Improved Key Recovery Attacks of Ascon

Shuo Peng[1,2,3], Kai Hu[2,1,3(✉)], Jiahui He[1,3], and Meiqin Wang[2,1,3]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
`{pengshuo, hejiahui2020}@mail.sdu.edu.cn`

[2] Quan Cheng Shandong Laboratory, Jinan, China
`{kai.hu, mqwang}@sdu.edu.cn`

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

Abstract. Ascon, a family of algorithms that support hashing and Authenticated Encryption with Associated Data (AEAD), is the final winner of the NIST Lightweight Cryptography Project. As a research hotspot, Ascon has received substantial third-party security evaluation. Among all the results of Ascon-128 (the primary recommendation of AEAD), the key recovery attack can only be achieved by reducing the initialization phase to 7 rounds or fewer, regardless of whether it violates the security claims made by the designers (i.e., misuse of the nonce or exceeding data limits 2^{64}). In this paper, we, from two aspects (misuse-free setting and misused setting), improve the key recovery attack on Ascon-128 using the cube attack method. In one part, we present a faster method to recover the superpolies for a 64-dimensional cube in the output bits of the 7-round initialization, enabling us to recover the secret key with a time complexity of $2^{95.96}$ and a data complexity of 2^{64}. Our 7-round key recovery attack, based on the full key space, greatly improves the time complexity, making it the best result to date. Additionally, we utilize several techniques to extend state recovery to key recovery, answering the open problem of transitioning from full state recovery in the encryption phase to key recovery for Ascon-128 (ToSc Vol 4, 2022). By combining encryption phase state recovery with initialization phase key recovery, we can achieve 8-round and 9-round initialization phase key recovery in the nonce misuse scenario, with time complexities of 2^{101} and $2^{123.92}$, respectively. This represents an improvement of two rounds over previous results in the misused setting. Our first key recovery attack is also applicable to Ascon-128a, achieving the same result. In cases where the full state, prior to the encryption phase, can be recovered in other Ascon AEAD modes, our second key recovery attack will also be useful. It is worth noting that this work does not threaten the security of the full 12 rounds Ascon, but we expect that our results provide new insights into the security of Ascon.

Keywords: Ascon-128 · Cube attack · Superpoly recovery · Key recovery

1 Introduction

Ascon [11], designed by Dobraunig, Eichlseder, Mendel, and Schläffer, is a family of lightweight Authenticated Encryptions with Associated Data (AEAD) and hash functions. As the winner of CAESAR competition, it is part of the final portfolio of the CAESAR competition in the "lightweight applications" category. Additionally, in February 2023, the NIST Lightweight Cryptography Team announced its decision to standardize the Ascon family for lightweight cryptographic applications, recognizing its suitability for a wide range of use cases where lightweight cryptography is essential. Consequently, assessing the security of Ascon is of critical importance.

The Ascon family supports authenticated ciphers Ascon-128, Ascon-128a, and Ascon-80pq, hash functions Ascon-Hash and Ascon-Hasha, and extendable output functions Ascon-XOF and Ascon-XOFa. In this paper, we focus on the AEAD modes, mainly Ascon-128, the primary recommendation of AEAD mode which aims to provide integrity, confidentiality, and authenticity. The designers of Ascon claim AEAD variants provide 128-bit security of privacy and authenticity when unique nonce values are used for the encryption under the same key. The maximum available plaintext and associated data blocks to the attacker are limited to 2^{64} per key.

Since being proposed, the security of the Ascon family has been widely analyzed. Cryptanalysis of Ascon in the AEAD context can be divided into different categories by whether the security claims are satisfied during the attack process. When the security claims are followed, many results cannot reach 7 rounds [10,19,16] and most 7-round key recovery attacks are based on the weak key condition [23,15]. The only known 7-round result based on the full key space is proposed by the authors in [22] with a time complexity of 2^{123}. In cases where the security claims are violated, some 7-round initialization key recovery attacks exceed the data limitations, although they remain within the nonce-respecting setting [19,15]. When it comes to that nonce is misused(nonce is reused under the same key), the state recovery during the encryption phase is typically the primary target of attacks, as seen in studies such as [19,3,7]. In [19], Li et al. provided key recovery attacks using a cube-like technique on a round-reduced version of Ascon with 7 rounds initialization in a nonce-misused setting. Another key recovery attack mentioned in [7] targeted the finalization phase of Ascon-128a and Ascon-80pq.

Although extensive analyses have been conducted on Ascon-128, none of the identified attacks pose a significant threat to the initialization phase of Ascon-128 for more than 7 rounds. Also, most 7-round initialization results either violate the security claims or are restricted to a weak key space. The only known 7-round key recovery attack, based on security claims, was proposed by the authors in [22] and exhibits a substantial time complexity of 2^{123}. Given the low degree of the round function in Ascon, algebraic attacks may prove more effective in the attack of Ascon. In this paper, we present some improved key recovery results for the initialization phase of Ascon-128 based on the cube attack, considering both misused and misuse-free settings.

Contributions. This paper focuses on key-recovery attacks against Ascon-128. We improve the key recovery attack from two aspects (misuse-free and misused settings) on Ascon-128. The details of our results are shown in Table 1.

Key recovery in the misuse-free setting. In the misuse-free setting, unique nonce values are used for encryption under the same key, with data complexity not exceeding 2^{64}. This limitation on data and the respective nonce prevent key recovery attack on Ascon-128. To adhere to the security claims of Ascon-128, we employ a cube attack methodology, specifically targeting the 7-round initialization phase of Ascon-128 while utilizing a 64-dimensional cube. We first present a faster method to efficiently recover the superpolies of the selected 64-dimensional cube in the output bits of the 7-round Ascon initialization. Based on the recovered superpolies, we can execute a key recovery attack. As a result, our attack can recover the secret key with a time complexity of $2^{95.96}$ and a data complexity of 2^{64}. Our key recovery attack operates over the full key space and significantly improves the time complexity compared to previous works.

Key recovery in the misused setting. In [3], Baudrin et al. proposed a conditional cube attack capable of recovering the full state during the encryption phase of Ascon-128 within a practical time. However, extending this attack to achieve key recovery or forgery in Ascon-128 remains an open problem. To address this issue, we extend state recovery to key recovery by integrating state recovery during the encryption phase with key recovery during the initialization phase. Note a secret key is added to the state after the initialization phase, we exploit a cube distinguisher in the initialization phase. Then we traverse the key space to decrypt the recovered state until the end of the distinguisher. Since the correct key satisfies the conditions of the distinguisher, while the wrong key may not, the secret key space is reduced. To enhance the decryption process, we employ the partial sum technique, allowing us to append more rounds at the end of the distinguisher. For Ascon-128, we can recover the secret key with a complexity of 2^{101} for 8-round initialization and $2^{123.92}$ for 9-round initialization. Compared to the previous attack, which was limited to 7-round initialization, we have achieved an improvement of 2 rounds.

Outline. This paper is organized as follows: Section 2 presents some notations and definitions that will be used in this paper. The specification and some properties of Ascon are illustrated in Section 3. Section 4 introduces our key recovery in the misuse-free setting. The key recovery in the misused setting is shown in Section 5. Section 6 concludes the paper. The source codes and some results used in this paper are provided at `https://github.com/keyrecoveryofascon/keyrecoveryofascon`.

2 Preliminaries

Notations. Let $\mathbb{F}_2$ denote the finite field with two elements, while $\mathbb{F}_2^n$ denotes the vector space of dimension n over $\mathbb{F}_2$. We use bold italic lowercase letters to represent bit vectors. Given two vectors $\boldsymbol{u} = (u_0, u_1, \cdots, u_{n-1}), \boldsymbol{v} = (v_0, v_1, \cdots, v_{n-1}) \in \mathbb{F}_2^n$,

Table 1: Summary of key recovery attacks on Ascon AEAD. In the "Attack Type" column, "NR" denotes nonce-respecting attacks, while "NM" indicates nonce is misused. The "Var." column lists the Ascon variants, including Ascon-128, Ascon-128a, and Ascon-80pq. In the "Method" column, "Con." stands for conditional, "DL" refers to differential-linear, and "HDL" denotes higher-order differential-linear. The symbol † represents "weak-key space". "Valid.D" indicates whether the data complexity exceeds 2^{64}.

