Practical cube-attack against nonce-misused Ascon[†]

Jules Baudrin, Anne Canteaut and Léo Perrin

Inria, France
{jules.baudrin,anne.canteaut,leo.perrin}@inria.fr

Abstract. In this paper we present a practical cube attack against the full 6-round encryption in Ascon in the nonce-misuse setting. We precise right away that this attack does not violate the security claims made by the authors of Ascon, due to this setting.
Our cryptanalysis is a conditional cube attack that is capable of recovering the full capacity in practical time by carefully studying the monomials of highest degree in the ANF of the full Ascon permutation. Overall, it has a complexity of about 2^{40} adaptatively chosen plaintexts, and about 2^{40} calls to the permutation.
We have implemented the full attack and our experiments confirm our claims.

Keywords: Ascon · cube attack · algebraic attack · ANF

1 Introduction

As the NIST lightweight cryptography standardization effort reaches its final stage, it is crucial to investigate the security of the 10 finalists. In this paper, we focus our attention on Ascon, a family of lightweight primitives. It is also part of the final portfolio of the CAESAR[1] competition [CAE14] which ended in 2019 and had a "lightweight applications (resource constrained environments)" category.

The goal of the Ascon suite is to provide a solution with a "very low memory footprint in hardware and software, while still being fast, robust and secure" [DEMS19]. In the context of lightweight cryptography, the authors offer a trade-off between size, speed and security while focusing mainly on the size.

We focus here on one of the Authenticated Encryption with Associated Data mode (AEAD [Rog02]) of Ascon, namely Ascon-128, which motivates the attack model we choose. In this mode, Ascon aims to provide integrity and confidentiality both with authenticity in an effective integrated manner (Authenticated Encryption [BN00]). On top of that, it ensures the authenticity of associated public data (such as public headers).

Even if they are not studied here, let us note that other AEAD parameter sets are presented in the NIST submission, namely Ascon-128a and Ascon-80pq, as well a hash function, Ascon-Hash. Table 1 presents the values of the parameters, where $k, |S|, r, c, |N|, |T|$ stand for the key, state, rate, capacity, nonce and tag sizes respectively, while a and b are the different numbers of iterations of the permutation.

Due to the low degree of its round function, so-called *cube attacks* and other cryptanalysis directions related to higher-order differentials are tempting attack vectors. Following several works investigating such attacks against Ascon, we add new evidence that this intuition is

[†]The first author is funded by ANR grant ANR-20-CE48-0017 (SELECT).
[1]Competition for Authenticated Encryption: Security, Applicability, and Robustness

Table 1: Parameter sets for recommended versions of Ascon.

	k	$\lvert S\rvert$	r	c	$\lvert N\rvert$	$\lvert T\rvert$	a	b
Ascon-128	128	320	64	256	128	128	12	6
Ascon-128a	128	320	**128**	**192**	128	128	12	**8**
Ascon-80pq	**160**	320	64	256	128	128	12	6

correct. We focus on the terms of highest degree in the Algebraic Normal Form (see below) of the inner permutation of Ascon, and we identify strong and (until now) unknown patterns that allow us to mount a practical capacity-recovery attack against the full primitive. This result comes however with a strong caveat: it assumes significant nonce-misuse, a context in which the authors of Ascon have not made any security claim. Thus, our results do not violate their security claims.

Our paper is structured as follows. First, we present the necessary background (notation, basic concepts, specification of Ascon, and related works) in Section 2. Our attack model is then presented in Section 3, which is followed by the details of our recovery attack in Section 4. Section 5 concludes the paper.

2 Preliminaries

2.1 Notation

- $\mathbb{F}_2$ denotes the finite field with two elements: $\mathbb{F}_2 := \{0,1\}$, while $\mathbb{F}_2^n$ denotes the vector space of dimension n over $\mathbb{F}_2$.
- $\oplus$ indicates a bitwise addition (XOR) of two binary vectors (potentially over a vector space of dimension 1).
- $||$ is used to indicate the concatenation of two bitstrings.
- $[\![0, n-1]\!]$ denotes the set $\{0, \cdots, n-1\}$.
- **wt** stands for Hamming weight, in others words, given a vector of $\mathbb{F}_2^n$, $u = (u_0, \cdots, u_{n-1})$, $\mathbf{wt}(u)$ is defined as $\mathbf{wt}(u) := |\{i, u_i = 1\}|$.

Boolean functions

Given two vectors $u, x \in \mathbb{F}_2^n$, we denote x^u the monomial $x^u := \prod_{i=0}^{n-1} x_i^{u_i}$.

Any Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ can be uniquely represented by its *Algebraic Normal Form* (or ANF, see [Can16]) which is a polynomial over $\mathbb{F}_2[x_0, \cdots, x_{n-1}]/(x_0^2 + x_0, \cdots, x_{n-1}^2 + x_{n-1})$: $f(x) = \sum_{u \in \mathbb{F}_2^n} a_u x^u$ where each a_u lies in $\mathbb{F}_2$. The ANF of a vectorial Boolean function is the collection of the ANF of its coordinate functions.

The Möbius transform enables to go from the ANF to the table of values of a Boolean function (or the other way around), and it also enables to deduce part of the ANF from part of the values taken by the function, as summarized in the following proposition.

Proposition 1 (Computation of a coefficient (*e.g.* [Can16]))**.** *Let f be a Boolean function of n variables whose ANF is $\sum_{u \in \mathbb{F}_2^n} a_u x^u$. Then, for any $u \in \mathbb{F}_2^n$, the coefficient a_u satisfies: $a_u = \sum_{x \preccurlyeq u} f(x)$, where $x \preccurlyeq u$ stands for $x_i \leq u_i, \forall\ i \in [\![0, n-1]\!]$.*

Keyed Boolean functions

In the context of symmetric ciphers, it is often necessary to distinguish controllable *public variables* from inaccessible *secret variables*. We denote public variables v_i, and use different variable names (such as k_i for key variables, or a_i, b_i, c_i, d_i for capacity variables) which will be clarified when used. With such a distinction, when a Boolean function depends on m public variables and n key variables, we prefer to look at it as $f = \sum_{u \in \mathbb{F}_2^m} \alpha_u x^u$, where each α_u lies in $\mathbb{F}_2[k_0, \cdots, k_{n-1}]/(k_0^2 + k_0, \cdots, k_{n-1}^2 + k_{n-1})$.

For a given $u \in \mathbb{F}_2^n$, α_u is referred as the *coefficient of* x^u.

When referring to the degree of a Boolean function, we mean the degree in public variables: $\deg(f) := \max\{\mathbf{wt}(u), \alpha_u \neq 0\}$.

The cube notation

When referring to monomials in public variables, we sometimes use the usual bijection associating a monomial x^u to the set of indices $\{i, u_i = 1\}$. In that case, symbols such as C or $\mathcal{C}$ (for cube) are used to denote the set of indices.

2.2 Ascon

The design of Ascon is based on the permutation-oriented Sponge Duplex mode of operation [BDPV12, BDPA11a]. The authors thus mainly focused on the design of the permutation which also inherits from other well-analyzed and already-standardized designs (such as the permutation of the SHA-3 hash function [BDPA11b]). Figure 1 presents the AEAD encryption mode. The permutation is described below.

