IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.13154/tosc.v0.i0.0-0

Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon

Raghvendra Rohit[1], Kai Hu[2,5], Sumanta Sarkar[3] and Siwei Sun[4,6]

[1] Univ Rennes, Centre National de la Recherche Scientifique (CNRS), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes, France
raghvendra-singh.rohit@irisa.fr
[2] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
hukai@mail.sdu.edu.cn
[3] TCS Innovation Labs, Hyderabad, India, sumanta.sarkar1@tcs.com
[4] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, siweisun.isaac@gmail.com
[5] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, Shandong, China
[6] University of Chinese Academy of Sciences, Beijing, China

Abstract. Being one of the winning algorithms of the CAESAR competition and currently a second round candidate of the NIST lightweight cryptography standardization project, the authenticated encryption scheme Ascon (designed by Dobraunig, Eichlseder, Mendel, and Schläffer) has withstood extensive self and third-party cryptanalysis. The best known attack on Ascon could only penetrate up to 7 (out of 12) rounds due to Li et al. (ToSC Vol I, 2017). However, it violates the data limit of 2^{64} blocks per key specified by the designers. Moreover, the best known distinguishers of Ascon in the AEAD context reach only 6 rounds. To fill these gaps, we revisit the security of 7-round Ascon in the nonce-respecting setting without violating the data limit as specified in the design. First, we introduce a new superpoly-recovery technique named as *partial polynomial multiplication* for which computations take place between the so-called degree-d homogeneous parts of the involved Boolean functions for a $2d$-dimensional cube. We apply this method to 7-round Ascon and present several key recovery attacks. Our best attack can recover the 128-bit secret key with a time complexity of about 2^{123} 7-round Ascon permutations and requires 2^{64} data and 2^{101} bits memory. Also, based on division properties, we identify several 60 dimensional cubes whose superpolies are constant zero after 7 rounds. We further improve the cube distinguishers for 4, 5 and 6 rounds. Although our results are far from threatening the security of full 12-round Ascon, they provide new insights in the security analysis of Ascon.

Keywords: Ascon · Authenticated encryption · Cube attack · Division property · Partial polynomial multiplication

1 Introduction

Around 2000, Bellare and Namprempre introduced the notion of authenticated encryption (AE) – a type of symmetric-key primitive providing both confidentiality and authenticity. Its subsequent development is shaped by real-world applications and finally it evolves into the notion of authenticated encryption with associated data (AEAD) [Rog02, Rog04, RS06], where the authenticity of the associated data (some public information like packet headers) along with the message is also ensured.

The first major event in the cryptographic community for soliciting and evaluating AEADs was the CAESAR competition (the Competition for Authenticated Encryption: Security, Applicability, and Robustness), initially announced at the Early Symmetric-key Crypto workshop 2013 [Ear, CAE]. After several years of intensive analysis and comparison of the 57 submissions, the final portfolio was announced in February 2019, and the winning algorithms are categorized into three use cases listed as follows:

- Lightweight applications: Ascon [DEMS16] and ACORN [Wu16a];
- High-performance applications: AEGIS-128 [WP16] and OCB [KR16];
- Defense in depth: Deoxys-II [JNPS16] and COLM [ABD+16].

Ascon, the main target of this work, is a family of lightweight AEAD which has been selected as the primary choice for the lightweight use case in the final portfolio of the CAESAR competition. It was subsequently submitted to the LWC project – a public competition-like process to solicit, evaluate, and standardize authenticated encryption and hashing schemes suitable for highly constrained computing environments initiated by the US National Institute of Standards and Technology (NIST) [Nat19]. On August 30, 2019, Ascon was selected as one of the 32 second-round candidates out of the 57 initial submissions (only 56 were accepted as the first-round candidates) based on public feedbacks and internal reviews. As one of the winning algorithms of the CAESAR competition and second-round candidates of the NIST LWC project, Ascon has withstood extensive self-evaluation and third-party cryptanalysis, which are briefly summarized in the following.

Previous Cryptanalysis. Apart from the self-analysis provided by the designers [DEMS16], Ascon has gone through substantial third-party cryptanalysis. First of all, without considering the AEAD context, the security of the underlying permutation of Ascon was evaluated with respect to (impossible) differential cryptanalysis [Tez16], (zero-correlation) linear cryptanalysis [DEM15], differential-linear cryptanalysis [DEMS15, BDKW19], integral (based on division properties) or zero-sum distinguishing attacks [YLW+19, DEMS15, GRW16, Tod15], and subspace trail cryptanalysis [LTW18]. While these works do provide a deeper understanding of the security of Ascon permutation, generally they do not directly translate into meaningful attacks in the AEAD setting.

Cryptanalysis of Ascon in the AEAD context can be divided into two categories. In the first category, generic security analysis or comparison of a series of constructions with Ascon or its variants as special cases is conducted. For example, security analysis and bounds for the full-state keyed duplex with application to Ascon was discussed in [DMA17]. In [VV18], Vaudenay and Vizár analyzed the misuse resistance of Ascon along with other third-round CAESAR candidates. At ASIACRYPT 2014, Jovanovic et al. provided security proofs for sponge-duplex mode concluding that Ascon can process higher data rate without degradation in security [JLM14]. Later, in [SY15], Sasaki and Yasuda gave some suggestions on processing associated data efficiently in SpongeWrap-like modes (including Ascon), which can achieve the same security bounds as Jovanovic et al. [JLM14]. Moreover, Forler et al. discussed the reforgeability of Ascon and many other authenticated encryption algorithms in [FLLW17].

The second category is more relevant to our work, where concrete cryptanalysis specific to Ascon is performed, including state recovery attacks [DKM+17], differential-linear cryptanalysis [DEMS15], and cube-like attacks [LZWW17, DEMS15, LDW17]. A summary of the results are given in Table 1, from which we can see that the best claimed attack penetrates seven rounds of Ascon. It is worth noting that all 7-round attacks on Ascon in literature so far require some misuse of the target violating the security claims of the design and thus are invalid. Therefore, the best previous attack only reaches six rounds [LDW17].

Table 1: Summary of attacks and distinguishers on ASCON in the AEAD context

Type	#Rounds	Data	Time	Method	Validity	Source
Key recovery	4/12	2^{18}	2^{18}	Differential-linear	✓	[DEMS15]
	5/12	2^{36}	2^{36}	Differential-linear	✓	[DEMS15]
	5/12	2^{35}	2^{35}	Cube-like	✓	[DEMS15]
	5/12	2^{24}	2^{24}	Conditional cube	✓	[LDW17]
	6/12	2^{66}	2^{66}	Cube-like	✓	[DEMS15]
	6/12	2^{40}	2^{40}	Conditional Cube	✓	[LDW17]
	7/12	$2^{77.2}$	$2^{103.9}$	Conditional cube	✗	[LDW17]
	7/12	$2^{77.2}$	2^{77}	Conditional cube‡	✗	[LDW17]
	7/12	2^{33}	2^{97}	Cube-like	×	[LZWW17]
	7/12	2^{33}	2^{97}	Cube tester	×	[LZWW17]
	7/12	2^{64}	$\mathbf{2^{123}}$	**Cube**	✓	**Section 6**
Distinguisher	4/12	2^{9}	2^{9}	Degree	✓	[DEMS15]
	5/12	2^{17}	2^{17}	Degree	✓	[DEMS15]
	6/12	2^{33}	2^{33}	Degree	✓	[DEMS15]
	4/12	$\mathbf{2^{5}}$	$\mathbf{2^{5}}$	**Division Property**	✓	Section 7
	5/12	$\mathbf{2^{16}}$	$\mathbf{2^{16}}$	**Division Property**	✓	Section 7
	6/12	$\mathbf{2^{31}}$	$\mathbf{2^{31}}$	**Division Property**	✓	Section 7
	7/12	$\mathbf{2^{60}}$	$\mathbf{2^{60}}$	**Division Property**	✓	Section 7
Forgery	3/12	2^{33}	2^{33}	Differential	✓	[DEMS15]
	4/12	2^{101}	2^{101}	Differential	✗	[DEMS15]
	4/12	2^{9}	2^{9}	Cube-like	×	[LZWW17]
	5/12	2^{17}	2^{17}	Cube-like	×	[LZWW17]
	6/12	2^{33}	2^{33}	Cube-like	×	[LZWW17]
State recovery	5/6	2^{18}	2^{66}	Cube-like	×	[LZWW17]

✗: Invalid as the required data is beyond 2^{64}; ×: Invalid as the nonce is repeated; ‡ : Weak key setting

Our Contributions. Before listing the contributions, we would like to emphasize the security claims [DEMS16] made by the designers of ASCON and discuss their implications on previous cryptanalysis:

> "*The number of processed plaintext and associated data blocks protected by the encryption algorithm is limited to a total of* 2^{64} *blocks per key ...*" (see [DEMS16, Chapter 2, Page 9])
>
> "*In order to fulfill the security claims ..., implementations must ensure that the nonce (public message number) is never repeated for two encryptions under the same key ...*" (see [DEMS16, Chapter 3, Page 12])

The above two statements indicate that any attack requiring more than 2^{64} known/chosen data (plaintexts, associated data or nonces) blocks under a same key is invalid. Taking the 7-round attack given by Li et al. [LDW17] for example, it requires at least 2^{65} nonces with the same secret key, and since they employed several different cubes of dimension 65, their actual data complexity is more than 2^{65}. Accordingly, we conclude that this attack is invalid and the best known attack reaches only six rounds of ASCON.[1] In this work, complying with the security requirements, we present the first misuse-free key recovery and distinguishing attacks on 7-round ASCON. Our contributions are twofold.

Firstly, we propose a generic technique called *partial polynomial multiplication* for cube attacks. The technique enables to recover the superpoly of a given cube by multiplying

[1]We have confirmed this issue with both the designers of ASCON and the authors of [LDW17].

the simplified versions of the involved Boolean functions. More precisely, under certain conditions, the superpoly of a $2d$-dimensional cube at r-th round can be computed by multiplying some specific sets of partial polynomials (the so-called degree-d homogeneous parts) from previous rounds. We apply this technique on 7-round Ascon and could recover the superpolies of a 64 dimensional cube with time complexity of about 2^{123} 7-round Ascon permutations. We give the superpoly recovering procedure for different configurations of cube and non-cube variables to achieve a key-recovery attack with minimal time complexity. Our best attack can recover the 128-bit secret key with time and memory complexities of 2^{123} and 2^{101} bits, respectively.

Secondly, we identify several new cube distinguishers for Ascon in the AEAD setting using the division property. We show that there exist $2^{19.27}$ 60 dimensional cubes whose superpolies are constant zero after seven rounds. To the best of our knowledge, these are the first distinguishers for 7-round Ascon. For 4-, 5-, and 6-round Ascon, we find distinguishers with complexities $2^5, 2^{16}$, and 2^{31} which improves the best known cube distinguishers by a factor of 2^4, 2, and 2^2, respectively. All the source codes for verification are publicly available at `https://github.com/raghavrohit/ascon_cube_distinguishers`.

Outline. The rest of the paper is organized as follows. Section 2 provides an overview of useful techniques in the theory of Boolean functions, cube attacks, and division properties. The specification of Ascon and our attack models are described in Section 3. We introduce the notion of partial polynomial multiplication in Section 4 and give our key recovery attacks in Sections 5 and 6. Our MILP modeling for the division properties and the obtained distinguishers are discussed in Section 7. Finally, we conclude in Section 8 with some open problems.