Attack Type	Phase	Var.	#R	Data/Time	Method	Valid.D	Source
NR key-recovery	Init.	all	4/12	$2^{18}/2^{18}$	DL	✓	[10]
		all	5/12	$2^{36}/2^{36}$	DL	✓	[10]
		all	5/12	$2^{35}/2^{35}$	Cube	✓	[10]
		all	5/12	$2^{26}/2^{26}$	Con.DL	✓	[21]
		all	5/12	$2^{24}/2^{36}$	Con.cube	✓	[20]
		all	5/12	$2^{22}/2^{32}$	Con.HDL	✓	[16]
		all	6/12	$2^{66}/2^{66}$	Cube	✗	[10]
		all	6/12	$2^{40}/2^{40}$	Con.cube	✓	[20]
		128(a)	7/12	$2^{77.2}/2^{77}$	Con.cube†	✗	[20]
		all	7/12	$2^{64}/2^{97}$	Cube†	✓	[23]
		all	7/12	$2^{63}/2^{115.2}$	Cube†	✓	[23]
		128(a)	7/12	$2^{72.1}/2^{72.1}$	Con.cube†	✗	[15]
		128(a)	7/12	$2^{63.32}/2^{115}$	Con.cube†	✓	[15]
		80pq	7/12	$2^{72.1}/2^{72.1}$	Con.cube†	✗	[15]
		128(a)	7/12	$2^{72.1}/2^{104.7}$	Con.cube	✗	[15]
		80pq	7/12	$2^{72.1}/2^{104.7}$	Con.cube	✗	[15]
		all	7/12	$2^{64}/2^{123}$	Cube	✓	[22]
		128(a)	7/12	$2^{64}/2^{95.96}$	Cube	✓	Section 4
NM key-recovery	Init.	all	7/12	$2^{97}/2^{97}$	Cube-like	✗	[19]
		128	8/12	$2^{101}/2^{101}$	Con.cube	✗	Section 5
		128	9/12	$2^{101}/2^{123.92}$	Con.cube	✗	Section 5

$\boldsymbol{u} \preccurlyeq \boldsymbol{v}$(resp. $\boldsymbol{u} \succeq \boldsymbol{v}$) stands for $u_i \le v_i$ (resp. $u_i \ge v_i$), $\forall i \in \{0,\ldots,n-1\}$. The Hamming weight of vector $\boldsymbol{x} = (x_0, x_1, \cdots, x_{n-1})$, considered in $\mathbb{Z}$, is denoted as $wt(\boldsymbol{x})$, which is equal $\sum_{i=0}^{n-1} x_i$. 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. Let I be a set, the complementary set of I is denoted $\bar{I}$. we use $|I|$ to represent the size of I. All elements in I but not in J are denoted by $I - J$. Given a set $I \subseteq \{0,\ldots,n-1\}$ 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 $f : \mathbb{F}_2^n \to \mathbb{F}_2$ with Algebraic Normal Form

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} \alpha_{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{u}},$$

where $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$, $\alpha_{\boldsymbol{u}} \in \mathbb{F}_2$ and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$, the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{u}}$ in f is denoted by $\alpha_{\boldsymbol{u}}$ =Coe$_f(\boldsymbol{x}^{\boldsymbol{u}})$.

Lemma 1 **([6,5]).** *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 $Coe_f(\boldsymbol{x}^{\boldsymbol{u}}) = \sum_{\boldsymbol{x} \preccurlyeq \boldsymbol{u}} f(\boldsymbol{x})$ with $2^{wt(\boldsymbol{u})}$ evaluations of f.*

Lemma 2 **([6,5]).** *The set of all coefficients $\{\alpha_{\boldsymbol{u}} = \sum_{\boldsymbol{x} \preccurlyeq \boldsymbol{u}} f(\boldsymbol{x}) : \boldsymbol{u} \in \mathbb{F}_2^n\}$ of the ANF can be obtained from the truth table of f with so-called fast Möbius transform with about $n2^n$ XOR operations.*

Keyed Boolean functions. It is often necessary to distinguish controllable public variables from inaccessible secret variables. We denote public variables $\boldsymbol{x}$, and secret variables $\boldsymbol{k}$. With such a distinction, when a Boolean function depends on n public variables and m key variables, we look at it as

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

where $\boldsymbol{x} \in \mathbb{F}_2^n$, $\boldsymbol{k} \in \mathbb{F}_2^m$. To distinguish from the Boolean function without secret variables, we also denote the keyed Boolean functions as $f_{\boldsymbol{k}}(\boldsymbol{x})$. When referring to the degree of a keyed Boolean function, we mean the degree in public variables: $\deg(f) := \max\{\mathbf{wt}(\boldsymbol{u}), \alpha_{\boldsymbol{u}} \neq 0\}$. In equation 1, the coefficient $\mathrm{Coe}_{f_{\boldsymbol{k}}}(\boldsymbol{x}^{\boldsymbol{u}})$ actually is a Boolean function from $\mathbb{F}_2^m \to \mathbb{F}_2$ which maps $\boldsymbol{k}$ to $\alpha_{\boldsymbol{u}}(\boldsymbol{k})$.

Lemma 3 **([22]).** *For the Boolean function shown in equation 1, it takes $2^{m+wt(\boldsymbol{u})}$ evaluations of $f_{\boldsymbol{k}}$ to recover $Coe_{f_{\boldsymbol{k}}}(\boldsymbol{x}^{\boldsymbol{u}})$ for a certain u where $wt(\boldsymbol{u}) > log_2(m)$.*

Proof. According to Lemma 1, for any keyed Boolean function $f_{\boldsymbol{k}}$ and any given key, the value 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. If $wt(\boldsymbol{u}) > log_2(m)$, $2^{m+log_2(m)}$ XOR operations can be ignored.

Cube attack. The cube attack was proposed at EUROCRYPT 2009 by Dinur and Shamir to analyze black-box tweakable polynomials [9]. For a set $I \subseteq \{0, \ldots, n-1\}$ with its complementary set $\bar{I} = \{0, \ldots, n-1\} - I$, the above keyed Boolean function can be represented as

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

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 $\boldsymbol{k}$. Concerning the superpoly, we have the following lemma.

Lemma 4 **([9]).** *For a set $I \subseteq \{0, \ldots, n-1\}$ 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 $I = \{0, 1, \cdots, n-1\}$, then the superpoly of $\boldsymbol{x}^I$ in $f(\boldsymbol{x}, \boldsymbol{k})$ equals to $\mathrm{Coe}_f(\boldsymbol{x}^I)$.

Division property. Recovering the superpoly of $\boldsymbol{x}^I$ is challenging when the expressions of the $f_{\boldsymbol{k}}$ are complex and not readily available. Luckily, division properties proposed by Todo [24] have evolved into an effective and accurate method of probing the structure of Boolean functions. Its bit-based variants [26] together with their automatic search methods [29] have been found to have a great potential in accurately determining whether a monomial appears or disappears in an ANF that is not directly accessible [25,27,28,13]. 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 [25,27,28,13,14,17]. In fact, the division property has become a quite standard tool in assisting cube attacks. In this work, we adopt the Mixed Integer Linear Programming (MILP) approach to model the three-subset bit-based division property without unknown subset(3BDPwoU) [13], aiding in the recovery of the superpoly. A permutation can be decomposed into a sequence of basic operations, such as XOR, AND, and COPY. Therefore, it is sufficient to provide propagation rules of the 3BDPwoU for these basic operations. The concrete propagation rules and models of the 3BDPwoU are provided in Appendix A.

3 Ascon specification and useful properties

At a high level, 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. The Ascon AEAD adopts a MonkeyDuplex [4] mode with a stronger keyed initialization and keyed finalization phases as illustrated in Figure 1. The Ascon AEAD family consists of three members, Ascon-128, Ascon-128a, and Ascon-80pq. All three variants have 128-bit nonce and Ascon-80pq takes half of the IV positions to allow 32 more key bits. The rates of the Ascon-128 and Ascon-80pq are 64 bits while 128 bits for Ascon-128a. The parameters are summarized in Table 2 and all three variants provide 128-bit security. The core components of all variants are the two 320-bit permutations p^a and p^b, with the permutation p of a and b rounds, respectively.

Table 2: Ascon-AEAD 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
Ascon-80pq	320	64	160	128	128	12	6	a0400c06

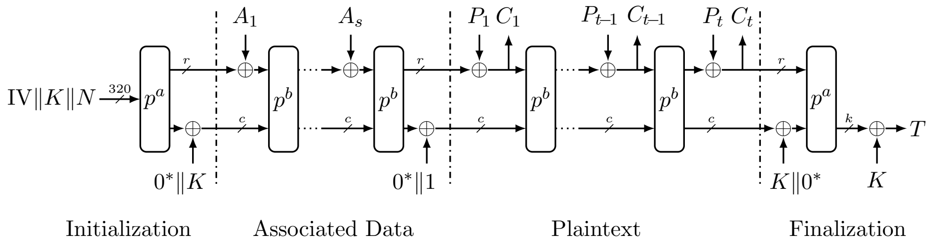

Fig. 1: Ascon AEAD encryption.

The permutation p is defined as the composition of a constant addition, a non-linear substitution layer, and a linear diffusion layer: $p = p_l \circ p_s \circ p_c$. The permutation operates on a 320-bit state, which is typically decomposed into five rows(Each row is a 64-bit word). 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 $[\cdot]$ and X^r, Y^r refer to the full state. We denote $X_i^r[j]$ as the j-th (column) bit of the i-th (row) 64-bit word, where $0 \le i \le 4$ and $0 \le j \le 63$. Alternatively, $X_i^r[j]$ is also denoted as $X^r[64 * i + j]$. The state is constructed as a Substitution Permutation Network (SPN), as illustrated in Figure 2. These operations in p are described as follows. *Addition of constants*(p_c). The constant addition step consists in XORing an

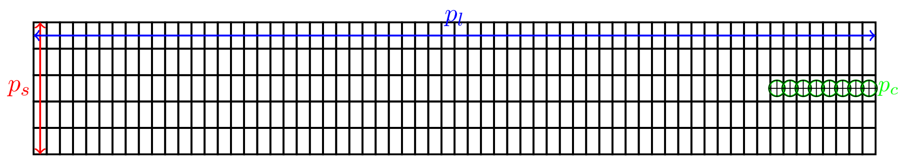

Fig. 2: The column-wise S-box layer, the row-wise linear layer and the constant addition.