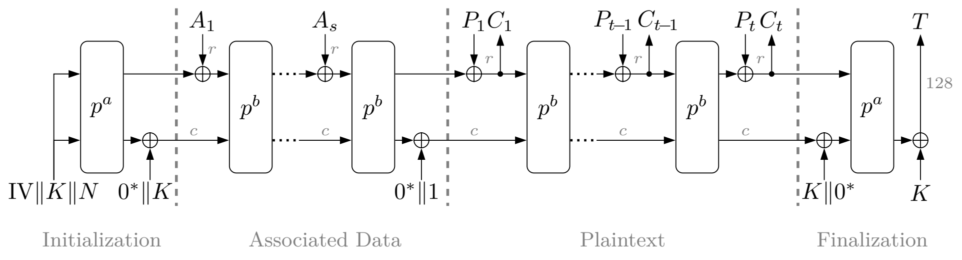


Figure 1: Ascon AEAD encryption[2].

The claimed level of security for Ascon-128 is 128 bits in terms of plaintext confidentiality and plaintext, data and nonce integrity under three hypotheses:

1. the single usage of each nonce;
2. the outputting of a decrypted plaintext only if the tag is correct;
3. the encryption of less than 2^{64} blocks using the same key.

The permutation

The permutation p is the core element of the design and the main object we study in this document. It is built on a Substitution Permutation Network (SPN) structure: p is the composition of a constant addition, a non-linear substitution layer and a linear diffusion transformation: $p = p_L \circ p_S \circ p_C$.

[2]All Ascon figures in this document are highly based (if not identical) on the ones found on the page of the authors [DEMS] and on the TikZ for Cryptographers repository [Jea16].

For an easier understanding, the state is usually decomposed into five 64-bit words, as follows: $S = X_0||X_1||X_2||X_3||X_4$, where X_0 is the most significant word, X_4 the least one.

Constant addition

The constant adding step consists in XORing an 8-bit constant to the word X_2. The constant only depends on the index of the current iteration. There are two different pools of constants to be used either at initialization/finalization or during encryption. They are based on incremented and decremented counters, thus easily computable.

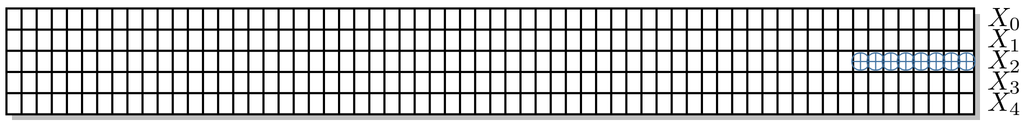


Figure 2: The round constant adding function p_C.

Substitution Layer

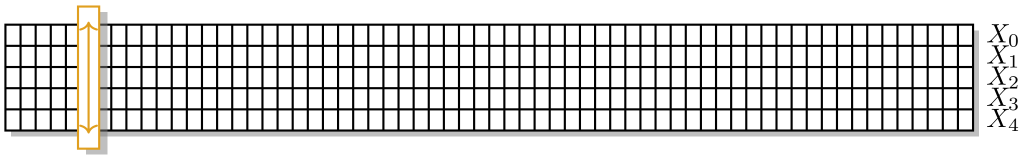


Figure 3: The S-box layer p_S.

The substitution layer is made of 64 parallel calls to a single Substitution-box (S-box) on each column of the state. The S-box used in Ascon is a quadratic 5-bit permutation. Figure 4 presents the algebraic normal form of the S-box in Ascon.

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

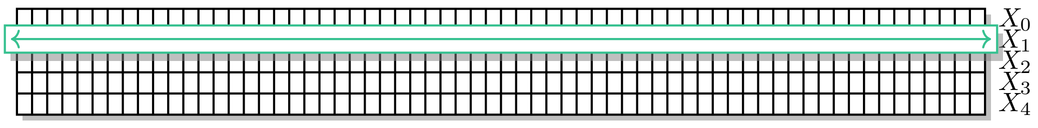
Figure 4: Algebraic normal form (ANF) of the S-box in Ascon.

The low algebraic degree of the S-box (it is quadratic) might lead to cube attacks. This plays a crucial role in the previous cube attacks against Ascon (see Section 2.3). It is also the case in the attack we present in Section 4.

Linear layer

The linear diffusion layer is made of five calls to five different linear functions Σ_i on each row of the state. For each $i \in \{0, 1, 2, 3, 4\}$, $\Sigma_i(X_i)$ is built as the XOR of the i^{th} row X_i

with two rotated versions of itself. The indices of rotations are fixed and only depend on i.



$$\begin{aligned}
X_0 &= X_0 \oplus \ (X_0 \ggg 19) \ \oplus \ (X_0 \ggg 28) \\
X_1 &= X_1 \oplus \ (X_1 \ggg 61) \ \oplus \ (X_1 \ggg 39) \\
X_2 &= X_2 \oplus \ (X_2 \ggg 1) \ \oplus \ (X_2 \ggg 6) \\
X_3 &= X_3 \oplus \ (X_3 \ggg 10) \ \oplus \ (X_3 \ggg 17) \\
X_4 &= X_4 \oplus \ (X_4 \ggg 7) \ \oplus \ (X_4 \ggg 41)
\end{aligned}$$

Figure 5: The linear layer p_L.

2.3 Related work

With the point of view introduced in Section 2, it is possible to study the ANF of a symmetric cipher “coefficient-wise”. In the case of Ascon, it already led to distinguishing attacks (such as *cube-testers* [ADMS09] or *zero-sum distinguishers* [BC11], see Table 3). We are here interested in using this point of view to mount recovery attacks, often named *cube attacks* (see Table 2).

Table 2: Summary of cube-like key-recoveries and state-recoveries against Ascon.

Attack type	Target / Scenario	Number of rounds	Data / Time	Method	Source
Key-recovery	Initialization - NR	5/12	$2^{19}/2^{35}$	Borderline cube	[DEMS15]
Key-recovery	Initialization - NR	6/12	$2^{34}/2^{66}$	Borderline cube	[DEMS15]
Key-recovery	Initialization - NR	5/12	2^{24}	Conditional cube	[LDW17]
Key-recovery	Initialization - NR	6/12	2^{40}	Conditional cube	[LDW17]
Key-recovery	Initialization - NR	7/12	$2^{77.2}$ / $2^{103.9}$	Conditional cube	[LDW17]
Key-recovery †	Initialization - NR	7/12	$2^{77.2}$ / 2^{77}	Conditional cube	[LDW17]
Key-recovery	Initialization - NR	7/12	2^{64} / 2^{123}	Cube	[RHSS21]
Key-recovery†	Initialization - NR	7/12	2^{64} / 2^{97}	Cube	[RS21]
Key-recovery†	Initialization - NR	7/12	2^{63} / $2^{115.2}$	Cube	[RS21]
Key-recovery	Initialization - NM	7/12	? / 2^{97}	Cube-like	[LZWW17]
State-recovery	Encryption - NM	5/6	? / 2^{66}	Cube-like	[LZWW17]
State-recovery	**Encryption - NM**	**6/6**	$\leq \mathbf{2^{40}}$	**Conditional cube**	**Section 4**

† stands for “Weak-key subspace”, NR for nonce-respecting and NM for nonce-misuse

Cube attacks were introduced by Dinur and Shamir [DS09] in order to analyze cryptosystems viewed as polynomial blackboxes. The goal of the attack is to obtain, by the use of chosen *cubes* (*i.e.* sets of public variables), enough linear equations in secret variables that it becomes possible to solve the resulting system.

