On the hardness of monomial prediction and zero-sum distinguishers for Ascon

Pranjal Dutta [⋆1], Mahesh Sreekumar Rajasree [⋆⋆2], and Santanu Sarkar[3]

[1] Chennai Mathematical Institute, India pranjal@cmi.ac.in
[2] IIT Kanpur, India mahesr@cse.iitk.ac.in
[3] IIT Madras, India santanu@iitm.ac.in

Abstract. In this paper, we focus on the *monomial prediction* problem in the quadratic setting: 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 functions. This 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. Essentially it exploits the fact that a set of monomials are *absent* from the polynomial representation of the cryptographic primitive. Monomial prediction, in a more general setting, over any finite field, is known to be NP-hard. Here, we show that this is true even in the quadratic setting. The proof method is very different from the previous ones and far from obvious. 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.

Keywords: cube testers · boolean function · monomial prediction · NP -hardness · Ascon.

1 Introduction

Dinur and Shamir [2] 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. 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

⋆ Supported by Google Ph.D. Fellowship.
⋆⋆ Supported by Prime Minister's Research Fellowship (PMRF).

that $t = \prod_{i \in I} v_i$. Then, it can be easily verified that for any constants $c_j, \forall j \notin I$,

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

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 [10] introduced the *division property*. Soon division property based cube attack became a hot topic in the community. In [5], 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].

A dive into complexity theory. From a theoretical perspective, how hard is monomial prediction for a given (blackbox) polynomial f? In [6], Kayal considered this very computational problem from a complexity theoretic point of view, which can be broadly (re)stated as follows: Given a blackbox 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, similar problem was also studied by Malod [7] in his PhD thesis, from an algebraic complexity theoretic lens.

Kayal termed this problem as CoeffSLP and showed that it is #P-complete (for a self-contained proof, see [6, Appendix A]). We recall that #P essentially captures the number of solutions of a given instance, and thus obviously a #P problem must be *at least as hard* as the corresponding NP problem. However, roughly speaking, for most of the stream ciphers, f can be thought as a composition of a bunch of *linear* and *quadratic* Boolean functions, and not *arbitrary* compositions. This makes the monomial prediction paradigm both theoretically and practically interesting! So, we restrict ourselves to this particular case and ask the complexity of the following problem defined below.

Problem 1. Given a composition of quadratic functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the coefficient of m in f.

Our contributions. Our paper revolves around the above problem. Though it may sound obvious to be NP-hard, the proof is far from the obvious. We consider a slightly general parameters and indeed show that Problem 1 is NP-hard; for details see Theorem 1 and its proof in section 3.

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. Ascon [3]

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 4, we present a new zero-sum distinguisher for 5-round Ascon with complexity 2^{14} which improves the best known cube distinguishers [8] by a factor of 2^2.

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 [5], the authors usesd 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 4, 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 4.

2 Preliminaries and Notations

2.1 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.

2.2 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).

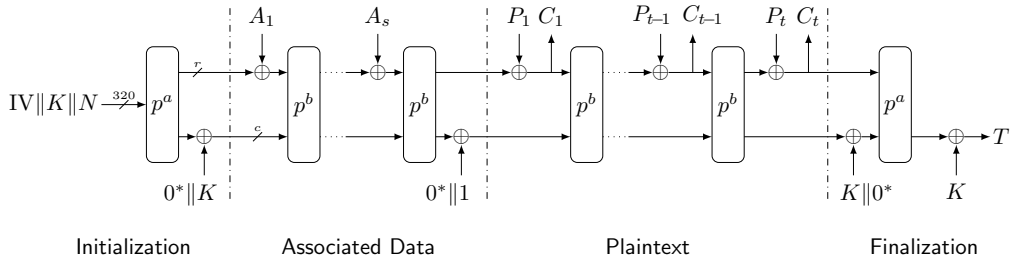

Fig. 1. Ascon's mode of operation (encryption phase)

Table 1. 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

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

Substitution layer (p_S). We apply a 5-bit Sbox 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 Sbox, respectively. Then the algebraic normal form (ANF) of the Sbox is given in Equation 1. Note that here x_i and y_i are the bits of word X_i and Y_i, respectively.

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

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.

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

3 On the hardness of monomial prediction: Answering Problem 1

To show the hardness, formally, we define the following language.

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

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 systems 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 NP-hard.