8-bit constant to the positions $56, \cdots, 63$ of the 64-bit word X_2 at each round. *Substitution layer*(p_s). The substitution layer is made of 64 parallel calls to a single Substitution box (Sbox) on each state column. The Sbox used in Ascon is a quadratic 5-bit permutation. Equation 2 presents the ANFs of the Sbox in Ascon.

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

Linear diffusion layer(p_l). The linear diffusion layer is made of five calls to five different linear functions on each row of the state, as illustrated in Equation 3.

$$\begin{aligned}
X_0 &\leftarrow \sum_0(Y0) = Y_0 + (Y_0 >>> 19) + (Y_0 >>> 28)\\
X_1 &\leftarrow \sum_0(Y1) = Y_1 + (Y_1 >>> 61) + (Y_1 >>> 39)\\
X_2 &\leftarrow \sum_0(Y2) = Y_2 + (Y_2 >>> 1) + (Y_2 >>> 6)\\
X_3 &\leftarrow \sum_0(Y3) = Y_3 + (Y_3 >>> 10) + (Y_3 >>> 17)\\
X_4 &\leftarrow \sum_0(Y4) = Y_4 + (Y_4 >>> 7) + (Y_4 >>> 41)
\end{aligned} \tag{3}$$

This paper focuses on cube attack for Ascon-128. Based on the ANFs of the Ascon Sbox, several useful properties can be derived, which are commonly used in the cube attack of Ascon-128. In the following, we introduce some properties of Ascon-128 that will be used in our attack. Note that only the nonce X_3^0 and X_4^0 can be viewed as public variables in Ascon-128. We refer to the variables of X_3^0 and X_4^0 when discussing the degree.

Property 1. Among the 5 output bits of the Sbox, $X_4[j]$ never multiplies with $X_2[j]$.

Property 2. For $r \in \{0, \cdots, 7\}$, the degree of $X_i^r[j]$ is at most 2^r.

Property 3. If only one of the words X_3^0 and X_4^0 is treated as public variables, with the other set as constant, the degree of $X_i^r[j]$ is at most 2^{r-1}.

Property 1, 2 and 3 can be easily deduced from the quadratic Sbox in Ascon. To reduce the degree of $X_i^r[j]$, a condition can be imposed on the public variables, namely *Conditional Cube* [18]. In [22], the authors provided a condition that X_3^0 equals X_4^0 for Ascon-128. They then modelled the 3BDPwoU for the initialization phase of Ascon-128 using MILP to compute upper bounds on the degree of $X_i^r[j]$. We present their 7-round results in the following property.

Property 4 (Adapted form[22]). Under the condition that the public variables X_3^0 and X_4^0 are equal, the algebraic degrees of $X_0^7[j], X_1^7[j], \cdots, X_4^7[j]$ where $j \in \{0, \cdots, 63\}$ are 59, 59, 60, 60, and 58, respectively.

The concept *Borderline cube* whose superpoly depends only on a (relatively) small number of key bits is proposed in [8]. When the cube variables are not multiplied together in the first round, and the degree of $X_i^r[j]$ is 2^{r-1}, if a secret variable k_i is not multiplied with the cube variables in the first round, then the superpoly of a 2^{r-1} dimensional cube in $X_i^r[j]$ will be independent of k_i. According to the property 1 and property 3, a borderline cube can be constructed when only X_4^0 is regarded as cube variables, which was observed in [10]. We write their observations as the following Lemma.

Lemma 5 (Adapted from [10]). *If only X_4^0 can be regarded as public variables denoted $x_i (i \in 0, 1, \cdots, 63)$, for $1 \le r \le 7$ 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 $X_i^{(r)}[j]$ for $i \in \{0, \cdots, 4\}$ and $j \in \{0, \cdots, 63\}$ can be fully determined by the 2^{r-1} key bits in $\left\{k_{i_0}, \cdots, k_{i_{2^{r-1}-1}}\right\}$.*

4 Our key recovery attack in the misuse-free setting

In this section, we present our key recovery attack of 7-round initialization on Ascon-128 adhering to the security claims made by the designers(the nonce respective setting and the data complexity does not exceed 2^{64}). Our attack mode is illustrated in Figure 3, employing a cube attack technique. The value in the first row of X_0^7 can be obtained by calculating $P_1 + C_1$. According to the inverse of the linear layer in Ascon, we can compute the value of Y_0^6(the first row in the state after 7-round Sbox operation.) by X_0^7. Initially, we recover the superpolies of chosen cube in $Y_0^6[j]$, for $0 \le j \le 63$. The values of these superpolies can be evaluated through several calls to the 7-round initialization process. Subsequently, the recovered superpolies enable us to deduce crucial key information.

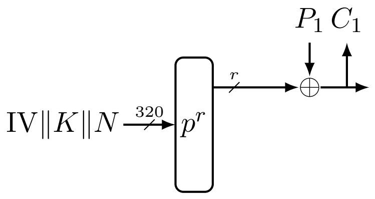

Fig. 3: Our attack mode.

4.1 Previous work in [22]

In [22], the authors provided the first valid key recovery attack(which follows Ascon's security claims) on the 7-round initialization. In their work, they proposed a useful technique called *partial polynomial multiplication*, which enables the recovery of the superpoly of a given cube by multiplying the simplified versions of the involved Boolean functions.

More precisely, for a keyed Boolean function $f(\boldsymbol{x},\boldsymbol{k}) = \sum_{\boldsymbol{u}\in\mathbb{F}_2^n} \alpha_{\boldsymbol{u}}(\boldsymbol{k})\boldsymbol{x}^{\boldsymbol{u}}$ where $\boldsymbol{x} \in \mathbb{F}_2^n$ and $\boldsymbol{k} \in \mathbb{F}_2^m$, if it can be expressed 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 $\deg(p^{(t)}) \le n/2$, $\deg(q^{(t)}) \le n/2$ and $\deg(\varepsilon) < n$ (here, We emphasize that $\deg(f)$ refers to the algebraic degree in $\boldsymbol{x}$) where n is even, then for $I = \{0,\cdots,n-1\}$

$$Coe_f(\boldsymbol{x}^I) = Coe_{\sum_{t=0}^{l-1} p^{(t)}q^{(t)}}(\boldsymbol{x}^I) = \sum_{t=0}^{l-1} Coe_{p^{(t)}q^{(t)}}(\boldsymbol{x}^I)$$

.

Since the degrees of $p^{(t)}$ and $q^{(t)}$ are no greater than $n/2$, only the degree-$n/2$ monomials in $p^{(t)}$ multiplying with the degree-$n/2$ monomials in $q^{(t)}$ might contribute to the coefficient of the monomial $\boldsymbol{x}^I$. Therefore, We only need to care the coefficients of degree-$n/2$ monomials in both $p^{(t)}$ and $q^{(t)}$ when computing the coefficient of $\boldsymbol{x}^I$, as shown in the following,

$$Coe_{p^{(t)}q^{(t)}}(\boldsymbol{x}^I) = \sum_{J \subsetneq I, |J|=n/2} Coe_{p(t)}(\boldsymbol{x}^J) Coe_{q(t)}(\boldsymbol{x}^{I-J})$$

The authors in [22] applied this technique to the key recovery of 7-round Ascon initialization, successfully recovering the superpoly of a 64-dimensional cube by multiplying the coefficients of degree-32 monomials in the bits of 6-th round state($X_i^r[j]$). With an overall time complexity of approximately 2^{123} calls for 7-round Ascon permutations, they successfully recovered the superpoly. In their work, the process of computing the ANFs of $X_i^6[j]$ dominates the overall time complexity of superpoly recovery, which requires approximately 2^{123} calls of 7-round Ascon permutation.

In this paper, we extend the *partial polynomial multiplication* technique proposed in [22] to a more general case and apply it to the superpoly recovery of the 7-round Ascon-128 initialization. Specifically, to recover the superpoly of a 64-dimensional cube, we compute it by multiplying the coefficients of monomials of an earlier round state (in our case, the 5-th round state, rather than the 6-th round state). Compared to the computation of the ANFs of $X_i^6[j]$, evaluating the ANFs of $X_i^5[j]$ is significantly faster. Another challenge addressed in this work is the increased frequency of multiplying the coefficients of two monomials. To mitigate this, instead of expanding the product into a polynomial in terms of $\boldsymbol{k}$, we exploit its product form, which reduces the overall computational time.