2 Notations and Preliminaries

Let A and B be two sets. The number of elements in A is written as $|A|$. The set of all elements in A but not in B is denoted by $A - B$. Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field with two elements and $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function whose algebraic normal form (ANF) is $f(\boldsymbol{x}) = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{u}}$, where $\boldsymbol{x} = (x_0, \cdots, x_{n-1})$, $a_{\boldsymbol{u}} \in \mathbb{F}_2$, and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i{}^{u_i}$. We denote the *coefficient* of the *monomial* $\boldsymbol{x}^{\boldsymbol{u}}$ in f by $a_{\boldsymbol{u}} = \mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}})$. Given a set $I \subseteq \{0, \cdots, n-1\}$ of indexes, $\boldsymbol{x}[I]$ denotes the set of variables $\{x_i : i \in I\}$ and $\boldsymbol{x}^I$ denotes the *monomial* $\prod_{i\in I} x_i$. Let $\boldsymbol{u} = (u_0, \cdots, u_{n-1})$ and $\boldsymbol{v} = (v_0, \cdots, v_{n-1})$ be two vectors in $\mathbb{F}_2^n$. We say $\boldsymbol{u} \preceq \boldsymbol{v}$ if $u_i \le v_i$ for all $i \in \{0, \cdots, n-1\}$. In addition, the Hamming weight of $\boldsymbol{u}$ is denoted by $wt(\boldsymbol{u})$. Note that we use "+" to denote all kinds of additions (of integers, field elements, Boolean functions, etc.), the actual meaning of a specific use instance should be clear from the context.

Lemma 1 ([Car10, Can16])**.** *Given an oracle access to the Boolean function f, the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{u}}$ in f for a particular $\boldsymbol{u}$ can be computed as* $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}}) = \sum_{\boldsymbol{x}\preceq\boldsymbol{u}} f(\boldsymbol{x})$ *with $2^{wt(\boldsymbol{u})}$ evaluations of f.*

Lemma 2 ([Car10, Can16])**.** *The set of all coefficients $\{a_{\boldsymbol{u}} = \sum_{\boldsymbol{x}\preceq\boldsymbol{u}} f(\boldsymbol{x}) : \boldsymbol{u} \in \mathbb{F}_2^n\}$ of the ANF of f can be obtained from the truth table of f with the so-called fast Möbius transform with about $n2^n$ XOR operations*[2]*.*

Definition 1. The degree of a Boolean function f, represented by $\deg(f)$, is defined as $max\{wt(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{F}_2^n$ and $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}}) \neq 0\}$.

[2]In [Can16], the complexity is given as $n2^{n-1}$. We note that this complexity discrepancy of the fast Möbius transform does not affect the dominant terms of the complexities of our attacks.

Definition 2. The degree-d homogeneous part of a Boolean function $f = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} x^{\boldsymbol{u}}$ is defined as the sum of all degree-d terms of f, and is denoted as

$$\hbar_d(f) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n, wt(\boldsymbol{u})=d} a_{\boldsymbol{u}} x^{\boldsymbol{u}}.$$

Example 1. For a Boolean function $f(x_0, x_1, x_2, x_3) = x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + x_0x_1 + x_0x_3 + x_1 + x_3 + 1$ the degree-3 homogeneous part of f is $\hbar_3(f) = x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3$ while the degree-2 homogeneous part of f is $\hbar_2(f) = x_0x_1 + x_0x_3$.

Keyed Boolean Functions. In the context of symmetric-key cryptanalysis, we typically regard each output bit of a keyed primitive with an m-bit secret key as a keyed Boolean function $f_{\boldsymbol{k}} : \mathbb{F}_2^n \to \mathbb{F}_2$ whose algebraic normal form is

$$f_{\boldsymbol{k}}(\boldsymbol{x}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k}) \boldsymbol{x}^{\boldsymbol{u}}, \boldsymbol{x} \in \mathbb{F}_2^n, \boldsymbol{k} \in \mathbb{F}_2^m, \tag{1}$$

where we regard $\boldsymbol{k}$ as a (secret) constant. In this setting, the coefficient $\mathrm{Coe}_{f_{\boldsymbol{k}}}(\boldsymbol{x}^{\boldsymbol{u}})$ can be represented as a Boolean function from $\mathbb{F}_2^m \to \mathbb{F}_2$ which maps $\boldsymbol{k}$ to $a_{\boldsymbol{u}}(\boldsymbol{k})$. In fact, the function mapping $(\boldsymbol{x}, \boldsymbol{k})$ to $f_{\boldsymbol{k}}(\boldsymbol{x})$ can be expressed as a Boolean function from $\mathbb{F}_2^{m+n}$ to $\mathbb{F}_2$. However, in our work, $\boldsymbol{k}$ (secret constants) and $\boldsymbol{x}$ (Boolean variables) are not treated equally. When we talk about the degree of a keyed Boolean function, the degree is defined with respect to $\boldsymbol{x}$. Moreover, we may use $f(\boldsymbol{x}, \boldsymbol{k})$ to denote the keyed Boolean function $f_{\boldsymbol{k}}(\boldsymbol{x})$ when there is no confusion. We use the following example to clarify potential notation confusions caused by keyed Boolean functions.

Example 2. For a keyed Boolean function $f(x_0, x_1, x_2, k_0, k_1, k_2) = k_0k_1k_2x_0x_1x_2 + k_1k_2x_0x_1x_2 + k_0x_0x_2x_3 + k_1x_0x_1 + k_2x_0x_1 + x_2 + k_2 = (k_0k_1k_2 + k_1k_2)x_0x_1x_2 + k_0x_0x_2x_3 + (k_1 + k_2)x_0x_1 + x_2 + k_2$ with a 3-bit secret key (k_0, k_1, k_2), the degree of f is 3 rather than 6. The set of all terms involved in f is

$$\{(k_0k_1k_2 + k_1k_2)x_0x_1x_2, k_0x_0x_2x_3, (k_1 + k_2)x_0x_1, x_2, k_2\}$$

rather than $\{k_0k_1k_2x_0x_1x_2, k_1k_2x_0x_1x_2, k_0x_0x_2x_3, k_1x_0x_1, k_2x_0x_1, x_2, k_2\}$. Also, we have $\mathrm{Coe}_f(x_0x_1x_2) = k_0k_1k_2 + k_1k_2$. The degree-3 homogeneous part of f is $\hbar_3(f) = (k_0k_1k_2 + k_1k_2)x_0x_1x_2 + k_0x_0x_2x_3$ while the degree-2 homogeneous part of f is $\hbar_2(f) = (k_1 + k_2)x_0x_1$.

According to Lemma 1, for any keyed Boolean function f and any given $\boldsymbol{k}$, the *value* of $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}})$ can be obtained with $2^{wt(\boldsymbol{u})}$ evaluations of f. Thus, the truth table of $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}})$ for all possible $\boldsymbol{k} \in \mathbb{F}_2^m$ can be obtained with $2^{m+wt(\boldsymbol{u})}$ evaluations of f. Then, applying Lemma 2, the ANF of the $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}})$ in $\boldsymbol{k}$ can be derived with about $m2^m = 2^{m+\log_2 m}$ XOR operations. Therefore, if $wt(\boldsymbol{u})$ is much larger than $\log_2 m$, the complexity for recovering the ANF of $\mathrm{Coe}_f(\boldsymbol{x}^{\boldsymbol{u}})$ for this particular $\boldsymbol{u}$ can be estimated as $2^{m+wt(\boldsymbol{u})}$ evaluations of f. For the sake of convenience, we write it as a Lemma.

Lemma 3. *For the Boolean function shown in Equation 1, it takes $2^{m+wt(\boldsymbol{u})}$ evaluations of f to recover $a_{\boldsymbol{u}}(\cdot)$ for a certain $\boldsymbol{u}$ where $wt(u) > log_2(m)$.*

Cube Attack and Division Property. The cube attack was proposed at EUROCRYPT 2009 by Dinur and Shamir to analyze black-box tweakable polynomials [DS09]. Given a keyed Boolean function $f(\boldsymbol{x}, \boldsymbol{k})$ with n-bit public input $\boldsymbol{x} \in \mathbb{F}_2^n$ and m-bit secret input $\boldsymbol{k} \in \mathbb{F}_2^m$, for a set $I \subseteq \{0, \cdots, n-1\}$ with its complementary set $\bar{I} = \{0, \cdots, n-1\} - I$, we have

$$f(\boldsymbol{x}, \boldsymbol{k}) = \boldsymbol{x}^I \cdot p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k}) + q(\boldsymbol{x}, \boldsymbol{k}),$$

where each term of $q(\boldsymbol{x}, \boldsymbol{k})$ misses some variables in $\boldsymbol{x}[I]$. We call $\boldsymbol{x}^I$ the cube term and $p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k})$ the superpoly of $\boldsymbol{x}^I$ in $f(\boldsymbol{x}, \boldsymbol{k})$. If we set the variables in $\boldsymbol{x}[\bar{I}]$ to some fixed constants, the superpoly $p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k})$ is a Boolean function of $\boldsymbol{k}$. How to recover the algebraic normal form of $p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k})$ in the key bits is a fundamental problem in cube attacks. Concerning the superpoly, we have the following lemma.

Lemma 4 ([DS09])**.** *For a set $I \subseteq \{0, \cdots, n-1\}$ and a keyed Boolean function*

$$f(\boldsymbol{x}, \boldsymbol{k}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k}) \boldsymbol{x}^{\boldsymbol{u}} = \boldsymbol{x}^I \cdot p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k}) + q(\boldsymbol{x}, \boldsymbol{k}),$$

we have $p_I(\boldsymbol{x}[\bar{I}], \boldsymbol{k}) = \sum_{\boldsymbol{x}[I] \in \mathbb{F}_2^{|I|}} f(\boldsymbol{x}, \boldsymbol{k})$.

In Lemma 4, if $I = \{0, \cdots, n-1\}$, then the superpoly of $\boldsymbol{x}^I$ in $f(\boldsymbol{x}, \boldsymbol{k})$ equals to $\mathrm{Coe}_f(\boldsymbol{x}^I)$. In this case, recovering the ANF of the superpoly of $\boldsymbol{x}^I$ is equivalent to recovering the coefficient $\mathrm{Coe}_f(\boldsymbol{x}^I)$.

A very useful cryptanalytic technique in literature is the division property initially proposed by Todo at EUROCRYPT 2015 [Tod15] as a generalization of integral cryptanalysis. Its bit-based variants [TM16] together with their automatic search methods [XZBL16] have been found to have a great potential in probing the structure of a Boolean function described as a sequence of composition of Boolean functions whose overall ANF is too complicated to compute [TIHM17, WHT+18, WHG+19, HLM+20]. In particular, bit-based division property can detect the presence or absence of a monomial in the target Boolean function, and therefore can be used to (partially) determine the algebraic structures of superpolies in cube attacks [TIHM17, WHT+18, WHG+19, HLM+20, HLLT20, HSWW20]. In fact, division property has become a quite standard tool in assisting cube attacks. In this work, we take the MILP (Mixed Integer Linear Programming) based approach to search for division properties and find cube distinguishers of Ascon. The technical details will be introduced on-site in Section 7 when immediately necessary.

3 Specification and Useful Properties of Ascon

Ascon (designed by Dobraunig, Eichlseder, Mendel, and Schläffer) is a family of AEAD algorithms [DEMS16]. At a high level, as depicted in Figure 1, the Ascon AEAD algorithm takes as input a nonce N, a secret key K, an associated data A and a plaintext or message M, and produces a ciphertext C and a tag T. The authenticity of the associated data and message can be verified against the tag T. Table 2 lists the variants of Ascon along with the recommended parameter sets.

