Cryptanalysis of a lightweight primitive submitted to the NIST standardization process: ASCON

Jules Baudrin

Master Thesis

Cryptanalysis of a lightweight primitive submitted to the NIST standardization process: Ascon

Submitted to University of Versailles Saint-Quentin-en-Yvelines (UVSQ) in partial fullfillment of the requirements for the Degree of Master in Applied Algebra

Jules Baudrin (M2 Algèbre appliquée, UVSQ)

supervised by Anne Canteaut & Léo Perrin (INRIA)

August 2021

Abstract

In this document I present the main aspects of my first research experience as an intern at Inria supervised by Anne Canteaut & Léo Perrin within the COSMIQ team from March to August 2021.

We focus on Ascon, a lightweight symmetric cipher submitted to the current NIST standardization process. We mostly analyze different properties which could lead to cube attacks. A presentation of Ascon and a non-exhaustive literature review are drawn as complements to the chronological overview of the work done during these six months. They altogether enable to assess the hindsight obtained month after month.

This internship is part of the requirements to get my Master's degree from UVSQ (Université de Versailles Saint-Quentin-en-Yvelines, M2 Algèbre appliquée).

Acknowledgments

First of all, I would like to thank Anne Canteaut & Léo Perrin for their support and involvement during this internship; I could not have hoped for a better first research experience! Thank you as well for your renewed confidence, I am really looking forward to spending the next three years at your side.

I also would like to thank every member of the COSMIQ team for their kindness and enthusiastic welcome; even with a single on-site day per week at first, you all contribute to make me feel as I belong!

Finally, I would like to thank Christina Boura and Yann Rotella who put me in contact with the team and who supported my application; this was a valued help. I really hope we will be able to work together in the future.

Contents

1 Context and overview

As stated in this document's title, our main concern during the last six months was to analyze the security of a lightweight primitive named Ascon (designed by Dobraunig, Eichlseder, Mendel and Schläffer [DEMS19]), which is one of the finalists of a current standardization process from the American National Institute of Standards and Technology (NIST, [NIS17]). However, before diving into this state-of-the-art subject, I would like to present the context around Ascon. The feeling I have after this 6-month experience is that the better I understood and clarified the overall setting, the easier it was for me to focus on what our questions really were, and to try to answer them.

This first chapter thus intends to give a proper survey to start with. It will be followed by a brief literature study of previous works about symmetric cryptanalysis (more precisely about integral and cube attacks which were our main focus). Then, I will give a detailed description of Ascon in order to finally summarize our research work and present a few ideas for future developments.

I Symmetric cryptography and cryptanalysis

For a few decades now, discussing and sharing information secretly and remotely has really become a daily matter: on a communication channel which may be accessible to unwanted listeners (such as the World Wide Web), one often needs to exchange data in such a way that only the legitimate recipient can understand it, even if someone else were to eavesdrop on the exchange. This need, *confidentiality*, is the core concept of cryptography since its early stage. In order to achieve it, the oldest methods known shared a common principle: before being able to communicate secretly, the sender and the recipient need first to agree on a common framework which includes a common secret (called *secret key*) shared only by the two protagonists.

Those methods are known as *symmetric encryption* or *secret-key encryption*, the former insisting on the common information shared beforehand while the latter insists on the fact that this information must be known only by them. On the contrary, the framework itself does not need to be secret. Even more, it is commonly accepted that in order to properly analyze the security of an encryption method, one must not base one's reasoning on some unknown elements of the framework. Rather, the only secret should be the key. This is known as *Kerckhoffs' principle*. Following this framework, the secret key (shared by the two parties) is used to both *encrypt* and *decrypt*, in other words: first to transform the message in an unintelligible shape (called *ciphertext*, this is done by the sender) and then to transform it back to its genuine shape (called *plaintext*, this is done by the recipient).

The goal of an adversary who would like to eavesdrop on the exchange is to discover as much information as possible about the original message sent without the initial knowledge of the key (if the key is known at the beginning, then nothing is hidden as the adversary knows as much as the recipient and both are able to decrypt the message). The most powerful attacks are key-recoveries which enable the adversary to gain during

communication the same power as the recipient, but there exist a lot of other ways to break cryptosystems.

In order to build more secure systems, it is necessary to analyze the security of the primitives designed. This is not an easy task for many reasons. First, we need formal definitions and understanding of what *security* means. To do so, a lot of parameters have to be taken into account: what are the resources of the adversary (computational power, data accessible...), which piece of information can or should not leak, or even what are we actually trying to achieve (we spoke about confidentiality but *integrity* and *authenticity* are also major concerns of cryptographers). Secondly, trust and confidence in a symmetric cryptosystem are gained by evaluating its resistance against a lot of attacks. The more unsuccessful ones, the more we gain trust. The effort can thus seem endless: billions of unsuccessful attacks will only mean that the system is secure *enough* but a single successful one could ultimately lead to an untrustworthy feeling... This is the role of *cryptanalysis*.

II Lightweight cryptography in the IoT era

Even though cryptography is thousand of years old, it was reserved until recently to political and military contexts. The first "computers" were invented in order to break military ciphers during World War II, and since then started what we can call modern cryptography: the scientific study of secret exchanges. Shannon's work [Sha49] is considered as one of the first major theoretical milestones in cryptography. Through his communication theory, he built the first mathematical approach to rigorously study ciphers and their security (and even actual primitive designs are still inspired by his work, as we will see later).

Since, thanks to the development of computer science and even more to the growth of the Internet, cryptography has really become an everyday matter with billions of messages shared daily and needing to be safely transmitted. Even though a lot of solutions have been proposed to carefully exchange data over the World Wide Web, most of them were designed to be used by computers or servers. This usage does not suit all the purposes anymore. More and more constrained devices are used in a lot of different areas (healthcare, embedded systems, "smart home", connected objects...): we entered what some call the *Internet of Things* era (IoT). All of these devices have in common the fact that they cannot afford as much computational power as desktop computers.

"As much computational power as desktop computers" seems blurry, yet we usually define in the same blurry way *lightweight cryptosystems* as primitives trying to provide authenticity, integrity and/or confidentiality at "smaller costs". The constraints and costs have many different natures [BP17]: some solutions offer smaller printed circuit boards, less power and/or RAM consumption, a small number of logical gates, optimizing either hardware or software implementations... This way, lightweight cryptosystems can, for example, secure devices (such as sensor networks or RFID tags) which often run on battery and whose main purpose is not data protection. In a more general manner, lightweight cryptography tries to find the best trade-off between size, speed and security, while taking into account some external constraints and requirements imposed by the future conditions of use.

III International competition and standardization process

In this general context where more and more cryptographic schemes live together, there is an incontestable need to guide their usage and to determine the best of them depending on the needs. Some national or

international organizations or projects (such as the European ECRYPT or the Japanese CRYPTREC) evaluate and recommend the usage of chosen ciphers, while some others (such as the American NIST) aim to standardize them, that is, to create a common documented frame of reference. These standards will then either be followed *de facto* (thanks to their dominant usage) or be legally regulated (we then speak of *de jure* standards). Through these processes, their intention is to identify in the most (yet not always) transparent way algorithms to be used broadly and securely. At the same time, they prevent the usage of hundreds of algorithms (some of which are insecure): a few of them are enough as long as they are being properly studied.

In order to involve academic communities around this same common goal, international competitions are organized by those institutes: expert researchers thus design and analyze algorithms, their guidance being a key element of the selection process and of the future standard's legitimacy. Regarding symmetric lightweight primitives, two recent competitions are of particular interest for us:

- CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness [CAE14]) which had a "lightweight applications (resource constrained environments)" category and ended in 2019; and
- the NIST's lightweight cryptography competition ([NIS17]) which started with its round-1 candidates in April 2019. They selected the candidates for the second round in August 2019 and the finalists were announced in March 2021. According to them, "the final round of the standardization process is expected to last approximately 12 months".

Ascon is a symmetric lightweight cipher which is in the final portfolio from CAESAR and a finalist in the NIST standardization process mentioned. In this now clarified context, it seems natural and interesting to study this primitive and its security. This was the purpose of this internship within Inria's COSMIQ team.

IV Internship's environment

Inria (Institut National de Recherche en Informatique et en Automatique - French Institute for Research in Computer Science and Control) is a French Public Scientific and Technical Research Establishment (EPST) which mainly focuses on computer science and applied mathematics.
The COSMIQ team (one of the 200 project-teams which build Inria's research ecosystem) gathers researchers working mainly on the design and analysis of cryptographic primitives through different angles: symmetric cryptography, code-based cryptography and quantum information theory.

With Anne Canteaut and Léo Perrin who supervise my internship, we focus (as already discussed) on symmetric cryptography.

2 Ascon

As described in the NIST submission [DEMS19], 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". In the context of lightweight cryptography, the authors offer a trade-off between size, speed and security while focusing mainly on the size.

To do so, they based their work on many modes (*i.e* ways using a block cipher to securely process plaintexts and ciphertexts longer than one block), primitives (low-level cryptographic algorithms) and cryptographic solutions already analyzed and/or standardized (such as the Sponge Duplex mode of operation [BDPV12, BDPA11a], or the famous SHA-3 hash function's permutation [BDPA11b]). Their idea is thus to provide a design "with comfortable security margin".

Ascon is a familly of lightweight primitives. We focus here on the Authenticated Encryption with Associated Data modes (AEAD) of Ascon (namely Ascon-128, Ascon-128a and Ascon-80pq) which motivate the attack models we will choose. This kind of primitives aims to provide integrity and confidentiality both with authenticity in an effective integrated manner, instead of using generic methods to combine encryption and message authentication codes (MAC). The former authenticated encryption (AE, [BN00]) model evolves into a new version whose goal is to ensure, on top of that, the authenticity of associated public data (such as public headers) along with the message's authenticity (AEAD, [Rog02]).

Even if they will not be studied here, let us note that two closely related hash functions are also presented in the NIST submission (Ascon-Hash and Ascon-Xof).

As we were mainly interested in the permutation during this internship, I will only briefly describe the way the whole mode works, before diving more deeply in the specifications of the permutation and the S-box.

I Mode of operation

An AEAD mode for Ascon is pretty straight forward. It takes as input a secret key K, a nonce N, associated data A and a plaintext P and outputs a cipher text C for P and an authentication tag (for A, P, and N). We will name the internal state S and its external and internal parts respectively S_r and S_c.

The permutation used in the Sponge construction is p and will be precisely described in Section II.

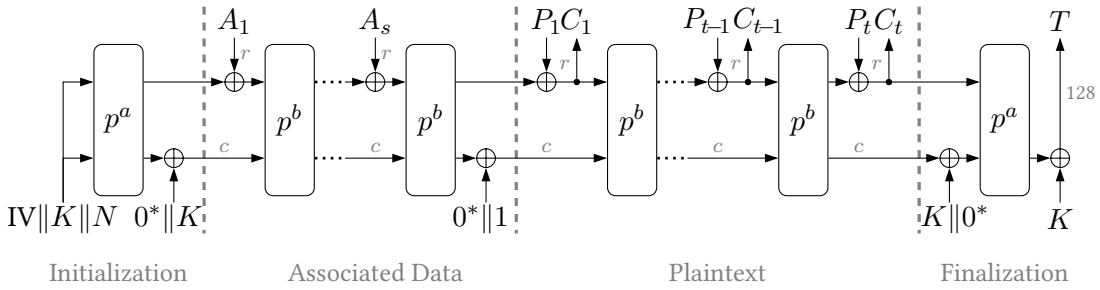


Figure 2.1: Ascon AEAD encryption[1]

Table 2.1 presents the parameters' values for the studied (and recommended) versions of Ascon where:

- a, b are the number of iterations of the permutation,
- k is the key size,
- r is the absorption/squeezing rate, that is, the number of bits either input and processed, or output,
- c is the capacity, that is, the size of the other part of the internal state which is inaccessible from outside in an ideal scenario.

	k	$\|S\|$	r	c	$\|N\|$	$\|T\|$	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

Table 2.1: Parameters' values for recommended versions of Ascon

Ascon-80pq works exactly as Ascon-128 but with a longer key (for post-quantum security, in order to resist an exhaustive key search with Grover algorithm). Ascon-128a also works exactly in the same way except it has a larger rate (and thus a smaller capacity) which forces to iterate a bit more (8 intermediate p iterations instead of 6 because the smaller capacity may imply a lower security).

As shown in Figure 2.1, Ascon's AEAD mode of operation follows four steps which are detailled just below.

I.1 Notation

- $\oplus$ will indicate a bitwise addition of two binary strings (XOR).
- $||$ will be used to indicate the concatenation of two bitstrings.
- $\leftarrow$ means "assign the value".
- 0^* means "enough 0 bits to fill up the state".
- $\lfloor x \rfloor_i, \lceil x \rceil^j$ indicate the respective truncation of the first i and last j bits of a bitstring (most and least significant i and j bits respectively).

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

I.2 Ascon's AEAD encryption workflow

1. Initialization:
   $S \leftarrow IV||K||N$
   $S \leftarrow p^a(S)$
   $S \leftarrow S \oplus (0^*||K)$
   Here, IV is an initial vector which only depends on the version used ($IV \leftarrow k||r||a||b||0^{160-k}$). The final XOR is used as a domain separator and to avoid some known attacks.

2. Associated data absorption:
   $S \leftarrow S \oplus (A_i||0^*)$
   $S \leftarrow p^b(S)$
   This step is repeated for every (padded) associated data blocks and is followed by a similar domain separation after the last block: $S \leftarrow S \oplus (0^*||1)$

3. Plain message absorption and encryption:
   $S \leftarrow S \oplus (P_i||0^*)$
   $C_i \leftarrow \lfloor S \rfloor_r$ ($\lfloor S \rfloor_{|P| \mod r}$ instead for the last block so that $|P| = |C|$)
   $S \leftarrow p^b(S)$ (does not occur for the last block)
   This step is repeated for every plain text block (with minor changes for the last one) and is also followed by a domain separation step: $S \leftarrow S \oplus (0^r||K||0^{320-r-k})$.

4. Finalization and tag creation:
   $T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$

In order to decrypt, the workflow is almost the same: after the two first identical steps each cipher blocks are input one after the other, the corresponding plain text is then output, and the cipher text is used to obtain the appropriate state to carry on.

Remarks I.1.

1. $|A|$ and $|P|$ can be arbitrary long but a padding must be applied in order to obtain messages of length divisible by r. It is a simple injective pad: always add 1 at the end of the message and then add the minimum number of 0 to fill the last block.

2. The nonce can be a simple counter, the only constraint is that the sender must ensure that it is used only once per key.

3. The XORs used as domain separators are there in order to protect against some key-recovery attacks and tag forgeries in case of an intermediate-state recovery (using manipulation with related keys/nonces/plaintexts). The two separators which use K are shifted one from the other to avoid XOR cancellation if only a single block is encrypted.

4. the IV constant plays a crucial role in Ascon's security. Its reasons to be may seem unclear to the reader for the moment, but I will detail them later.

II Ascon's permutation

We will now describe the permutation p which is the core element designed by the authors and which will be of major interest in the rest of this document. It is built on a Substitution Permutation Network (SPN) logic: p is the composition of a constant adding function, a non-linear substitution layer and a linear diffusion transformation: $p = p_L \circ p_S \circ p_C$, thus following Shannon's confusion-diffusion paradigm [Sha49].

In order to understand better the way p works, we use the following decomposition of $S = X_0||X_1||X_2||X_3||X_4$ into five 64-bit words (X_0 is the most significant, X_4 the least one).