4.2 Our superpoly recovery(offline phase)

In the above, we briefly describe the *partial polynomial multiplication* proposed in [22]. Actually, the partial polynomial multiplication can be extended to a more general case. For a keyed Boolean function $f(\boldsymbol{x}, \boldsymbol{k}) = \sum_{\boldsymbol{u} \in \mathbb{F}_2^n} \alpha_{\boldsymbol{u}}(\boldsymbol{k}) \boldsymbol{x}^{\boldsymbol{u}}$ where $\boldsymbol{x} \in \mathbb{F}_2^n$ and $\boldsymbol{k} \in \mathbb{F}_2^m$, if it can be represented as the following form

$$f(\boldsymbol{x}, \boldsymbol{k}) = \varepsilon(\boldsymbol{x}, \boldsymbol{k}) + \sum_{t=0}^{l-1} (\prod_{s=0}^{i-1} p_{(s)}^{(t)}(\boldsymbol{x}, \boldsymbol{k})) \tag{4}$$

such that $\deg(p_{(s)}^{(t)}) \le n/i$, and $\deg(\varepsilon) < n$ where n is divisible by i. This representation of $f(\boldsymbol{x}, \boldsymbol{k})$ contains l monomials of $p(\boldsymbol{x}, \boldsymbol{k})$, i.e., the product of i polynomials $p(\boldsymbol{x}, \boldsymbol{k})$. Then for $I = \{0, \cdots, n-1\}$

$$Coe_f(\boldsymbol{x}^I) = Coe_{\sum_{t=0}^{l-1}(\prod_{s=0}^{i-1} p_{(s)}^{(t)})}(\boldsymbol{x}^I) = \sum_{t=0}^{l-1} Coe_{\prod_{s=0}^{i-1} p_{(s)}^{(t)}}(\boldsymbol{x}^I) \tag{5}$$

Similarly, in $p(\boldsymbol{x}, \boldsymbol{k})$, only the coefficient of degree-(n/i) monomial can contribute to $\mathrm{Coe}_f(\boldsymbol{x}^I)$. Thus, we can only focus on the coefficient of degree-(n/i) monomials in each $p(\boldsymbol{x}, \boldsymbol{k})$.

An R-round Ascon $f(\boldsymbol{x},\boldsymbol{k})$ can be further decomposed into a composition of simple vectorial Boolean functions as follows,

$$f(\boldsymbol{x},\boldsymbol{k}) = f^{(R-1)} \circ f^{(R-2)} \circ \cdots f^{(0)}(\boldsymbol{x},\boldsymbol{k}),$$

where quadratic round function $f^{(r)} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ $(0 \le r \le R-1)$. In the condition of $R \le 7$, if we choose a proper number r_0, where $r_0 \in \{0,\cdots,R-1\}$, $f(\boldsymbol{x},\boldsymbol{k})$ can be represented the sum of monomials of X^{r_0}, which is the form as Equation 4, where now $p(\boldsymbol{x},\boldsymbol{k})$ is a bit of X^{r_0} and $i = 2^{R-r_0}$. Thus, the coefficient of $\boldsymbol{x}^I$ in $f(\boldsymbol{x},\boldsymbol{k})$ can be computed by Equation 5.

In our superpoly recovery process for Ascon-128, we set $r_0 = 5$ and i will be 4. Consequently, the state Y_0^6 can be expressed as a polynomial composed of monomials of X^5. Known the representation of $f(\boldsymbol{x},\boldsymbol{k})$ as Equation 4, the superpoly recovery process can be divided into two parts: (1) compute the ANFs of all bits in X^5 with respect to variables $\boldsymbol{x}$ and $\boldsymbol{k}$, and (2) compute the superpoly according Equation 5.

The superpoly recovery process is performed entirely offline and needs to be done only once for all (secret key). For sake of clarity, we take the first bit of Y_0^6 as an example to demonstrate how we recover the superpoly of $\boldsymbol{x}^I$ in $Y_0^6[0]$.

Choose cube. According to Lemma 5, when all bits of X_3^0 are set to 0 and all bits of X_4^0 are treated as cube variables, the superpoly corresponding to this cube in 7the -th round depends solely on the 64 bits secret key, namely X_1^0. In this configuration, the word X_4^0 is viewed as cube variables denoted as $x_0, x_1, \cdots, x_{63}$, while X_1^0 and X_2^0 represent secret variables, denoted as $k_0, k_1, \cdots, k_{127}$ (with only $k_0, k_1, \cdots, k_{63}$ are involved in the superpoly). We specifically chose this 64-dimensional borderline cube for the superpoly recovery because a smaller number of variables simplifies both the superpoly and key recovery processes. Consequently, in our superpoly recovery, the cube set $\boldsymbol{x}[I]$ we choose is $\{x_0, x_1, \cdots, x_{63}\}$.

Compute all monomials of $X_i^5[j]$ containing $\boldsymbol{x}^I$ in $Y_0^6[0]$. $Y_0^6[0]$ can be expressed as a polynomial composed of monomials of X^5. According to property 3, the degree of $X_i^5[j]$ is 16 and the degree of $Y_0^6[0]$ is 64. Therefore, only the monomial of X^5 with four factors may contain $\boldsymbol{x}^I$, which will contribute to $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$. Without loss of generality, we denoted the monomial of X^5 containing $\boldsymbol{x}^I$ as the following form,

$$X^5[p]X^5[q]X^5[s]X^5[t]$$

Where p, q, s and t are different values in $\{0,1,\cdots,319\}$. While there are too many monomials of X^5 in the representations of $Y_0^6[0]$, we use the MILP tool and the three-subset bit-based division property without unknown subset(3BDPwoU) [13] to help us finding all monomials containing $\boldsymbol{x}^I$. The details of the MILP model rule for 3BDPwoU are shown in the appendix A.

We first compute all monomials with four factors($X_i^5[j]$) using Gurobi [1], by solving a MILP model for the 6-th round and 7-th round (exclude the linear layer). Subsequently, for each obtained monomial, we test whether it can contain $\boldsymbol{x}^I$ by solving a MILP model for the first five rounds of Ascon.

In total, we identify 121 monomials that contribute to $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$. The corresponding source code and results are available in the github repository at https://github.com/keyrecoveryofascon/keyrecoveryofascon/tree/main/forward_extension.

Step1: Compute the coefficients of degree-16 monomials in $X_i^5[j]$. According to Lemma 5, for a given $J = \{i_0, \cdots, i_{15}\} \subseteq I = \{0, \cdots, 63\}$, the coefficient of $\boldsymbol{x}^J$ in $X_i^5[j]$ is a Boolean function in variables $\{k_{i_0}, \cdots, k_{i_{15}}\}$, involving 15 secret key variables. The truth table for this Boolean function can be obtained after $2^{16} \times 2^{16} = 2^{32}$ evaluations of the 5-round Ascon permutation, where for each possible value of $(k_{i_0}, \cdots, k_{i_{15}}) \in \mathbb{F}_2^{16}$, we evaluate the coefficient value of $\boldsymbol{x}^J$ based on Lemma 1. Since one evaluation of the 5-round Ascon permutation gives the coefficient value of $\boldsymbol{x}^J$ in the full state $X_i^5[j]$, after $2^{16} \times 2^{16} = 2^{32}$ evaluations of the 5-round Ascon permutation, we obtain the 320 truth tables for the coefficients of $\boldsymbol{x}^J$ in $X_i^5[j]$.

By applying the fast Möbius transform to the 320 truth tables, we obtain the ANFs of $Coe_{X_i^5[j]}(\boldsymbol{x}^J)$ for $i \in \{0, \cdots 4\}$ and $j \in \{0, \cdots, 63\}$, with about $320 \times 16 \times 2^{16}$ XOR operations. In summary, the time complexity of recovering the ANF of $Coe_{X_i^5[j]}(\boldsymbol{x}^J)$ for $i \in \{0, \cdots 4\}$ and $j \in \{0, \cdots, 63\}$ is dominated by 2^{32} evaluations of the 5-round Ascon permutation. Since there are totally $\binom{64}{16} \approx 2^{48.8}$ different sets $J \subseteq I$ with $|J| = 16$, it takes $2^{48.8+32} \approx 2^{80.8}$ calls to the 5-round Ascon permutation to obtain the ANFs of $Coe_{X_i^5[j]}(\boldsymbol{x}^J)$ ($i \in \{0, \cdots, 4\}$ and $j \in \{0, \cdots, 63\}$) for all possible $J \subseteq I$ with $|J| = 16$. Then we store each $Coe_{X_i^5[j]}(\boldsymbol{x}^J)$ as a 2^{16}−bit string into a hash table $\mathbb{T}_i[j]$ at address $addr(\boldsymbol{x}^J) \in \mathbb{F}_2^{64}$. Finally, we obtain 320 tables $\mathbb{T}_i[j]$ ($i \in \{0, \cdots, 4\}$ and $j \in \{0, \cdots, 63\}$), which requires about $320 \times \binom{64}{16} \times 2^{16} \approx 2^{73.12}$ bits of memory.

Step2: Compute $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$. In the above, we identify 121 monomials that contribute to $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$. To compute the coefficient of $\boldsymbol{x}^I$ in $Y_0^6[0]$, we need to calculate the contribution of each monomial and then sum them. Next, we outline the process for calculating the contribution of a single monomial, using $X^5[p]X^5[q]X^5[s]X^5[t]$ as an illustrative example.