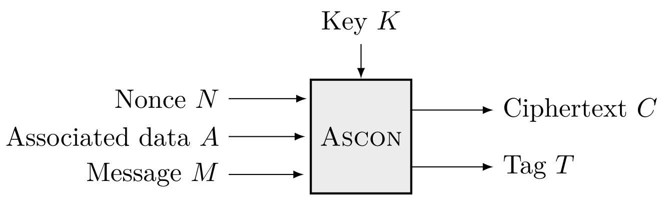

Figure 1: The high-level structure of the encryption algorithm of Ascon

Ascon adopts a MonkeyDuplex [Dae12] mode with a stronger keyed initialization and keyed finalization phases as illustrated in Figure 2. The underlying permutations p^a and p^b are iterative designs, whose round function p is based on the substitution permutation network design paradigm and consists of three simple steps p_C, p_S, and p_L. We now describe the round function p and each step in detail.

Table 2: Ascon variants and their recommended parameters

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

The round function $p = p_L \circ p_S \circ p_C$ operates on a 320-bit state arranged into five 64-bit words. The input state to the round function at r-th round is denoted by $X_0^r \| X_1^r \| X_2^r \| X_3^r \| X_4^r$ while the output state after p_S is given by $Y_0^r \| Y_1^r \| Y_2^r \| Y_3^r \| Y_4^r$. A bit of these words is denoted by $[\cdot]$. For instance, $X_i^r[j]$ represents the j-th bit (starting from left) of word i at r-th round for $j = 0, \cdots, 63$. Alternatively, $X_i^r[j]$ is also denoted as $X^r[64i + j]$. The steps p_C, p_S, and p_L (with round superscript removed for simplicity) are visualized in Figure 3, Figure 5 and Figure 6 [3], respectively and described as follows.

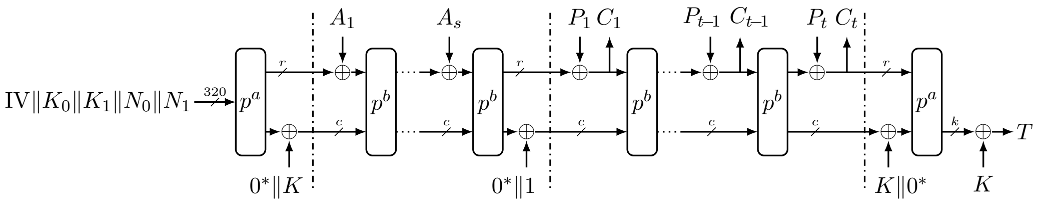

Figure 2: The encryption algorithm of Ascon

Addition of constants (p_C). An 8-bit constant is XORed to the bit positions $56, \cdots, 63$ of the 64-bit word X_2 at each round.

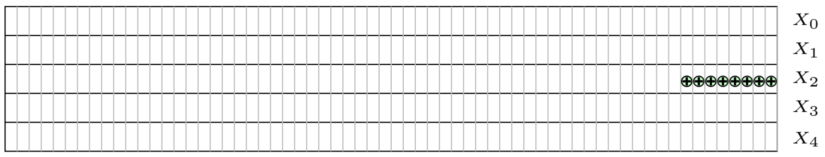

Figure 3: Addition of constant in word X_2 of state

Substitution layer (p_S). Update each slice of the 320-bit state by applying the 5-bit Sbox defined by the following algebraic normal forms:

$$\begin{cases} Y_0[j] \leftarrow X_4[j]X_1[j] + X_3[j] + X_2[j]X_1[j] + X_2[j] + X_1[j]X_0[j] + X_1[j] + X_0[j] \\ Y_1[j] \leftarrow X_4[j] + X_3[j]X_2[j] + X_3[j]X_1[j] + X_3[j] + X_2[j]X_1[j] + X_2[j] + X_1[j] + X_0[j] \\ Y_2[j] \leftarrow X_4[j]X_3[j] + X_4[j] + X_2[j] + X_1[j] + 1 \\ Y_3[j] \leftarrow X_4[j]X_0[j] + X_4[j] + X_3[j]X_0[j] + X_3[j] + X_2[j] + X_1[j] + X_0[j] \\ Y_4[j] \leftarrow X_4[j]X_1[j] + X_4[j] + X_3[j] + X_1[j]X_0[j] + X_1[j] \end{cases} \tag{2}$$

The substitution layer is typically implemented using the bitsliced form rather than applying 5-bit operation on each slice (Equation 2). This is illustrated in Figure 4 where the operations are performed on each of the five 64-bit word.

By studying the ANF of the Sbox in Equation 2, the following properties are given.

[3]Thanks to TikZ for Cryptographers [Jea16]. Ascon figures are adapted from [DEMS]

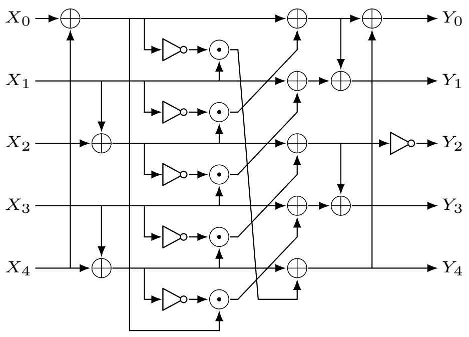

Figure 4: Bitsliced implementation of the substitution layer

Property 1. Among the 5-bit output of the Sbox, $X_4[\cdot]$ never multiplies with $X_2[\cdot]$.

Property 2. For each bit of the output, $X_3[\cdot]X_1[\cdot]$ and $X_3[\cdot]X_2[\cdot]$ always appear together or do not appear.

Property 3. If $X_3[j] = X_4[j]$, then $Y_2[j]$ and $Y_3[j]$ are independent of $X_3[j]$ and $X_4[j]$.

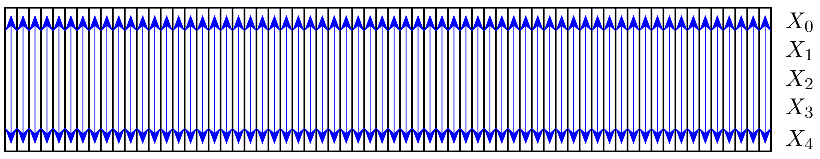

Figure 5: The substitution layer p_S

Linear diffusion layer (p_L). Apply a linear transformation Σ_i to each 64-bit word Y_i with $0 \leq i < 5$, where Σ_i is defined as

$$\begin{cases} X_0 \leftarrow \Sigma_0(Y_0) = Y_0 + (Y_0 \ggg 19) + (Y_0 \ggg 28) \\ X_1 \leftarrow \Sigma_1(Y_1) = Y_1 + (Y_1 \ggg 61) + (Y_1 \ggg 39) \\ X_2 \leftarrow \Sigma_2(Y_2) = Y_2 + (Y_2 \ggg 1) + (Y_2 \ggg 6) \\ X_3 \leftarrow \Sigma_3(Y_3) = Y_3 + (Y_3 \ggg 10) + (Y_3 \ggg 17) \\ X_4 \leftarrow \Sigma_4(Y_4) = Y_4 + (Y_4 \ggg 7) + (Y_4 \ggg 41) \end{cases} \tag{3}$$

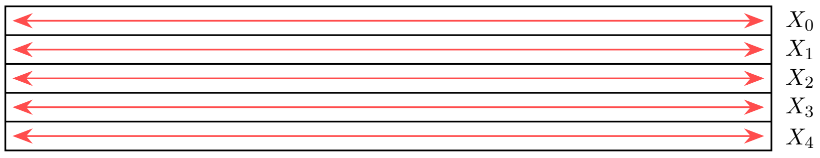

Figure 6: The linear diffusion layer p_L

Configurations for Our Attacks. The overall configuration for our attacks is visualized in Figure 7, where only one block of message is involved and there is no associated data. In our key-recovery attack, for each nonce $N_0||N_1$, we call the ASCON oracle to encrypt a random plaintext P and obtain the corresponding ciphertext C, then X_0^7 can be calculated

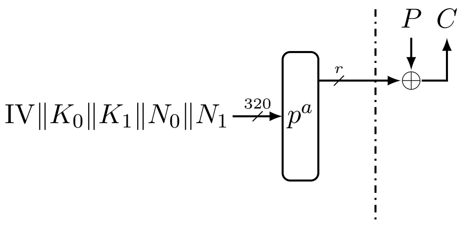

Figure 7: Our attack model

by $P + C$. Since the linear operation Σ_0 acts on word 0, we apply Σ_0^{-1} to X_0^7 and obtain Y_0^6. Moreover, since

$$X_0^7 = \Sigma_0(Y_0^6) = Y_0^6 + (Y_0^6 \ggg 19) + (Y_0^6 \ggg 28)$$

the algebraic degrees of $X_0^7[j]$ and $Y_0^6[j]$ are equal and it is simpler for us to recover the superpoly for a certain cube term in $Y_0^6[j]$ than $X_0^7[j]$. Therefore, in this paper, we always focus on recovering the superpoly of $Y_0^6[j], 0 \leq j < 64$. When without ambiguity, the 7-round Ascon output means $Y_0^6[j]$. For the distinguishers, the attack setting is similar. The only difference is that instead of recovering a superpoly, we find cubes whose superpolies are constant zero. In the following, we only give the key recovery attacks and distinguishers for Ascon-128 in detail. However, they are equally applicable to Ascon-128a.

4 New Technique for Superpoly Recovery: Partial Polynomial Multiplication

In this section, we formally introduce the partial polynomial multiplication technique for superpoly recovery. Let $f(\boldsymbol{x}, \boldsymbol{k})$ be a keyed Boolean function with algebraic normal form

$$f(\boldsymbol{x}, \boldsymbol{k}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k}) \boldsymbol{x}^{\boldsymbol{u}}, \boldsymbol{x} \in \mathbb{F}_2^n, \boldsymbol{k} \in \mathbb{F}_2^m.$$

Here, by $\deg(f)$ we mean the algebraic degree in $\boldsymbol{x}$. If $f(\boldsymbol{x}, \boldsymbol{k})$ can be written in the following form

$$f(\boldsymbol{x}, \boldsymbol{k}) = \varepsilon(\boldsymbol{x}, \boldsymbol{k}) + \sum_{t=0}^{l-1} p^{(t)}(\boldsymbol{x}, \boldsymbol{k}) q^{(t)}(\boldsymbol{x}, \boldsymbol{k})$$

such that $\deg(p^{(t)}) \leq n/2, \deg(q^{(t)}) \leq n/2$ and $\deg(\varepsilon) < n$ where n is even, then for $I = \{0, \cdots, n-1\}$,

$$\mathrm{Coe}_f(\boldsymbol{x}^I) = \mathrm{Coe}_{\sum_{t=0}^{l-1} p^{(t)} q^{(t)}}(\boldsymbol{x}^I) = \sum_{t=0}^{l-1} \mathrm{Coe}_{p^{(t)} q^{(t)}}(\boldsymbol{x}^I), \tag{4}$$

where we have

$$\begin{aligned} \mathrm{Coe}_{p^{(t)} q^{(t)}}(\boldsymbol{x}^I) &= \mathrm{Coe}_{\hbar_{\frac{n}{2}}(p^{(t)}) \hbar_{\frac{n}{2}}(q^{(t)})}(\boldsymbol{x}^I) \\ &= \sum_{J \subseteq I, |J| = \frac{n}{2}} \mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J) \mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^{I-J}). \end{aligned} \tag{5}$$

To summarize, ultimately, we only need to multiply the degree-$\frac{n}{2}$ parts of $p^{(t)}$ and $q^{(t)}$ for $0 \leq t < l$ to determine the coefficient of $\boldsymbol{x}^I$ in f. Putting it another way, the knowledge of all the coefficients of degree-$\frac{n}{2}$ terms is enough to compute $\mathrm{Coe}_f(\boldsymbol{x}^I)$. Combining Equation (4) and (5) gives

$$\mathrm{Coe}_f(\boldsymbol{x}^I) = \sum_{t=0}^{l-1} \sum_{J\subseteq I, |J|=\frac{n}{2}} \mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^{I-J}). \tag{6}$$

Assuming we have the oracle access to the keyed Boolean functions $p^{(t)}(\boldsymbol{x},\boldsymbol{k})$ and $q^{(t)}(\boldsymbol{x},\boldsymbol{k})$ for $0 \le t < l$, we can recover the algebraic normal form of $\mathrm{Coe}_f(\boldsymbol{x}^I)$ in two steps as follows. First, we compute the algebraic normal forms of $\mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^J)$ for all $J \subseteq I$ with $|J|= \frac{n}{2}$. This step is equivalent to recover the degree-$\frac{n}{2}$ homogeneous parts of $p^{(t)}(\boldsymbol{x},\boldsymbol{k})$ and $q^{(t)}(\boldsymbol{x},\boldsymbol{k})$.

