbb{Q}[\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}]$ and $deg(h_k) < 3\cdot\ell$. Similar to the previous proof, we can show that the coefficient of the monomial $y_1 z_1 \cdots y_n z_n$ for $k = 1, \ell = n$ is $\mathsf{HC}(\boldsymbol{x})/n^{n-1}$. However, if we were to work with polynomials over $\mathbb{Q}[\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}]$, the presence of n^{n-1} factor in the denominator would affect the value in the numerator, making it impossible to extract the *exact* number of Hamiltonian cycles. To avoid this, we need to work with polynomials over $\mathbb{F}_p[\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}]$ rather than $\mathbb{Q}[\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}]$ where p is a prime *strictly greater* than $n!$. The reason for choosing such a prime is that the number of Hamiltonian cycles in a graph with n vertices is at most $n!$. These changes are sufficient to show that the problem is $\#\mathbf{P}$-hard. For part B of the proof, i.e., it can be computed by a $\#\mathbf{P}$ machine, can be achieved by interpolation, as described in [5, Appendix A]. □

IV. New Zero-Sum Distinguishers for Ascon

In this section, we present several distinguishers for 5-round Ascon-128. As shown in Figure 1, X_0^0 is set to IV whereas, X_1^0, X_2^0 are set to key bits (secret bits) and X_3^0, X_4^0 to nonce bits (public bits). Since, the ciphertext C_1 is obtained by XOR-ing the message P_1 to X_0^5, we will consider zero-sum distinguisher at X_0^5 positions only.

In [12], the authors found a distinguisher with complexity 2^{16} for 5-round Ascon by setting $X_3^0 = X_4^0$ and finding an upper bound on the algebraic degree in nonce variables using division property. In our experiments, we also set $X_3^0 = X_4^0$.

The road-map. We first show that there is a degree-15 monomial that *is not present* in $X_0^5[1]$, and thus, we find a 15 sized cube as a zero-sum distinguisher. Observe that it is impossible to construct the exact polynomials of X_0^5 with respect to the key and cube variables because the number of key variables is 128, and this naturally results in a huge number of monomials.

Instead of finding the *exact* polynomial in $X_0^5[1]$, which we denote as $f(\boldsymbol{x},\boldsymbol{v}) \in \mathbb{F}_2[\boldsymbol{x},\boldsymbol{v}]$, we will come up with an *approximate polynomial* denoted as $g(\boldsymbol{v}) \in \mathbb{F}_2[\boldsymbol{v}]$. Observe that g contains only cube variables. The approximate polynomial and the exact polynomial has the following relationship.

Property 1. *If the exact polynomial, $f(\boldsymbol{x},\boldsymbol{v})$ has a monomial $p\cdot q$ where $p \in \mathbb{F}_2[\boldsymbol{v}]$ and $q \in \mathbb{F}_2[\boldsymbol{x}]$, then the approximate polynomial, $g(\boldsymbol{v})$ will contain p.*

In other words, if a monomial p is missing from $g(\boldsymbol{v})$, then it is guaranteed that $p\cdot q$ for any $q \in \mathbb{F}_2[\boldsymbol{x}]$ would not be present in $f(\boldsymbol{x},\boldsymbol{v})$. But, this does not say anything about the presence or absence of $p\cdot q'$ in $f(\boldsymbol{x},\boldsymbol{v})$ where $q' \in \mathbb{F}_2[\boldsymbol{v}]$, i.e., q' contains only nonce variables. We do not have to worry about such terms because $p\cdot q'$ can be ignored by setting the appropriate nonce variables to 0, to satisfy $q' = 0$.

To build these approximate polynomials, we will work over the ring of integers $\mathbb{Z}$, *rather than* $\mathbb{F}_2$. We start by building the exact polynomial over $\mathbb{Z}$ up-to 2 rounds, i.e., we will consider both key and cube variables as integer variables and replace XOR with integer $+$ and $\cdot$ with integer $\times$. This will give rise to two issues.

1) Firstly, the coefficients of the monomials will blow-up. This can be handled by reducing the polynomial modulo 2.
2) Secondly, the monomials are no longer multi-linear. This also can fixed by reducing the polynomial by modulo $v_i^2 - v_i, \forall i$.

Observe that as an alternative, we can simply work over $\mathbb{F}_2$, find the exact polynomials and then convert them into polynomials over $\mathbb{Z}$. But, this approach seems to be time consuming in SAGE [16].