1. Firstly, we compute the coefficients of all degree-32 monomials in $X^5[p]X^5[q]$ and $X^5[s]X^5[t]$. Here, we use x^L where $L \subseteq I$ and $|L| = 32$ to denote a degree-32 monomial. For $X^5[p]X^5[q]$ and $X^5[s]X^5[t]$, we use the hash tables $\mathbb{L}_{p,q}$ and $\mathbb{L}_{s,t}$ to store the coefficients of the degree-32 monomials. The coefficient of $\boldsymbol{x}^L$ in $X^5[p]X^5[q]$ (resp. $X^5[s]X^5[t]$) is stored at address $addr(x^L) \in \mathbb{F}_2^{64}$ of the hash table $\mathbb{L}_{p,q}$ (resp. $\mathbb{L}_{s,t}$).
   For clarity, we take $X^5[p]X^5[q]$ as an example to show how we compute the coefficients of all degree-32 monomials in them and $X^5[s]X^5[t]$ is similar. For a degree-16 monomial $\boldsymbol{x}^{J_1}$ in $X^5[p]$ and a degree-16 monomial $\boldsymbol{x}^{J_2}$ in $X^5[q]$, they will produce a degree-32 monomial if $|J_1 \cup J_2| = 32$. We then store the coefficient of $\boldsymbol{x}^{J_1}$ and $\boldsymbol{x}^{J_2}$ at the address $addr(\boldsymbol{x}^{J_1 \cup J_2}) \in \mathbb{F}_2^{64}$ of the hash table $\mathbb{L}_{p,q}$. It is worth noting that instead of storing it as a polynomial in terms of $\boldsymbol{k}$, we just store the address of the coefficient of $\boldsymbol{x}^{J_1}$ and $\boldsymbol{x}^{J_2}$.

There are $\binom{64}{16}$ degree-16 monomials in $X^5[p]$. For each degree-16 monomial in $X^5[p]$, there are $\binom{48}{16}$ degree-16 monomials in $X^5[q]$ that will be multiplied with it to produce a degree-32 monomial. Thus, the time complexity is $\binom{64}{16} \times \binom{48}{16} \approx 2^{48.80+41.04} = 2^{89.84}$ memory accesses of hash table $\mathbb{L}_{p,q}$. For a degree-32 monomial, there will be $\binom{32}{16}$ terms in its coefficient. The memory complexity is $\binom{64}{32} \times \binom{32}{16} \times (64+64) \approx 2^{60.67+29.16+7} = 2^{96.83}$. Considering $X^5[s]X^5[t]$ and there are 121 monomials to be handled, the time complexity is $2^{97.76}$ and the memory complexity is $2^{104.85}$. Note that there are many duplicate items of $X^5[p]X^5[q]$ and $X^5[s]X^5[t]$ in the 121 monomials. The complexity can be reduced further, but here we omit it.

2. Then, we compute the coefficient of $\boldsymbol{x}^I$ in $X^5[i]X^5[j]X^5[s]X^5[t]$. For a degree-32 monomial in $X^5[i]X^5[j]$, only one degree-32 monomials in $X^5[s]$ $X^5[t]$ can multiply with it to produce a degree-64 monomial, namely $\boldsymbol{x}^I$. Let $\boldsymbol{x}^{L_1}$ be a degree-32 monomial in $X^5[i]X^5[j]$ and $\boldsymbol{x}^{L_2}$ be a degree-32 monomial in $X^5[s]X^5[t]$, where $|L_1 \cup L_2| = 64$. We can obtain the coefficient of x^I provided by multiplying the coefficient of $\boldsymbol{x}^{L_1}$ with the coefficient of $\boldsymbol{x}^{L_2}$. Also, we use the product form rather than expand it into a polynomial. It will cost $2 + 2\binom{32}{16}$ memory accesses (one memory access of $\mathbb{L}_{p,q}$ and $\mathbb{L}_{s,t}$, $2\binom{32}{16}$ memory accesses of $\mathbb{T}_i[j]$). There are $\binom{64}{32}$ degree-32 monomials in $X^5[i]X^5[j]$. The time complexity in this procedure is $\binom{64}{32} \times 2 \times \binom{32}{16} \approx 2^{60.67+29.16+1} = 2^{90.84}$ memory access(one memory access of $\mathbb{L}_{p,q}$ and $\mathbb{L}_{s,t}$ can be ignored). Considering 121 monomials, the time complexity is $2^{97.76}$.

If all coefficients of $\boldsymbol{x}^I$ in the 121 monomials of X^5 are computed, $\mathrm{Coe}_{Y_0^6[0]}(\boldsymbol{x}^I)$ equals the sum of them. In total, in this step, the time complexity is $2^{98.76}(2^{97.76}+2^{97.76})$ memory accesses and the memory complexity is $2^{104.85}$ bits.

Generate the comparison tables for key candidates. With the 64 recovered superpolies $\mathrm{Coe}_{Y_0^6[0]}\left(\boldsymbol{x}^I\right), \mathrm{Coe}_{Y_0^6[1]}\left(\boldsymbol{x}^I\right), \ldots, \mathrm{Coe}_{Y_0^6[63]}\left(\boldsymbol{x}^I\right)$, 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 $\left(\mathrm{Coe}_{Y_0^6[0]}\left(\boldsymbol{x}^I\right), \ldots, \mathrm{Coe}_{Y_0^6[63]}\left(\boldsymbol{x}^I\right)\right)$. 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.

Experiment. Since the superpoly of a 64-dimensional cube in an output bit of the 7-round initialization is too large to be practically recovered, we conduct an experiment to recover the superpoly of a 32-dimensional cube in an output bit of the 6-round initialization. In our experiment, we impose the following condition on the public variable:

$$X_3^0[j] = \begin{cases} X_4^0[j] + 1 & \text{if } j = 0 \text{ or } j = 1, \\ X_4^0[j] & \text{if } j \in \{2, 3, \ldots, 63\}. \end{cases}$$

The degree of X_0^6 is 30 when $X_3^0[j] = X_4^0[j]$ for $j \in \{0, 1, \ldots, 63\}$. However, when the first two bits of $X_3^0[j]$ and $X_4^0[j]$ are not equal, the degree of X_0^6 becomes 32. It appears that the appearance of degree-32 monomials is mainly due to the first two bits of $X_3^0[j]$ and $X_4^0[j]$ are not equal. The main reason for this condition is that the superpoly of a 32-dimension cube may be sparse. The cube set we have chosen is $\{x_0, x_1, x_2, x_3, x_{36}, x_{37}, \cdots, x_{63}\}$ and the superpoly involves 64 bits secret key. While the superpoly can be recovered at a time complexity of 2^{96} evaluations according to Lemma 1, we are able to recover it within 36 hours using our superpoly recovery method. The experiment was conducted on a server equipped with 36 Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and 534GB of memory. Some of the programs were run in parallel, as long as the memory is enough. The corresponding source code and the recovered superpoly are available in the GitHub repository at `https://github.com/keyrecoveryofascon/keyrecoveryofascon`.

4.3 Key recovery(online phase)

Considering the cube set $\boldsymbol{x}[I] = (x_0, x_1, \ldots, x_{63}) \in \mathbb{F}_2^{64}$ that we have chosen, we begin by selecting a random 64-bit plaintext P and encrypt it using Ascon, obtaining the ciphertext C. The first 64-bit output after 7 rounds Ascon, namely X_0^7 is computed as $P + C$. According to the inverse of the linear layer in Ascon, Y_0^6 can be obtained from X_0^7. By summing all Y_0^6 over all $\boldsymbol{x} \in \mathbb{F}_2^{64}$, we obtain the 64-bit cube sum, denoted as $(z_0, z_1, \ldots, z_{63})$. Then the key candidates are obtained from $\mathbb{H}[(z_0, z_1, ..., z_{63})]$. Based on the assumption that the superpoly for 7-round Ascon is (almost) balanced Boolean functions, on average, only one key candidate is suggested. Such an assumption is common and also used in the previous works [20,22]. The complexity of this step is 2^{64} queries to Ascon. The remaining 64-bit key can be obtained by an exhaustive search, which requires another 2^{64} queries. The total complexity in the online phase is then 2^{65} queries for 7-round Ascon permutations.

4.4 Complexity evaluation

Time complexity. In our attack, the process of superpoly recovery(offline phase) dominates the main complexity. In Step 1, the time complexity is $2^{80.8}$ calls to the 5-round permutation, which can regarded as $5/7 \times 2^{80.8} = 2^{80.38}$ calls to 7-round permutation. In Step 2, the time complexity is $2^{98.76}$ memory accesses. In a conventional method, one memory access can be regarded as one Sbox operation. In this paper, we regard one memory access to a big table as one single round of Ascon permutation, considering the worst case. But our results are still a significant improvement over previous work. Thus, the time complexity of our attack $2^{98.76}/7 \approx 2^{95.96}$ calls of 7-round permutation. Following computing 64 superpolies in a parallel fashion used in [22], the total time complexity is $2^{95.96}$ calls for 7-round Ascon permutation.

Memory complexity. In our attack, the memory complexity is dominated by the process of the superpoly recovery in Subsection 4.2. In the superpoly recovery process, step 1 requires $2^{73.12}$ bits memory while step 2 requires a memory of $2^{104.85}$ bits. When generating the comparison tables for key candidates, the memory complexity is 2^{70} bits of memory. Therefore, the total memory complexity in our attack is around $2^{104.85}$.