Lemma 5. *Given an oracle access to the Boolean function $p(\boldsymbol{x},\boldsymbol{k}) = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k})\boldsymbol{x}^{\boldsymbol{u}}$ with $\boldsymbol{x} \in \mathbb{F}_2^n$ and $\boldsymbol{k} \in \mathbb{F}_2^m$, the coefficients of all degree-d terms of $p(\boldsymbol{x},\boldsymbol{k})$ can be recovered with $\binom{n}{d} \cdot 2^{m+d}$ evaluations of p.*

Proof. It follows from Lemma 3 and the fact that there are at most $\binom{n}{d}$ terms in $\hbar_d(f)$. □

According to Lemma 5, the complexity for recovering all the coefficients of degree-$\frac{n}{2}$ terms of $p^{(t)}(\boldsymbol{x},\boldsymbol{k})$ for some $t \in \{0,\cdots,l-1\}$ is about $\binom{n}{n/2} \cdot 2^{m+n/2}$ evaluations of $p^{(t)}$. Moreover, if we know that $p^{(t)}(\boldsymbol{x},\boldsymbol{k})$ involves only $m_t < m$ bits of $\boldsymbol{k}$, the time complexity can be reduced to $\binom{n}{n/2} \cdot 2^{m_t+n/2}$ evaluations of $p^{(t)}$. Similarly, we can recover the degree-$\frac{n}{2}$ homogeneous part of $q^{(t)}$.

Next, with the knowledge of all the coefficients of degree-$\frac{n}{2}$ terms of $p^{(t)}(\boldsymbol{x},\boldsymbol{k})$ and $q^{(t)}(\boldsymbol{x},\boldsymbol{k})$, we show how to compute

$$\sum_{J\subseteq I, |J|=\frac{n}{2}} \mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^{I-J}).$$

For each $J \subseteq I$ with $|J|= n/2$, if we know that $\mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^J)$ involves only $m_t < m$ bits of $\boldsymbol{k}$ in total and the set of key bits is $\{k_{i_0},\cdots,k_{i_{m_t-1}}\}$, then we can represent the algebraic normal form of $\mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)$ (or $\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^J)$) as a 2^{m_t}-bit sequence $\boldsymbol{\nu} = (\nu_0,\cdots,\nu_{2^{m_t}-1}) \in \mathbb{F}_2^{m_t}$ such that $\nu_i = 1$ if and only if the monomial $\prod_{j=0}^{m_t-1} k_{i_j}^{\mathtt{bin}_{m_t}(i)[j]}$ appears in $\mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)$, where $\mathtt{bin}_{m_t}(i)[j]$ denote the j-th bit of the m_t-bit binary representation of the integer i. We then store $\mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)$ as a 2^{m_t}-bit string described as above into the hash table $\mathbb{T}_{p^{(i)}}$ at address $\mathtt{addr}(\boldsymbol{x}^J) = (\mu_0,\cdots,\mu_{n-1}) \in \mathbb{F}_2^n$ with $\mu_i = 1$ if and only if $i \in J$. After processing all possible $J \subseteq I$ with $|J|= \frac{n}{2}$, $\mathbb{T}_{p^{(t)}}$ contains all coefficients of the degree-$\frac{n}{2}$ terms of $p^{(t)}$, which requires $\binom{n}{n/2} \cdot 2^{m_t}$ bits of memory. Similarly, we can construct the hash table $\mathbb{T}_{q^{(t)}}$ for the coefficients of all degree-$\frac{n}{2}$ terms of $q^{(t)}$. Then $\sum_{J\subseteq I,|J|=\frac{n}{2}} \mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^{I-J})$ can be computed with a complexity about $\binom{n}{n/2}2^{2m_t}$ memory accesses. In summary, to compute the coefficient or the superpoly of $\boldsymbol{x}^I$ in f:

$$\mathrm{Coe}_f(\boldsymbol{x}^I) = \sum_{t=0}^{l-1} \sum_{J\subseteq I, |J|=\frac{n}{2}} \mathrm{Coe}_{p^{(t)}}(\boldsymbol{x}^J)\mathrm{Coe}_{q^{(t)}}(\boldsymbol{x}^{I-J}),$$

we require $2\sum_{t=0}^{l-1}\binom{n}{n/2} \cdot 2^{m_t}$ bits of memory, $\sum_{t=0}^{l-1}\binom{n}{n/2}2^{2m_t}$ memory accesses, and $\binom{n}{n/2} \cdot 2^{m_t+n/2}$ evaluations of $p^{(t)}$ and $q^{(t)}$ for $0 \le t < l$.

Note that in practice the evaluation of $p^{(t)}$ and $q^{(t)}$ for many different t's may be executed in parallel through an oracle access to a vectorial Boolean function with $p^{(t)}$ and $q^{(t)}$ as its coordinate functions. Therefore, the complexity for this part should be analyzed on a case by case basis.

5 Key-Recovery Attack on 7-Round Ascon

In this section, we explain our key-recovery attack procedure on 7-round Ascon. Our attack is divided into two phases: an *offline phase* where we recover the superpolies of a 64-degree cube based on the partial polynomial multiplication technique and an *online phase* where we recover the secret key. We first describe the configuration of the initial state and some core observations related to our attack and then give details of the offline and online phases.

5.1 Initial State Configuration

We start with the initial state as depicted in Figure 8, where the 64 cube variables are set in X_4^0 and shown in green, and X_0^0 is filled with the constant $IV =$ `0x80400c0600000000`. The key bits $(k_0, \cdots, k_{63})$ and $(k_{64}, \cdots, k_{127})$ are loaded in X_1^0 and X_2^0, respectively. Throughout this section, X_3^0 is fixed to an arbitrary constant and here we set it to $(0, \cdots, 0)$ for the sake of simplicity. In this setting, the key bits are treated as symbolic secret *constants* and the 64 bits of X_4^0 are treated as Boolean variables $\boldsymbol{x} = (x_0, \cdots, x_{63})$. With

1	0	0	0	0	0	0	0	0	1	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_0^0
k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	$\cdots$	k_{50}	k_{51}	k_{52}	k_{53}	k_{54}	k_{55}	k_{56}	k_{57}	k_{58}	k_{59}	k_{60}	k_{61}	k_{62}	k_{63}	X_1^0
k_{64}	k_{65}	k_{66}	k_{67}	k_{68}	k_{69}	k_{70}	k_{71}	k_{72}	k_{73}	k_{74}	k_{75}	k_{76}	k_{77}	$\cdots$	k_{114}	k_{115}	k_{116}	k_{117}	k_{118}	k_{119}	k_{120}	k_{121}	k_{122}	k_{123}	k_{124}	k_{125}	k_{126}	k_{127}	X_2^0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_3^0
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	$\cdots$	x_{50}	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}	x_{60}	x_{61}	x_{62}	x_{63}	X_4^0

Figure 8: Initial state with cube variables in X_4^0

this configuration, every state bit $X_i^r[j]$ (also denoted as $X^r[64i + j]$ hereafter) can be regarded as a keyed Boolean function $f(\boldsymbol{x}, \boldsymbol{k})$ whose algebraic normal form is

$$f(\boldsymbol{x}, \boldsymbol{k}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}(\boldsymbol{k}) \boldsymbol{x}^{\boldsymbol{u}}, \boldsymbol{x} \in \mathbb{F}_2^{64}, \boldsymbol{k} \in \mathbb{F}_2^{128},$$

where the coefficient of $\boldsymbol{x}^{\boldsymbol{u}}$ is symbolically represented as a Boolean function $a_{\boldsymbol{u}} : \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2$.

We now give the core observations based on which our superpoly recovery is performed. Note that some observations have been used in [DEMS15], but for the convenience of reference, we still state them as lemmas.

Lemma 6. *For any $r \in \{1, \cdots, 7\}$, $i \in \{0, \cdots, 4\}$, and $j \in \{0, \cdots, 63\}$, the degree of $X_i^r[j]$ is upper bounded by 2^{r-1}.*

Proof. The round function is quadratic and $X_i^1[j]$ is affine with respect to the initial configuration given in Figure 8. □

Lemma 7 (Adapted from [DEMS15])**.** *For $1 \leq r \leq 7$ and $I = \{i_0, i_1, \ldots, i_{2^{r-1}-1}\} \subseteq \{0, 1, \ldots, 63\}$, the coefficient of the monomial $\boldsymbol{x}^I = \prod_{i \in I} x_i$ in $X^r[i]$ for any $i \in \{0, \cdots, 319\}$ can be fully determined by the 2^{r-1} key bits in $\{k_{i_0}, \cdots, k_{i_{2^{r-1}-1}}\}$.*

Proof. We prove it by induction on r. When $r = 1$, $I = \{i_0\}, 0 \leq i_0 < 64$. According to Property 1 and Lemma 6, the coefficient of $\boldsymbol{x}^I = x_{i_0}$ in the polynomial $X^1[i], 0 \leq i < 320$, is either k_{i_0} or $k_{i_0} + 1$. Assuming that this lemma holds for $r = l < 7$, we are going to show that it also holds for $r = l + 1$.