Theorem 1 (Hardness result). *Given a composition of quadratic functions f and a monomial m, deciding whether $(f,m) \in L$ is NP-hard.*

Remark 1. This proof can be adapted to work over finite field $\mathbb{F}$ or integer ring $\mathbb{Z}$.

Proof. The proof is motivated from algebraic complexity theory and uses the *Hamiltonian Cycle polynomial*, HC_n, defined below, which is a well-known VNP-complete[4] polynomial over $\mathbb{F}_2$ [1,7]. 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 well-known that the problem `Odd Hamiltonian Cycle` – deciding whether a given graph $G = (V,E)$ has an odd number of Hamiltonian cycles, is NP-hard. We now show a reduction from `Odd Hamiltonian cycle`$\leq_{\mathsf{P}} L$ this implying that our problem is NP-hard.

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:

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

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 1 (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*

[4] The class VNP, Valiant's NP, is known as the *algebraic* NP class in the algebraic complexity theory.

(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) $\text{coef}_{y_1 \cdots y_n \cdot z_1 \cdots z_n}(f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) = \mathsf{HC}_n(\boldsymbol{x})$, *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 1. 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:

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

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:

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

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:

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

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:

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

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

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

$$\begin{aligned}(g_2(g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k &= \sum_{r=1}^{n}(x_{i,r}y_iz_r)\cdot(x_{r,j}y_rz_j)\\ &= y_iz_j\cdot\sum_{1\leq r\leq n} x_{i,r}x_{r,j}y_rz_r\ ,\end{aligned}$$

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 1. 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

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

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:

$$(g_n(\ldots(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\ldots)_1$$
$$= y_1 z_1 \cdot \sum_{1 \le m_1, \ldots, m_{n-1} \le n} x_{1,m_1} x_{m_1,m_2} \cdots x_{m_{n-2},m_{n-1}} x_{m_{n-1},1} \cdot \left(\prod_{s=1}^{n-1} y_{m_s} z_{m_s} \right) .$$

The coefficient of the monomial $y_1 z_1 \cdots y_n z_n$ 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*, $y_1 \cdots y_n z_1 \cdots z_n$ is also generated with the $\boldsymbol{x}$-monomial. This finishes the proof of the part 2. □

Since, Lemma 1 is now proved, the NP-hardness follows, as well. □

4 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 [8], 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 $X_0^5[1]$, we will come up with an *approximate polynomial* which contains the cube variables only. The approximate polynomial has the following property.

Property 1. If the exact polynomial 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 will contain p.

In other words, if a monomial p is missing from the approximate polynomial, then it is guaranteed that $p \cdot q$ for any $q \in \mathbb{F}_2[\boldsymbol{x}]$ would not be present in the exact output polynomial. But, this does not say anything about the presence or absence of $p \cdot q'$ in the exact polynomial where $q' \in \mathbb{F}_2[\boldsymbol{v}]$. 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 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 seem to be time consuming in SAGE [9].

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 at: `https://github.com/Mahe94/ascon_monomial_detection.git`.

In Table 2, 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 2 was 0 for every key.
2. For 13 sized cubes, the sum was 0 with *high probability*.

All indices mentioned in Table 2 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 [8], by a factor of 2^2.

References

1. Bürgisser, P.: Completeness and reduction in algebraic complexity theory, vol. 7. Springer Science & Business Media (2000) 5
2. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 278–299. Springer (2009) 1
3. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1. 2. submission to NIST (2019) (2020) 2
4. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for Three-Subset Division Property without Unknown Subset. Journal of Cryptology **34**(3), 1–69 (2021) 2

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

Table 2. List of cubes for 5-round Ascon-128

5. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 446–476. Springer (2020) 2, 3
6. Kayal, N.: Algorithms for Arithmetic Circuits. In: Electron. Colloquium Comput. Complex. vol. 17, p. 73 (2010) 2
7. Malod, G.: Polynômes et coefficients. Ph.D. thesis, Université Claude Bernard-Lyon I (2003) 2, 5
8. Rohit, R., Hu, K., Sarkar, S., Sun, S.: Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. IACR Transactions on Symmetric Cryptology pp. 130–155 (2021) 3, 8, 9
9. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 7.6) (2017), `https://www.sagemath.org` 9
10. Todo, Y.: Structural evaluation by generalized integral property. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 287–314. Springer (2015) 2