Data complexity. Since our attack utilizes only a 64-dimensional cube, it requires 2^{64} data to compute the cube sum. Consequently, the data complexity is 2^{64}, which satisfies the security claim of Ascon.

5 Our key recovery in the misused setting

In [3], Baudrin et al. proposed a conditional cube attack on Ascon-128 that is capable of recovering the full state prior to the encryption phase within a practical time. However, the extension of this attack to achieve key recovery or forgery in Ascon-128 remains an unresolved open problem. In this paper, we provide an answer of the problem by executing a key recovery attack on Ascon-128 under the condition of the full state prior to the encryption phase being recovered. Our attack is in the misused setting, specifically, the nonce will be reused, and the data exceeds 2^{64}. We first present our key recovery attack on the 8-round initialization of Ascon-128 in Subsection 5.1. Then, in Subsection 5.2, we introduce our key recovery attack when the initialization phase is reduced to 9 rounds.

5.1 Key recovery on the 8-round initialization

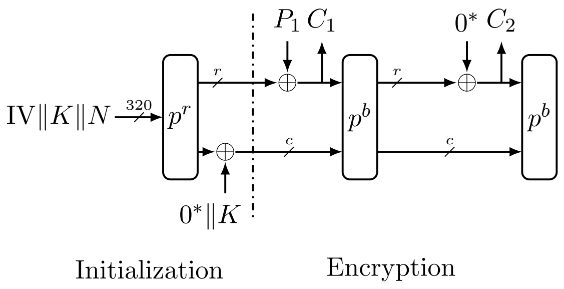

Fig. 4: Our attack mode.

In our attack process, we focus on both the initialization phase and the encryption phase, as illustrated in Figure 4. We omit the associated data phase since the associated data is public and will never affect our attack results. For convenience, the state prior to the encryption phase is denoted as E. Accordingly, the bit in i−th row and j−th column is $E_i[j]$. The details of our attack process on 8-round initialization are as follows.

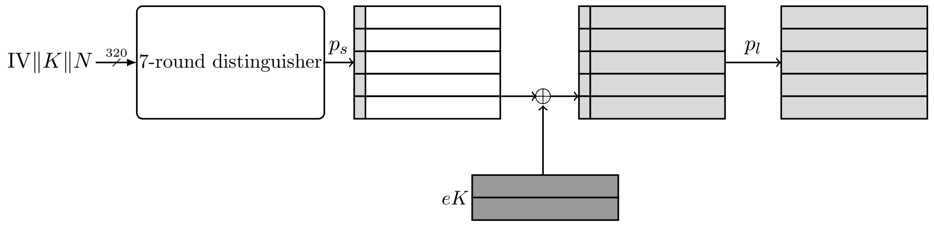

Fig. 5: Key recovery of 8-round initialization of Ascon-128.

Choose a distinguisher for the initialization phase. In [22], Rohit et al, detected the degree upper bound of Ascon in the condition of $X_3^0 = X_4^0$. The 7-round results are shown in Property 4. When a cube size $|I|$ is larger than the degree upper bound ($|I| > deg(f)$) in $X_i^7[j]$, the superpoly of the monomial $\boldsymbol{x}^I$ in $X_i^7[j]$, namely $\mathrm{Coe}_{X_i^7[j]}(\boldsymbol{x}^I)$, must be 0, which distinguishes the 7-round Ascon permutation from random. According to Property 4, we choose a 61 dimension cube in the condition of $X_3^0 = X_4^0$. Then, the value of all superpolies of the 61 dimension cube in all bits of $X_i^7[j]$ is 0. Namely, $\mathrm{Coe}_{X_i^7[j]}(\boldsymbol{x}^I) = \sum_{\boldsymbol{x}[I]\in\mathbb{F}_2^{|I|}} f(\boldsymbol{x},\boldsymbol{k}) = 0$. This gives us a distinguisher for 7-round initialization. In this distinguisher, the number of nonces we used is 2^{61} to compute the cube sums.

Recover the state prior to the encryption phase. For the 2^{61} nonces, we will recover all states prior to the encryption phase, which is denoted as E. The authors in [3] proposed a method to recover the value of E for Ascon-128 with a time complexity less than 2^{40}. To obtain the values of E for 2^{61} different nonces, we employ the method introduced in [3]. In total, we can recover all E at a time complexity less than 2^{101} calls to an 8-round permutation in the initialization phase and a 6-round permutation in the encryption phase.

Recover the secret key. Since the secret key is added to the 4-th and 5-th words after the initialization phase, we can guess the secret key and decrypt E to the end of the chosen distinguisher. The correctness of the guess is determined by whether the guessed key satisfies the end condition of the distinguisher.

As shown in Figure 5, we append one additional round at the end of the distinguisher, targeting the 8-round initialization. The constant operation p_c is simple and does not affect our attack, so we omit it. Furthermore, since both the linear layer p_l and key addition are linear, we swap the order of these two operations in the final round and introduce the equivalent keys eK.

By recovering all information of eK, we can deduce the original secret key using the linear layer of the Ascon permutation. The substitution layer of the Ascon permutation consists of 64 parallel Sbox calls on each state column, allowing the equivalent key bits to be guessed separately. For example, we first guess 2 equivalent key bits in the first column to determine the input to the first Sbox in the 8-th round, i.e., $X_i^7[0]$ for $i \in \{0,\ldots,4\}$.

For 2^{61} states E, the time complexity is 2^{63} Sbox operations. If the guessed bits are correct, the sum of 2^{61} values of $X_i^7[0]$ should be zero for $i \in \{0,\ldots,4\}$, forming a 5-bit filtering condition. As a result, only one candidate remains for the guessed equivalent key bits. For the 64 Sboxes, this procedure requires 2^{69} Sbox operations.

Complexity. In the attack process, the state recovery dominates the complexity. Thus, the data complexity and time complexity are around 2^{101}. The memory complexity is $2^{61+8.32} = 2^{69.32}$ bits.

5.2 Key recovery on 9-round initialization

In the above, we introduce our key recovery target the 8-round initialization by appending one round at the end of a 7-round distinguisher of the initialization. Actually, we can recover the secret key for the 9-round initialization at a complexity less than an exhaustive search by appending two rounds after the 7-round distinguisher. There is a challenge in that the diffusion of the inverse of p_s and p_l in Ascon is relatively fast. Equation 6 presents the ANFs of the inverse of the Sbox which can be easily obtained by SageMath [2].

$$\begin{aligned}
X_0[j] =& Y_4[j]Y_3[j]Y_2[j] + Y_4[j]Y_3[j]Y_1[j] + Y_4[j]Y_3[j]Y_0[j] + Y_3[j]Y_2[j]Y_0[j] \\
&+ Y_3[j]Y_2[j] + Y_3[j] + Y_2[j] + Y_1[j]Y_0[j] + Y_1[j] + 1, \\
X_1[j] =& Y_4[j]Y_2[j]Y_0[j] + Y_4[j] + Y_3[j]Y_2[j] + Y_2[j]Y_0[j] + Y_1[j] + Y_0[j], \\
X_2[j] =& Y_4[j]Y_3[j]Y_1[j] + Y_4[j]Y_3[j] + Y_4[j]Y_2[j]Y_1[j] + Y_4[j]Y_2[j] \\
&+ Y_3[j]Y_1[j]Y_0[j] + Y_3[j]Y_1[j] + Y_2[j]Y_1[j]Y_0[j] + Y_2[j]Y_1[j] \\
&+ Y_2[j] + 1 + a_1, \\
X_3[j] =& Y_4[j]Y_2[j]Y_1[j] + Y_4[j]Y_2[j]Y_0[j] + Y_4[j]Y_2[j] + Y_4[j]Y_1[j] + Y_4[j] \\
&+ Y_3[j] + Y_2[j]Y_1[j] + Y_2[j]Y_0[j] + Y_1[j], \\
X_4[j] =& Y_4[j]Y_3[j]Y_2[j] + Y_4[j]Y_2[j]Y_1[j] + Y_4[j]Y_2[j]Y_0[j] + Y_4[j]Y_2[j] \\
&+ Y_3[j]Y_2[j]Y_0[j] + Y_3[j]Y_2[j] + Y_3[j] + Y_2[j]Y_1[j] + Y_2[j]Y_0[j] \\
&+ Y_1[j]Y_0[j].
\end{aligned} \tag{6}$$

For the inverse of the linear layer, the state Y_i^{r-1}(state before the linear layer) can be regarded as $M_i \times X_i^r$, where i refers to the i-th row, $i \in \{0, 1, \cdots, 4\}$. The binary matrices M_i used in the inverse of the linear layer are 64×64 circulant matrices, which are shown in Appendix B.

Similarly to the key recovery of the 8-round case, we change the order of the last linear layer in 9-th round and the addition key operation by introducing eK. Thus, the last linear layer can be ignored, and the state before the last linear layer, denoted C, can be calculated by the linear layer of Ascon.

Firstly, we try to write out explicitly the mapping f from $C_i[j]$ to any one bit of $X^{(7)}$. without loss of generality, here we take the bit in the first row and first column $X_0^7[0]$ as an example. Since the Boolean function of the mapping f is too complicated, we split it into three steps, clearly indicating how each step maps to the next.