For $I = \{i_0, i_1, \ldots, i_{2^{l+1}-1}\}$, we consider the coefficient of monomial $x_{i_0} \cdots x_{i_{2^{l+1}-1}}$ in $X^{l+1}[j]$. $X^{l+1}[j]$ can be expressed as a quadratic function g of the bits of X^l. Let

$$\mathcal{D} = \{(i, j) : 0 \leq i < j < 320 \text{ and } X^l[i]X^l[j] \text{ is a term of } g\}.$$

Since $\deg(X^l[s]) \leq 2^l$ for $0 \leq s < 320$, and $\deg(x_{i_0} \cdots x_{i_{2^{l+1}-1}}) = 2^{l+1}$, we have

$$\mathrm{Coe}_{X^{l+1}[j]}(x_{i_0} \cdots x_{i_{2^{l+1}-1}}) = \sum_{(s,t)\in\mathcal{D}} \sum_{J\subseteq I} \mathrm{Coe}_{X^l[s]}(x^J)\mathrm{Coe}_{X^l[t]}(x^{I-J}).$$

According to the induction hypothesis, $\mathrm{Coe}_{X^s[t]}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{X^s[t']}(\boldsymbol{x}^{I-J})$ can be fully determined by $\{k_j : j \in J\}$ and $\{k_j : j \in I-J\}$ respectively. Therefore, $\mathrm{Coe}_{X^{l+1}[j]}(x_{i_0} \cdots x_{i_{2^{l+1}-1}})$ is a function of variables in $\{k_i : i \in J \cup (I-J) = I\}$. □

5.2 Offline Phase: Superpoly Recovery

Before going any further, we emphasize that this process is completely offline and is done once for all (secret keys). Let $I = \{0, 1, \ldots, 63\}$ then $\boldsymbol{x}^I = \prod_{i=0}^{63} x_i$. We are going to recover the superpolies of the cube term $\boldsymbol{x}^I$ for $Y^6[i]$ (recall that we can ignore Σ_0 at the 7-th round of Ascon). For the sake of concreteness, we present the detailed procedure for recovering the superpoly of x^I for $X^6[0]$, which is equally applicable to $X^6[i]$ for all $i \in \{0, \cdots, 319\}$. As Lemma 4 shows, if we choose $\boldsymbol{x}^I$ as the cube term, then $\mathrm{Coe}_{X^6[0]}(\boldsymbol{x}^I)$ is just the superpoly of the cube term $\boldsymbol{x}^I$. To recover the algebraic normal form of $\mathrm{Coe}_{X^6[0]}(\boldsymbol{x}^I)$, we apply the method presented in Section 4 to the following equation derived from the algebraic normal forms of the Ascon Sbox given in Equation 2:

$$Y_0^6[0] = X_4^6[0]X_1^6[0] + X_3^6[0] + X_2^6[0]X_1^6[0] + X_2^6[0] + X_0^6[0]X_1^6[0] + X_1^6[0] + X_0^6[0]. \quad (7)$$

Since $\deg(Y_0^6[0]) \leq 64$ and $\deg(X_4^6[0] + X_2^6[0] + X_0^6[0]) \leq 32$ due to Lemma 6, applying Equation 6 to Equation 7 gives

$$\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I) = \alpha + \beta + \gamma, \quad (8)$$

where

$$\begin{cases} \alpha = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X_4^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J}) = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X^6[256]}(\boldsymbol{x}^J)\mathrm{Coe}_{X^6[64]}(\boldsymbol{x}^{I-J}) \\ \beta = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X_2^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J}) = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X^6[128]}(\boldsymbol{x}^J)\mathrm{Coe}_{X^6[64]}(\boldsymbol{x}^{I-J}) \\ \gamma = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X_0^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J}) = \sum\limits_{J\subseteq I, |J|=32} \mathrm{Coe}_{X^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X^6[64]}(\boldsymbol{x}^{I-J}) \end{cases}. \quad (9)$$

Therefore, to recover the algebraic normal form of $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$ (regarded as a Boolean function with variables k_i, $0 \leq i < 64$), we need to recover the algebraic normal forms of α, β, and γ, which in turn can be derived from the algebraic normal forms of the coefficients of all degree-32 terms of $X_i^6[j]$.

Step 1: Computing the ANFs of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. For a given $J = \{i_0, \cdots, i_{31}\} \subseteq I = \{0, \cdots, 63\}$, Lemma 7 tells us that the coefficient of $\boldsymbol{x}^J$ in $X^6[j]$ is a Boolean function in variables $\{k_{i_0}, \cdots, k_{i_{31}}\}$ rather than $\{k_0, ..., k_{63}, k_{64}, \cdots, k_{127}\}$.

The truth table of this Boolean function can be obtained after $2^{32} \times 2^{32} = 2^{64}$ evaluations of the 6-round Ascon permutation, where for each possible value of $(k_{i_0}, \cdots, k_{i_{31}}) \in \mathbb{F}_2^{32}$, we evaluate the coefficient value of $\boldsymbol{x}^I$ based on Lemma 1. Since one evaluation of the 6-round Ascon permutation gives $X^6[j]$ for all $j \in \{0, \cdots, 319\}$, after $2^{32} \times 2^{32} = 2^{64}$ evaluations of the 6-round Ascon permutation we get the 320 truth tables for the coefficients of $\boldsymbol{x}^J$ in $\{X^6[j] : 0 \leq j < 320\}$.

By applying the fast Möbius transform given in Lemma 2 to the 320 truth tables, we obtain the algebraic normal forms of $\mathrm{Coe}_{X^6[j]}(\boldsymbol{x}^J)$ for all $j \in \{0, \cdots, 319\}$ with about $320 \times 32 \times 2^{32}$ XOR operations. In summary, the time complexity of recovering the algebraic normal forms of $\mathrm{Coe}_{X^6[j]}(\boldsymbol{x}^J)$ for all $j \in \{0, \cdots, 319\}$ is dominated by 2^{64} evaluations of the 6-round Ascon permutation. Since there are totally $\binom{64}{32} \approx 2^{60.7}$ different $J \subseteq I$ with $|J|= 32$, it takes $2^{60.7+64} \approx 2^{124.7}$ calls to the 6-round Ascon permutation to obtain the algebraic normal forms of $\mathrm{Coe}_{X^6[j]}(\boldsymbol{x}^J)$ for all possible $J \subseteq I$ with $|J|= 32$ and all $j \in \{0, \cdots, 319\}$. Then we store each $\mathrm{Coe}_{X^6[j]}(\boldsymbol{x}^J)$ as a 2^{32}-bit string into a table $\mathbb{T}_j$ at address $\mathtt{addr}(\boldsymbol{x}^J) \in \mathbb{F}_2^{64}$ as described in Section 4. Finally, we obtain 320 tables $\mathbb{T}_0, \cdots, \mathbb{T}_{319}$ which requires about $320 \times 2^{60.7} \times 2^{32} = 2^{101}$ bits of memory.[4] Using these tables, α, β, γ, and thus $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$ can be computed with a complexity of $3\times\binom{64}{32}\times 2^{2\times 32} \approx 2^{126.3}$ memory accesses according to the analysis of Section 4. For 64 superpolies, the complexity would be $2^{132.3}$ memory accesses. In Section 6.1, we will further show several techniques to reduce this complexity to $2^{123.28}$ 7-round Ascon.

Step 2: Generating the comparison tables for key candidates. With the 64 recovered superpolies $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I), \mathrm{Coe}_{Y_0^6[1]}(\boldsymbol{x}^I), \ldots, \mathrm{Coe}_{Y_0^6[63]}(\boldsymbol{x}^I)$, we can define a vectorial Boolean function $F : \mathbb{F}_2^{64} \to \mathbb{F}_2^{64}$ mapping $(k_0, k_1, \ldots, k_{63})$ to $(\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I), \ldots, \mathrm{Coe}_{Y_0^6[63]}(\boldsymbol{x}^I))$. Then, we store each $(k_0, k_1, \ldots, k_{63}) \in \mathbb{F}_2^{64}$ into a hash table $\mathbb{H}$ at address $F(k_0, k_1, \ldots, k_{63})$, which requires about $2^{64} \times 64 = 2^{70}$ bits of memory.

5.3 Online Phase: Key Recovery

For a cube set $\{\boldsymbol{x} = (x_0, x_1, \ldots, x_{63}) \in \mathbb{F}_2^{64}\}$ set as Figure 8, we choose one random 64-bit plaintext P, call the Ascon to encrypt P and obtain the corresponding C. Then the first 64-bit output of 7-round Ascon can be evaluated as $P + C$. Summing all $P + C$ under all $\boldsymbol{x} \in \mathbb{F}_2^{64}$, we get the 64-bit cube sum, denoted as $(z_0, z_1, \ldots, z_{63})$. Then the key candidates are just obtained from $\mathbb{H}[(z_0, z_1, \ldots, z_{63})]$. On the average only one key candidate is suggested. The complexity of this step is 2^{64} queries of Ascon. The remaining 64-bit key $(k_{64}, k_{65}, \ldots, k_{127})$ can be obtained by an exhaustive search, which requires another 2^{64} queries. The total complexity in online phase is then 2^{65} 7-round Ascon permutations queries.

6 Improved Key-Recovery Attacks

In this section, we present some techniques (specific to Ascon only) which can reduce the number of memory accesses and give the complexity analysis of key-recovery in 7-round Ascon permutations.

6.1 Techniques for Improving the Superpoly-recovery Complexity

Combine the Similar Monomials. Our first technique is based on combining the common terms in the degree-2 homogeneous part of $Y_0^6[j]$. Recall Equations 8 and 9, where we have computed α, β and γ separately. However, we can rearrange $X_4^6[0]X_1^6[0] + X_2^6[0]X_1^6[0] + X_0^6[0]X_1^6[0]$ as $X_1^6[0]\left(X_4^6[0] + X_2^6[0] + X_0^6[2]\right)$, which simplifies Equation 8 as

$$\begin{aligned}\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I) &= \mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0])X_0^6[0]}(\boldsymbol{x}^I) \\ &= \sum_{J\subset I, |J|=32} \mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0])}(\boldsymbol{x}^{I-J}).\end{aligned} \tag{10}$$

This reduces the time complexity of recovering all 64 superpolies by a factor of 3.

[4]The actual memory is $256 \times 2^{60.7} \times 2^{32} = 2^{100.7}$ as the quadratic term involving $X_3^6[j]$ does not appear in $Y_0^6[j]$ for all $j \in \{0, \cdots, 63\}$. This means we do not need to compute the tables $\mathbb{T}_{192}, \cdots, \mathbb{T}_{255}$.

Choose New Initial State. Our second technique utilizes Property 2 of the Sbox (related to X_3^0 rather than X_4^0) and algebraic degree bounds of the superpolies. The new initial state is depicted in Figure 9 where we regard $X_3^0[i], 0 \leq i < 64$ as the cube variables and set X_4^0 as the zero constant. From Property 2, the following lemma could be deduced (previously used in [DEMS15] to attack 5- and 6-round Ascon). We state it here for completeness.

1	0	0	0	0	0	0	0	0	1	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_0^0
k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	$\cdots$	k_{50}	k_{51}	k_{52}	k_{53}	k_{54}	k_{55}	k_{56}	k_{57}	k_{58}	k_{59}	k_{60}	k_{61}	k_{62}	k_{63}	X_1^0
k_{64}	k_{65}	k_{66}	k_{67}	k_{68}	k_{69}	k_{70}	k_{71}	k_{72}	k_{73}	k_{74}	k_{75}	k_{76}	k_{77}	$\cdots$	k_{114}	k_{115}	k_{116}	k_{117}	k_{118}	k_{119}	k_{120}	k_{121}	k_{122}	k_{123}	k_{124}	k_{125}	k_{126}	k_{127}	X_2^0
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	$\cdots$	x_{50}	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}	x_{60}	x_{61}	x_{62}	x_{63}	X_3^0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_4^0

Figure 9: Initial state with cube variables in X_3^0

Lemma 8. (Adapted from [DEMS15]) *For $1 \leq r \leq 7$ and $I = \{i_0, i_1, \dots, i_{2^{r-1}-1}\} \subseteq \{0, 1, \dots, 63\}$, the coefficient of the monomial $\boldsymbol{x}^I = \prod_{i \in I} x_i$ in $X^r[i]$ for any $i \in \{0, \cdots, 319\}$ can be fully determined by the 2^r equivalent key bits in $\{k_{i_0} + k_{i_0+64}, \cdots, k_{i_{2^{r-1}-1}} + k_{i_{2^{r-1}-1}+64}\}$.*

In the following, we always denote $\kappa_i = k_i + k_{i+64}$. Given $I = \{0, 1, \dots, 63\}$ and $\boldsymbol{x}^I = \prod_{j=0}^{63} x_j$, the $\mathrm{Coe}_{Y_0^6[i]}(\boldsymbol{x}^I)$ is a polynomial of $\{\kappa_0, \kappa_1, \dots, \kappa_{63}\}$ (follows from Lemma 8). According to Equation 10, we need to compute $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0])}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0])}(\boldsymbol{x}^{I-J})$ for certain $J \in I, |J| = 32$, each of which requires 2^{32} 6-round Ascon permutations. However, if the upper bound on the degree of $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ is d, then the number of monomials in $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ is at most $\sum_{i=0}^{d} \binom{32}{i} \leq 2^{32}$. Thus, we only need to consider the keys with Hamming weight at most d. The complexity of constructing the truth table for $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ then reduces to $2^{32} \cdot \sum_{i=0}^{d} \binom{32}{i}$. For $d < 32$, this complexity can be reduced.