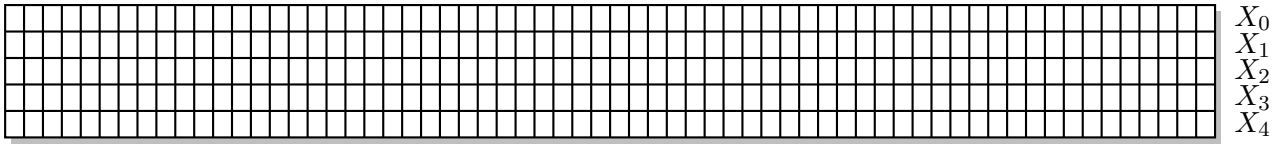


Figure 2.2: Ascon's state S

- The constant adding step consists of XORing an 8-bit constant to the word X_2. The constant only depends on the index of the iteration that takes place. There are different pools of constants depending on how many iterations we are computing (p^a or p^b), and they are easily computable: when $a = 12$, the most significant half of the constant is decremented from 15 to 4 throughout the rounds while the least significant one is incremented from 0 to 11.

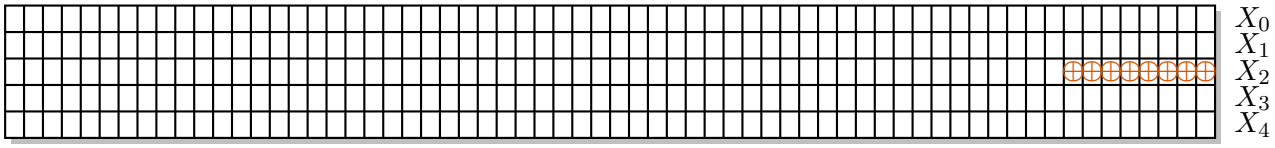


Figure 2.3: The round constant adding function p_C

- The substitution layer is made of 64 parallel uses of a single S-box on each column of the state. Those operations can be done at the same time considering the fact that the S-box has a bit-slicing design using only bitwise XOR, AND or NOT. This also implies what the authors call a "natural side-channel protection" as no look-up table is necessary. We will discuss the S-box in details in Section III.

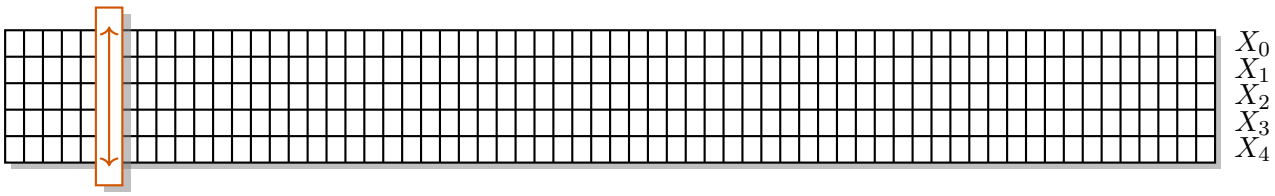
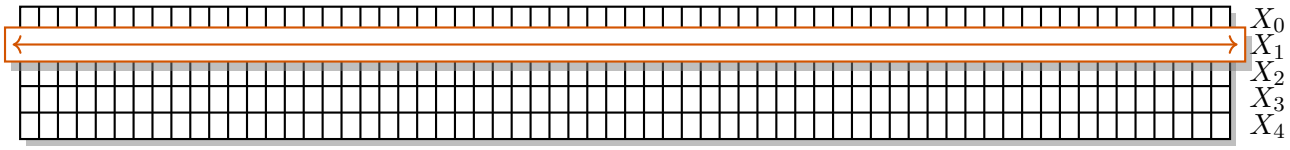


Figure 2.4: The S-box layer p_S

- The linear diffusion layer is made of five calls to five different linear functions Σ_i on each row of the state. For each $i \in \{0, 1, 2, 3, 4\}$, $\Sigma_i(X_i)$ only consists on the XOR of the i^{th} row X_i with two rotated versions of itself. The indexes 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 2.5: The linear layer p_L

This choice was inspired by SHA-2's Σ [NIS15] which was replaced by five distinct versions instead of one. This choice was made in order to provide a better diffusion (more and more differential and linear active S-boxes at the end of each round), while keeping a relatively minimal cost (rotations are "almost free in hardware and relatively cheap in software").

About the choice of the rotation constants, there are no other motivation given than that it achieves "a good diffusion after 3 rounds of Ascon". Nevertheless, it is interesting to note that other designs were inspired by Ascon and that the authors chose to change those constants: two of them in the case of DryGascon [Rio19] and all of them in the case of Sycon v1.0. [SMS19][2]. However, Sycon v1.0. was not selected for the second round of the NIST competition (a new version called Sycon [MSST21] is supposed to compensate the "lack in good diffusion property" with a new linear layer of the form $X = (X \oplus (X \lll i) \oplus (X_4 \lll 2i)) \lll j)$ whereas DryGascon went to round 2 without being selected as a finalist.

III Ascon's S-box

As providing non-linearity is often costly, a lot of designs use small non-linear functions which are applied locally on the full state to achieve this purpose. It enables at the same time an easier study of the substitution layer.

Ascon follows the same principle. Its S-box is a 5-bit permutation, in other words $S\colon \mathbb{F}_2^5 \to \mathbb{F}_2^5$.
It is well-known (see Theorem I.4) that any S-box $S\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ can be uniquely represented as a collection of multivariate polynomials of $\mathbb{F}_2[x_1, \cdots, x_n]/(x_1^2 + x_1, \cdots, x_n^2 + x_n)$ known as its *Algebraic Normal Form* (ANF). Figure 2.6 presents the algebraic normal form of Ascon's S-box and we can observe that it is indeed the only source of non-linearity as its algebraic degree is equal to 2.

[2]In the case of DryGascon, the S-box used is the same as Ascon's. With Sycon v1.0. and Sycon, the S-box was changed; it could explain the search for a compatible linear layer.

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

Figure 2.6: Algebraic normal form (ANF) of Ascon's S-box

The choice of an S-box is of primary matter in a design and it was an important part of the process according to the authors. The design of Ascon's S-box was inspired by the already well-studied Keccak's χ ([DR02]). Actually, this S-box is *affine equivalent* to χ and the authors thus managed to correct some unwanted properties (higher linear and differential branch numbers, no more fix-point, higher dependency in the input bits). It was built all at once with the linear layer to provide good enough diffusion. They decided to go with an "alright" S-box in terms of linear/differential properties instead of a "perfect" one to minimize the cost (a performance/security trade-off). But part of the defects are compensated by the number of iterations. It is invertible and has no fix point. It has a low algebraic degree (2) which was chosen to facilitate masking and threshold implementation, which are side-channel countermeasures. This low algebraic degree might lead to integral attacks (presented in the following chapter). However, the column-wise way of applying the S-boxes, together with the row-wise injection of data was thought in order to avoid some of those attacks (preventing the control of all entries of a single S-box by an adversary).

IV Advantages and security claims

The claimed level of security by the designers of Ascon's AEAD mode is 128 bits for each of the versions 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.

A lot of choices were made in order to protect against side-channel attacks (domain separation against implementation attacks which could lead to recover intermediate states, low algebraic degree to facilitate masking, no use of look-up tables thanks to bitslicing...) which is of primary matter for the targeted usage.

Dobraunig *et al.* claim that Ascon achieves "high security and robustness in practice with a very low area footprint in both software and hardware implementations, particularly for short messages". The software implementation is straight-forward, light and quick while optimizations and trade-offs are possible on the hardware implementation. After five years of competition and analysis, Ascon still seems to have a reasonable security margin and is a viable option to be used in the context of lightweight symmetric cryptography.

A lot of third-party studies were made on Ascon to support this claim. Some of them are now presented in the following chapter. It is not an exhaustive description of the existing attacks on Ascon, but rather a selection of the ones which are, directly or indirectly, related to the study we made.

3 Previous works and related topics

In Chapter 2, we quickly mentioned a fundamental tool which will be of major interest in the following topics: the *Algebraic Normal Form*. In this section, I will start by defining properly what is the ANF of a Boolean function, as well as the other mathematical tools needed to fully understand the rest of this document. Next, I will present integral and cube-like attacks which are closely related to the study we made (and which will be precisely presented in the following chapter) but also some variants already used to put to the test Ascon's reliability.

I Mathematical background

Notation I.1. In the following sections:

- n will be a non-zero integer ($n \in \mathbb{N} \setminus \{0\}$).
- We will denote $\mathbb{F}_2$ the finite field of size two ($\mathbb{F}_2 = \{0, 1\}$).
- We will denote $[\![1, n]\!]$ the set $\{1, \cdots, n\}$.
- We will also use an exponential notation when looking at the AND of multiple variables: $x^u := \prod_{i=0}^{n-1} x_i^{u_i} \in \mathbb{F}_2[x_0, \cdots, x_{n-1}]/(x_0^2 + x_0, \cdots, x_{n-1}^2 + x_{n-1})$, where $u = (u_0, \cdots, u_{n-1}) \in \mathbb{F}_2^n$.
- If x and y are two Boolean vectors of size n, $x \preccurlyeq y$ will mean: $x_i \leq y_i$ for all $i \in [\![1, n]\!]$.
- $f \equiv 0$ will mean that f is the null function.

The first objects we are interested in are of course S-boxes, which are a generalization of Boolean functions.

Definition I.2 (Boolean function)**.** A Boolean function of n variables is a function of the form: $f : \mathbb{F}_2^n \to \mathbb{F}_2$.

Definition I.3 (S-box)**.** An S-box is a collection of a finite number of Boolean functions of n variables. If the collection is of size $m \in \mathbb{N} \setminus \{0\}$, an S-box can be viewed as a vectorial Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ where each individual Boolean function is called a *coordinate function*.

As the domain and codomain are finite ($\mathbb{F}_2^n$, $\mathbb{F}_2^m$ respectively), it is natural to describe our functions through their truth tables (a table in which we associate each antecedent to its image). However, there exist other representations of Boolean functions (each of them having some advantages and drawbacks). Thereafter, we will only use the algebraic normal form of Boolean functions.

Theorem I.4 (Algebraic Normal Form [Can16])**.** *Let f be a Boolean function of n variables. Then, there exists a unique multivariate polynomial in $\mathbb{F}_2[x_0, \cdots, x_{n-1}]/(x_0^2 + x_0, \cdots, x_{n-1}^2 + x_{n-1})$ such that:*

$$f(x_0, \cdots, x_{n-1}) = \sum_{u \in \mathbb{F}_2^n} a_u x^u, \quad a_u \in \mathbb{F}_2, \ \forall\, u \in \mathbb{F}_2^n.$$

This multivariate polynomial is called the *algebraic normal form* (ANF) of f.

We can of course consider the ANF of an S-box by collecting all the ANF of its coordinates.

If we know the truth table of an S-box, we can determine its ANF thanks to the *binary Möbius transform* (see [Can16]). However in practice, this transformation is often costly (about $n2^{n-1}$ operations and $n = 320$ in the case of Ascon's permutation), and very hard to carry out: we need to already know the entire truth table, which is very restrictive. The following proposition gives a way to gain partial information about the ANF of a function, only by having access to some of its values.

Proposition I.5 (ANF's coefficient computation [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: $a_u = \sum_{x\preccurlyeq u} f(x)$.*

We are also able to define the *degree* of a Boolean function as the degree of the largest monomial appearing in its ANF:

$$\deg(f) := \max_{u\in\mathbb{F}_2^n,\ a_u\neq 0} wt(u).$$

and the degree of an S-box will naturally be the largest degree of its coordinates.

However, in our case, we also are interested in some *derived functions* ([HSWW20]) from f which are obtained by fixing some variables to known constants. The number of variables thus goes down and the remaining ones, depending on their public or private access rights, cannot be considered the same way. That is why we will look at our polynomials in rings like $A[v_0, \cdots, v_{63}]/(v_0^2 + v_0, \cdots, v_{63}^2 + v_{63})$ where $A = \mathbb{F}_2[k_0, \cdots, k_{128}]/(k_0^2 + k_0, \cdots, k_{127}^2 + k_{127})$, thus dissociating public variables (v_i in this example) from secret ones (k_i here). The number of variables and their names will be clarified depending on the context. The *derived degree* will be the one in which we will be interested, that is, the degree with respect to the public variables.

It is known since Shannon ([Sha49], page 711) that, by building a sufficiently large system of equations (in a known-plaintext kind of attack) in which the unknowns are the secret variables (namely, the key bits), it is possible, in theory, to break a cryptosystem algebraically by solving the aforementioned system. This is the main reason why we need non-linear parts inside a primitive and why it is important to study the role of those components in the growth of the degree, round after round. The described attack would imply the knowledge of the full ANF of the permutation and the solving of a system of very high degree which is, in practice, infeasible. However, there exist some attacks which exploit the fact that the functions used inside primitives are not randomly chosen, but instead, built round after round by composing low-degree layers. I present in the next sections some of these methods which influenced our work and the way we looked at Ascon.

II Higher-order differentials and integral attacks

We now focus on a *chosen-plaintext attack* (CPA) in which an adversary will gain information, not from single messages or pairs independently (like with differential or linear attacks) but rather from the whole pool of chosen messages at once. *Integral attacks* (as they are now commonly called) are usually based on two kinds of knowledge:

- the way the growth of the degree evolves throughout the rounds; and/or

- the internal structure and the precise understanding of each sub-function composed or concatenated to build the full permutation.

Different names have been given to this kind of attacks: SQUARE attack because of the first cipher attacked by Daemen *et al.* [DKR97], *saturation attack* as we will "saturate" a word in input (meaning that we will make it take all possible values while keeping the other words fixed, name due to Lucks [Luc02]), *multiset attack* as we will consider the set of chosen messages not as a set of finite word sequences but instead as the finite sequence of each word's multiset (this is introduced by Biryukov & Shamir [BS01]), and finally integral attack in reference to the analytical tool (we here sum over a finite integral domain, name due to Knudsen and Wagner [KW02] in an attempt to formalize in a unified way all those attacks).

The following results directly lead to some higher-order differential[1] distinguishers.

Definition II.1 (Derivative along a vector)**.** Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let $a \in \mathbb{F}_2^n$. We define the derivative of F along a as $D_aF\colon x \mapsto F(x) \oplus F(x \oplus a)$.

Proposition II.2. *[Lai94] Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let V be a vector subspace of $\mathbb{F}_2^n$ and $(a_1, \cdots, a_d)$ a basis of V. Then $(D_{a_1} \circ \cdots \circ D_{a_d})F(x) = \sum_{v \in V} F(x \oplus v)$ for any $x \in \mathbb{F}_2^n$.*

We can therefore define the derivative of F along V (as the previous value does not depend on the choice of the basis).

Definition II.3 (Derivative along a vector subspace)**.** Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let $V \subset \mathbb{F}_2^n$ be a vector subspace. Let $(a_1, \cdots, a_d)$ be a basis of V. We define the derivative of F along V as $D_VF = (D_{a_1} \circ \cdots \circ D_{a_d})F$.

Proposition II.4. *[Lai94] Let $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let V be a vector subspace of $\mathbb{F}_2^n$.*
Then, $\deg(D_VF) \leq \deg(F) - \dim(V)$. In particular, if the dimension of V is strictly higher than $\deg(F)$, then $D_VF \equiv 0$.

Thus, in order to distinguish the studied function from randomly chosen ones with similar properties, one only needs a bound on the function's degree. It will then cost $2^{\deg(f)+1}$ chosen messages.