The attack is thus composed of two stages:

1. Firstly, the offline phase. The goal is to focus on coefficients associated to given monomials (in given output coordinates) which are (almost surely) known to be linear; either from linearity tests or theoretical arguments. Then, thanks to a complete offline access to the black-box (both public and secret variables can be set), the linear expressions can be recovered by interpolation.

Table 3: Summary of the cube-like distinguishers against Ascon.

Attack type	Target / Scenario	Number of rounds	Data / Time	Method	Source
Distinguisher	Permutation	12/12	2^{130}	Zero-sum	[DEMS15]
Distinguisher	Initialization / NR	6/12	2^{33}	Cube-tester	[DEMS15]
Distinguisher	Encryption / NM	6/12	2^{33}	Cube-tester	[DEMS15]
Distinguisher	Initialization / NR	6/12	2^{31}	Cube-tester	[RHSS21]
Distinguisher[†]	Initialization / NR	6/12	2^{17}	Cube-tester	[RS21]
Distinguisher	Initialization / NR	7/12	2^{60}	Cube-tester	[RHSS21]
Distinguisher[†]	Initialization / NR	7/12	2^{33}	Cube-tester	[RS21]

† stands for "Weak-key subspace", NR for nonce-respecting and NM for nonce-misuse

2. Secondly, the online phase. Its objective is to recover the actual values of the previously-targeted coefficients. This is done by querying the online oracle (a limited-access blackbox where only public variables can be set). Thanks to the Möbius transform (see Proposition 1), the value of any coefficient can be recovered in this setting. Finally, with pairs of linear equations/values, an adversary can solve the system and recover (part of) the unknown bits.

Because finding linear coefficients is often highly-restrictive and/or costly, cube attacks evolved into other directions, as illustrated by the following works on Ascon.

Adapting the work [DMP+15] of Dinur *et al.* , Dobrauning *et al.* [DEMS15] mount a cube-attack on 5 rounds of the initialization by using cubes which only depend on a small number of key bits. It enables them to recover the key in a divide-and-conquer manner but the attack is however costly to adapt to a 6-round initialization, because of the quick growth of the number of variables involved in a coefficient.

Li *et al.* [LDW17] continue testing the resistance of Ascon against cube-like attacks by adapting and generalizing the conditional cube attacks against Keccak introduced by Huang *et al.* [HWX+17]. They search for cubes such that all the output coefficients share a linear divisor and manage to find some, without having to recover the entire expression of the coefficients. If such a common divisor exists, an adversary is able to determine its value from the output as it influences the value of the cube-sum vector. Replacing the common linear divisor by a family of linear polynomials in which lies at least one divisor for each coefficient, they adapt their 5/6-round attack to 7 rounds.

Rohit *et al.* present in their paper [RHSS21] the first 7-round misuse-free key-recovery cube-attack on Ascon, reaching the limited number of 2^{64} encrypted blocks of data. In [RS21], 7-round initializations are studied with the weak-key spaces point of view, reaching a number of encrypted blocks stricly lower than the bound imposed by the authors of Ascon.

All of the aforementioned works study the nonce-respecting scenarios and thus focus on the initialization. However, implementation errors will eventually happen and sometimes with high risk: we show in Section 4 that, in the event of a lot of reuse of nonces, confidentiality is compromised.

3 Nonce-misuse setting and attack model

3.1 The attack model

The nonce-misuse scenario assumes, **contrary to the recommendations of the authors**, that a key/nonce pair can be reused many times to encrypt plaintexts. In this situation, the state after initialization can be considered fixed once and for all. We also suppose that

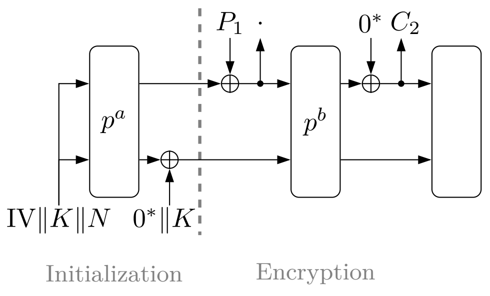


Figure 6: Nonce-misuse attack model.

no associated data is processed. In such a scenario, an adversary can interact with the cipher as follows:

- He can input any **chosen** block on the first row (known as the *rate*, or the outer part of the state). We can thus suppose that, each time, he asks the oracle, first to encrypt a chosen block, and then to immediately encrypt the all-zero block.
- He can recover any ciphertext associated to a chosen plaintext. He can thus omit the first ciphertext and consider the second ciphertext (associated to the all-zero block) as the output of the cipher.

Figure 6 illustrates the situation.

In the nonce-misuse scenario, the goal of the adversary is to recover the *capacity*, that is, the unknown inner part of the state after initialization. If an adversary manages to recover the capacity, then confidentiality is no longer ensured: knowing the first block of associated data (as it is usually the case) as well as the capacity is sufficient to recover all the next plaintexts.

Let us note that a state-recovery does not directly enable a key-recovery (for example by going through the initialization backward) or a tag forgery (by going through the finalization): the XOR with the key at the end (resp. beginning) of the initialization (resp. finalization) prevents such attacks.

3.2 Properties of the nonce-misuse scenario

Four rows of 64 bits need to be recovered. We name them, a, b, c, d from top to bottom. a_i, with $i \in [\![0, 63]\!]$, (resp. b_i, c_i, d_i) represents the ith bit of the first (resp. second, third and fourth) row of the capacity. We use v_i to represent the ith controlled input bit (*i.e.* the ith public variable). Using the point of view introduced in Section 2, we look at the ANF of a coordinate as a "keyed Boolean function": $f = \sum_{u \in \mathbb{F}_2^n} \alpha_u v^u$ where each α_u lies in $\mathbb{F}_2[a_0, \cdots, d_{63}]/(a_0^2 + a_0, \cdots, d_{63}^2 + d_{63})$.

Table 4: ANF of Column i after initialization (left) and after the first S-box layer (right).

$\mathbf{v_i}$	$(a_i+1)\mathbf{v_i}$	$+$	$a_ib_i + a_id_i + a_i + b_i + c_i$
a_i	$\mathbf{v_i}$	$+$	$a_ib_i + a_ic_i + b_ic_i + a_i + b_i + c_i + d_i$
b_i	0	$+$	$c_id_i + a_i + b_i + d_i + 1$
c_i	$(c_i + d_i + 1)\mathbf{v_i}$	$+$	$a_i + b_i + c_i + d_i$
d_i	$a_i\mathbf{v_i}$	$+$	$a_id_i + a_i + c_i + d_i$

Property 1. In the nonce-misuse scenario, the highest degree of a monomial (in any of the 320 output coordinates) at round t, $t \in [\![1,6]\!]$ is 2^{t-1}. Moreover, let us suppose that a highest-degree monomial v^u appears in a coordinate: $\alpha_u \neq 0$. Then, it was obtained as one or more products of two highest-degree terms one round before. α_u can then be seen as a sum, each term of the sum being a product of two coefficients of highest-degree terms one round before.

Property 1 is a consequence of the fact that, during the first S-box layer, no public variable is multiplied with another public variable, as public variables are input row-wise, while the S-box is applied column-wise. So after one round the highest-degree is still 1. Afterwards, the bound doubles at each round because the S-box is quadratic.