To compute the values of d, we use the division property method similar to [WHT$^+$18]. Since $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ is a polynomial of $\{\kappa_j : j \in J\}$, it is not trivial to model $\kappa_j, i \in J$ into the MILP model. However, we can write each monomial in $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ as

$$\prod_{j \in J} \kappa_j = \prod_{j \in J} (k_j + k_{j+64}) = \sum_{J' \subseteq J} \left(\prod_{j' \in J'} \kappa_{j'} \prod_{j'' \in J - J'} \kappa_{j''+64} \right).$$

The above equation shows that the degree of $\prod_{j \in J} \kappa_j$ is equivalent to the degree of $\prod_{j' \in J'} \kappa_{j'} \prod_{j'' \in J-J'} \kappa_{j''+64}$. The later can be modeled easily in MILP. The upper bound returned by the division property algorithm on the degrees of $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ for all $J, |J| = 32$ and $0 \leq i < 320$ is 15. Thus, the time complexity to compute $\mathrm{Coe}_{X^6[i]}(\boldsymbol{x}^J)$ is given by $\sum_{i=0}^{15} \binom{32}{i} \cdot 2^{32} \approx 2^{30.78+32} = 2^{62.78}$. Accordingly, the complexity of computing the degree-32 terms of $X^6[i], 0 \leq i < 320$ is then reduced to $2^{60.7+62.78} = 2^{123.48}$ 6-round Ascon permutations.

In the superpoly recovery of $Y_0^6[0]$ (Equation 10), we have assumed that there are 2^{32} monomials in both $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J})$ for each J. As a result, the complexity of computing $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J})$ is roughly estimated as $2^{32+32} = 2^{64}$ memory accesses. Now we know that both of them have at most $\sum_{i=0}^{15} \binom{32}{i} \approx 2^{30.78}$ monomials. Hence, the time complexity is reduced to at most $2^{61.56}$. Finally, the time complexity of computing Equation 10 is reduced to $2^{122.26}$ memory accesses. In other words, it takes $2^{122.26}$ memory accesses to recover one superpoly.

Computing 64 Superpolies in a Parallel Fashion. For all 64 superpolies $\mathrm{Coe}_{Y_0^6[i]}(x^I), 0 \leq i < 64$, the complexity would be $2^{128.26}$ memory accesses if we recover them one by one. However, we can recover the 64 superpolies in a parallel way. For each J in Equation 10, we can represent $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{X_1^6[0])}(\boldsymbol{x}^{I-J})$ in two bit strings of length $2^{30.78}$, denoted by S and S', respectively. For $0 \leq i < 2^{30.78}$, $S[i]$ and $S'[i]$ are the coefficients (0 or 1) of the monomials in $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{X_1^6[0])}(\boldsymbol{x}^{I-J})$, respectively, which are ordered by the lexicographic order of the monomials. Then the coefficient of the corresponding monomials of $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)\mathrm{Coe}_{X_1^6[0])}(\boldsymbol{x}^{I-J})$ can be computed as $S[i] \cdot S[j]$ for all $0 \leq i, j < 2^{30.78}$. For the 64 superpolies, we can process 64 different pairs of (S, S') at the same time by arranging 64 different S's and S''s in the bit-slicing fashion. Then recovering the 64 superpolies still takes $2^{122.26}$ memory accesses.

The online phase differs slightly now as we set cube variables in X_3^0 instead of X_4^0. After the offline phase, we recover 64-bit equivalent keys $\{\kappa_0, \kappa_1, \ldots, \kappa_{63}\}$. To recover the remaining 64-bit key information, we do an exhaustive search over the 64-bit key space $\{k_0, k_1, \ldots, k_{63}\}$. For each guess of $\{k_0, k_1, \ldots, k_{63}\}$, we first compute $k_{64+i} = k_i + \kappa_i$ for $i \in \{0, 1, \ldots, 63\}$ and then determine the right key by testing a plaintext and ciphertext pair.

Discussion about the Recovered Superpolies. Since the superpolies are too large to be practically recover, our attacks are actually based on an assumption that the superpolies for 7-round Ascon are (almost) balanced Boolean functions. Such an assumption is common and also used in the conditional cube attacks on 7-round Ascon [LDW17].

If the superpolies are highly biased, then we cannot extract 64 keys in the online phase because many different keys will lead to the same cube sum. To reflect some behaviors of the 7-round superpolies, we tested 161827 superpolies for 5-round Ascon with randomly selected 16-dimensional cubes and computed the truth tables of the superpolies, among 2^{16} different key values (According to Lemma 7, only 16 key bits are involved) there are averagely 32591 entries with value 1 and 32945 entries with value 0. In other words, many superpolies for 5-round Ascon are almost balanced functions. So we expect the 7-round superpolies to have the same behavior.

In an extreme case, the superpolies may be constant then our key-recovery attack will degenerate to a distinguisher. There are some key-recovery attacks (based on superpolies recovery) on stream ciphers like Trivium [CP08], Grain-128a [ÅHJM11] and Acorn [Wu16b] that have proven to be distinguishing attacks only [YT19, HLM+20]. However, for Trivium and Acorn, the degeneration happens as the degree upper bound of recovered superpoly is 1. For the degeneration case of Grain-128a, the degree upper bound is 14 and at most 21 key bits are involved in the supperpoly. In case of 7-round Ascon, the degrees are upper bounded by 30, and 64 equivalent key bits are involved. Also, note that currently all degradations are observed in NLFSR-based stream ciphers. For SPN ciphers with much stronger diffusion and confusion, intuitively the risk of degeneration should be low.

6.2 Evaluation of the Attack Complexities

Time Complexity. Firstly, we analyze the complexity in the offline phase, which is dominated by the step 1 obviously. With the techniques in Section 6.1, the complexities of step 1 are reduced to $2^{123.48}$ 6-round Ascon permutations for computing degree-32 part of $X^6[i], 0 \leq i < 320$ and $2^{122.26}$ memory accesses for the multiplication of the partial polynomials, respectively.

But the final complexity should be evaluated in 7-round Ascon permutations. One 6-round Ascon permutation is about $\frac{6}{7} \approx 2^{-0.2}$ 7-round Ascon permutation. Then the complexity of computing degree-32 part of $X^6[i], 0 \leq i < 320$ is about $2^{123.28}$ 7-round Ascon permutations.

To estimate the complexity of the multiplication of the partial polynomial, we need to compute the scale factor between the memory access (denoted by N_{mem}) and the 7-round Ascon permutations (denoted by $\mathsf{T}_{\text{Ascon}}$). Therefore, to compute $\mathsf{T}_{\text{Ascon}}$ from N_{mem}, we define the scale factor η, satisfying $\mathsf{T}_{\text{Ascon}} \approx \eta \times \mathsf{N}_{\text{mem}}$.

In a conventional method, we can regard an Sbox operation as one memory access and ignore the cost of the linear layer. Thus one round Ascon equals approximately 64 memory accesses and 7-round Ascon equals $64 \times 7 \approx 2^{8.8}$ memory accesses, i.e., $\eta = 2^{-8.8}$. Then the complexity of computing Equation 10 phase is equivalent to about $2^{122.26-8.8} = 2^{113.46}$ 7-round Ascon permutations.

Since that Ascon is designed for bit-sliced implementation, we still use another scale for the transformation. Note that the bitsliced implementation of Ascon has eleven 64-bit XORs for the Sbox layer (Figure 4) and ten 64-bit XORs for the linear layer (Equation 3). Thus, there are $7\times(11+10) \approx 2^{7.2}$ 64-bit XORs in total for 7 rounds. We ignore the ANDs and NOTs because the XORs are heavier in general. Since all memory operations in our attack are 64-bit vector, we assume one memory access equals one 64-bit XOR operation. Accordingly, we have $\eta = 2^{-7.2}$. Then the complexity of computing Equation 10 in the offline phase is about $2^{115.06}$ 7-round Ascon permutations. At last, comparison between different operations is always a difficult task. We can also compare a memory access to one single encryption, then the time complexity for computing Equation 10 is about $2^{122.26}$ encryptions. Note the complexity of the multiplication is considered in the worst case where we always assume that the monomials in $\mathrm{Coe}_{X_4^6[0]+X_2^6[0]+X_0^6[0]}(\boldsymbol{x}^J)$ and $\mathrm{Coe}_{X_1^6[0]}(\boldsymbol{x}^{I-J})$ will appear if we cannot make sure that they do not appear.

Overall, the time complexity in the offline phase is dominated by computing the degree-32 part of $X^6[i], 0 \leq i < 320$, i.e., $2^{123.28}$ 7-round Ascon permutations.

The complexity in the online phase is 2^{65} which consists of one evaluation of the cube sum and the exhaustive search on 64 key bits. In the end, the overall time complexity is dominated by $2^{123.28}$ 7-round Ascon permutations.

Memory Complexity. The memory complexity is dominated by the 320 tables $\mathbb{T}_0, \mathbb{T}_1, \ldots, \mathbb{T}_{319}$. Each $\mathbb{T}_i, 0 \leq i < 320$ contains $2^{60.7}$ 2^{32}-bit strings, so the memory complexity is about $320 \times 2^{60.7} \times 2^{32} \approx 2^{101}$ bits.

Remark. In this paper, we regard one memory access to a big table as one 64-bit XOR operation, which is sometimes controversial. However, even we compare a memory access to one single encryption, the time complexity is still less than exhaustive search, though marginal. We hope the technique of partial polynomial multiplication can inspire further improvements.

7 Distinguishers for Round-reduced Ascon Based on Divison Property

In this section, we present several distinguishers on round-reduced Ascon by exploiting the properties of Sbox and using the three-subset bit-based division property (3SBDP) [HLM+20]. We first give an efficient MILP model for the 3SBDP propagation rules of Ascon by adopting the arithmetic circuit approach. Next, we use this model to find cubes whose superpolies are constant zero.

7.1 Efficient MILP Modeling of Ascon

Let $x, y_1, y_2, \cdots, y_n$ be binary variables. The 3SBDP propagation of a cipher can be modeled with three basic operations, namely bitwise COPY, bitwise AND and bitwise XOR [HLM+20]. To model these operations in MILP, the following linear inequalities are sufficient.

- $x \xrightarrow{\text{COPY}} (y_1, \cdots, y_n) : x \geq y_i$ for $1 \leq i \leq n$, and $y_1 + y_2 + \cdots + y_n \geq x$
- $(y_1, \cdots, y_n) \xrightarrow{\text{AND}} x : x = y_i$ for $1 \leq i \leq n$
- $(y_1, \cdots, y_n) \xrightarrow{\text{XOR}} x : x = y_1 + y_2 + \cdots + y_n$

In Ascon, the state is initially loaded with an IV which has certain bits set as constant 1. Further, the constant 1 is XORed to part of state via round constant bits. Hence, to model the division trails of *XOR with constant 1*, we propose a new propagation rule in Proposition 1.