This property can also be used (once with f^a and once with f^{-b} for well-chosen values of a and b) to build *zero-sum distinguishers*. This method enables to distinguish the permutation from a random function, because as stated in [BC11], "it is expected that a randomly chosen function does not have many zero-sum". In order to lower the cost (or to enable attacks on more rounds), it is sufficient to find tighter bounds on the degree of a composition of functions [BC11, BC13]. It is done in [BC11] using Walsh spectrum analysis. By using an improvement to add "a free round in the middle", one can finally mount a distinguishing attack on 11 rounds of ASCON's permutation (which is affine-equivalent to Keccak's one) for 2^{85} and on the 12-round version for 2^{130}. But as stated by Dobraunig *et al.* [DEMS15], this non-random behavior does not seem to lower the cipher's security as no attack exploits this property yet.

In a more general manner, with integral attacks, we are interested in the evolution of some multiset properties through the different layers of the cipher. Here, we only assume that all S-boxes are bijective.

[1] In $\mathbb{F}_2$, as addition and substraction are the same operation, the terminology oscillates between integral and differential, depending on the way we look at it.

Definition II.5 (Multiset properties)**.** Let us consider messages of length $n = k \times m$ bits where k is the number of S-boxes of a substitution layer. Let W be a multiset composed of 2^m m-bit values. We say that W has the multiset property:

- $\mathcal{C}$ if W contains a single value (constant) occuring 2^m times.
- $\mathcal{P}$ (for permutation, sometimes called $\mathcal{A}$ for all) if W contains each 2^m possible value, once.
- $\mathcal{B}$ if the multiset is balanced, meaning that XORing over W gives 0^m: $\bigoplus_{w \in W} w = 0$.
- $\mathcal{E}$ if all the values occur an even number of times and $\mathcal{D}$ (dual) if W has property $\mathcal{E}$ or $\mathcal{P}$.

With these definitions we can look at the previous distinguisher as the transformation of a subspace V which has property $\mathcal{P}$ to a set $f(V)$ which has property $\mathcal{B}$. Zero-sums can be seen as paths from $\mathcal{B}$ to $\mathcal{B}$. These distinguishers (in their initial versions) do not use any internal structure to study this evolution, only the knowledge of a bound on the degree. It can however be interesting to use the internal structure, as shown by the following proposition.

Proposition II.6 (Evolution of multiset properties [BS01])**.** *Let S (state) be a sequence of k multisets composed of 2^m m-bit values.*

1. *If S has property $\mathcal{C}^{i-1}\mathcal{P}\mathcal{C}^{k-i}$ then it is preserved by a layer of S-boxes.*
2. *If S has property $\mathcal{D}^k$ then it is preserved by a layer of S-boxes.*
3. *If S has property $\mathcal{D}^k$ then it is transformed into property $\mathcal{B}^k$ by a linear or affine layer on $n = m \times k$ bits.*
4. *If S has property $\mathcal{C}^{i-1}\mathcal{P}\mathcal{C}^{k-i}$ then it is transformed into property $\mathcal{D}^k$ by an affine layer.*

Thanks to these results, one can mount different kinds of attacks:

- One can use internal properties in order to follow more accurately the evolution of multiset properties and thus build either distinguishers, or attacks on the last round for round-reduced versions. Well-known examples are the 3-round distinguisher of Square [DKR97], AES [DR02], Crypton [DBRP99] and their corresponding 4-round attacks. They use an input sequence verifying $\mathcal{C}^{i-1}\mathcal{P}\mathcal{C}^{k-i}$. After three rounds, the intermediate state has the distinguishing property $\mathcal{B}^k$. Regarding the attack on the last subkey, the adversary will find the subkey, byte by byte, by trying for each one of them all the 2^8 possibilities. Once a "bet" on a key byte is made, one can inverse the last-round, and find, from a ciphertext byte, the value of the corresponding 3-round intermediate byte. It is then possible to compute the integral (XOR) of the intermediate bytes corresponding to all the possible values of the chosen ciphertext's byte. This integral will always be null for the correct guess of the key and has an about $\frac{1}{256}$ probability of being null for an incorrect key (assuming a random behavior in this case). Therefore, the expected number of probable byte is $1 + \frac{255}{256} \approx 2$. This reduces the key space to about 2^{16} after carrying out this test for each of the 16 bytes. This attack can be adapted to 5 or 6 rounds of the aforementioned ciphers.
- On the other hand, even without having any information on the cipher except its general structure, one can still discover most of the data needed to encrypt or decrypt messages without finding the key or the exact internal structure: Biryukov & Shamir present in their article [BS01] a way to find "an equivalent representation of all the elements in the scheme [which] may be different from the original definitions of these elements".

Finally, we can also look at integral attacks from another angle, in the spirit of Proposition I.5:

Proposition II.7. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Let $V = \langle e_{i_1}, \cdots, e_{i_k} \rangle$ where $(e_0, \cdots, e_{n-1})$ is the canonical basis of $\mathbb{F}_2^n$ and $i_j \in [\![0, n-1]\!]$ for all $j \in [\![1, k]\!]$. Let I be the set $\{i_1, \cdots, i_k\}$. Then, looking at the ANF, we get:*

$$f(x_0, \cdots, x_{n-1}) = D_V f(x_0, \cdots, x_{n-1}) x^{v_I} + g(x_1, \cdots, x_{n-1}), \quad \text{with } v_I = \bigoplus_{j=0}^{k} e_{i_j}$$

and where no monomial present in g is divisible by x^{v_I}.

We thus clearly observe that higher-order differential distinguishers are based on the absence of some monomials in the ANF (all monomials divisible by x^{v_I} when we choose a specific I for which $D_V f \equiv 0$, and more generally all monomials of too high degree when a bound is known according to Proposition II.4). The search for specific monomials whose coefficients are constant[2], that is, the search for monomials whose presence is independent from the key used, is the main idea behind integral distinguishers. Cube attacks (which are presented in Section III) are a way to derive integral distinguishers into key-recovery attacks. The *division property* was introduced in order to study more precisely the ways integrals propagate. It will be presented in Section IV).

III Cube attacks

III.1 Classical cube attacks

Cube attacks were introduced by Dinur and Shamir [DS09] in order to analyze cryptosystems viewed as polynomial blackboxes. They designed a CPA method which enables the cryptanalyst to attack Boolean polynomials (for example, the algebraic normal form of an encryption function) by using fewer calls to the blackbox than an interpolation method. Dinur *et al.* [DMP+15] then applied it to attack some round-reduced versions of Keccak.

The goal of the attack is to obtain, by the use of chosen cubes (*i.e* sets of public variables), multiple linear combinations of secret variables, and then perform a classical linear system solving. The name "cube" comes from the $d-1$-dimensional Boolean cube of possible values taken by the chosen variables. The main element to observe is that the derivative of a polynomial of degree d with respect to a $d-1$-dimensional vectorial subspace is either linear or constant (according to Proposition II.4).

This attack is composed of two stages. During the first stage (the offline one) the attacker is allowed to query the blackbox choosing both the public variables (*i.e* the plain-text, the nonce, the IV...) and the secret ones (the secret key). This process is independent of the actual choice of the key and thus only needs to be done once per cryptosystem. The second phase is the actual attack (done online) during which the attacker can only query the system by choosing public variables.

A multivariate Boolean polynomial p can be uniquely written as $p = x^{v_I} \times D_{V_I} p + q$ where I is a subset of $[\![0, n-1]\!]$, x^{v_I} the corresponding monomial, V_I the corresponding vector subspace and q a polynomial which is not divisible by x^{v_I} (see Proposition II.7). This higher-order differential is often called *superpoly* in the context of cube attacks. The value of the evaluated higher-order differential (which is in practice a

[2]We focused on null higher-order differentials/coefficients but constant 1 also gives rise to a distinguisher.

polynomial in unknown variables) can thus be obtain by summing over V_I (see Proposition I.5). Thus, if we are able to find proper sets I such that $D_{V_I}p$ are affine combinations of secret values, then we can hope for an effective key-recovery attack. This is the goal of the offline phase.

When a tight bound on the degree is known, we can discover linear combinations more easily (Proposition II.4 states that the higher-order differential will have a degree equal to 0 or 1). But it may also exist a lot of linear or constant superpolys related to terms of degree less than $d-1$. They are in general harder to find but, according to Proposition I.5, computing their actual value requires less data. This could lead, if they are found, to stronger attacks. In order to search for some of them, Dinur and Shamir use a method by testing the linearity of a derivative with respect to a random subspace and adjusting the subspace's dimension as long as the derivative is not linear.

Once that one is (almost) sure that the derivative is linear (which can be tested through a BLR linearity test), this linear combination is recovered by interpolation (let us keep in mind that during the offline phase the secret variables can be chosen by the attacker, so, as $\sum_{v\in V_I} p(v,x) = \sum_{i=1}^{r} a_i x_i + c$ (v_i are public variables and x_i unknown ones), we get $c = \sum_{v\in V_I} p(v,0)$ and then $a_i = \sum_{v\in V_I} p(v,e_i) + c$ where e_i is the vector which has a single coordinate equal to one at index i).

Finally, during the online phase, the secret variables are already set ($\sim$ is here to emphasize it) and unknown to the attacker but by querying the blackbox for $p(v,\tilde{x})$ and summing over the cube V_I, one is now able to obtain a bit b which verifies $\sum_{i=1}^{r} a_i\tilde{x}_i + c = b$. If one has enough independent combinations, one will then recover the values of the secret variables (or at least reduce the key space).

III.2 Practical attacks on Keccak and Ascon

This enables a key-recovery attack on five rounds of Keccak. In order to find linear higher-order differentials, we focus on $d-1$-dimensional cubes (where d is the expected degree of p, here $d=32$) as the corresponding derivative can only be linear or constant. So, by picking 31 out of the 128 public variables (nonce variables in the case of Ascon's initialization) randomly, the other ones being set to 0, and testing each coordinate's derivative, one can expect to find enough linearly independent combinations. For example in the case of Keccak, Dinur *et al.* ([DMP+15]) were able to find 117 relations thanks to 19 cubes, so the cost of their attack is about $19\times 2^{31} \approx 2^{35}$ (the exhaustive search for the last 11 variables is negligible compared to the 2^{35} queries).

For six rounds, we would need to consider cubes of dimension 63 which seems to make the offline phase more difficult if we want it to be done in reasonable time. In order to go one round further, Dinur *et al.* managed to invert the last round in different scenarios.

Those techniques can be adapted to attack a reduced-version of the initialization phase of Ascon: as stated by Dobraunig *et al.* [DEMS15], the initialization is the only part where a nonce-respecting attacker "can keep some inputs of the permutation constant and deterministically influence others" (the nonces being chosen). Using input bits from associated data or plain-texts is possible but with a nonce-respecting condition this would also mean changing the nonce at each query and thus changing each time the state after initialization, which is problematic as it cannot be fully recovered in a sponge mode. The 5-round attack on the initialization

can be implemented as explained above.

A *cube tester* [ADMS09] (another name for what we previously called an integral distinguisher) can also be created for a 6-round initialization by using a structural property of Ascon : as the S-box is the only non-linear transformation during one round and as it is applied column by column on the state, by choosing the cube variables on the same row, we are assured that, after round 1, no bit of the state contains any degree-2 monomial in the cube variables. With such a choice and after 6 rounds, the state bits cannot exceed a degree of 32 in the cube variables. By choosing a degree-33 cube on the same row, one can build a cube-tester: summing over the cube will always give a null output sum.

III.3 Bordeline cubes

Another cube-like attack on a 5/6-round initialization is described by Dobraunig *et al.* [DEMS15]. It is based on *borderline cubes* and on previous studies [DMP$^+$15, FKM08]. The logic of the attack is to first select small cubes whose superpolys (which are not necessarily linear anymore) only depend on a small amount of key bits. If the cubes are well-chosen, the subkeys can be discovered independently from the rest of the key. Attacking in a divide-and-conquer manner is thus possible. To do so, first compute the cube sum for each possible choice of subkey (offline phase) and then, compare them to the online sum. In the case of Ascon's 5-round initialization, it is expected to find on average a single possibility for each independent subkey [DEMS15]. This thus leads to a key-recovery attack with a time complexity of $2 \times 4 \times 2^{16} \times 2^{16} = 2^{35}$ (two key-words made of four 16-bit subkeys each, 2^{16} possibilities for each subkey, and a cube-sum cost of 2^{16}) for a 5-round initialization and about 4×2^{64} for the 6-round version (if there is actually, on average, a single possibility for each 32-bit subkeys).

III.4 Generalized conditional cube attacks

Li *et al.* [LDW17] continued testing the resistance of Ascon against cube-like attacks in two ways: first by adapting and generalizing conditional cube attacks against Keccak used by Huang *et al.* [HWX$^+$17] and then by developing a new technique called *cube-like key-subset*. The idea behind those attacks is, as for bordeline cubes, to enable the use of non-linear higher-order differentials. To do so, linear common divisors shared by all the output bits' differentials (with respect to a fixed cube) are searched. If such a divisor exists, we are able to determine its value from the cube sums (*cf.* Property 4 of [LDW17]) as it will influence the value of each output cube-sum. By carefully choosing 256 16-dimensional cubes, the authors are able to recover the secret key from a 5-round initialization in about $256 \times 2^{16} = 2^{24}$ queries (against 2^{35} of the original cube attacks, time and data share here the same complexity). Their selected cubes all follow the same logic: under a simple linear key bit condition (such as $k_i = 0$ or $k_i + k_j = 0$, which is in fact a common divisor of the current superpolys), it is guaranteed that the monomial does not appear in any output coordinate. If the monomial does appear, one is guaranteed that the condition is not respected. Using another cube for the complementary condition, the authors actually manage to recover the full key, bit by bit. The same goes for the 6-round version with a complexity of $256 \times 2^{32} = 2^{40}$ (against a previous theoretic complexity of 2^{66}).

III.5 Cube-like key-subset technique

The same logic is used for the attack against a 7-round initialization: the idea is to determine conditions on the key bits which influence the disappearance of a term of order 65 in the output bits of the seventh round, forcing the cube sum to be equal to 0. To do so, the authors worked on the ANF of two rounds of Ascon

and managed to find conditions on the secret key which "delete" the term[3] of degree more than 2 after two rounds. These conditions now constitute a system of linear equations (called *partial divisors*) and replace the unique common divisor of the previous method. If the conditions are satisfied by the key, the term of degree 65 will not appear in the output and the cube sums will then be equal to 0. On the other hand, the probability of finding 64 null cube sums with a random key is, according to Assumption 2 [LDW17], small (2^{-64}, as we recover 64 cube sums at once with the output of the first row). This can thus be used to detect whether the conditions are satisfied or not (or equivalently, to detect whether the key belongs to a certain subset of the key space or not).

In fact, those subsets (which each determines a particular choice of a cube) are regrouped, thanks to the use of *control cube variables*, into bigger subsets of size 2^{127} or 2^{126}. Thus, each time a key belongs to one of these bigger subsets, 8 bits of data are discovered (the linear systems are composed of 8 equations). On the other hand, if after trying all subsets regrouped into a bigger one, one did not find any match (no computation leads into finding 64 null cube sums), one can assume that the key does not belong to this big subset and this way finds, more or less, 1 or 2 information bits of the key.

Finally, an adversary accumulates some one-bit conditions which enable to lower the cost of the exhaustive search. In the best-case scenario (which appends for 1 key out of 2048), one obtains 52 bits of information through a process costing about 2^{76} and finally adds an exhaustive search on the 76 missing bits for another 2^{76}; the complete cost is thus about 2^{77}. In the worst case, the complexity climbs up to about $2^{103.92}$.

III.6 First misuse-free key-recovery attack on 7-round Ascon

The problem with the previous attack described is that it uses cubes of dimension 65, meaning that it requires 2^{65} chosen data (here nonces) to be used with the same key. However, Dobraunig *et al.* claimed in their submission that "the number of processed plaintext and associated data blocks protected by the encryption algorithm is limited to a total of 2^{64} blocks per key". This problem was first pointed out by Rohit *et al.* [RHSS21] who present the first misuse-free key-recovery attack on 7-round Ascon using cube techniques.