Corollary 1. *Let α_u be the coefficient of v^u in any output coordinate after round t. Let us suppose that $\mathbf{wt}(u) = 2^{t-1}$. Then, α_u can be viewed as a sum of products of 2^{t-1} coefficients of degree-1 terms after one round.*

Because all the coefficients of degree-1 terms after one round are either linear in unknown variables or constant, Corollary 1 gives an immediate upper-bound on the degree in the unknown variables of α_u: 2^{t-1}.

Moreover, by looking at the ANF after the first S-box layer given in Table 4, we observe that each product appearing in a coefficient associated to a term of degree 1 after one round has the following structure:

$$\prod_{i,u_i=1} \ell_i \text{ , where } \ell_i \in \{a_i \oplus 1,\ 1,\ c_i \oplus d_i \oplus 1,\ a_i\}.$$

We denote $e_i := c_i \oplus d_i \oplus 1$ for any $i \in [\![0,63]\!]$.

In particular, targeting highest-degree cube can only enable the recovery of the bits of a and $e := c \oplus d$. If we want to recover b, c or d, sub-leading terms can be used. We study those using the following property (and its corollary).

Property 2. In the nonce-misuse scenario, a sub-leading monomial (in any of the 320 output coordinates) at round t, $t \in [\![1,6]\!]$ has degree $2^{t-1} - 1$. Moreover, let us suppose that a sub-leading monomial v^u appears in a coordinate: $\alpha_u \neq 0$. Then, it was obtained as one or more products of one of the following forms:

- a product of two highest-degree terms one round before sharing a public variable as common divisor, or
- a product of a highest-degree term and a sub-leading term one round before.

α_u can then be seen as a sum, each term of the sum being either a product of two coefficients of highest-degree terms or a product of one coefficient of highest-degree term by one coefficient of a sub-leading term one round before.

Corollary 2. *Let α_u be the coefficient of v^u in any output coordinate after round t. Let us suppose that $\mathbf{wt}(u) = 2^{t-1} - 1$. Then, α_u can be viewed as a sum of products of 2^{t-1} coefficients of terms after the second constant addition[3], each product having one of the following forms:*

- *a product of 2^{t-1} coefficients of terms of degree 1 (among the 2^{t-1} terms of degree 1, two are identical, but the two respective coefficients may differ), or*
- *a product of one constant term and $2^{t-1} - 1$ coefficients of terms of degree 1.*

[3]Four constant terms (after the first linear layer) are flipped by the second constant addition.

From there, because the coefficients of highest-degree and sub-leading terms only depend on the coefficients of highest-degree and sub-leading terms after the first linear layer or after the second constant addition, we observe that only the first two constant additions can influence them; the others can thus be omitted. The first addition occurs just after initialization, flipping four unknown bits. Recovering those bits or their flipped values being equivalent, we can omit the first addition (at the (null) cost of flipping four recovered bits).

The situation for highest-degree terms is then completely cyclic, as stated in Property 3.

Property 3 (Rotation invariance)**.** Let us consider the nonce-misuse scenario with the first constant addition being omitted. Let us suppose that $\alpha_u v^u$ appears in the ANF of the jth output coordinate of Row i at round t, and that $\mathbf{wt}(u) = 2^{t-1}$.

Then, for any $k \in [\![0, 63]\!]$, $\beta_{\tilde{u}} v^{\tilde{u}}$ appears in the ANF of the $(j+k)$th[4] output coordinate of Row i at round t; where $\tilde{u}_i = 1 \iff u_{i-k} = 1$ and $\beta_{\tilde{u}}$ is built, based on α_u, by shifting all the indices of the variables by k modulo 64.

Finally, the second constant addition flips four constant terms after the first linear layer. It can thus influence the coefficients of sub-leading terms. However, it is easy to keep track of this influence, as we will see later.

4 Capacity-recovery attack against the full encryption

The attack we propose is made of three steps. All of the steps are based on the same principle: targeting chosen cubes for their particular properties, and then, recovering information about the capacity from the computation of the corresponding cube-sums. However, depending on the stage, the properties we look for are different.

The choice of cubes in the first stage is independent from the capacity, unlike the two remaining stages in which choices are made depending on the information we have already recovered.

The attack we present is thus an **adaptative chosen-plaintext attack** against nonce-misused Ascon.

4.1 First step: Recovering most of the bits a_i and e_i

In this first step, our goal is to recover most of the bits a_i and e_i by targeting cubes of size 32. Those are the only bits present in the coefficients of any monomial of degree 32 (see Corollary 1).

To do so, following Li, Dong & Wang's lead [LDW17], we mount a conditional cube attack. In their paper, they look for cubes such that all the coefficients (in the output coordinates) share a linear common divisor ℓ (only dependent on unknown variables): $f_i = \ell \tilde{f}_i$ for all $i \in [\![0, 63]\!]$.

Here, we focus on a slightly weaker property: we look for cubes such that all the coefficients can be written as: $f_i = \ell_1 F_{i,\ell_1} + \ell_2 F_{i,\ell_2}$, where ℓ_1 and ℓ_2 are linear polynomials (in unknown variables).

In order to find such cubes, we study the first two rounds of Ascon.

In the nonce-misuse setting and after one round, monomials of degree 1 (in public variables) have different coefficients depending on the row they appear on: either $a_i \oplus 1$, 1, e_i or a_i (see Table 4). After L_1 (the first linear layer), the situation is similar because p_L is applied row-wise. Proposition 2 follows immediately.

Proposition 2. *Let us fix a pair of indices (i_0, i_1). Let us suppose that all $v_{i_0} v_{i_1}$ present in the ANF after S_2, are obtained through multiplications of some v_{i_1} present in the ANF*

[4]The addition is computed modulo 64.

after one round by some v_{i_0} coming ***only from Row* j**. *Then, all the coefficients of monomials $v_{i_0}v_{i_1}$ present in the ANF after 2 rounds share the only coefficient of v_{i_0} on Row j after L_1 as common divisor: either $a_{i_0} \oplus 1$, 1, e_{i_0} or a_{i_0}, depending on j.*

The situation for highest-degree terms being completely cyclic in this setting (see Proposition 3), we fix v_0 to play the role of v_{i_0} in Proposition 2. v_0 is referred as the *primary* variable.[5] We can now split the 63 remaining variables into five disjoint sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5$:

1. $\mathcal{S}_1$ contains the variables v_i that are **only** multiplied by some v_0 which was present on **Row 0** after L_1. If v_i belongs to this set, all coefficients of all v_0v_i (present[6]) after S_2 share $a_0 \oplus 1$ as a common divisor.

2. $\mathcal{S}_2$ contains the variables v_i that are **only** multiplied by some v_0 which was present on **Row 4** after L_1. If v_i belongs to this set, all coefficients of all v_0v_i (present) after S_2 share a_0 as a common divisor.

3. $\mathcal{S}_3$ contains the variables v_i that are **only** multiplied by some v_0 which was present on **Row 3** after L_1. If v_i belongs to this set, all coefficients of all v_0v_i (present) after S_2 share e_0 as a common divisor.

4. $\mathcal{S}_4$ contains the v_i that are **not multiplied** by any v_0 during S_2.

5. $\mathcal{S}_5$ contains the remaining variables, that is, the ones that, either are multiplied by some v_0 coming only from Row 1, or by some v_0 coming from multiples rows.