Proposition 1 (MILP model for XOR+1)**.** *Let x, y be binary variables and $x \xrightarrow{XOR+1} y$ be the three-subset division trail of $y = x + 1$. Then the following inequality is sufficient to describe the propagation of $y = x + 1$.*

$$y \geq x.$$

We now proceed to model the 3SBDP of Ascon using the aforementioned rules. Algorithm 1 describes the MILP model for Ascon reduced to r rounds. Below, we explain the individual components of Algorithm 1 and give explicit linear inequalities in Appendix A.

Modeling Sboxes. Ascon utilizes the same Sbox throughout multiple rounds. However, to have an accurate and efficient propagation of division trails, we model the exact vectorial Boolean functions in each round. Let $x_0, \cdots, x_4$ and $y_0, \cdots, y_4$ be binary variables. We denote the Sbox modeling by $\mathsf{SB}([x_0, x_1, x_2, x_3, x_4], [y_0, y_1, y_2, y_3, y_4])$, and the corresponding inequalities can be generated with the convex hull computation method [SHW+14]. Note that depending on $x_j = 0$ or 1, the Sbox is modeled accordingly. For instance, if $x_0 = 1$, then we model the 4-bit to 5-bit vectorial Boolean function given by $\mathsf{SB}([0, x_1, x_2, x_3, x_4], [y_0, y_1, y_2, y_3, y_4])$. This approach gives the flexibility to assign the constant 0 or 1 to a state variable which in turn allows the precise modeling. Lines 6, 9, 19, and 22 in Algorithm 1 depict the Addition of constants and Substitution layer. The exact modeling of $\mathsf{SB}(\cdot)$ function is provided in Algorithm 3.

Modeling the Linear Layer. The linear layer takes the entire state as an input and mix the 64-bit words by performing XOR operations (Equation 3). Thus, it can be simply modeled with COPY and XOR rules. This is denoted by $\mathsf{L}([y_0, \cdots, y_{319}], [x_0 \cdots, x_{319}])$ in Lines 13 and 26 of Algorithm 1. The exact modeling of $\mathsf{L}(\cdot)$ function is provided in Algorithm 4.

Verification. For verifying the correctness of our model, we computed the ANF of each state bit and matched with the output of Sage. The source codes are available at https://github.com/raghavrohit/ascon_cube_distinguishers in case reader wants to verify the models.

Algorithm 1: MILP model for 3SBDP of Ascon

```
1  Input: Rounds: $r$, $IV$, key variables: $k_0, \cdots, k_{127}$, nonce variables: $v_0, \cdots, v_{127}$
         and empty MILP model $\mathcal{M}$
   Output: MILP model $\mathcal{M}$
2  \\ Round 0
3  for $i = 0$ to 63 do
4      $\mathcal{M}.addVar \leftarrow y_i^0, y_{i+64}^0, y_{i+128}^0, y_{i+192}^0, y_{i+256}^0$
5      if $i \leq 55$ then
6          SB($\mathcal{M}, [IV[i], k_i, k_{i+64}, v_i, v_{i+64}], [y_i^0, y_{i+64}^0, y_{i+128}^0, y_{i+192}^0, y_{i+256}^0]$)   ▷ Sbox layer
7      else
8          SB($\mathcal{M}, [IV[i], k_i, rc_{i-56}^0 + k_{i+64}, v_i, v_{i+64}], [y_i^0, y_{i+64}^0, y_{i+128}^0, y_{i+192}^0, y_{i+256}^0]$)   ▷ Add constants and Sbox layer
9      end
10 end
11 $\mathcal{M}.addVar \leftarrow x_0^0 \cdots, x_{319}^0$
12 L($\mathcal{M}, [y_0^0, \cdots, y_{319}^0], [x_0^0, \cdots, x_{319}^0]$)   ▷ Linear layer
13 \\ Round 1 to $r-1$
14 for $j = 1$ to $r-1$ do
15     for $i = 0$ to 63 do
16         $\mathcal{M}.addVar \leftarrow y_i^j, y_{i+64}^j, y_{i+128}^j, y_{i+192}^j, y_{i+256}^j$
17         if $i \leq 55$ then
18             SB($\mathcal{M}, [x_i^{j-1}, x_{i+64}^{j-1}, x_{i+128}^{j-1}, x_{i+192}^{j-1}, x_{i+256}^{j-1}], [y_i^j, y_{i+64}^j, y_{i+128}^j, y_{i+192}^j, y_{i+256}^j]$)   ▷ Sbox layer
19         else
20             SB($\mathcal{M}, [x_i^{j-1}, x_{i+64}^{j-1}, rc_{i-56}^j + x_{i+128}^{j-1}, x_{i+192}^{j-1}, x_{i+256}^{j-1}], [y_i^j, y_{i+64}^j, y_{i+128}^j, y_{i+192}^j, y_{i+256}^j]$)   ▷ Add constants and Sbox layer
21         end
22     end
23     $\mathcal{M}.addVar \leftarrow x_0^j, \cdots, x_{319}^j$
24     L($\mathcal{M}, [y_0^j, \cdots, y_{319}^j], [x_0^j, \cdots, x_{319}^j]$)   ▷ Linear layer
25 end
```

7.2 Finding Good Cubes for Ascon

Our aim here is to find cubes with dimension less than 64 whose superpolies are constant zero. We restrict ourselves to at most 63 dimensional cubes as the prescribed data limit by designers for a single key is 2^{64}. The best known cubes that satisfy this limit can reach 4, 5 and 6-round Ascon and have dimensions 9, 17 and 33, respectively [DEMS15]. To the best of our knowledge, there are no distinguishers on 7-round Ascon. Thus, it is worth investigating whether there exists cubes which can distinguish the output of 7-round Ascon with data complexity less than 2^{64} encryptions.

To answer the above question, we first recall Property 3 of the Sbox. If we set $X_3^0 = X_4^0$ in Equation 2, then both Y_2^0 and Y_3^0 become independent of words X_3^0 and X_4^0. This means if we take $N_0 = N_1$, i.e., the nonce variables as $v_i = v_{i+64}$ for $i = 0, \cdots, 63$ and use them as cube variables, then after round 1, no cube variable v_i is present in words X_2^1 and X_3^1. In words X_0^1, X_1^1 and X_4^1, the cube variables are linear. Since the algebraic degree of round function is 2, the degree in cube variables is at most 64 after 7 rounds. The fact that two words after round 1 are independent of cube variables suggests that the algebraic degree in cube variables may be less than $4, 8, 16, 32$ and 64 after 3, 4, 5, 6 and 7 rounds, respectively.

Upper Bounds of Degree. To compute the upper bounds of algebraic degree in cube variables, we set $N_0 = N_1$ and model the 3SBDP of Ascon following Algorithm 1. We then compute the degree upper bound of each state bit using Algorithm 2.

```
Algorithm 2: MILP model for computing the upper bound on degree
   Input: Rounds: $r$, $IV$, key variables: $k_0, \cdots, k_{127}$, nonce variables: $v_0, \cdots, v_{127}$, bit
          position $target$ and empty MILP model $\mathcal{M}$
   Output: degree upper bound of $x_i^{r-1}$
 1 Set $v_i = v_{i+64}$ for $i = 0, \cdots, 63$                    ▷ Cube variables condition
 2 Model $r$-round Ascon using Algorithm 1
 3 for $i = 0$ to $319$ do
 4     if $i == target$ then
 5         $\mathcal{M}.addConstr(x_i^{r-1} = 1)$
 6     else
 7         $\mathcal{M}.addConstr(x_i^{r-1} = 0)$
 8     end
 9 end
10 $\mathcal{M}.setObjective(\max \sum_{i=0}^{63} v_i)$
11 return Objective value
```

In Table 3, we list the obtained upper bounds for each state words till 7 rounds. For 7-round, the upper bound is 59 which means the superpoly of any 60 dimensional cube (with $N_0 = N_1$) is constant zero. There exists $\binom{64}{60} \approx 2^{19.27}$ such cubes. Further, we can distinguish 4, 5 and 6-round with data $2^8, 2^{16}$ and 2^{31}, respectively, which improves the existing results.

Table 3: Upper bounds on the algebraic degree of Ascon in cube variables

Round r	Bits in word				
	X_0^r	X_1^r	X_2^r	X_3^r	X_4^r
2	2	1	1	2	2
3	3	3	4	4	3
4	7	8	7	7	6
5	15	15	13	14	15
6	30	29	29	30	30
7	59	59	60	60	58

Distinguisher for 4-round with complexity 2^5. Our previous distinguisher for 4-round requires 2^8 data. However, we find that there exists a set of six cube variables $\{v_i, v_{i+8}, v_{i+16}, v_{i+17}, v_{i+34}, v_{i+63}\}$[5] which do not multiply with each other after round 2. Choosing any 5 out of 6 gives a cube distinguisher with 32 nonces.

Experimental Verification. We have experimentally verified all our distinguishers for 4, 5 and 6 rounds using the Ascon reference C code. The codes are also publicly available at `https://github.com/raghavrohit/ascon_cube_distinguishers`.

[5]For $i \geq 1$ indices should be taken modulo 64.

8 Conclusion and Open Problem

In this work, we have presented the first cube-based key recovery attack on 7-round Ascon without violating the data limit per key specified by the designers. The main technique employed in this attack is the so-called partial polynomial multiplication, enabling the recovery of superpolies by considering simplified versions of the target Boolean functions. Our best attack can recover the 128-bit secret key with a time complexity of 2^{123} and requires 2^{64} data and 2^{101} bits memory. Moreover, based on division properties, we identified the first 7-round misuse-free cube distinguishers for Ascon and some 4-, 5-, and 6-round distinguishers with reduced complexities. All our results are equally applicable to Ascon-128a because of the same core permutation.

We believe that the partial polynomial multiplication technique can find applications in other contexts too. Furthermore, in our key-recovery attacks, we have assumed the worst case for the number of cube terms present in a state bit and their corresponding number of monomials in the superpoly. Any improvement in both or either of these two will reduce the time and memory complexities, and thus requires further investigation. Finally, is there any key-recovery attack using less than 2^{64} data? We hope to get the answer in future.

9 Acknowledgement

The authors would like to thank the reviewers of ToSC 2021 for providing us with insightful comments to improve the quality of the paper. Raghvendra Rohit was partially supported by the French Agence Nationale de la Recherche through the CryptAudit project under Contract ANR-17-CE39-0003. Siwei Sun and Kai Hu are supported by the National Key Research and Development Program of China (No. 2018YFA0704702, 2018YFA0704704), the Major Scientific and Technological Innovation Project of Shandong Province, China (No. 2019JZZY010133), the Chinese Major Program of National Cryptography Development Foundation (MMJJ20180102), and the National Natural Science Foundation of China (62032014, 61772519).

References

[ABD$^+$16] Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, and Kan Yasuda. Colm v1. *Candidate for the CAESAR Competition. See*, 2016. https://competitions.cr.yp.to/round3/colmv1.pdf.

[ÅHJM11] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.*, 5(1):48–59, 2011.

[BDKW19] Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. DLCT: A new tool for differential-linear cryptanalysis. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 313–342, 2019.

[CAE] CAESAR: Call for Submission. http://competitions.cr.yp.to/.