Step 1: Express $X_0^7[0]$ by Y^7. We first express $X_0^7[0]$ in a polynomial of Y^7, according to the ANF of the inverse of the Sbox.

$$\begin{aligned} X_0^7[0] = {} & Y_4^7[0]Y_3^7[0]Y_2^7[0] + Y_4^7[0]Y_3^7[0]Y_1^7[0] + Y_4^7[0]Y_3^7[0]Y_0^7[0] \\ & + Y_3^7[0]Y_2^7[0]Y_0^7[0] + Y_3^7[0]Y_2^7[0] + Y_3^7[0] \\ & + Y_2^7[0] + Y_1^7[0]Y_0^7[0] + Y_1^7[0] + 1 \end{aligned}$$

There are five bits involved in the ANF of $X_0^7[0]$. If we have known the $Y_i^7[0]$ for $0 \le i \le 4$, we can compute the value of $X_0^7[0]$.

Step 2: Express $Y_i^7[0]$ for $0 \le i \le 4$ by X^8.

$$Y_i^7[0] = M_i[0] \cdot (X_i^8[0], X_i^8[1], \ldots, X_i^8[63])^T = \sum_{j \in I_i} X_i^8[j],$$

where I_i is a set of indices corresponding to the coefficient of $M_i[0]$($M_i[j]$ is the j-th row of M_i). According to M_i, $Y_i^7[0]$ can be computed by the sum of 31, 33, 33, 33 and 35 bits of $X_i^8[0]$ respectively, where $i \in \{0, 1, \cdots, 4\}$.

Step 3: Express $X_i^8[j]$ for $0 \le i \le 4$ by Y^8.

According to the ANF of the inverse of Sbox, each bit of $X_i^8[j]$ can be represented by $Y^{(8)}$ easily. Here we only take $X_0^8[j]$ as an example and others are similar.

$$\begin{aligned} X_0^8[j] = {} & Y_4^8[j]Y_3^8[j]Y_2^8[j] + Y_4^8[j]Y_3^8[j]Y_1^8[j] + Y_4^8[j]Y_3^8[0]Y_0^8[j] \\ & + Y_3^8[j]Y_2^8[j]Y_0^8[j] + Y_3^8[j]Y_2^8[j] + Y_3^8[j] \\ & + Y_2^8[j] + Y_1^8[j]Y_0^8[j] + Y_1^8[j] + 1, \end{aligned}$$

In the end, an equivalent key eK is added to Y^8 at the position of 3-th and 4-th rows. Thus, we can build a relation between the Y^8 and C. Towards the first three rows of Y^8, we can directly obtain $Y_i^8[j] = C_i[j]$ where $i \in \{0, 1, 2\}$ and $j \in \{0, \cdots, 63\}$. For the 4-th row of Y^8, we can represent them by $Y_3^8[j] = C_3[j] + eK_0[j]$, where $j \in \{0, \cdots, 63\}$. For the 5-th row of Y^8, we can represent them by $Y_4^8[j] = C_4[j] + eK_1[j]$, where $j \in \{0, \cdots, 63\}$.

As a result, we can further represent $Y_i^7[0]$ for $0 \le i \le 4$ by C and eK. Here, we only take $Y_i^7[0]$ as an example and others are similar.

$$\begin{aligned} Y_0^7[0] &= \sum_{j \in I_0} X_0^8[j] \\ &= \sum_{j \in I_0} Y_4^8[j]Y_3^8[j]Y_2^8[j] + \sum_{j \in I_0} Y_4^8[j]Y_3^8[j]Y_1^8[j] + \sum_{j \in I_0} Y_4^8[j]Y_3^8[j]Y_0^8[j] \\ &+ \sum_{j \in I_0} Y_3^8[j]Y_2^8[j]Y_0^8[j] + \sum_{j \in I_0} Y_3^8[j]Y_2^8[j] + \sum_{j \in I_0} Y_3^8[j] \\ &+ \sum_{j \in I_0} Y_2^8[j] + \sum_{j \in I_0} Y_1^8[j]Y_0^8[j] + \sum_{j \in I_0} Y_1^8[j] + 1, \qquad (7) \\ &= \sum_{j \in I_0} (C_4[j] + ek_1[j])(C_3[j] + ek_0[j])(C_2[j] + C_1[j] + C_0[j]) \\ &+ \sum_{j \in I_0} (C_3[j] + ek_1[j])C_2[j]C_0[j] + \sum_{j \in I_0} (C_3[j] + ek_1[j])C_2[j] \\ &+ \sum_{j \in I_0} (C_3[j] + ek_1[j]) + \sum_{j \in I_0} C_2[j] + \sum_{j \in I_0} C_1[j]C_0[j] + \sum_{j \in I_0} C_1[j] + 1 \end{aligned}$$

According to Equation 7, $Y_0^7[0]$ involves $2\times|I_0| = 62$ bits of eK. It is similar for $Y_i^7[0]$ where $i \in \{1, 2, 3, 4\}$, the involved eK is 33, 33, 33 and 35 bits respectively, which are shown in Figure 6. If we decrypt C to $X_i^7[j]$ directly, many key bits are involved which will cause the complexity increasing quickly. Fortunately, the technique, *partial sum*, can be used to reduce the complexity.

The partial-sum technique was initially introduced by Ferguson et al. in [12] to reduce the time complexity of integral attacks. In Ascon, the small Sboxes are applied separately and the output of $M_i \times X_i^r$ is a linear combination of $X_i^r[j]$. Then, the secret key bits are divided into relatively independent parts and can be guessed one after another, which will reduce the complexity. When a part of the secret key is guessed, the corresponding value obtained by decrypting C, can be compressed with the information and stored for further calculation.

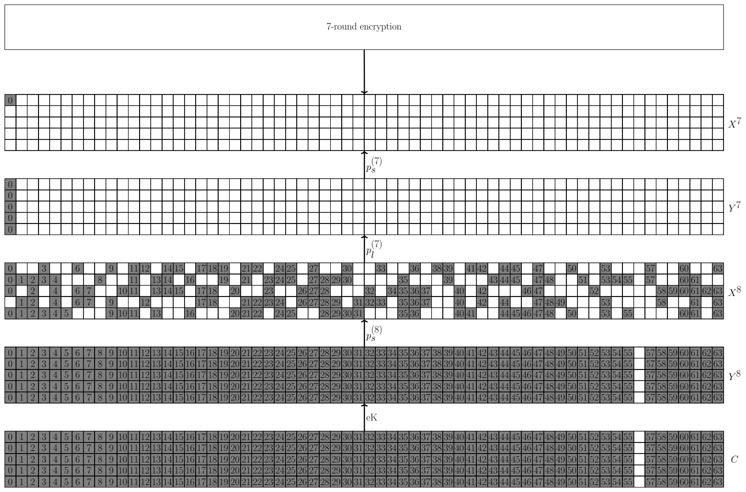

Fig. 6: Key recovery of 9-round initialization.

Guess strategy. According to Figure 6, we can identify only 63 columns are involved to calculate the sum of $X_0^7[0]$, which can be guessed separately. Firstly, for each 2-bit guess, decrypt through $p_s^{(8)}$ and $p_l^{(7)}$ and calculate the contribution to $Y_i^7[0]$ for $0 \leq i \leq 4$. We treat the cost of the operation at once as 2 times 5-bit Sbox computations. Since there are 2^{61} C distributed over 315 bits, the average amount of data reduced each time a 2-bit key is guessed is $2^{0.97}$. The maximum complexity of the step to guess the key of the last column is $2^{127.97}$ 5-bit Sbox computations. We then decrypt $Y_i^7[0]$ to $X_i^{(7)}[0]$ for $0 \leq i \leq 4$. Due to there are 2^{126} bit guessed equivalent key, the time complexity is $2^{(126+5)} \times 2$ 5-bit sbox computation.

The key space by a factor of 2^{-1} for the 126-bit guess. Actually, the superpolies of the 61 dimension cube in $X_i^7[0]$ for $0 \leq i \leq 4$ are all 0. The key space can be reduced by 2^{-5}. For the remaining key space, we just exhaustive search all possibilities.

Complexity. In the state recovery phase, we need recover the 2^{61} states of E, with a time complexity of less than $2^{61} \times 2^{40} = 2^{101}$, which can be omitted. In the guessing equivalent key phase, the complexity is dominated by decrypting $Y_i^7[0]$ to $X_i^{(7)}[0]$ for $0 \leq i \leq 4$, which involves around 2^{132} 5-bit Sbox computations. There are 64 5-bit Sbox computations in p_s of Ascon and we only regard a p_s operation as equivalent to one round permutation. The corresponding complexity represented by 9-round initialization is $2^{132}/(64 \times 9) \approx 2^{122.83}$. Since there are 2^{123} possibilities in the remaining key space, 2^{123} exhaustive search are needed. The total time complexity is $2^{122.83}+2^{123} \approx 2^{123.92}$ calls for 9-rounds permutation of Ascon.

Since there are 2^{61} states to be recovered, the data complexity is 2^{101} and the memory complexity is $2^{61+8.32} = 2^{69.32}$ bits.