In Table 5 are stored the sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5$. The indices of the variables in each set are deduced by shifting (modulo 64) the index of the primary variable by the offsets given.

Table 5: The sets $\mathcal{S}_i$.

Set	Size	Offsets (relative to the primary variable)
$\mathcal{S}_1$	6	9, 12, 18, 19, 21, 28
$\mathcal{S}_2$	5	7, 24, 41, 43, 52
$\mathcal{S}_3$	5	17, 35, 40, 46, 55
$\mathcal{S}_4$	22	1, 4, 5, 6, 8, 14, 15, 16, 26, 27, 30, 34, 37, 38, 48, 49, 50, 56, 58, 59, 60, 63
$\mathcal{S}_5$	25	2, 3, 10, 11, 13, 20, 22, 23, 25, 29, 31, 32, 33, 36, 39, 42, 44, 45, 47, 51, 53, 54, 57, 61, 62

Now, in order to build a cube of size 32, 31 indices have to be chosen, as we already selected v_0.

Proposition 3. *Let $\mathcal{S}'$ be a subset of $\mathcal{S}_1 \cup \mathcal{S}_3 \cup \mathcal{S}_4$. Let f_i be the coefficient associated to the monomial $M = \prod_{j \in \{0\} \cup \mathcal{S}'} v_j$ in the ith output coordinate after 6 rounds ($i \in [\![0, 63]\!]$). Then, f_i can be written as $f_i = (a_0 \oplus 1) f_{a,i} + e_0 f_{e,i}$.*

Proof. According to Corollary 1, a coefficient associated to a degree-32 monomial after 6 rounds can be seen as a sum of products of coefficients of degree-2 monomials after 2 rounds. For a coefficient associated to M, in each of these products, exactly one coefficient of degree 2 corresponds to a coefficient associated to a term v_0v_i with $i \in \mathcal{S}'$ which was present after 2 rounds. But according to the choice of variables, all the coefficients of terms v_0v_i with $i \in \mathcal{S}'$ are either divisible by $a_0 + 1$ or e_0. The result follows immediately. □

[5]Everything can be applied to another choice of primary variable by simply shifting all the indices.

[6]We can generalize by saying that $a_0 \oplus 1$ is a factor of all the coefficients of all v_0v_i, not only the non-null coefficients, as $a_0 \oplus 1$ also divides 0.

Remarks 1.

1. With the same reasoning, we cannot find a cube of size 32 and guarantee that all the coefficients share a common divisor. Indeed $|\mathcal{S}_4 \cup \mathcal{S}_j| < 31$ for any $j \in \{1, 2, 3\}$.

2. Interestingly, this can however be done when looking at round-reduced initializations. For example, cube $C_1(t)$, which is presented in Section 4 and in Table 7 of [LDW17], have a linear factor which comes from such a choice of variables.

By changing the sets in which we choose the remaining variables, we obtain a similar result.

Proposition 4. *Let $\mathcal{S}'$ be a subset of $\mathcal{S}_{\mathbf{2}} \cup \mathcal{S}_3 \cup \mathcal{S}_4$. Let f_i be the coefficient associated to monomial $M = \prod_{j \in \{0\} \cup \mathcal{S}'} v_j$ in the ith output coordinate after 6 rounds. Then, f_i can be written as $f_i = \mathbf{a_0} \tilde{f}_{a,i} + e_0 \tilde{f}_{e,i}$.*

In Ascon, when computing a cube-sum, a cube-sum vector of size 64 is obtained. It corresponds to the 64 cube-sums associated to a fixed monomial, for each output coordinate. With a choice of cube corresponding to Proposition 3, we can sometimes recover information about the capacity without having to compute the exact polynomial expressions of the 64 coefficients. Indeed, if the corresponding cube-sum vector is not the zero vector, then we are assured that $a_0 \oplus 1 = 1$ or $e_0 = 1$.

However *in theory*, without further studying $f_{a,i}, f_{e,i}$ for all $i \in [\![0, 63]\!]$, we can only use the aforementioned property.

But, *in practice*, we can learn a lot more from the value of the cube-sum vector. Indeed, our experimental trials show that, when $a_0 \oplus 1 = 1$ or $e_0 = 1$, the cube-sum vector is (almost) never all-zero. This means that a "practical reciprocal" can be used and thus enables to recover information about the capacity more often than with the theoretical argument only.

4.1.1 Choosing the cubes $\mathcal{C}_1$ and $\mathcal{C}_2$ and fixing our main assumptions

In order to mount the first stage of our attack and recover most of the bits a_i and e_i, we only need two cubes, $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively chosen according to Propositions 3 and 4.

In Table 7 we give the indices of the variables of the cubes $\mathcal{C}_1$ and $\mathcal{C}_2$.

It has to be observed that the choices of variables were not done in order to obtain some particular results. Rather, it was done using the following algorithm.

1. Take the primary variable v_0.

2. Take the 22 variables of $\mathcal{S}_4$.

3. Take the 5 variables of $\mathcal{S}_3$.

4. Take the remaining 4 variables in $\mathcal{S}_1$ for $\mathcal{C}_1$ (resp in $\mathcal{S}_2$ for $\mathcal{C}_2$) in ascending order.

Table 7: Cubes $\mathcal{C}_1$ and $\mathcal{C}_2$.

Cube	Primary variable	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$
$\mathcal{C}_1$	0	9, 12, 18, 19	-	17, 35, 40, 46, 55	1, 4, 5, 6, 8, 14, 15, 16, 26, 27, 30, 34, 37, 38, 48, 49, 50, 56, 58, 59, 60, 63	-
$\mathcal{C}_2$	0	-	7, 24, 41, 43	17, 35, 40, 46, 55	1, 4, 5, 6, 8, 14, 15, 16, 26, 27, 30, 34, 37, 38, 48, 49, 50, 56, 58, 59, 60, 63	-

We did not study other choices of cubes, as those ones are enough to mount the first step of our capacity-recovery attack. However, we do not expect other choices corresponding to Proposition 3 or 4 to behave in a drastically different manner.

Now that $\mathcal{C}_1$ and $\mathcal{C}_2$ are fixed, and before explaining the first stage of our attack, we state the assumptions under which the attack is based, present the experimental results obtained and specify why those results underpin our hypotheses.

Assumption 1. *Let us consider the nonce-misuse scenario. Let us suppose that the cube-sum associated to the cube $\mathcal{C}_1$ (see Table 7) is the zero vector. Let us suppose that we guess that $a_0 \oplus 1 = 0$ and $e_0 = 0$. Then, the probability of making a wrong guess is negligible.*

Assumption 2. *Let us consider the nonce-misuse scenario. Let us suppose that the cube-sum associated to the cube $\mathcal{C}_2$ (see Table 7) is the zero vector. Let us suppose that we guess that $a_0 = 0$ and $e_0 = 0$. Then, the probability of making a wrong guess is negligible.*

4.1.2 Underpinning our assumptions

In order to support our assumptions, we present and comment the experimental results we obtained.

Observation 1. *Let us consider the following experiment: take a random capacity, then compute the cube-sum vector associated to the cube $\mathcal{C}_1$.*

If the cube-sum vector is all-zero, then guess that $a_0 \oplus 1 = 0$ and $e_0 = 0$. If the cube-sum is not null, do not guess anything.