To do so, they use initial configurations in which one nonce row is set to 0 while the other one will contain 64 cube variables. This way, the maximal degree in the cube variables is still 1 after the first round and the coefficients of the linear terms only depend on one key bit (either k_i or $k_i \oplus k_{i+64}$ depending on the configuration). These configurations ensure that the coefficients of *the* monomial of degree 64 after the seventh round will necessarily be a sum of products of coefficients in front of monomials of degree 32 after the sixth round. Thus, by using their so-called *partial polynomial multiplication* technique, they manage to recover the ANF of the superpolys of $x^{v[\![0,63]\!]}$ for each output coordinate by first computing the truth tables and the ANF of the coefficients of all the degree-32 terms after round 6. They are now able to evaluate these superpolys for each possible 64-bit subkey (let us keep in mind that these superpolys only depend on 64 keybits), store those values in a hash table and compare the online cube sum in order to derive suggested subkeys. According to them, there is, on average, only a single subkey which will be suggested. The remaining unknown 64-bit subkey can finally be obtained by an exhaustive search.

[3]There is only a single term of degree 2 after round 1. It is due to the choice of the 65 cube variables: 64 of them are on the same row, leaving only two variables in the same column. This leads to a single term of degree 2.

The total costs of the refined version of this algorithm are dominated, for time complexity, by $2^{123.28}$ 7-round Ascon evaluations, for memory complexity, by 2^{101} bits and for data complexity, 2^{64} nonces used with a single key during the online phase.

IV Generalized integral attacks using the division property

Integral and cube attacks have in common the way of obtaining meaningful information on a cipher's permutation (by saturating part of the input bits), especially by distinguishing a non-random behavior linked to constant higher-order differentials. A very powerful cryptanalytic technique introduced by Todo [Tod15] enabled to study more precisely these specific behaviors. By looking, on the one hand, at the evolution of some general properties of input and output multisets, and, on the other hand, by focusing on the evolution of the degree round after round, the newly introduced division property enabled to build new and more powerful distinguishers. This tool and its variants (for example the three-subset bit-based division property [HLM$^+$20], 3SBDP) are now more and more understood and studied from various angles (through *parity sets* [BC16], *division trails* [XZBL16], *monomial trails* [HSWW20], etc.) and with the help of MILP (Mixed Integer Linear Programming) solvers, it finally became kind of a standard cryptanalytic approach.

Regarding Ascon, the division property was used by Rohit *et al.* in their already cited paper [RHSS21] for two reasons:

- First, to obtain a more precise study of the complexity of their cube attack: by bounding more tightly the maximum degree of the coefficients of degree-32 terms after the sixth round, they manage to obtain a better approximation of the actual complexity.
- Secondly, they build new integral distinguishers with the help of the three-subset bit-based division property (3SBDP), which improve the existing ones in the case of 4-, 5- and 6-round Ascon . They also found, with this same technique, the first 7-round Ascon distinguishers for a cost of 2^{60} in data and time complexity.

Both of the parts depicted above were obtained through the already mentioned division property/MILP models association.

Those works, especially Rohit *et al.*'s article, were the starting point of our work on Ascon , which is presented in the following chapter.

4 Our work

I Motivations and choices made at the beginning of the study

Let us now focus on the studies we made over the months and which actually kept us busy most of the time (except the first month which was mostly used for the literature study). As already mentioned, the starting point was mainly Rohit *et al.*'s article [RHSS21] and we thus used and analyzed Ascon in an AEAD context. In this paper, they present a cube attack on Ascon's initialization which is the only part where a nonce-respecting attacker "can keep some inputs of the permutation constant and deterministically influence others". In this configuration, the nonces (which are not given in the specifications and which can be set freely[1] by an adversary) are chosen, no associated data are used and a single plaintext is involved. As we can see on Figure 4.1, as C is output and P is known (in a chosen-plaintext situation), one can recover the first row output by p^a by computing $P \oplus C$.

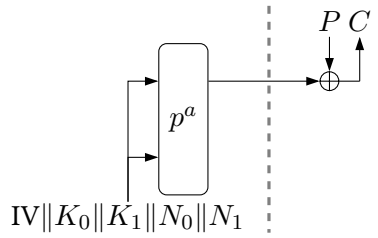


Figure 4.1: Ascon attack model inspired from [RHSS21]

A key-recovery attack on a 7-round version can thus be mounted: by choosing the cube formed by all variables corresponding to N_1, the associated superpoly only depends on the keybits of K_0. Therefore **assuming that all the output coordinates are indeed of maximum degree in the cube variables** (*i.e* of degree 64 after seven rounds), one can expect that the value vector of the superpolys computed for some keybits $k_0, \cdots, k_{64}$ will match the actual value vector only for the correct choice of key bits.

This attack's description leads to the following remark which first guided our work:

Remark I.1. This attack only works **if the cube monomial actually appears** after seven rounds in all of the 64 output coordinates. Otherwise, some or all of the superpolys are null, which means that we will not be able to recover the 64-bit subkey (in fact we can expect to find on average $2^{64-|\{\text{null superpolys}\}|}$ subkeys which will match the actual value vector). In the extreme case where all the superpolies are null, the key-recovery attack degenerates to a distinguisher.

This remark motivated the search for the actual degree of the output coordinates in the cube variables or more information on this degree, its bounds, its evolution throughout the rounds...

[1] the only constraint being to not use the same nonce twice

In order to simplify this study, we used multiple versions of Ascon, modifying either the IV, the addition of constants, the S-box, etc. It enabled us to get a better understanding of the way the actual Ascon works. Those modifications, their purposes, and the insights obtained thanks to them are described in Section II, III and IV. In Section V, I describe some tracks we followed on Ascon's other possible representations. Finally, Section VI gives a summary of studies on different initialization scenarios while Section VIII describes our final study on a nonce-misuse kind of attack. All those topics are presented in a chronological order, in order to observe how ideas came and went along the way. They however are all motivated by the search for a better understanding of the degree's growth throughout the rounds. I will try to show with the 6-month hindsight we have now, how those subjects are linked one to another.

II First steps in studying and understanding Ascon

II.1 First results

As I explained earlier, the first initialization scenario we focused on was the one where all the accessible variables are located on the fifth row r_4 (N_1) while IV and N_0 (rows r_0 and r_3) stayed fixed and the two key rows were unknown variables. We will respectively denote v_i and k_i, k_{i+64} the variable and the two key bits located in column C_i. In order to use the array representation of the state, we will denote $c_{i,j}$ the output coordinate located in column C_j on row r_i; $c^r_{i,j}$ when the number of rounds r is needed.

In this setting and after one round, each coordinate of a given column only contains linear terms (column C_i can only contain a linear term of the form $q(k_i, k_{i+64}) \times v_i$ after p_S and two other similar terms after p_L) and/or a constant term. This is because the S-box is applied column-wise, so multiplications can only occur between constants and degree-1 monomials. If *the*[2] monomial of degree 64 were to appear in some or all coordinates of the first row after seven rounds, it necessarily means that the sequence of maximal degrees after each round has the following first terms: $1, 2, 4, 8, 16, 32, 64$. Indeed, the S-box is quadratic, so the highest degree, with this initialization and after seven rounds, is upper-bounded by $2^{7-1} = 64$. In order to obtain a term of such degree after seven rounds, it is necessary to also obtain the highest possible degree for all the previous rounds as the multiplication of two Boolean monomials gives rise to a monomial of degree $d \leq \deg_1 + \deg_2$ with an equality when the two monomials are coprime[3] (because in the Boolean world: $x^2 = x$).

These observations lead us to only focus on terms of highest possible degree throughout the rounds. This further meant that we may skip the round constant addition p_C (which only affects constant terms and thus does not affect highest-degree monomials in the next rounds) and even more, focus only on the quadratic part of the S-box (which is the only part from which a growth of degree is obtained).[4]

Furthermore, we can observe that the role of the IV (as well as the round constant additions) is to break a cyclic behavior of the permutation:

Proposition II.1. *Let use consider Ascon's initialization with a null IV up to r rounds, $r \in [\![1, 7]\!]$. Let $V = v_{i_0} v_{i_1} \cdots v_{i_{2^{r-1}-1}}$ be a monomial of highest possible degree. Then, for any $(i, j) \in [\![0, 4]\!] \times [\![0, 63]\!]$, V is present*

[2] Our polynomials belong to $A[v_0, \cdots, v_{63}]/(v_0^2 + v_0, \cdots, v_{63}^2 + v_{63})$ where $A = \mathbb{F}_2[k_0, \cdots, k_{128}]/(k_0^2 + k_0, \cdots, k_{127}^2 + k_{127})$, that is why there is only a single monomial of degree 64.

[3] *i.e* they do not have any variable in common

[4] For the first round which acts differently, it is necessary to keep both the addition of the round constant and the linear part of the S-box.

in coordinate $c_{i,j}$ if and only if $v_{i_0+k}v_{i_1+k}\cdots v_{i_{2^{r-1}-1}+k}$ is present in coordinate $c_{i,j+k}$, where variables indexes are considered modulo 64.

Thus, by using a null IV, if we were able to determine whether the degree-64 monomial were present or not in the ANF of an output coordinate in the first row, we would be able to obtain the same result for all the other coordinates of the first row ($v_0v_1\cdots v_{63}$ being equal to all its cyclic shifts). In order to study the terms of highest degree independently from the column they are in, we decided to first study Ascon with a null IV.

Those simplifications (no round constant, quadratic part of the S-box, null IV) will be used further for several results. They will be mentioned when needed.

II.2 Column dependencies

With Ascon, one needs to know 11 columns from the former round in order to compute a single column (because of the 11 shifts used in the linear layer p_L). To compute those 11 columns, one needs to know 45 columns of the second-to-last round and the full third-to-last state is needed to determine those 45 columns. Therefore, in order to find information about a new round for one column, one needs to know almost the full state two rounds before (two thirds of it). It thus seems unfeasible to get meaningful knowledge on future rounds from just a partial knowledge of the current state. The choice of a null IV does not change this fact (which only depends on the linear layer).

However, if we are interested in the ANF of a column (and not its actual value), by using a null IV and no addition of round constants, the ANF of column i can be directly computed from any other column's ANF. This can be explain by the fact that the cyclic behavior generalizes from highest-degree terms (Proposition II.1) to all terms when round constants are null. It further means that any information about some monomials (presence, absence...) of column C_0 can be then extended to similar knowledge on all the others columns.

II.3 On the loss of degree

Based of the fact that a product of two monomials will lead to a monomial of degree $\deg_1 + \deg_2$ if and only if they are co-prime, I looked at some basic probabilities in order to get better intuitions on how things could work in Ascon. The probability of randomly picking a monomial of degree d which is co-prime to a fixed degree-d monomial is $\frac{\binom{64-d}{d}}{\binom{64}{d}}$. Assuming that degree-$d$ monomials appear at random in each coordinate of a single column (which is absolutely not the case, at least on the first rounds), it means that:

- the probability of going from two terms of degree 4 to one of degree 8 was about 77%,
- 32% from 8 to 16,
- $4‰$ from 16 to 32,
- $\frac{1}{\binom{64}{32}} \simeq 2^{-60.7}$ from 32 to 64, as once a term of degree 32 is fixed, there is only a single choice (the complementary monomial).

This seems to lead to two *intuitions*:

- Until after six rounds, there are no "loss of degree", meaning that there actually are terms of degree 32 in output of the sixth S-box layer (and the highest degrees are also reached at each previous rounds as

seen before). This is due to the fact that "enough" multiplications occur during these rounds compared to the probability.

- However, there might not be a term of degree 64 after round 7 because of an insufficient number of random multiplications. Moreover, even if one multiplication between two right terms is enough, it seems unlikely that the permutation was built in order to force the appearance of the degree-64 term, in this particular setting.

To investigate on the former intuitions, I recovered, thanks to the use of Sage, the list of the degree-4 terms after the third S-box layer (and later on the degree-8 terms after the fourth one) which confirmed the "obvious" fact that some of them actually existed.

Afterward, using the five lists corresponding to the five coordinates of a fixed column, I was groping for a degree-32 term. This search was based on two points:

- As seen before, a degree-32 term after round 6 necessarily comes from a product of two terms of degree 16, which come from four terms of degree 8, coming from eight disjoint terms of degree 4 after round 3. So it seems interesting to try following the trail from terms of degree 4 to the term of degree 32.
- During the linear layer, a row is modified to become the sum of three rotated versions of itself, including the null rotation. Thus, in a fixed column, we are guaranteed that some of the monomials present will result from multiplications which occurred between previous coordinates of this fixed column.

Thus, I first fixed a chosen *trail*: I decided that the among the eight disjoint terms, one should come from $c_{0,0}$, one from $c_{1,0}$, one from $c_{2,0}$, two from $c_{3,0}$ and three from $c_{4,0}$ which is a valid choice. Using this trail in this order, and filtering the lists each time a new degree-4 term was chosen, I managed to find a combination of eight terms which could lead to the appearance of a degree-32 term after round 6 in coordinate $c_{0,0}$. This is in line with the appearance of degree-32 terms after round 6 and a first "loss of degree" at round 6 or later.

However, it does not prove that this degree-32 term will surely appear in the ANF after round 6: We also have to take into account that it might be canceled by the appearance of the same monomial with the same coefficient coming from another trail (or from a sum of terms from different trails with different coefficients but in front of the same monomial). This kind of phenomenon does happen from time to time, however they seem to be marginal and do not seem to threaten the existence of some degree-32 term after round 6.

Here, coefficients in front of the monomials were not taken into account, but they should. The study presented in Section VIII is closely related to the preliminary work I just presented. Unlike this study, it aims at focusing on the coefficients which are of important matter: as polynomials in key variables, they contain meaningful information in the context of a key-recovery.

III Statistical and combinatorial study

Another way of grasping Ascon and its permutation was to study statistical and combinatorial data. The first idea was to look at very basic statistics such as the number of variables present in highest-degree terms, the number of times a monomial appears in a single coordinate, row or column... The intuition behind this was that it could be possible to benefit from the fact that Ascon does not have optimal diffusion. In particular, this should be visible in the way the highest-degree terms evolve throughout the rounds, and we may be able

to use this insight to study the absence or presence of some specific monomials.

To gather those data more effectively I implemented some Sage and C programs benefiting either from powerful libraries or powerful computations with very raw data structures; the truthfulness of the results being verified, when possible, on the other implementation.

The C implementation was actually less precise that the one done on Sage: it did not handle the coefficients in front of the monomials and thus did not manage to cancel some terms when the coefficients were nullified by a XOR of two equal quantities. As a result, the sets studied were not the actual sets of maximal-degree terms but instead bigger sets *containing* all the maximal terms. In the quest for non-appearing monomials after 6 rounds, this is not a problem: the resulting set of absent monomials is smaller than the actual one but they would all still be absent.

In order to be a little bit more precise, I managed to correct "by hand" the cancellations occurring during the second round of Ascon in the null-IV scenario: it resulted in $+10\%, +1.5\%, +1.4\%, +1.4\%, +17\%$ terms after the third round on each coordinate of a fixed column. The remaining errors are due to cancellations occurring during round 3 which I did not handle. It however seems to predict that these cancellations will occur in proportion less and less as the rounds go (cancellation during the second linear layer: 2 terms in y_0 (11%) and 6 in y_1 (50%)) which seems to go in the same way our intuition above went.

Thanks to all this preliminary work I managed to extract some data which are now presented.