6 Conclusion

In this paper, we improve the key recovery attack on Ascon-128 both in misuse-free and misused settings. In a misuse-free setting, we present a faster method to recover the superpolies for a 64-dimension cube in the output bits of 7-round initialization, by which we can recover the full key with a time complexity of $2^{98.76}$ and a data complexity of 2^{64}. Additionally, we employ several techniques to extend state recovery to key recovery, answering the open question of how to transition from full state recovery in the encryption phase to key recovery for Ascon-128 (ToSc Vol 4, 2022). We achieve a 8-round and 9-round key recovery in the nonce misuse scenario, with complexities of 2^{101} and $2^{123.92}$ for Ascon-128, respectively. This represents an improvement of two rounds over previous results. We believe our attacks will be helpful in understanding the security of Ascon.

Acknowledgment. We sincerely thank the anonymous reviewers for providing valuable comments to help us improve the overall quality of the paper. This research is supported by the National Key R&D Program of China(Grant No. 2024YFA1013000), the National Natural Science Foundation of China (Grant No. 62032014, U2336207), Department of Science & Technology of Shandong Province (No.SYS202201), Quan Cheng Laboratory (Grant No. QCLZD202301, QCLZD202306). Kai Hu is also supported by the National Natural Science Foundation of China (62402283), Natural Science Foundation of Jiangsu Province (BK20240420), and Program of Qilu Young Scholars of Shandong University.

References

1. Gurobi optimization. https://www.gurobi.com
2. Sagemath. https://www.sagemath.org/

3. Baudrin, J., Canteaut, A., Perrin, L.: Practical cube attack against nonce-misused ascon. IACR Trans. Symmetric Cryptol. **2022**(4), 120–144 (2022). https://doi.org/10.46586/TOSC.V2022.I4.120-144, https://doi.org/10.46586/tosc.v2022.i4.120-144
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-based encryption, authentication and authenticated encryption. Directions in Authenticated Ciphers pp. 159–170 (2012)
5. Canteaut, A.: Lecture notes on cryptographic boolean functions. Inria, Paris, France **3** (2016)
6. Carlet, C., Crama, Y., Hammer, P.L.: Boolean functions for cryptography and error-correcting codes. (2010)
7. Chang, D., Hong, D., Kang, J., Turan, M.S.: Resistance of ascon family against conditional cube attacks in nonce-misuse setting. IEEE Access **11**, 4501–4516 (2023). https://doi.org/10.1109/ACCESS.2022.3223991, https://doi.org/10.1109/ACCESS.2022.3223991
8. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced keccak sponge function. 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 34. pp. 733–761. Springer (2015)
9. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 278–299. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_16, https://doi.org/10.1007/978-3-642-01001-9_16
10. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of ascon. In: Nyberg, K. (ed.) Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9048, pp. 371–387. Springer (2015). https://doi.org/10.1007/978-3-319-16715-2_20, https://doi.org/10.1007/978-3-319-16715-2_20
11. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021). https://doi.org/10.1007/S00145-021-09398-9, https://doi.org/10.1007/s00145-021-09398-9
12. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of rijndael. In: Schneier, B. (ed.) Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer (2000). https://doi.org/10.1007/3-540-44706-7_15, https://doi.org/10.1007/3-540-44706-7_15
13. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In: Canteaut, A., Ishai, Y. (eds.) 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. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_17, https://doi.org/10.1007/978-3-030-45721-1_17

14. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 12491, pp. 537–566. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_18, https://doi.org/10.1007/978-3-030-64837-4_18
15. Hu, K.: Improved conditional cube attacks on ascon aeads in nonce-respecting settings - with a break-fix strategy. IACR Cryptol. ePrint Arch. p. 743 (2024), https://eprint.iacr.org/2024/743
16. Hu, K., Peyrin, T., Tan, Q.Q., Yap, T.: Revisiting higher-order differential-linear attacks from an algebraic perspective. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14440, pp. 405–435. Springer (2023). https://doi.org/10.1007/978-981-99-8727-6_14, https://doi.org/10.1007/978-981-99-8727-6_14
17. 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: Moriai, S., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 12491, pp. 446–476. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_15, https://doi.org/10.1007/978-3-030-64837-4_15
18. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round keccak sponge function. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II 36. pp. 259–288. Springer (2017)
19. Li, Y., Zhang, G., Wang, W., Wang, M.: Cryptanalysis of round-reduced ASCON. Sci. China Inf. Sci. **60**(3), 38102 (2017). https://doi.org/10.1007/S11432-016-0283-3, https://doi.org/10.1007/s11432-016-0283-3
20. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ASCON. IACR Trans. Symmetric Cryptol. **2017**(1), 175–202 (2017). https://doi.org/10.13154/TOSC.V2017.I1.175-202, https://doi.org/10.13154/tosc.v2017.i1.175-202
21. Liu, M., Lu, X., Lin, D.: Differential-linear cryptanalysis from an algebraic perspective. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 247–277. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_9, https://doi.org/10.1007/978-3-030-84252-9_9
22. Rohit, R., Hu, K., Sarkar, S., Sun, S.: Misuse-free key-recovery and distinguishing attacks on 7-round ascon. IACR Trans. Symmetric Cryptol. **2021**(1), 130–155 (2021). https://doi.org/10.46586/TOSC.V2021.I1.130-155, https://doi.org/10.46586/tosc.v2021.i1.130-155
23. Rohit, R., Sarkar, S.: Diving deep into the weak keys of round reduced ascon. IACR Trans. Symmetric Cryptol. **2021**(4), 74–99 (2021).

https://doi.org/10.46586/TOSC.V2021.I4.74-99, https://doi.org/10.46586/tosc.v2021.i4.74-99

24. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) 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. Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_12, https://doi.org/10.1007/978-3-662-46800-5_12
25. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 250–279. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_9, https://doi.org/10.1007/978-3-319-63697-9_9
26. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 357–377. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_18, https://doi.org/10.1007/978-3-662-52993-5_18
27. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 275–305. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1_10, https://doi.org/10.1007/978-3-319-96884-1_10
28. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) 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. Lecture Notes in Computer Science, vol. 11923, pp. 398–427. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_14, https://doi.org/10.1007/978-3-030-34618-8_14
29. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) 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. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24, https://doi.org/10.1007/978-3-662-53887-6_24
30. Yan, H., Lai, X., Wang, L., Yu, Y., Xing, Y.: New zero-sum distinguishers on full 24-round keccak-f using the division property. IET Inf. Secur. **13**(5), 469–478 (2019). https://doi.org/10.1049/IET-IFS.2018.5263, https://doi.org/10.1049/iet-ifs.2018.5263

Appendix

A MILP model for the three-subset bit-based division property [13]

In the following, we will give the model method of 3BDPwoU trail propagation for some basic operations.

COPY operation. For the COPY operation, 3BDPwoU MILP model $u \rightarrow (v_1, v_2)$ can use the following constraints

$$\begin{cases} u, v_1, v_2 \text{ are binary variables} \\ u = v_1 \vee v_2 \end{cases}$$

XOR operation. For the XOR operation, 3BDPwoU MILP model $(u_1, u_2) \rightarrow v$ can use the following constraints

$$\begin{cases} u_1, v_2, v \text{ are binary variables} \\ v = u_1 + u_2 \end{cases}$$

AND operation. For the AND operation, 3BDPwoU MILP model $(u_1, u_2) \rightarrow v$ can use the following constraints

$$\begin{cases} u_1, u_2, v \text{ are binary variables} \\ u_1 = v, u_2 = v \end{cases}$$

For the nonlinear layer Sbox, we can model the exact vectorial Boolean functions of Sbox in each round.

B The binary matrices M_i

The binary matrices M_i used in the inverse of the linear layer are 64×64 circulant matrices, as follows.

$$\begin{aligned}
M_0 = \mathrm{cir}([&1,0,0,1,0,0,1,0,0,1,0,1,1,0,1,1,0,1,1,1,0,1,1,0,1,1,0,1,0,0,1,0,\\
&0,1,0,0,1,0,1,1,0,1,1,0,1,1,0,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,1]),\\
M_1 = \mathrm{cir}([&1,1,1,1,1,0,0,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,1,1,0,1,1,1,1,0,\\
&0,0,0,1,0,0,0,1,0,0,0,1,1,1,0,1,1,0,0,1,0,1,1,1,0,1,0,0,1,1,0,0]),\\
M_2 = \mathrm{cir}([&1,0,1,0,1,0,1,1,0,0,1,1,0,1,1,1,0,1,1,0,1,0,0,1,0,0,1,1,1,0,0,0,\\
&1,0,1,1,1,1,0,0,1,0,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,1]),\\
M_3 = \mathrm{cir}([&0,1,1,0,1,0,1,1,0,1,0,0,1,0,0,0,0,1,1,0,0,1,1,1,1,0,1,1,1,1,0,1,\\
&1,1,0,1,1,1,0,0,1,0,1,0,1,0,0,1,1,1,0,0,0,1,0,0,0,0,1,0,0,1,0,1]),\\
M_4 = \mathrm{cir}([&1,1,1,1,1,1,0,0,0,1,1,1,0,1,0,0,1,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,\\
&0,0,0,1,1,0,0,0,1,1,0,0,1,1,1,1,1,0,1,0,0,1,0,1,0,0,0,0,1,1,0,1]).
\end{aligned}$$