Our experiment gave the following results: 4000 capacities were tested. A guess was made about $\frac{1}{4}^{\text{th}}$ of the time (1037/4000) and 100% of the guesses were actually right.

Observation 1 suggests that it is possible to detect each time that $a_0 \oplus 1 = 0$ and $e_0 = 0$. But, more importantly, it raises no false-detection under Assumption 1.

Observation 2 gives a similar result for the cube $\mathcal{C}_2$.

Observation 2. *Let us consider the following experiment: take a random capacity, then compute the cube-sum vector associated to the cube $\mathcal{C}_2$.*

If the cube-sum vector is all-zero, then guess that $a_0 = 0$ and $e_0 = 0$. If the cube-sum is not null, do not guess anything.

Our experiment gave the following results: 4000 capacities were tested. A guess was made about $\frac{1}{4}^{\text{th}}$ of the time (1002/4000) and 100% of the guesses were actually right.

Without having access to the algebraic expression of any of the polynomials $f_{a,i}$, $f_{e,i}$, $\tilde{f}_{a,i}$, $\tilde{f}_{e,i}$, it is hard (if not impossible) to guarantee that an all-zero cube-sum vector for $\mathcal{C}_1$ implies $a_0 = 1$ and $e_0 = 0$ (or $a_0 = 0$ and $e_0 = 0$ for $\mathcal{C}_2$). However, our experiments show that an all-zero cube-sum vector has a really low probability of happening if $a_0 = 0$ or $e_0 = 1$ (or $a_0 = 1$ or $e_0 = 1$ for $\mathcal{C}_2$). In Figure 7, we present the distribution of the Hamming weights of the cube-sum vectors for random capacities, depending on the cube tested and the actual values of the targeted bits of the capacity. We compare them to the probability mass function of a binomial distribution with parameters $n = 64,\ p = 0.5$.

Overall, we observe, when the values of the bits are not the targeted values, that the cube-sum is never all-zero. More importantly, we observe that the distributions of the Hamming weights of the vectors, when (a_0, e_0) is not equal to the targeted pair of values, behave as the distribution of the Hamming weights of random vectors: they seem to follow a binomial distribution with parameters $n = 64,\ p = 0.5$. In particular, they have an average around half of the size of the vector, namely, 32 and a low variance around it. The observation of these bell-shaped plots is actually due to another observation made on the batches of cube-sum vectors.

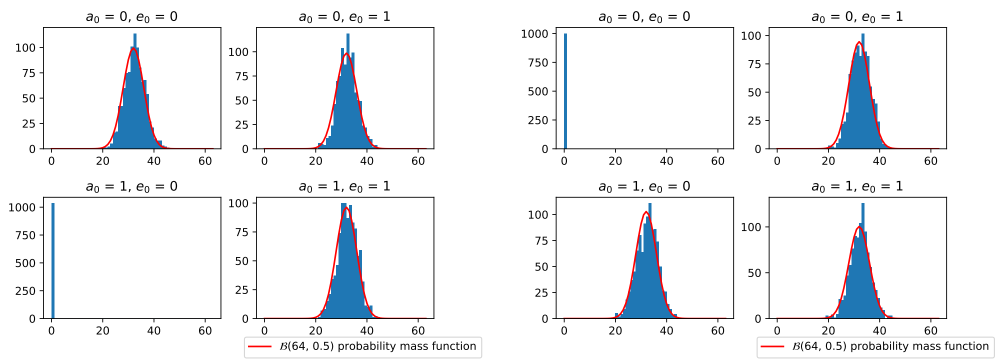


Figure 7: Distribution of the Hamming weight of cube-sum vectors for $\mathcal{C}_1$ (left) and $\mathcal{C}_2$ (right) depending on the values of a_0/e_0, for 4000 capacities chosen uniformly at random.

Observation 3. *Let us consider the experiments described in Observations 1. Let us suppose that* $(a_0, e_0) \neq (1, 0)$*. Then, the probability of each bit of the vector to be set is about* $\frac{1}{2}$*.*

Observation 4. *Let us consider the experiments described in Observations 2. Let us suppose that* $(a_0, e_0) \neq (0, 0)$*. Then, the probability of each bit of the vector to be set is about* $\frac{1}{2}$*.*

Figure 8 illustrates Observations 3 and 4.

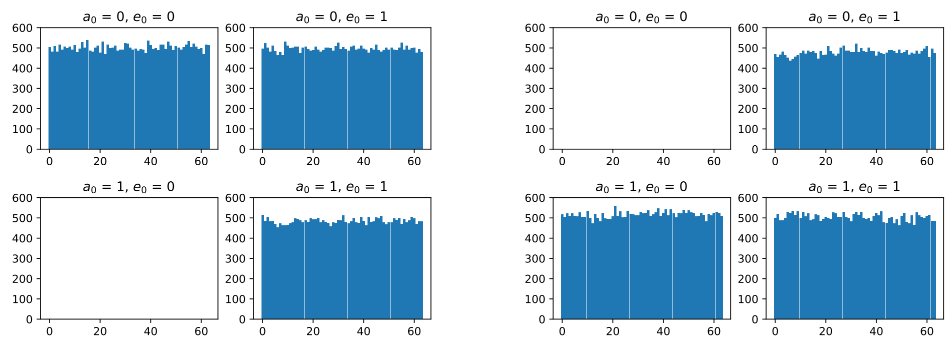


Figure 8: Number of times each coefficient (on the output row) associated to cubes $\mathcal{C}_1$ (left) and $\mathcal{C}_2$ (right) is equal to 1, depending on the values of a_0/e_0, for 4000 capacities chosen uniformly at random.

Thus, for a fixed cube $\mathcal{C}_1$ or $\mathcal{C}_2$, in each of the three cases in which no guess can be made, each coefficient looks balanced. In other words, $f_{a,i}$, $f_{e,i}$ and $f_{a,i} + f_{e,i}$ (resp. $\tilde{f}_{a,i}$, $\tilde{f}_{e,i}$ and $\tilde{f}_{a,i} + \tilde{f}_{e,i}$) look balanced.

When no guess is supposed to be made, if the Hamming weight of the cube-sum vector was able to take any extreme value with a relatively high probability, we feel that we would have been able to observe it. It was not the case. As the capacities were picked uniformly at random for our experiments, we expect this survey to be quite representative of the overall situation, and from this, Assumptions 1 and 2 are made.

4.1.3 Mounting the first stage of the attack

From an adversary's point of view, combining Observations 1 and 2 under Assumptions 1 and 2 enables to recover, 1 or 2 bits of the capacity, depending on their values, through the following method.

1. First, compute the cube-sum vector associated to $\mathcal{C}_1$. If $a_0 = 1$ and $e_0 = 0$, then the cube-sum will be the zero vector and the guess made will be right. Otherwise, if $(a_0, e_0) \neq (1, 0)$, the cube-sum will not be all-zero and no wrong guess will be made, according to Assumption 1.
2. First, compute the cube-sum vector associated to $\mathcal{C}_2$. If $a_0 = 0$ and $e_0 = 0$, then the cube-sum will be the zero vector and the guess made will be right. Otherwise, if $(a_0, e_0) \neq (0, 0)$, the cube-sum will not be all-zero and no wrong guess will be made, according to Assumption 2.
3. Finally, if both cube-sums are distinct from the zero vector, guess that $e_0 = 1$. This guess is always, and under no assumption, right.