III.1 Number of variables

- It seems pointless to study only the number of variables present in the maximal-degree monomials of each row as they are all here after the second S-box layer (and even after the first one in the null-IV scenario).
- Regarding the columns, they all contain all the variables in at least one maximal-degree monomial after the third linear layer.
- After the fourth linear layer, all the variables will be present in at least one maximal-degree monomial of each of the coordinates.

All those results seemed to indicate that it was pointless to continue in this way, I thus chose other approaches.

III.2 Number of distinct maximal monomials

I computed the number of distinct highest-degree monomials after the first, second and third S-box layer (which are the same as after the respective first, second and third linear layers which do not add any new monomial): 64, 1602 and 364140. In comparison with $\binom{64}{1} = 64, \binom{64}{2} = 2016, \binom{64}{4} = 635376$, we observe that, in proportion, more and more monomials are missing: 100% were present after the first S-box layer, 79.5% after the second one and finally 57.3% after the third one. This tendency will certainly not stop after the fourth one, as among the 364140 monomials of degree 4, all will not multiply together, and most importantly, they are quite a lot to share common divisors, as we will see right now.

III.3 Variables distribution

I looked at the distribution of the variables among the highest-degree terms and throughout the rounds. After three rounds of Ascon, the distribution is still highly biased (see Figure 4.2): some variables are still missing whereas the variables present from the previous rounds have probability of appearing significantly higher than the newcomers. At that time, I had the feeling that we could study in the same way the distribution of binomials among degree-4 terms, one round after, in order to follow more precisely what happens inside Ascon's S-box. But this would mean being able to distinguish binomials which already existed a round before from the one created among a degree-4 term by a new multiplication. Again, the need to track the trail of the possible multiplications arose (see Section VIII). This study was unfortunately not done, instead we looked at the reasons which could explain the absence of some binomials.

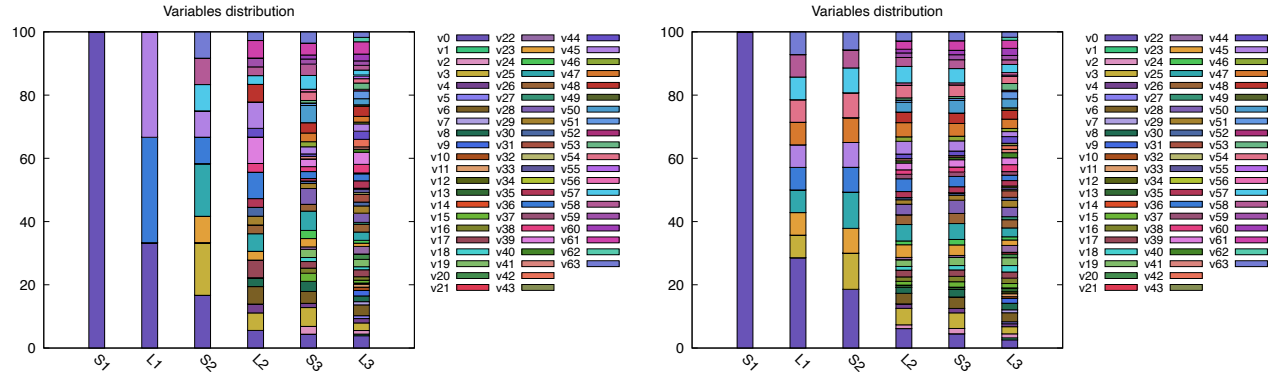


Figure 4.2: Distribution of the public variables among the highest-degree terms in coordinate $c_{0,0}$ (left) and column C_0 (right)

III.4 Study of the binomials

We now focus a little bit more on the study of the binomials. We will investigate the reason of the missing 384 binomials after round 2 in the null-IV case. As shown on Figure 4.3, we find for every variable, 12 binomials that are missing in a cyclic manner. The missing couples are, for a fixed i, $x_i x_{i+j}$, where:

$$j \in \{5, 8, 14, 15, 27, 30, 34, 37, 49, 50, 56, 59\} = \{5, 8, 14, 15, 27, 30, -30, -27, -15, -14, -8, -5\}.$$

The second writing is done modulo 64. It can be explained by the fact that $x_i x_{i+j}$ is missing if and only if $x_{i+j} x_{(i+j)-j}$ is missing. For a fixed variable x_i, the missing binomials in which it is involved can be separated in three groups:

- When j belongs to $\{8, 14, 15, 49, 50, 56\}$, x_i and x_{i+j} never appear in the same column before the second S-box layer, thus no multiplication occur between them and thus no appearance.
- When j belongs to $\{5, 30, 34, 59\}$, the two variables appear together in a single column but on the same row, leading to no multiplication through the S-box S and thus no appearance.
- Finally when j belongs to $\{27, 37\}$, the two variables appear together in a single column, on row 0 and 2 respectively but as $x_0 x_2$ is one of the two missing binomials in the ANF of the S-box, it leads to no multiplication and thus no appearance.

It seems very tempting to try generalizing the process used here in order to find, for example, missing degree-8 terms after the 4^{th} round. The main problem we will face trying to adapt recursively the technique

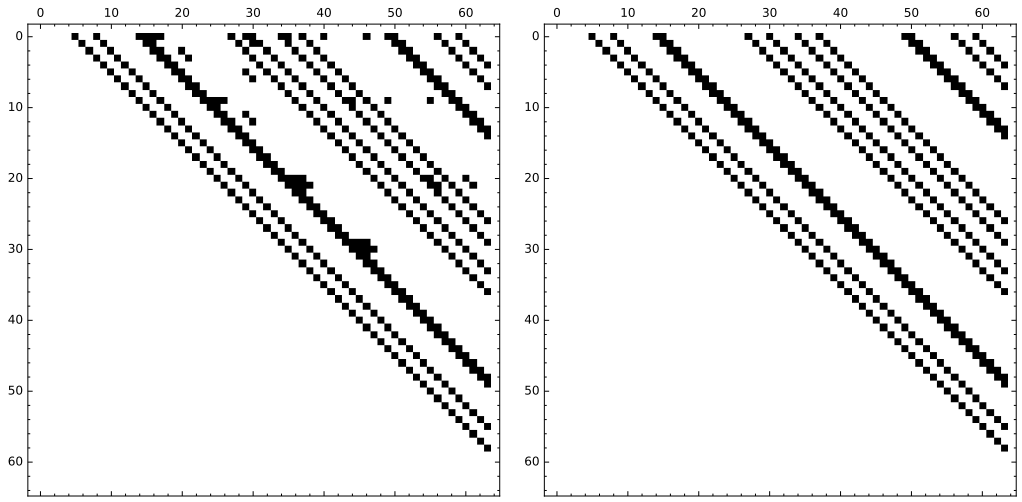


Figure 4.3: Missing binomials after 2 rounds of Ascon in the usual (left) and null (right) IV scenarios

used here is the following: in terms of missing binomials, as the entire set of variables is present after the first round, the absence of a single binomial is equivalent to the non-multiplication of the two concerned variables. For missing degree-4 terms or more, this is not the case anymore: the absence of a degree-4 term is actually due to the non-multiplication of several pairs of binomials, each of these non-multiplications being explained either by the absence of a binomial or an impossible multiplication between the two binomials (we moreover do not take into account cancellation due to XORs of similar values!). This explains why we cannot just replace variables by binomials and expect to claim the absence of a degree-4 term thanks to a single non-multiplication as it was the case for degree-2 terms.

Nevertheless, the second round scheme is a "non-multiplication between columns scheme" and it could be useful later on. At the end of the first round, each coordinate of each column contains the variable indexed by the column. So the missing binomials after round 2 give in fact information about a general round behavior: if $x_i x_j$ is a missing binomial, it means that the content of columns i and j will never multiply each other after one round of Ascon. In other words, if, in an appropriate context, one needs to know that a multiplication between two terms does not happen, one may for example use the fact that these two terms appear exclusively and respectively in columns i and j. We could be more precise by just studying the missing degree-2 terms in the general ANF (in 320 variables), thus replacing impossible multiplications between columns by impossible multiplications between coordinates.

From another point of view, we can also try to take advantages of the knowledge of those missing binomials after round 2. As a degree-4 term $abcd$ can only come from the multiplication of two binomials, $abcd$ will NOT appear after round 3 if and only if:

1. ab does not multiply cd AND
2. ac does not multiply bd AND
3. ad does not multiply bc.

As we already stated, the non-multiplication of two terms can either comes from the absence of one of them or from an impossible multiplication due to their position. The second case does not interest us here as it uses knowledge about existing binomials. As we know the missing binomials, we can build missing degree-4

term by choosing them such that for each condition 1, 2 and 3, one of the two binomials involved is missing. By renaming appropriately the variables, there exist only two kinds of possible choices:

- Choice 1: ab, ac and ad are missing.
- Choice 2: ab, ac and bc are missing.

Choice 1 is easy to find: when the first variable is fixed, we only need to loop through the 3-subsets of the offsets given above. For choice 2, once the first variable a is fixed, we need to investigate if, for a choice of variable corresponding to an offset b, there exists another variable c such that both binomials ac and bc do not appear. The only choice for (b, c) are $(15, 30), (15, 49), (34, 49)$ (in "offsets" notation according to a). d can finally be chosen freely in the set of the 61 other variables. In the end, we find 16256 missing 4-term monomials after round 3.

In the same way, a *sufficient* condition for the absence of $abcdefgh$ after round 4 can be determined only thanks to missing binomials: if $ab, ac, ad, ae, af, ag, ah$ do not appear after round 2, $abcdefgh$ will certainly not appear after round 4. With this method, we find $64 \times \binom{12}{7} = 50688$ missing degree-8 terms after round 4, since the set $\{j \mid x_i x_{i+j} \text{ is missing}\}$ has size 12.

These methods (choice 1 of round 3 and the sufficient condition for round 4) cannot lead to a list of missing terms after round 5 in the same way because $(16 - 1) \geq 12$.

I also tried to look at other sufficient conditions for the absence of $abcdefgh$ thanks to the absence of other sets of binomials. There exist 105 decompositions of a single 8-uple into four binomials and for any binomial, there exist 15 decompositions that contain it. Therefore, as $7 \times 15 = 105$, 7 is the minimum number of absent binomials needed to guarantee the absence of an 8-uple. There is no other 7-binomial condition than the $ab, ac, ad, ae, af, ag, ah$ condition we used above. There does not exist any 8-binomial condition that does not contain any of the 7-condition used before.

IV Study of the general ANF

Another way I followed in order to get more insights on Asconwas to study the general ANF. By general ANF, I mean the function $p\colon \mathbb{F}_2^{320} \to \mathbb{F}_2^{320}$ which depends on 320 input bits compared to the derived function (see Section I) used before which had 64 public variables and 128 key variables. This permutation is the core of Ascon and its study, even if it is surely more difficult than the studies of its derived versions, is a goal in itself. I did not managed to get very relevant results on it, however some observations made at that time are still interesting and could be useful in some ways, if properly used.

Figure 4.4 presents how the highest-degree terms of a column's coordinates depend on all the 320 inputs variables: each time a monomial of degree 4 after two rounds (or of degree 8 after three rounds) is divisible by one of the 320 variables, the corresponding square is blackened. This way, we obtain a global picture of how the dependency actually looks like. Note that we present here only a single column as the cyclic behavior of Ascon stands with the general ANF: we can determine the dependency of the highest-degree terms of each other column by shifting the figures cyclically.

We can make a few remarks on these figures:

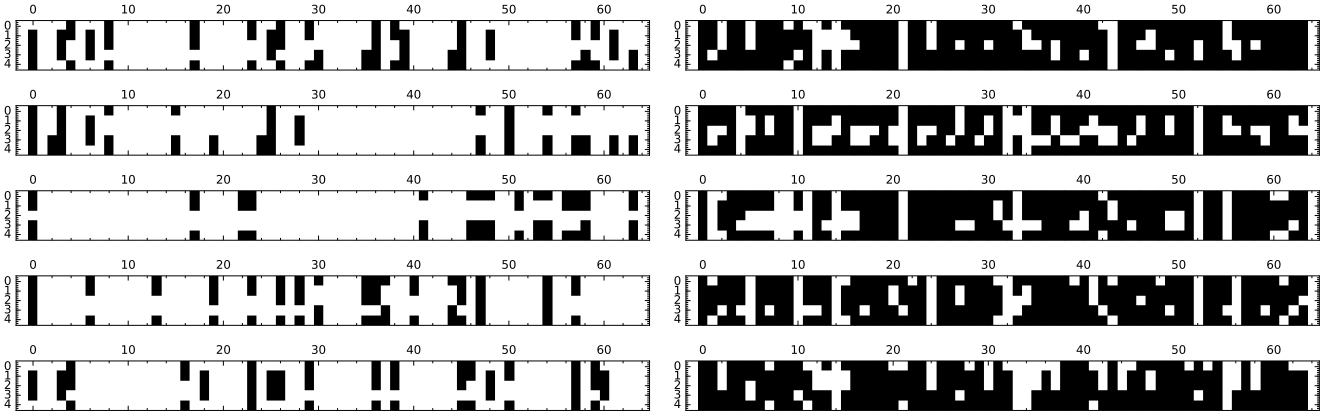


Figure 4.4: Dependencies of a column's highest-degree terms after two (left) and three (right) rounds

- The dependencies after two rounds are so sparse that we could use it to track the next existing maximal-degree terms: it either can give an overview of the possible terms of degree 4 in each coordinate after two rounds, but also, maybe, to use knowledge about a current state and "push it" two rounds forward.
- We can observe, among the impossible combinations on two rounds, a lot of empty columns either shared by the five coordinates or some of them. It means that column C_0 will never depend on those columns (this is only due to indexes which are impossible to obtain when summing two shift values of a same row). A more remarkable point is the empty row on the diagram corresponding to row r_2: it means that the highest-degree terms on row r_2 will never depend on the highest-degree terms on the same row two rounds before.
- On the other hand, after three rounds, a column is now dependent of almost every variable (or the highest-degree terms in a column depend on almost every previous highest-degree terms three rounds before). However we can distinguish full columns that will not be taken into account while computing three more rounds, leading to some impossible combinations.

This kind of study was also made on other decompositions of Ascon's permutation which are presented in the following section. It gave comparable results but they seemed often less exploitable.

V Different representations of Ascon's permutation

V.1 Splitting the S-box into two parts

In order to better understand the evolution of the degree throughout the rounds, we decided to look at other representations of Ascon's permutation. Ascon's S-box seems quite sparse (among the five coordinates, there are only 11 binomials in the ANF and only 8 distinct ones out of the 10 possible) and it is the only source of non-linearity. We thought it could be possible to look at it in such a way it would minimize those two quantities, by using linear changes of variables on the domain or codomain.