Next, we get rid of the key variables by evaluating the polynomial at $x_i = 1, \forall i$. Again, the coefficients may blow-up which is handled by replacing all non-zero coefficients with 1. Observe that these new polynomials have Property 1. We will apply the rest of the 3 rounds on these approximate polynomials to get the approximate polynomials for $X_0^5[1]$. In our experiment, while considering cube indices $0, 1, \ldots, 14$, the approximate polynomial for $X_0^5[1]$ *does not* contain the monomial $\prod_{i=0}^{14} v_i$. This gives us a zero-sum distinguisher for 5-round with complexity 2^{15}. Observe that this experiment is essentially solving Problem 1 for 5-round Ascon with the monomial m being $\prod_{i=0}^{14} v_i$. The source code is available on GitHub and can be provided upon request.

In Table II, we provide more cubes which can serve as a distinguisher for 5-round Ascon-128. We start by randomly guessing a few cubes (i.e., a subset of bit indices in X_3^0 and X_4^0) of size ℓ that is *strictly smaller* than 16, and set the rest of the bits of the nonce (non-cube bits) to 0. For each $u \in \{0,1\}^\ell$, we set the cube variables to u and run 5-round Ascon-128. The output X_0^5 with respect to each u is summed up to get the cube sum. We analyse the cube sum at X_0^5 bits for 2^{15} randomly generated keys and observe the following:

1) For all the 14 sized cubes, the sum at the output mentioned in Table II was 0 for every key.
2) For 13 sized cubes, the sum was 0 with *high probability*.

All indices mentioned in Table II have an offset of 0. Since, the 14 sized cubes are giving 0 as the cube-sum for all 2^{15} randomly chosen keys, we can use them as a distinguisher for 5-round Ascon with very high confidence, owing a complexity of 2^{14} and beating the best known cube distinguisher [12], by a factor of 2^2.

References

[1] I. Dinur and A. Shamir, "Cube attacks on tweakable black box polynomials," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2009, pp. 278–299.

TABLE II
LIST OF CUBES FOR 5-ROUND ASCON-128

Rounds	Cube size	Cube indices ($X_3^0 = X_4^0$)	Output indices (X_0^5)
5	13	0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 16	4
		0, 1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 16	4
		0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16	4
5	14	0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14	1, 4
		0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16	4, 15, 24, 36
		0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 18	4

[2] Y. Todo, "Structural evaluation by generalized integral property," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 287–314.

[3] K. Hu, S. Sun, M. Wang, and Q. Wang, "An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2020, pp. 446–476.

[4] Y. Hao, G. Leander, W. Meier, Y. Todo, and Q. Wang, "Modeling for three-subset division property without unknown subset," *Journal of Cryptology*, vol. 34, no. 3, pp. 1–69, 2021.

[5] N. Kayal, "Algorithms for arithmetic circuits," in *Electron. Colloquium Comput. Complex.*, vol. 17, 2010, p. 73.

[6] G. Malod, "Polynômes et coefficients," Ph.D. dissertation, Université Claude Bernard-Lyon I, 2003.

[7] H. Zhang, B. Juba, and G. Van den Broeck, "Probabilistic generating circuits," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 447–12 457.

[8] J. Harviainen, V. P. Ramaswamy, and M. Koivisto, "On inference and learning with probabilistic generating circuits," in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 829–838.

[9] M. J. Dworkin *et al.*, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.

[10] H. Wu and T. Huang, "Tinyjambu: A family of lightweight authenticated encryption algorithms (version 2)," *Submission to the NIST Lightweight Cryptography Standardization Process*, 2021.

[11] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1. 2. submission to nist (2019)," 2020.

[12] R. Rohit, K. Hu, S. Sarkar, and S. Sun, "Misuse-free key-recovery and distinguishing attacks on 7-round ascon," *IACR Transactions on Symmetric Cryptology*, pp. 130–155, 2021.

[13] D. Gerault, T. Peyrin, and Q. Q. Tan, "Exploring differential-based distinguishers and forgeries for ascon," *Cryptology ePrint Archive*, 2021.

[14] L. G. Valiant, "Completeness for parity problems," in *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3595, 2005, pp. 1–8.

[15] P. Bürgisser, "Completeness and reduction in algebraic complexity theory," vol. 7, 2000.

[16] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 7.6)*, 2017, `https://www.sagemath.org`.