In the end, at the cost of at most 2 cube-sums of size 32, an adversary can make a guess on the value of e_0 or guesses on the values of a_0 and e_0 and he will be, under Assumptions 1 and 2, always right.

The index of the recovered bits is subject to the choice of the primary variable, but according to the cyclic behavior of highest-degree terms (see Proposition 3), everything we just explained remains identical if we choose another primary variable. Under assumptions adapted from Assumptions 1 and 2 for all the other choices of primary variable, an adversary can thus guess, with no error, all the bits e_i and, in average, half of the bits a_i.

Overall, for a cost of $64 \times 2 = 128$ cube-sums of degree 32, we can expect to recover no fewer than 64 bits and up to 128 bits without making any wrong guesses. The worst case happens when $e_i = 1$ for all $i \in [\![0, 63]\!]$, whereas the best case happens when $e_i = 0$ for all $i \in [\![0, 63]\!]$. This translates to a cost majored by $128 \times 2^{32} = 2^{39}$ both in time and data, and a negligible cost in memory.

4.2 Recovering the remaining a_i

During the first phase, we manage to recover most of the bits a_i and e_i. But, in order for the next stage to work properly, it is necessary to recover some of the not-yet-recovered values of a_i, that is, the values of a_i in columns in which $e_i = 1$.

To do so, we mount a cube-like attack by targeting cubes of size 32. Contrary to the first step, the choices of cubes here depend on the capacity. So, **this stage is adaptative**.

Here, the choice of the cubes follows a simple and unique rule: at least one of the indices of the chosen cubes needs to be the index of an unknown a_i.

Once a cube C is chosen, the polynomial expressions of the associated coefficients are computed. Contrary to the first stage, the knowledge of most of the bits of a and e enables us to lower the difficulty of recovering the expressions, which seemed impossible before.

For example, it can be done by initializing a state after S_1 with the necessary variables v_i, a_i and the necessary values of the already-recovered a_i, e_i. The computation is then done by keeping, after each round, only the highest-degree terms which divide C, as they are the only ones able to influence the coefficients associated to C. Also, only the quadratic part of the S-box (the quadratic terms of the S-box) are necessary, as the linear terms of the S-box do not enable the maximal (and necessary) growth of the degree in each round.

This recovery method can be considered as an implemented version of the *Partial Polynomial Multiplication* method introduced by Rohit *et al.* [RHSS21].

Once the coefficients are recovered, the cube-sum vector can be computed, and then, the system of equations can be established and solved.

A trade-off between time and memory used offline (coefficients recovery, solving of the system) and data used online to compute the cube-sum is possible. Indeed, when choosing a cube, the more unknowns there are, the less cubes will be necessary to recover the remaining a_i, as they are thus recovered by groups. However, when the number of unknowns grows, the expressions of the coefficients becomes more intricate: the number of terms grows (see Proposition 5) making the recovery last longer and more costly in term of memory. Moreover the degrees of the coefficients also increase as they are obviously upper-bounded by the number of unknowns. Both phenomena have a tendency of making the solving of the system harder.

In our experiments, we tried to limit the number of unknowns variables to 4 or 5. In these cases, the recovery of the expression of a coefficient is fast (a few seconds on a laptop) and does not require too much RAM (16-Go RAM laptop). With 4 or 5 unknowns, it even seems that the systems are quickly solved and can be solved without using (and thus computing) all of the 64 polynomials.

However, these costs are subjected to the Hamming weight of e: if e has a low Hamming weight, a lot of unknown variables have to be used, because not enough variables were recovered during the first stage. Nevertheless, the probability of having a vector e of Hamming weight greater than or equal to 23 (meaning being able to use $32 - 23 = 9$ unknown variables or less) is high: $2^{-64}\left(\sum_{i=23}^{64}\binom{64}{i}\right) \approx 0.992$. In more than 91% of the cases 5 unknowns can be used. Furthermore, choosing the cubes at this stage, one after the other, enables to use the bits recovered at the very moment and thus lowers the difficulty of finding the next ones.

This enables the recovery of the remaining a_i at the online cost of less than 64 cube-sum computations of degree-32 cubes.

It can happen that the very last bits of a are hard to recover because finding non-constant coefficients is harder when most of the bits are known. If it is the case, they can remain unknown for the moment as only a few values of a_i when $e_i = 1$ are necessary for the next stage to work properly. The remaining unknown values would then be recovered during the third step, together with the bits of b and c.

4.3 Recovering most of the bits b_i and c_i for all i

Reaching the third step, we expect that all the bits e_i are recovered, as well as almost all a_i. We now focus on recovering bits b_i and c_i for all i. When e_i is known, because $e_i = c_i \oplus d_i \oplus 1$, it is sufficient to recover c_i in order to recover both c_i and d_i.

In order to recover all the bits of b and c, we mount a cube-like attack by targeting **sub-leading cubes of size 31**. **This stage is also adaptative** and depends on the capacity.

According to Corollary 2, a coefficient of a degree-31 monomial after 6 rounds can be seen as a sum of products, each product depending on coefficients of degree-1 terms after one round and sometimes on constant terms after the second constant addition. But, all the coefficients of degree-1 terms after one round depend only on variables a_i, e_i. Choosing variables v_i such that all the corresponding values of a_i, e_i are known enables to look at the corresponding coefficients of degree-1 terms as constant coefficients. Corollary 2 can then be simplified.

Corollary 3. *Let α_u be the coefficient of v^u in any output coordinate after round t. Let us suppose that $\mathbf{wt}(u) = 2^{t-1} - 1$. Moreover, let us suppose that a_i and e_i are known for all i such that $u_i = 1$. Then, α_u can be viewed as a sum, each term of the sum having one of the following forms:*

- *a 0 or 1 constant, if the term corresponds to a former product of 2^{t-1} coefficients of terms of degree 1, or*

- *a quadratic polynomial in b_i, c_i, b_ic_i for all $i \in [\![0, 63]\!]$ (and possibly some a_i, a_ib_i, a_ic_i for the few values of i for which a_i is still unknown) if the term corresponds to a former product of one constant term and $2^{t-1} - 1$ coefficients of terms of degree 1 after the second constant addition.*

The second point of Corollary 3 comes from the expressions of the constant terms in the ANF after the first S-box layer (see Table 4), once d_i is expressed in terms of variable c_i and the known value of e_i.

From there, we observe, when a cube of size 31 is chosen such that all the corresponding a_i and e_i are known, that the coefficients will be at most quadratic. Moreover, their polynomial expressions will be simple: if all the values of a_i were recovered during the previous stage, they will depend on up to 128 linear terms and 64 quadratic terms. If r values of a_i remain unknown at this stage, the polynomials will depend on up to $128 + r$ linear terms and $64 + 2r$ quadratic terms.

In order to avoid constant coefficients, the only constraint is to select among the cube variables, at least one v_i such that $e_i = 1$, because of Proposition 5.

Proposition 5. *Let us consider the nonce-misuse scenario. Let v^u be a monomial of degree 16 in public variables. Let α_u be a coefficient associated to v^u in any coordinate after 5 rounds. Then, each product (Corollary 1) appearing in the algebraic expression of α_u has at least one variable e_i (with i such that $u_i = 1$) as divisor.*

Proof. This can be proven by keeping track, round after round, of the coordinates in which all coefficients associated to highest-degree monomials which divide v^u, have the searched property. We thus obtain Table 8. □

Table 8: Bounds on the number of e_i variables appearing in each product (Corollary 1) of the coefficient of any 2^{t-1}-degree monomial at round t.