First, I looked at some changes of variables on the codomain. By computing all the 31 non-null linear combinations of the S-box coordinates, I tried to select five linearly independent ones, while minimizing the number of binomials appearing. It led to the only possible minimizing choice: y_2 and $y_0 + y_4$ (with a single binomial each) and $y_3, y_4, y_0 + y_1 + y_4$ (with two binomials each). It actually coincides with the choice of replacing y_0 by $y_0 + y_4$ and y_1 by $y_0 + y_1 + y_4$. With this invertible change of variables which we will note A,

we can actually write that $S = A^{-1} \circ A \circ S = A^{-1} \circ (AS)$. And as A^{-1} is a linear function we can actually look at one round of Ascon, as $L' \circ S' = LA^{-1} \circ AS$. The ANF of S' and L' are given below:

$$
\begin{aligned}
y_0 &= x_0 + x_1x_2 + x_2 + x_4 \\
y_1 &= x_1x_3 + x_1 + x_2x_3 + x_3 \\
y_2 &= x_1 + x_2 + x_3x_4 + x_4 + 1 \\
y_3 &= x_0x_3 + x_0x_4 + x_0 + x_1 + x_2 + x_3 + x_4 \\
y_4 &= x_0x_1 + x_1x_4 + x_1 + x_3 + x_4 \\
&- \\
y_0 &= (x_0 + x_4) + (x_0 + x_4) \ggg 19 + (x_0 + x_4) \ggg 28 \\
y_1 &= (x_0 + x_1) + (x_0 + x_1) \ggg 61 + (x_0 + x_4) \ggg 39 \quad \text{other unchanged}
\end{aligned}
$$

Figure 4.5: ANF of S' and partial ANF of L'

As we can see, this representation of the permutation enables to get rid of the products appearing twice. In a way, it allows a better understanding of how Ascon works: the S-box actually includes a linear transformation which is the reason of some binomials appearing twice. One can look at the simplified S-box S' as the part which produces the binomials and at the more complex linear layer L' as the part which duplicates and diffuses some of the binomials.

V.2 Changes of variables

As the example above shows, it seems that some ways of looking at the S-box can simplify it a bit. We just split the S-box into two parts and add the second one to the linear layer but another idea is to completely change our basis. By choosing a single invertible change of variables for the domain and codomain, we can see the S-box differently. In particular, we may find changes of variables through which the S-box will be sparser. As the S-box only handles 5 bits, we can try a brute-force approach to select the best possible changes (at the cost of $|\mathrm{GL}_5(\mathbb{F}_2)| = \prod_{k=0}^{4} (2^5 - 2^k) = 9999360$ tries).

We are interested in minimizing both the number of binomials and the number of distinct binomials appearing in the ANF of the S-box. From the first experimental results it seemed that the sum of those two values cannot be less than 16 but these cases are actually already better than the S-box original representation ($11 + 8 = 19$). Two special cases seemed to be of particular interest: the $9 + 7$ and the $11 + 5$ which are the extreme cases.

After looking more carefully at the $9+7$ and $11+5$ cases, we observed that each of them occurred 120 times. It was intriguing to find the same amount for both cases but it can actually be explained quite easily. Instead of looking at all the possible changes of variables (*i.e* . at all of $A \circ S \circ A^{-1}$ where A is an invertible matrix), we can instead look at them up to permutations (*i.e* consider $\bar{A}$ instead of A, where $\bar{\cdot}$ represents the class modulo $\mathcal{S}_5$). Indeed, it is of no particular interest for us to consider two changes which are in the same class as two distinct changes: changing the variables' labels and the order of the S-box coordinates accordingly does not enable us to look at S differently and leave the amount of binomials and distinct binomials unchanged.

As $|\mathcal{S}_5| = 5! = 120$, we can observe that there actually exists a single $9+7$ case, a single $11+5$ case, and four intermediate $10+6$ cases. Two representations of the respective $9+7$ and $11+5$ cases are given below. They were chosen among the 120 possibilities in order to avoid useless changes of labels.

$$
\begin{aligned}
y_0 &= x_0 + x_1x_2 + x_2 \\
y_1 &= x_0 + x_1x_2 + x_1x_3 + x_1 + x_2x_3 + x_2 + x_3 \\
y_2 &= x_1 + x_2 + x_3x_4 + x_4 + 1 \\
y_3 &= x_0x_3 + x_0x_4 + x_0 + x_1 + x_2 + x_3x_4 + x_3 + x_4 \\
y_4 &= x_0x_1 + x_1 + x_3 + x_4
\end{aligned}
\qquad
A = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$

Figure 4.6: Representation of S minimizing the number of binomials $(9+7)$: ANF of $A \circ S \circ A^{-1}$

$$
\begin{aligned}
y_0 &= x_0 + x_1x_2 + x_2 \\
y_1 &= x_0 + x_1x_2 + x_1 + x_2x_3 + x_2 + x_3 \\
y_2 &= x_0 + x_1x_2 + x_1 + x_2x_3 + x_3x_4 + x_3 + x_4 + 1 \\
y_3 &= x_0x_4 + x_0 + x_2 + x_3x_4 + x_3 + x_4 \\
y_4 &= x_0x_1 + x_0x_4 + x_0 + x_1 + x_2 + x_3x_4 + x_3
\end{aligned}
\qquad
A = \begin{pmatrix}
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1
\end{pmatrix}
$$

Figure 4.7: Representation of S minimizing the number of distinct binomials $(11+5)$: ANF of $A \circ S \circ A^{-1}$

V.3 Different changes in the domain and codomain

Later on, we also studied linear changes of variables in its most general way, namely PSQ^{-1} where P and Q are linear invertible functions. Indeed, in the study we wanted to look at, there was no point of using the same change in both domain and codomain. The space of all changes of this form is however way bigger and we did not use the same brute-force approach. It was clear from experimental trials that we could find better results than before: a $8+5=13$ case was for example found by chance. But we also had a $5+5=10$ case at hand since the beginning! As a matter of fact, a circuit representation is given in the specifications (also reproduced here, see Figure 4.8) in order to describe the bit-slice implementation of Ascon. On this figure, we can actually observe two linear transformations (Q and P), which give rise to the mentioned representation containing five binomials, all of which distinct from the others.

With this point of view, $\tilde{S}$ is of course the new S-box and the new linear layer will be $\tilde{L} = Q \circ L \circ P$. In order to use this representation, we however need to go through Q^{-1} before the first round and through P^{-1} after the last round in order to follow the right input-output couple. The study of the S-box is way simplified by these considerations: the only product appearing in each row is the product of the two following rows (in descending order). However the linear layer is not operating row by row anymore: a coordinate can depend (according to the row number) on coordinates from two, three or four rows after the new linear layer. After looking at this representation in the same spirit as we looked at the general ANF in Section IV, it is still not clear how this representation can be used.

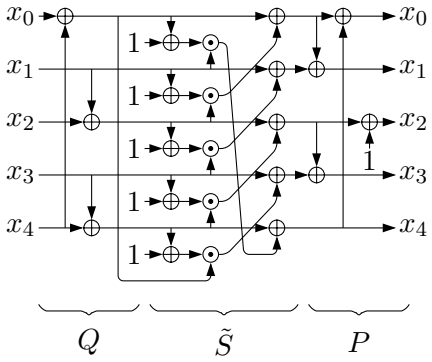


Figure 4.8: Ascon's circuit representation

On the other hand, this representation is the "best" representation we can hope for if we want to minimize both the number of binomials and the numbers of distinct binomials in its ANF.

Proof.

- Minimum number of binomials. By contradiction, let us suppose that there exist P, Q such that $P\tilde{S}Q$ contains at most four binomials in its ANF. One of the coordinate is therefore at most linear in the input variables. It is also the case of $P\tilde{S} = (P\tilde{S}Q)Q^{-1}$ (as the former input variables are replaced by linear combinations of themselves). $P\tilde{S}$ having a linear coordinate means that there exists a linear combination of the coordinates of $\tilde{S}$ canceling the quadratic terms. This is absurd as the binomials in each of them are, "by definition" of a polynomial ring, linearly independent. So the minimum number of binomials is therefore 5.

- Minimum number of distinct binomials. By contradiction, let us suppose that there exist P, Q such that $P\tilde{S}Q$ contains at most four distinct binomials in its ANF. Then, there exists a linear component S_b of $P\tilde{S}Q$: we can look at the quadratic part of the coordinates as Boolean vectors of size at most 4 indicating whether the quadratic terms are present or not; 5 vectors of a 4-dimensional vector space are necessarily linearly dependent. This further means that $\mathcal{E}(S_b + \varphi_a) = \begin{cases} 0 & \text{if } a \neq b \\ 2^5 & \text{if } a = b \end{cases}$ (where $\mathcal{E}$ is the bias of a function and φ_a the linear Boolean function corresponding to $a \in \mathbb{F}_2^5$, see [Can16] for more details). In particular, $2^5 = 32$ appears in the LAT of $P\tilde{S}Q$. But the values of the LAT of $P\tilde{S}Q$ are among the ones in the LAT of Ascon's S-box (as they are affine equivalent, see Proposition 2.16 of [Can16]). 32 does not appear in the LAT of Ascon's S-box (the LAT can be found in Table 11 of [DEMS19]), so this is absurd. The minimum number of distinct binomials is therefore 5.

□

VI Other initializations scenarios

After spending a lot of time switching from the general ANF to the derived one related to the initialization presented before (the nonce row N_0 (r_3) being fixed, while the public input variables were set on the row $N_1(r_4)$, now referred as Scenario 1), we decided to look at other possible initialization scenarios:

- Scenario 2: It is the "reverse" one where N_1 stays constant while the public variables are set in N_0.
- Scenario 3: We insert each given public variable v_i in both N_0 and N_1 in column C_i (v_i is inserted in coordinates $c_{3,i}$ and $c_{4,i}$)

$$C_i = \begin{pmatrix} \mathrm{IV}_i \\ k_i \\ k_{i+64} \\ 0 \\ v_i \end{pmatrix} \quad C_i = \begin{pmatrix} \mathrm{IV}_i \\ k_i \\ k_{i+64} \\ v_i \\ 0 \end{pmatrix} \quad C_i = \begin{pmatrix} \mathrm{IV}_i \\ k_i \\ k_{i+64} \\ v_i \\ v_i \end{pmatrix}$$

Figure 4.9: Initializations of Scenarios 1, 2 and 3 (from left to right)

For each of these three scenarios, we use either a null IV or the genuine one. All those different initialization possibilities give rise to different 64-variable functions with different properties. Scenarios 1, 2 and 3 are used by Rohit *et al.* in their article [RHSS21].

These new scenarios (2 and 3) seemed at first sights more appropriate to our study. As a matter of fact, Scenario 2 enables to find a constant coordinate after the first S-box layer: no nonce variable appears after one round on row r_2. Consequently, fewer binomials are present after two rounds, as products involving this coordinate will not lead to binomials anymore. On another subject, the coefficients in front of the linear terms after the first round are only dependent of the 64 first key variables (those on row r_1). This means that, as long as the highest-degree terms are obtained **only** by multiplications of highest-degree terms, those coefficients of highest-degree terms will only depend on 64 bits of the key. This is the main point used by Rohit *et al.* in order to mount their attack on one half of the key (which is followed by an exhaustive search for the other half).

With Scenario 3, there are now two coordinates per column with no linear terms (on rows r_2 and r_3). As a result, we observe that only three products (x_0x_1, x_0x_4, x_1x_4) of coordinates can lead to binomials after two rounds and thus only three of the output coordinates (x_0, x_3, x_4) actually contain quadratic terms. After the third round, again, only three products lead to degree-4 terms and only two coordinates (x_2, x_3) will contain them. Finally, only one product will lead to degree-8 terms which is contained in a single coordinate's ANF: after the fourth S-box layer, a single coordinate contains degree-8 terms. It further means that after the fifth, sixth and seventh S-box layer, we surely will not find any terms of degree 16, 32, or 64. This is the reason why Rohit *et al.* are able to find better distinguishers that the previous one using this initialization scenario. Unlike in Scenario 2, the coefficients in front of highest-degree terms may depend on the full range of keybits.

Afterward, we spent more time analyzing Scenario 3 which seems to be more promising than Scenario 2. In Scenario 3, the linear terms after the first S-box layer are independent from the IV value. The study was here made with a null IV but the same stays true with the genuine IV.

VI.1 About Rohit *et al.*'s distinguishers

By keeping track of upper bounds on the degree and taking into account the multiplications in the computation of each coordinate, we can find upper bounds on the degree for each coordinate, round after round. This is the same method as before, only a bit more precise. This way, we obtain a table (see Table 4.1) similar to Table 3 of [RHSS21] (Part 7.2): only a single value is different (we are a bit less precise). While they used the division property and a MILP model to find it, it seems to be explained in a more straight-forward manner as done above. This leads to distinguishers costing 2^{60} chosen data (only row r_0 is output and all monomials of degree 60 are absent).

Round	0 (Init.)	1	2	3	4	5	6	7
Row r_0	0	1	2	3	7	15	30	59
Row r_1	0	1	1	3	8	15	29	59
Row r_2	0	0	1	4	7	13	29	60
Row r_3	1	0	2	4	7	14	30	60
Row r_4	1	1	2	3	6	15	30	**59**[5]

Table 4.1: Upper bounds of coordinates in each row using Scenario 3

VI.2 Focusing on terms of degree 8 during the fourth round

Looking at Scenario 3, we can also have a deeper understanding of round 4 and its maximal-degree terms: not only are they present on a single row, but also their structure is very strict. We were able to find a list of 72 binomials such that each 8-tuple will at least contain one of them. In other words, one fourth of these 8-tuples can be very easily determined.

As already mentioned, our way of finding upper bounds on the degree of coordinates was based on two points:

- upper bounds of the degree for each coordinate at the previous round; and
- multiplications occurring between coordinates of a single column during the S-box layer.

We can be more precise. After the third S-box layer, coordinate $c_{2,i}$ will only contain terms of degree 2 coming from the product x_3x_4 and coordinate $c_{3,i}$ will only contain terms of degree 2 coming from products x_0x_3 and x_0x_4.

After the fourth S-box layer, coordinate $c_{1,i}$ will only contain degree-4 terms coming from the product x_2x_3.

Based on those possible choices, we can observe that each 8-tuple associated to a degree-8 term (present on row r_1) contains at least one out of 72 binomials: one half of a 8-tuple necessarily comes from coordinate $c^3_{2,i}$ after the third linear layer. But each possible 4-tuple comes from three different columns (the current one and two others according to the shifts of row r_2) and one half of each 4-tuple comes from coordinate $c^2_{3,i}$,$c^2_{3,i+1}$, or $c^2_{3,i+6}$, depending of where they come from. After the second linear layer, coordinate 3 (of a fixed column) contains 24 binomials, meaning that every first half of an 8-tuple necessarily contains at least one of the $3 \times 24 = 72$ mentioned binomials.

Below are listed all binomials in coordinate $c^2_{3,0}$. For the full list, one just needs to shift all the variables by 1 and 6 (in a cyclic manner, modulo 64).

(29 51) (7 28) (0 28) (10 51) (10 29) (17 58) (0 19) (28 41) (24 45) (10 38) (19 41) (17 29)
(7 19) (38 51) (0 41) (45 58) (24 36) (17 45) (17 38) (10 17) (0 7) (36 58) (17 24) (17 36)

Figure 4.10: List of all binomials in coordinate $c^2_{3,0}$ in Scenario 3

[5]This value is replaced by 58 in the mentioned article.

These results seem to indicate that, in Scenario 3, the growth of the degree is slower that in other scenarios (as seen in the previous part), but the diffusion and the mixing among the highest-degree terms seems also to be less significant. We were tempted to use this knowledge to better understand the coefficients in front of those degree-8 terms. For example we could be tempted to say that, knowing an existing degree-8 term which is divisible by $x_a x_b$, and knowing the coefficient in front of $x_a x_b$ at the end of round 2, we would be able to guarantee that the coefficient after round 4 would be divisible by it. This is not true because product $x_a x_b$ in the degree-8 terms could come from two different halves. The following study (presented in Section VIII) aims at finding techniques to search for coefficients of some chosen monomials.

Moreover, the study we just made here on the fourth round in Scenario 3 is very specific and it seems that we cannot use it to go a few rounds further. The main reason is that, during round 5 and beyond, the highest-degree terms are not obtained by the multiplication of two highest-degree terms anymore (there is only a single coordinate in which terms of degree 8 are present). This is a good news for distinguishers, a worse one if we want to obtain information about further rounds. A lot of case-by-case analysis would need to be done (if it can be done).