[Can16] Anne Canteaut. Lecture notes on cryptographic Boolean functions, 2016. https://www.rocq.inria.fr/secret/Anne.Canteaut/.

[Car10] Claude Carlet. Boolean functions for cryptography and error-correcting codes., 2010. https://www.researchgate.net/profile/Claude_Carlet/publication/228720083_Boolean_Functions_for_Cryptography_and_Error_Correcting_Codes/links/5405a6e10cf23d9765a71b9c.pdf.

[CP08] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[Dae12] Joan Daemen. Permutation-based Encryption, Authentication and Authenticated Encryption. DIAC 2012, 2012.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 490–509, 2015.

[DEMS] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon Resources. https://ascon.iaik.tugraz.at/resources.html. Accessed Nov 2020.

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 371–387, 2015.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. *Candidate for the CAESAR Competition. See also*, 2016. http://ascon.iaik.tugraz.at.

[DKM$^+$17] Ashutosh Dhar Dwivedi, Milos Kloucek, Pawel Morawiecki, Ivica Nikolic, Josef Pieprzyk, and Sebastian Wójtowicz. SAT-based cryptanalysis of authenticated ciphers from the CAESAR competition. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017*, pages 237–246, 2017.

[DMA17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 606–637, 2017.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 278–299, 2009.

[Ear] Early Symmetric Crypto workshop (ESC 2013). https://www.cryptolux.org/mediawiki-esc2013/index.php/ESC_2013.

[FLLW17] Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Reforgeability of authenticated encryption schemes. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, pages 19–37, 2017.

[GRW16] Faruk Göloglu, Vincent Rijmen, and Qingju Wang. On the division property of S-boxes. *IACR Cryptol. ePrint Arch.*, 2016:188, 2016.

[HLLT20] Phil Hebborn, Baptiste Lambin, Gregor Leander, and Yosuke Todo. Lower bounds on the degree of block ciphers. *IACR Cryptol. ePrint Arch.*, 2020:1051, 2020.

[HLM$^+$20] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128aead. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020.

[HSWW20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, pages 446–476, 2020.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in Sponge-based authenticated encryption modes. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 85–104, 2014.

[JNPS16] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41. *Candidate for the CAESAR Competition. See*, 2016. https://competitions.cr.yp.to/round3/deoxysv141.pdf.

[KR16] Ted Krovetz and Phillip Rogaway. Ocb (v1.1). *Candidate for the CAESAR Competition. See*, 2016. https://competitions.cr.yp.to/round3/ocbv11.pdf.

[LDW17] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional cube attack on round-reduced ASCON. *IACR Trans. Symmetric Cryptol.*, 2017(1):175–202, 2017.

[LTW18] Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. Searching for subspace trails and truncated differentials. *IACR Trans. Symmetric Cryptol.*, 2018(1):74–100, 2018.

[LZWW17] Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. Cryptanalysis of round-reduced ASCON. *Sci. China Inf. Sci.*, 60(3):38102, 2017.

[Nat19] National Institute of Standards and Technology. Lightweight Cryptography (LWC) Standardization project, 2019. https://csrc.nist.gov/projects/lightweight-cryptography.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107, 2002.

[Rog04] Phillip Rogaway. Nonce-based symmetric encryption. In *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, pages 348–359, 2004.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 373–390, 2006.

[SHW$^+$14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 158–178, 2014.

[SY15] Yu Sasaki and Kan Yasuda. How to incorporate associated data in Sponge-based authenticated encryption. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 353–370, 2015.

[Tez16] Cihangir Tezcan. Truncated, impossible, and improbable differential analysis of ASCON. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, pages 325–332, 2016.

[TIHM17] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 250–279. Springer, 2017.

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

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 287–314, 2015.

[VV18] Serge Vaudenay and Damian Vizár. Can CAESAR beat Galois? - robustness of CAESAR candidates against nonce reusing and high data complexity attacks. In *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, pages 476–494, 2018.

[WHG$^+$19] Senpeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. MILP-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 398–427. Springer, 2019.

[WHT$^+$18] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 275–305. Springer, 2018.

[WP16] Hongjun Wu and Bart Preneel. AEGIS: A fast authenticated encryption algorithm (v1.1). *Candidate for the CAESAR Competition. See*, 2016. https://competitions.cr.yp.to/round3/aegisv11.pdf.

[Wu16a] Hongjun Wu. ACORN: a lightweight authenticated cipher (v3). *Candidate for the CAESAR Competition. See*, 2016. https://competitions.cr.yp.to/round3/acornv3.pdf.

[Wu16b] Hongjun Wu. Acorn: A lightweight authenticated cipher (v3), 2016. https://competitions.cr.yp.to/round3/acornv3.pdf.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

[YLW$^+$19] Hailun Yan, Xuejia Lai, Lei Wang, Yu Yu, and Yiran Xing. New zero-sum distinguishers on full 24-round Keccak-f using the division property. *IET Inf. Secur.*, 13(5):469–478, 2019.

[YT19] Chen-Dong Ye and Tian Tian. Revisit division property based cube attacks: Key-recovery or distinguishing attacks? *IACR Trans. Symmetric Cryptol.*, 2019(3):81–102, 2019.

A MILP Models for Underlying Components of Ascon

In Algorithms 3 and 4, we give the exact inequalities for constructing the MILP model of Ascon's Sbox and linear layer, respectively.

Algorithm 3: MILP model for the SB function

```
Input: $\mathcal{M}$, $[x_0, \cdots, x_4], [y_0, \cdots, y_4]$
\\ Binary variables
$\mathcal{M}.addVar \leftarrow p_i$ for $0 \le i \le 6$
$\mathcal{M}.addVar \leftarrow q_i$ for $0 \le i \le 11$
$\mathcal{M}.addVar \leftarrow r_i$ for $0 \le i \le 6$
$\mathcal{M}.addVar \leftarrow s_i$ for $0 \le i \le 7$
$\mathcal{M}.addVar \leftarrow t_i$ for $0 \le i \le 7$
$\mathcal{M}.addVar \leftarrow a_i$ for $0 \le i \le 10$

\\ COPY operation
$\mathcal{M}.addConstr \leftarrow x_0 \ge p_i$, for $i = 0 \le i \le 6$ ; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{6} p_i \ge x_0$
$\mathcal{M}.addConstr \leftarrow x_1 \ge q_i$, for $i = 0 \le i \le 11$ ; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{11} q_i \ge x_1$
$\mathcal{M}.addConstr \leftarrow x_2 \ge r_i$, for $i = 0 \le i \le 6$ ; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{6} r_i \ge x_2$
$\mathcal{M}.addConstr \leftarrow x_3 \ge s_i$, for $i = 0 \le i \le 7$ ; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{7} s_i \ge x_3$
$\mathcal{M}.addConstr \leftarrow x_4 \ge t_i$, for $i = 0 \le i \le 7$ ; $\mathcal{M}.addConstr \leftarrow \sum_{i=0}^{7} t_i \ge x_4$

\\ AND and XOR operations
$\mathcal{M}.addConstr \leftarrow a_0 = t_0$ ; $\mathcal{M}.addConstr \leftarrow a_0 = q_0$
$\mathcal{M}.addConstr \leftarrow a_1 = r_0$ ; $\mathcal{M}.addConstr \leftarrow a_1 = q_1$
$\mathcal{M}.addConstr \leftarrow a_2 = p_0$ ; $\mathcal{M}.addConstr \leftarrow a_2 = q_2$
$\mathcal{M}.addConstr \leftarrow y_0 = a_0 + a_1 + a_2 + s_0 + r_1 + q_3 + p_1$

$\mathcal{M}.addConstr \leftarrow a_3 = s_1$ ; $\mathcal{M}.addConstr \leftarrow a_3 = r_2$
$\mathcal{M}.addConstr \leftarrow a_4 = s_2$ ; $\mathcal{M}.addConstr \leftarrow a_4 = q_4$
$\mathcal{M}.addConstr \leftarrow a_5 = r_3$ ; $\mathcal{M}.addConstr \leftarrow a_5 = q_5$
$\mathcal{M}.addConstr \leftarrow y_1 = a_3 + a_4 + a_5 + t_1 + s_3 + r_4 + q_6 + p_2$

$\mathcal{M}.addConstr \leftarrow a_6 = t_2$ ; $\mathcal{M}.addConstr \leftarrow a_6 = s_4$
$\mathcal{M}.addConstr \leftarrow y_2 \ge a_6 + t_3 + r_5 + q_7$

$\mathcal{M}.addConstr \leftarrow a_7 = t_4$ ; $\mathcal{M}.addConstr \leftarrow a_7 = p_3$
$\mathcal{M}.addConstr \leftarrow a_8 = s_5$ ; $\mathcal{M}.addConstr \leftarrow a_8 = p_4$
$\mathcal{M}.addConstr \leftarrow y_3 = a_7 + a_8 + t_5 + s_6 + q_8 + r_6 + p_5$

$\mathcal{M}.addConstr \leftarrow a_9 = t_6$ ; $\mathcal{M}.addConstr \leftarrow a_9 = q_9$
$\mathcal{M}.addConstr \leftarrow a_{10} = p_6$ ; $\mathcal{M}.addConstr \leftarrow a_{10} = q_{10}$
$\mathcal{M}.addConstr \leftarrow y_4 = a_9 + a_{10} + t_7 + s_7 + q_{11}$
return $\mathcal{M}$
```

Algorithm 4: MILP model for the L function

```
   Input: $\mathcal{M}$, $[y_0, \cdots, y_{319}], [x_0, \cdots, x_{319}]$
1  \\$\sigma$ operation on each 64-bit word
2  $\sigma(\mathcal{M}, [y_0, \cdots, y_{63}], [x_0, \cdots, x_{63}], 19, 28)$
3  $\sigma(\mathcal{M}, [y_{64}, \cdots, y_{127}], [x_{64}, \cdots, x_{127}], 61, 39)$
4  $\sigma(\mathcal{M}, [y_{128}, \cdots, y_{191}], [x_{128}, \cdots, x_{191}], 1, 6)$
5  $\sigma(\mathcal{M}, [y_{192}, \cdots, y_{255}], [x_{192}, \cdots, x_{255}], 10, 17)$
6  $\sigma(\mathcal{M}, [y_{256}, \cdots, y_{319}], [x_{256}, \cdots, x_{319}], 7, 41)$
7  return $\mathcal{M}$
```

Algorithm 5: MILP model for the σ function

```
1  Input: $\mathcal{M}$, $[y_0, \cdots, y_{63}], [x_0, \cdots, x_{63}], m, n$
2  \\ Binary variables
3  $\mathcal{M}.addVar \leftarrow a_i$ for $0 \leq i \leq 63$
4  $\mathcal{M}.addVar \leftarrow b_i$ for $0 \leq i \leq 63$
5  $\mathcal{M}.addVar \leftarrow c_i$ for $0 \leq i \leq 63$
6  \\ COPY operation
7  for $i = 0$ to 63 do
8      $\mathcal{M}.addConstr \leftarrow y_i \geq a_i$ ; $\mathcal{M}.addConstr \leftarrow y_i \geq b_i$ ; $\mathcal{M}.addConstr \leftarrow y_i \geq c_i$
9      $\mathcal{M}.addConstr \leftarrow a_i + b_i + c_i \geq y_i$
10 end
11 \\ XOR operation
12 for $i = 0$ to 63 do
13     $\mathcal{M}.addConstr \leftarrow x_i = a_i + b_{(64-m+i) \bmod 64} + c_{(64-n+i) \bmod 64}$
14 end
15 return $\mathcal{M}$
```