Round	1	2	3	4	5	6
Row r_0	0	0	1-2	1-4	**1**-7	2-13
Row r_1	0	1	1-2	0-3	**1**-6	3-15
Row r_2	x	1	0-1	1-2	**3**-7	3-15
Row r_3	1	0-1	0-1	1-3	**2**-8	2-15
Row r_4	0	0	1	2-4	**1**-7	2-13

When choosing C, if all indices are associated to zero bits of e, then, after 5 rounds, all the coefficients associated to degree-16 monomials dividing C will be zero. Thus all the coefficients of C after 6 rounds will be constant. If it is not the case, it is expected that some of the products of the coefficients will not be constant, and thus it is expected that some of the coefficients will not be constant. The number of bits set to 1 in e associated to the chosen cube variables influences the offline cost: if the number of indices associated to bits set to 1 in e is small, then the recovery of the expressions is fast because only a small amount of the products will not be equal to zero. On the contrary, if this number is high, then the amount of non-zero products is higher, the intermediate coefficients are less sparse, the recovered expressions tend to be more independent, and they thus enable the recovery of more information.

In the end, at least 128 equations are needed to recover the 128 bits of b and c, so at least two sets of 64 coefficients and two cube-sums of size 31 have to be computed. Indeed, contrary to the case of 32-degree coefficients, the dependency of 31-degree coefficients is not only limited to variables b_i, c_i for i such that $u_i = 1$. This is because, through the first linear layer, constant terms are shuffled (contrary to terms of degree 1 which are only copied in other coordinates).

From our experiments, using about 5 to 10 cubes is enough to decrease the number of unknown bits from 128 to less than 10.

4.4 Finalizing the recovery

When the three steps are done, a small number of unknown capacity bits is expected to remain. They can be recovered through an exhaustive search.

In the end, the complexity of the attack is dominated by the cost of the cube-sums of degree-32 monomials, that is, by $(128 + 64)2^{32} \approx 2^{7.6+32} < 2^{40}$ both in time and data.

5 Conclusion

To the best of our knowledge, this document presents the first state-recovery attack on the full 6-round encryption of ASCON, under the assumption that a key-nonce pair is reused many times. The main technique we use is our new way of searching for conditional cubes. It is based on the splitting of the pool of public variables, depending on the structure of their coefficients after two rounds. Moreover, this method gives a new point of view on previous choices of conditional cubes for ASCON, such as the cube $C_1(t)$ in the work of Li, Dong & Wang [LDW17].

The first step of our approach enables the recovery of 64 to 128 bits of the internal state. Thanks to this knowledge, the computation of the previously-inaccessible part of the ANF is now possible. The recovery of the remaining bits thus follows.

Overall, our adaptative chosen-plaintexts attack can recover the full capacity, that is, 256 unknown bits, with an online time and data complexity lower than 2^{40}. In order to speed-up the adaptative offline choices, or lower the cost in data, some trade-off were presented in this paper.

Our attack works: we have implemented it and it indeed successfully returns the secret capacity bits with the expected complexity.

It is already known that the nonce-misuse scenario enables the recovery of any plaintext at the cost of an adapted query for each block that needs decrypting [VV18]. In our attack, once the inner-state has been recovered, an adversary can recover all the next plaintexts from the ciphertexts, without any further interaction with the encryption oracle.

Nevertheless, this result does not go against the claims made by the authors of ASCON [DEMS19], nor does it unsettle the confidence gained through the previous studies, as it is clearly stipulated in the specifications that the nonce must not be "repeated for two encryptions under the same key".

However, we believe that such attacks can be interesting because implementation errors cannot be completely ruled out.

Finally, we would like to mention possible counter-measures against this kind of recovery based on cube-attack. For example, by adding one more round to the permutation while processing the associated data and encrypting the plaintext blocks, it becomes much more complex to recover information. Indeed in the nonce-misuse scenario, and after seven rounds, only a single highest-degree term could be exploited with a cost in data of 2^{64}, reaching the limitation given by the authors for nonce-respecting scenarios.

Acknowledgments

The first author is funded by ANR grant ANR-20-CE48-0017 (SELECT).

An independent team of cryptographers from NIST analyzed ASCON and obtained results that have some overlap with ours [CKT22].

References

[ADMS09] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Orr Dunkelman, editor, FSE 2009, volume 5665 of LNCS, pages 1–22, Leuven, Belgium, February 22–25, 2009. Springer, Heidelberg, Germany.

[BC11] Christina Boura and Anne Canteaut. Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, SAC 2010, volume 6544 of LNCS, pages 1–17, Waterloo, Ontario, Canada, August 12–13, 2011. Springer, Heidelberg, Germany.

[BDPA11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions, 2011. https://keccak.team/sponge_duplex.html.

[BDPA11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, 2011. https://keccak.team/keccak.html.

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, SAC 2011, volume 7118 of LNCS, pages 320–337, Toronto, Ontario, Canada, August 11–12, 2012. Springer, Heidelberg, Germany.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, ASIACRYPT 2000, volume 1976 of LNCS, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[CAE14] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, March 2014. https://competitions.cr.yp.to/caesar.html.

[Can16] Anne Canteaut. Lecture Notes on Cryptographic Boolean Functions, 2016. https://www.rocq.inria.fr/secret/Anne.Canteaut/poly.pdf.

[CKT22] Donghoon Chang, Jinkeon Kang, and Meltem Sönmez Turan. A New Conditional Cube Attack on Reduced-Round Ascon-128a in a Nonce-misuse Setting, 2022. https://csrc.nist.gov/Events/2022/lightweight-cryptography-workshop-2022.

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

[DEMS15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In Kaisa Nyberg, editor, CT-RSA 2015, volume 9048 of LNCS, pages 371–387, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany.

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Technical report, National Institute of Standards and Technology, 2019. https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In Elisabeth Oswald and Marc Fischlin, editors, EUROCRYPT 2015, Part I, volume 9056 of LNCS, pages 733–761, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, EUROCRYPT 2009, volume 5479 of LNCS, pages 278–299, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[HWX+17] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional cube attack on reduced-round Keccak sponge function. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, EUROCRYPT 2017, Part II, volume 10211 of LNCS, pages 259–288, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

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

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

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

[RHSS21] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-free key-recovery and distinguishing attacks on 7-round Ascon. IACR Trans. Symm. Cryptol., 2021(1):130–155, 2021.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, ACM CCS 2002, pages 98–107, Washington, DC, USA, November 18–22, 2002. ACM Press.

[RS21] Raghvendra Rohit and Santanu Sarkar. Diving deep into the weak keys of round reduced ascon. IACR Trans. Symmetric Cryptol., 2021(4):74–99, 2021.

[VV18] Serge Vaudenay and Damian Vizár. Can caesar beat galois? - Robustness of CAESAR candidates against nonce reusing and high data complexity attacks. In Bart Preneel and Frederik Vercauteren, editors, ACNS 18, volume 10892 of LNCS, pages 476–494, Leuven, Belgium, July 2–4, 2018. Springer, Heidelberg, Germany.