Finally, I also tried to get more information about the degree-8 terms in Scenario 1 in a way similar to the one used for Scenario 3 but it was actually not so easy (after the first S-box layer, there are four coordinates per column in which a linear term is present compared to three coordinates in Scenario 3). I was not able to find such a small set for binomials dividing the terms of degree 8. However, the different amount of terms of degree 8 after the fourth S-box layer in the different scenarios can be partially explained thanks to a study which is similar to the one made to find the list of binomials.

We decided at that time to change a bit our direction of study.

VII Analysis of Ascon's cyclicity and the respective role of IV and round constants

VII.1 Cyclic properties and anomalies

As already stated, when used with a **null-IV** (or an all-one IV) and **no constant addition** at each round, Ascon is very structured. In fact, by knowing (or assuming) that a monomial is in the ANF of a coordinate, one can deduce that it is also "present" in the other coordinates of the same row in a cyclic way: if x_a appears in the ANF of $c_{i,j}$ then x_{a+k} will surely be in the ANF of $c_{i,j+k}$. This works in the same manner for monomials of higher degree. This also explains Proposition II.1, as round constants do not influence the presence of highest-degree terms in the ANF during the first rounds.

Using this fact, we were able to have a more optimized way to study the maximal-degree terms in the first rounds: instead of keeping track of all the maximal-degree terms in all the columns, we only kept track of the ones in column C_0 and deduced the ones in column C_j by shifting the ones in C_0. This enables us to find our first upper bounds on the number of maximal-degree terms after four rounds of the permutation in the different scenarios (they are here given row per row):

Here, we did not take into account possible deletions due to XOR cancellation, implying that the previous numbers are upper bounds on the number of monomials of degree 8.

Scenario	1		3	
Row r_0	47759848	1.079%	0	0%
Row r_1	65511919	1.480%	10983660	0.248%
Row r_2	21397928	0.483%	0	0%
Row r_3	39453281	0.891%	0	0%
Row r_4	31045646	0.701%	0	0%

Table 4.2: Upper bounds of the number of monomials of degree 8 in Scenarios 1 and 3 and the respective proportions compared to $\binom{64}{8}$

Following the study of the cyclicity of the simplified Ascon, we decided to look at the true Ascon permutation and tried to find some cyclic properties, or at least to observe if some of them also occur in this context. With **a null IV and no round constant**, Ascon's initialization has the following cyclic property:

$$p^b(0^*, R_i(k), R_i(M)) = R_i \circ p^b(0^*, k, M) \quad \forall\, k, M \in \{0,1\}^{128}, \forall\, i \in [\![0, 63]\!]$$

where R_i is a rotation of i positions. This is not the case anymore in the true setting as the IV and the round constants intend to break this kind of properties, sometimes known as *self-similarity* properties [BDLF10, BB02].

In order to study the cyclic/non-cyclic behaviors on the true Ascon's permutation, I implemented a program to compute cube sums. With a fixed IV and key, we can look at the outputs of Scenario 3 as multivariate polynomials of 64 variables. With this approach, we can be interested in looking at the *value* of the coefficient of a monomial. This coefficient is a multivariate polynomial in the key variables. Its actual value can be obtain thanks to Möbius transform (see Proposition I.5).

Studying maximal-degree terms in the first rounds (on which I already worked) enabled us to understand the first details on the role of the IV and the round constant.

In the case of Scenario 3, the IV does not play any role in the coefficients of the maximal-degree terms in the first rounds as it only modifies the constant terms after the first S-box layer (and not the coefficients of the linear terms). However, the constant c_0 does play a role on the linear terms as it flips four keybits of k_1 before any S-box or linear layer. With the **real IV and no round constant**, the maximal-degree terms through the first rounds are actually acting cyclically. The only constant being able to modify this behavior for the first rounds is c_0 and it actually breaks the cyclic behavior. We can show that after the second S-box layer, a term of degree 4 will be present in all of the columns in a cyclic manner. However, if the corresponding coefficients depend on some bits of k_1, then their values will never be the same for all the 64 cyclic siblings; as c_1 will always flip four fixed bits of k_1, while k_1 is rotated. The number of columns in which the value is different can give some information on the way the coefficient of the monomial was built (which multiplications between which rows, which original row for each variable, etc). It is also important to note that the cyclic behavior of a monomial depends on the row on which it is found: the presence of a monomial in two rows of a single column does not imply the same cyclic behavior as the coefficients in front of them can be different (it depends on the building trail of the monomials).

Even if c_0 flips a few bit of the key, this is not of major interest as (not) knowing a keybit or its flipped version does not change anything and we can actually compensate the cyclicity problem it creates:

Proposition VII.1. *Let R be a rotation of the columns of a 5×64 matrix. Let E be r initialization rounds of* Ascon *, with $r \in [\![1, 4]\!]$, as in Scenario 3. Let $x_{(i,j)}$ be any output coordinate of $E_{R(k \oplus c_0) \oplus c_0} \circ R$ (k and c_0 are viewed as 5×64 matrices) and $y_{(i,j)}$ be the corresponding output coordinate of $R \circ E_k$. Then, $x_{(i,j)}$ and $y_{(i,j)}$ have the same maximal-degree monomials in their respective ANF: in other words, the same terms of degree 1, 2, 4 or 8 if E is respectively a 1-, 2-, 3- or 4-round initialization.*

We then studied the influence of the IV and the round constants on the cyclicity. Now that the role of the constants on the terms of maximal degree is understood for the few first rounds, we looked up how they could influence terms of smaller degrees. By intuition, it is expected that the constants can break the cyclic behavior of monomials of smaller degrees: if we look at monomials of degree inferior to the upper bound, then it is very likely that they were created through at least a multiplication of a constant and a monomial (only *likely* as the multiplication of degree-d terms can lead to a monomial of degree smaller than $2d$ if they are not coprime).

I firstly worked on a null-IV version in order to isolate the role of the round constants. While looking at monomials of degree 7 after S4 and their cyclic "siblings", every time a unusual behavior occurred (*i.e* each time that a sibling appeared when the referential one did not or the other way around) we were able to observe the presence of c_1: anytime an anomaly happened, three others happened according to c_1's pattern (0x00000000000000e1, which has a Hamming weight of 4).

We only talked here about c_1 as no other round constant can play a role on the coefficients of terms of degree 7 after the fourth S-box layer: indeed, if a round constant modifies a coefficient of a term, then it necessary means that a constant was multiplied by a monomial leading to a "loss of degree". But a constant-monomial multiplication involving c_2 during the third S-box layer necessarily leads to a loss of at least 2, and in the same manner a constant-monomial involving c_3 during the fourth S-box layer necessarily leads to a loss of at least 4 which thus cannot be the case. Moreover, for the terms of degree $d-1$, if an anomaly actually occurs then the monomial cannot result from the multiplication of two monomials of degree d as they would not be influenced by c_1.

This also implies that round-constant dependency may be studied in a case-by-case manner, by focusing on both the degree of the monomial and the index of the round studied.

From the ANF of the S-box, we can figure out which part of the linear terms after the second S-box layer are influenced by any of the constant (depending of course of the column we are looking at). Indeed, a round constant is always added on the second row before any S-box, which means that the linear terms which will have a cyclic anomaly are the ones on row r_0 coming from x_1x_2 (with a linear part in x_1) and the ones on row r_1 coming from x_3x_2 or x_2x_1 (with a linear part in x_3 or x_2).

One remarkable detail is that a single constant can influence the cyclic behavior of a monomial multiple times. This is due to the different building trails a monomial can follow: following different trails leads to different coefficients in front of the monomial which may be flipped thanks to c_1.

Indeed, as we already mentioned, when we choose a monomial which was built through a multiplication between a constant on row r_2 and a linear term during the second S-box layer, then it will almost always lead to the appearance of c_1's pattern among the cyclic anomalies. After a few investigations, with the choice we just described, most of the coefficients will depend of a single constant coming from r_2, leading to the

appearance of c_1's pattern among the anomalies. On the other hand, when, through the two first linear layers, shifts are combined "correctly", it can happen that those coefficients depend on more than a single constant coming from r_2, leading to the appearance of other patterns among the anomalies.

Contrary to the role of c_1, the role of the IV seems to be more delicate to study. It is easier to obtain a multiplication between a linear combination of multiple IV-bits and a linear term: after the first S-box layer, an IV-variable is present among three out of five constants of the corresponding column, it will then propagate thanks to the following linear layer in ten other coordinates. Thus many of the degree-7 monomials' coefficients will depend of a former linear combination of multiple IV bits. This seems to be the reason why, even when the IV is the only part responsible for cyclic anomaly of a monomial (when no round constants are used) it is more difficult to observe its action through the cyclic anomalies' "portrait".

VII.2 Visual results on the influence of the IV and of the round constants

Using 100000 random keys-nonces pairs, I compared the state after multiple versions of the permutation.

First I compared a null-IV version with a null-IV version without any round constant in order to understand the global influence of the addtion of round constants, p_C. Figure 4.11 depicts the results as heat-maps of differences after 3, 4 and 5 rounds: each time the two values of a coordinate were different, the corresponding cell was incremented.

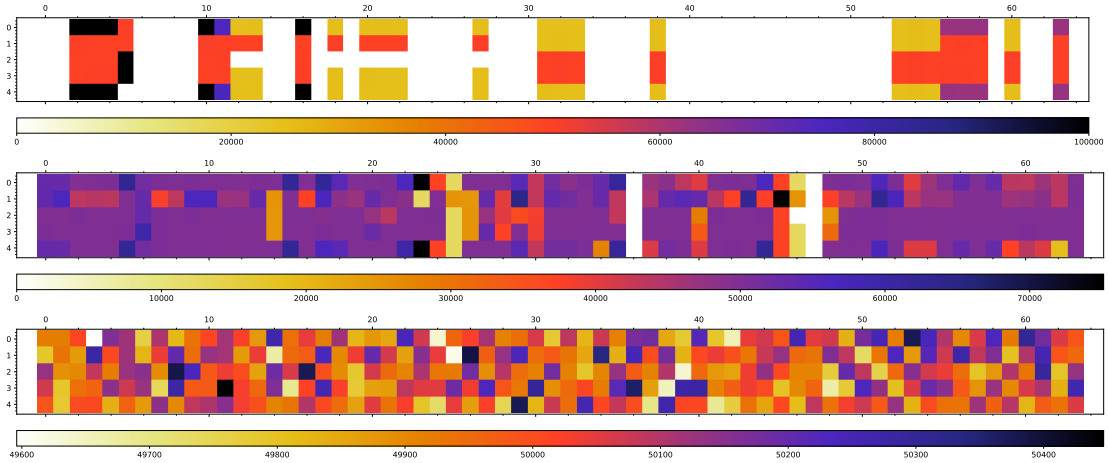


Figure 4.11: Comparison of states with or without round constants after 3, 4 and 5 rounds

We can observe that until four rounds (included), we can still find columns which are not altered by the constant additions. This is not the case anymore after five rounds: for each cell the number of equalities between the two versions is between 49600 and more than 50400, in other words a probability of equality around 50%.

We can observe the same way the influence of the IV on the permutation by comparing two versions with or without the real IV, both with no round constant (see Figure 4.12).

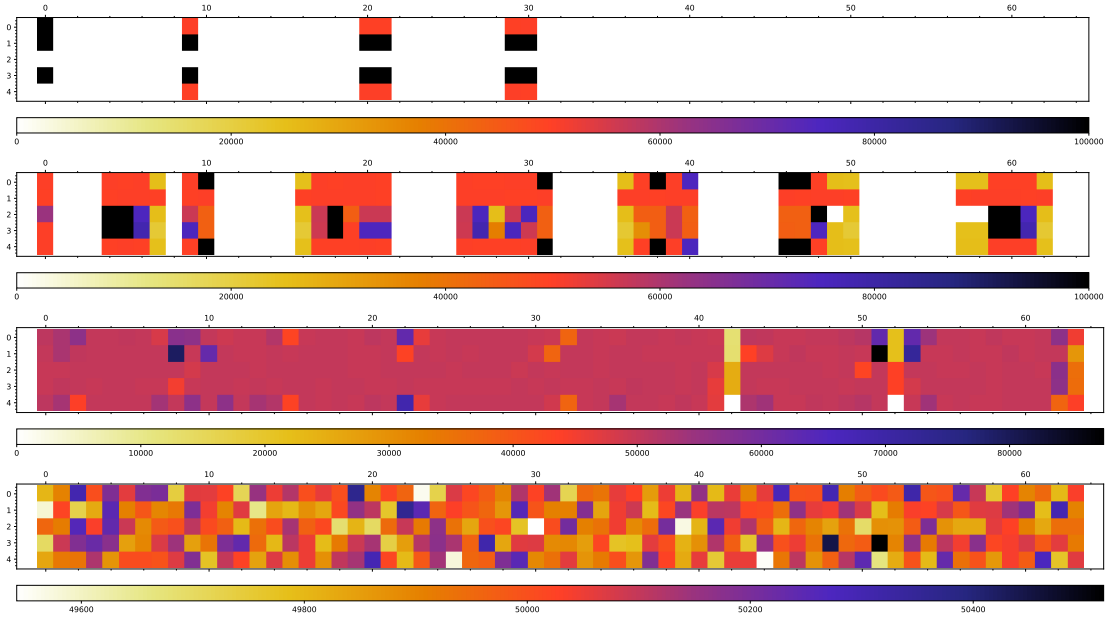


Figure 4.12: Comparison of states with or without round constants after 1, 2, 3 and 4 rounds

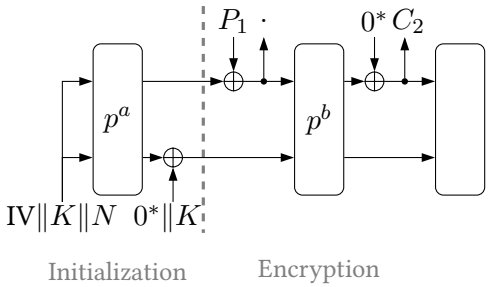


Figure 4.13: Nonce-misuse Scenario

VIII Searching for a superpoly in a nonce-misuse scenario

As we did not manage to get hold of Ascon through our previous studies[6], we decided to make a fresh start by changing our way of looking at it. In the following, we consider Ascon in a nonce-misuse scenario. In other words, we suppose that **a nonce is used multiple times with the same key**. We do not try to attack the initialization anymore, instead, we look at the second phase (as shown on Figure 4.13).

In this context, we consider a chosen-plaintext attack where an adversary can choose plaintext blocks (he thus controls the first row of the external state) and then recovers the next cipherblock (the attacker thus aims at recovering the first row after 6 rounds). As Ascon uses a Sponge construction, the goal of the adversary is to gain information about the capacity after the initialization. Discovering the full intermediate state (external state known, internal state recovered) could enable to get a lot of information: **if the XOR with the key were absent**, it could be possible to invert p^a, which is a permutation independent of any unknown variables. This would enable the recovery of the initial state which includes the secret key. In the case of Ascon, this does not seem to be a security problem as the XOR with the key at the end of the initialization[7] (which is, this time, a permutation depending of 128 unknown variables) prevents from recovering backward the initial state.

[6] There are some exploitable observations, but we did not manage to use them properly (yet).

[7] This looks like a Davies-Meyer construction, where the input is XORed to the image in order to make the inversion more difficult.

It is also important to note that we divert a bit from the authors' recommendations: they warned that Ascon's security claim was only in a nonce-respecting scenario.

Given a fixed key-nonce pair, our goal is to recover the coefficients (a multivariate polynomial in the capacity's bits after the initialization) of enough terms in order to gain information on the capacity. By knowing the actual values of the coefficients and the corresponding multivariate polynomials, we can hope for an easily-solvable system. We focused on highest-degree terms (degree 32 after 6 rounds), as they are now better understood.

The entire capacity cannot be determined with this method. Indeed, after one round, the coefficients of linear terms are actually independent of the second row r_2. This is a pity in our case as it does not enable the recovery of the full capacity. However this is a surprising property which could maybe used for another purpose.

We have some data at our disposal:

- Let us denote $M = \{i_0, \cdots, i_{31}\},\ i_j < i_{j+1}\ \forall\ j \in [\![0, 30]\!]$ the set of indexes associated to our cube's variables, C_M the corresponding vector subspace of dimension 32 (cube) in $\mathbb{F}_2^{64}$, and $\mathcal{M} = \prod_{j=0}^{31} v_{i_j}$ the corresponding monomial of degree 32.
- For every coordinate $(y, x),\ y \in [\![0, 4]\!],\ x \in [\![0, 63]\!]$, we know the list of all degree-4 terms (and the respective 4-tuples) after L_3 (the third linear layer). In particular, we know the list of all 4-tuples after L_3 whose coordinates are among the ones in our cube M:

$$Q_{y,x} = \{\{i_0^0, i_1^0, i_2^0, i_3^0\}, \cdots, \{i_0^{u_{y,x}}, i_1^{u_{y,x}}, i_2^{u_{y,x}}, i_3^{u_{y,x}}\}\}, \quad i_j^h < i_{j+1}^h\ \forall\ j \in [\![0, 2]\!]\ \forall\ h \in [\![0, u_{y,x}]\!],$$

$$i_j^h \in M\ \forall\ j \in [\![0, 3]\!]\ \forall\ h \in [\![0, u_{y,x}]\!].$$

- We also know the general ANF of coordinate $c_{(0,0)}$ after S_3 (the third S-box layer) and we thus have the list of all terms of degree 8 in it. We will call them *trails* as our monomial was actually obtained by following one or more of those trails. The set of all trails is named $\mathcal{P}$:

$$\mathcal{P} = \{\{p_0^0, \cdots, p_7^0\}, \cdots, \{p_0^w, \cdots, p_7^w\}\},\ p_j^h < p_{j+1}^h\ \forall\ j \in [\![0, 6]\!],\ \forall\ h \in [\![0, w]\!],$$

$$p_j^h \in [\![0, 4]\!] \times [\![0, 63]\!]\ \forall\ j \in [\![0, 6]\!]\ \forall\ h \in [\![0, w]\!].$$

Let us consider $\sum_{\substack{m \in \mathbb{F}_2^{64} \\ wt(m)=32}} a_m x^m$ the degree-32 homogeneous component of coordinate $c_{(0,0)}$ after 5.5 rounds (as we can partially invert the linear layer on the first row, we can compute the cube sum (*i.e* the actual value of a coefficient) after 5.5 rounds from the cube sum after 6 rounds:

$$a_M = \left[\sum_{v \in C_M} p_L^{-1} \circ p^6(v, c)\right]_{(0,0)} = \left[p_L^{-1}\left(\sum_{v \in C_M} p^6(v, c)\right)\right]_{(0,0)},$$

where $[\cdot]_{(0,0)}$ denotes the "coordinate $(0, 0)$ of the state"). We are interested in finding the expression of a_M. To do so, let us introduce our set of solutions $\mathcal{S}$ (see below). As a single degree-32 term can come from different products of eight degree-4 terms, it is not sufficient to only keep track of the 8-tuples of degree-4 terms. That

is the reason why, in our solutions, we will keep the information of the degree-4 terms which are multiplied, as well as the trail they followed:

$$\mathcal{S} = \{(s_0, \cdots, s_7, p_0, \cdots, p_7) \mid (p_0, \cdots, p_7) \in \mathcal{P},\ s_i \in Q_{p_i}\ \forall\, i \in [\![0, 7]\!],\ \bigcup_{i=0}^{6} s_i = M\}$$

Terms of degree 32 can only be obtained by successive products of maximal-degree terms from round 2 to round 6. In an ideal case, if each intermediate highest-degree term is only obtained by a single product, we are guaranteed that $a_M = \prod_{i=0}^{7} d_{s_i,p_i}$ where d_{s_i,p_i} is the coefficient associated to monomial $x^{v_{s_i}}$ ($x^{v_{s_i}} = \prod_{e \in s_i} x_e$) in coordinate p_i after three rounds. By intuition, this kind of behaviors should not happened often. Indeed, we might expect that at least one of the intermediate highest-degree terms is obtained through two (or more) different multiplications between highest-degree terms from the previous round. This makes the search for the superpoly way more difficult. However, with our ideal case described above (which coincides with the case $|\mathcal{S}| = 1$), the superpoly can be recovered as the product of eight known multivariate polynomials corresponding to the eight coefficients of terms of degree 4 which were multiplied to build our targeted monomial.

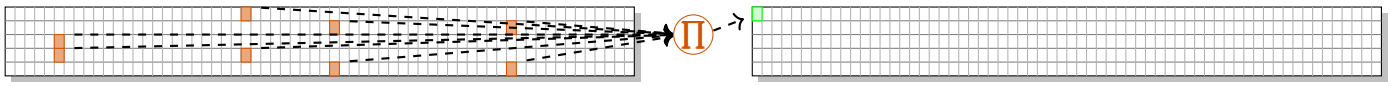


Figure 4.14: Part of the superpoly recovery: finding monomials coming from a single product

In order to study this *straight-forward monomial* behavior (no detour is made to go from the third round to the sixth, only a single trail is followed and a single product occurs), we want to compute $\mathcal{S}$, from the knowledge of $\mathcal{P}$ and $Q_{x,y}, \forall x, y$.

Algorithm 1 describes the way we look at our problem. Once a trail is chosen, the algorithm allows to list all the possible input degree-4 terms which lead to our targeted monomial by following the specified trail.

More precisely, this function takes as input the eight sets of degree-4 terms corresponding to the choice of our monomial and trail (eight sets $Q_{p_0^k}, \cdots, Q_{p_7^k}$ with the already introduced notation). It will return the (possibly empty) list of all degree-4 terms combinations (one term per coordinate) whose multiplication results in our targeted monomial. It works as follow: for any possible choice of a degree-4 term in the first coordinate, it first filters the remaining coordinates' sets (and thus excludes all terms which are not coprime with the selected one). Then it checks if the filtered sets are not empty. If none of them is empty (which means that are we are not in a dead-end (yet)), the selected degree-4 term is added to the current result and the function is called recursively with the current result and the list of the remaining filtered sets. Finally, the results of all recursive calls are stored and returned by the function.

As we can see this function actually terminates: if the list of sets is empty (which can only occur if eight choices were made, *i.e*, when a possible combination was found), the list of the eight degree-4 terms is returned. Otherwise, dead-end are skipped thanks to the **continue** command, and partial results are added one after the other. Note that the final returned value is a nested list containing lists in a tree fashion (parent nodes return the (possibly empty list) of all results found in children nodes).

Algorithm 1: A recursive function for searching for trails, **rec**

```
Input: a list of sets of possible 4-tuples, sets_possible_4tuples; a partial list of some chosen degree-4
       terms, result.
Output: An updated list of found trails.
if sets_possible_4tuples is empty then
    return result;
end
return_value ← [ ];
for q ∈ sets_possible_4tuples[0] do
    new_sets ← filter(q, sets_possible_4tuples[1 : ]);
    if one of the sets in new_sets is empty then
        continue;
    end
    result.insert(q);
    return_value.insert(rec(new_sets, result));
end
return return_value;
```

Using this recursive algorithm on each trail of the *hull* (the entire set of trails leading to a targeted coordinate), one can determine if the targeted monomial is either a straight-forward monomial or not.

The search algorithm is practical for 2-round trails of size 4, *i.e* a trail with four input coordinates which multiply together through two rounds of the permutation. It allows to list all the straight-forward degree-4 monomials after round 3. It takes only a few seconds on a personal laptop. From the knowledge of these straight-forward degree-4 monomials, it is also possible to obtain the number of products which lead to a random degree-16 monomial after five rounds in a comparable amount of time. I think it could be optimized for the search for degree-32 terms with some future refinements.

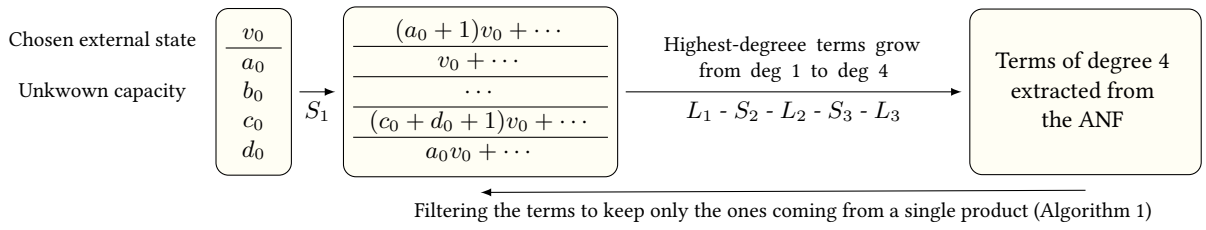


Figure 4.15: First stage: Recovering all the straight-forward degree-4 terms after L_3

Regarding the recovery of terms of degree 32 with the same method (using 3-round trails of size 8 instead of 2-round trails of size 4), it is now way too long to recover the superpoly of a random term. However, it might be possible if they were chosen properly (it is still unclear what "properly" should mean and which properties of such terms could speed up the process). There are at least two reasons it becomes impractical:

- The prefilter which eliminates terms whose support is not included in the support of our targeted monomial is way less efficient: instead of discarding each term which contains one of $64 - 16 = 48$ useless variables, it only discards each term which contains one of $64 - 32 = 32$ variables. This leads to bigger lists of monomials which could be multiplied together and this has an unfortunate avalanche effect...

- In addition, the number of trails to investigate is now way bigger: it grows from 363 to about 430 000.

Nevertheless, this kind of process could lead to cube-like attacks, following these steps:

1. Find a list of straight-forward monomials and recover their superpolys thanks to this method;
2. Compute the corresponding cube sum and start building a system of equations;
3. Finally recover most of the capacity by solving the system.

This motivated us to follow in this way.

With an optimized algorithm (in order to get rid of most of Sage's expensive structures and to privilege lists and sets of integers), I managed to study more precisely how degree-16 terms were built. I identified some terms of degree 16 after five rounds which were built only through a single multiplication: our first straight-forward monomials for round 5!

This property is very intriguing and many new questions arose at that moment and guided the end of my internship:

- How can we explain this behavior? Is it due to the trail? to the degree-4 terms? to both? at which extent?
- Is it possible assemble those kinds of monomials to build bigger ones?
- Can they be used in order to mount distinguishers and/or attacks?
- Is it reflecting a kind of "bad diffusion behavior"?

In order to investigate those questions, we first searched for precise definitions and results on our straight-forward monomials. Then we had to find a method in order to generate a lot of them. All of this is presented in the following sections.

VIII.1 An attempt at formalizing straight-forward monomials

In this part is presented an attempt at formalizing the notion of straight-forward monomials. The word *trail* is used in order to emphasize the link with the work by Hu *et al.* [HSWW20]. It is still necessary to determine exactly and precisely the relationship between the two notions.

Definition VIII.1 (Trail)**.** Let k, t be two integers. Let us consider $x_0, \cdots, x_{k-1} \in [\![0, 63]\!]$ and $y_0, \cdots, y_{k-1} \in [\![0, 4]\!]$ and the corresponding coordinates c_{y_i,x_i} for all $i \in [\![0, k-1]\!]$. Let $(Y, X) \in [\![0, 4]\!] \times [\![0, 63]\!]$ be associated to coordinate $c_{Y,X}$. We define a *trail over t rounds* of Ascon as: $(\{c_{y_0,x_0}, \cdots, c_{y_{k-1},x_{k-1}}\},\ c_{Y,X})$, such that the product of coordinates c_{y_i,x_i} for all $i \in [\![0, k-1]\!]$ is present in the ANF of coordinate $c_{Y,X}$ after t rounds. We will denote it: $\{c_{y_0,x_0}, \cdots, c_{y_{k-1}}\} \xrightarrow{t} c_{Y,X}$. Otherwise we will name it a *false trail* and denote it $\{c_{y_0,x_0}, \cdots, c_{y_{k-1}}\} \overset{t}{\not\to} c_{Y,X}$.

Definition VIII.2 (Straight-forward monomial)**.** Let $c_{Y,X}$ be a fixed coordinate after t rounds. We define a *straight-forward monomial* of $c_{Y,X}$ over t rounds as a monomial $\mathcal{M} = \prod_{i=0}^{k-1} v_i$ present in the ANF of $c_{Y,X}$ after t rounds which results from a **single trail** over t rounds and a **single product of variables**.

Definition VIII.3 (Predecessor)**.** Let $\mathcal{M}$ be a monomial present in the ANF of coordinate $c_{Y,X}$. We define one of its sets of *1-round predecessors* as the family composed of:

- two coordinates which are multiplied and present in the ANF of $c_{Y,X}$ over one round;
- two monomials whose product is equal to $\mathcal{M}$ and such that each of them is contained in one of the two given coordinates.

In other words, a set of 1-round predecessors is given as: $((c_{y_1,x_1}, \mathcal{M}_1), (c_{y_2,x_2}, \mathcal{M}_2))$ such that $\mathcal{M}_i$ is present in the ANF of c_{y_i,x_i}, with $i \in \{1,2\}$, $\mathcal{M}_1\mathcal{M}_2 = \mathcal{M}$, and $\{c_{y_1,x_1}, c_{y_2,x_2}\} \xrightarrow{1} c_{Y,X}$.

We can define in the same way sets of "older" predecessors: a set of t-round predecessors can be viewed as 2^t-tuple of monomial/coordinate pairs, whose coordinates constitute a trail over t rounds towards $c_{Y,X}$.

Example VIII.4. In the nonce-misuse scenario described, v_0 is input in coordinate $c_{0,0}$, and v_{25} is input in coordinate $c_{0,25}$. Through the first S-box layer and the first linear layer, v_{25} is diffused into coordinate $c_{0,1}$, while v_0 is still present in $c_{0,0}$[8].

Through the second S-box layer, coordinates $c_{0,0}$ and $c_{1,0}$ are multiplied together, their product appears in coordinate $c_{0,0}$. The second linear layer does not change anything, so we have $\{c_{0,0}, c_{1,0}\} \xrightarrow{1} c_{0,0}$. Thus, $((c_{0,0}, v_0), (c_{1,0}, v_{25}))$ is a set of 1-round predecessors of v_0v_{25} in $c_{0,0}$ at the end of round 2. $((c_{4,0}, v_0), (c_{1,0}, v_{25}))$ is another one, using the trail $\{c_{4,0}, c_{1,0}\} \xrightarrow{1} c_{0,0}$.

Proposition VIII.5. *Let $\mathcal{M}$ be a straight-forward monomial of $c_{Y,X}$ following $\{c_{y_0,x_0}, \cdots, c_{y_{k-1}}\} \xrightarrow{t} c_{Y,X}$. The coefficient $a^t_{\mathcal{M},Y,X}$ (which is a polynomial in secret variables) of $\mathcal{M}$ in $c_{Y,X}$ can be directly deduced from the knowledge of the coefficients associated to each variable v_i in coordinate c_{y_i,x_i}. More precisely: $a^t_{\mathcal{M},Y,X} = \prod_{i=0}^{k-1} a^0_{v_i,y_i,x_i}$, where $a^0_{v_i,y_i,x_i}$ is the coefficient in front of v_i in c_{y_i,x_i} at initialization.*

Proof. By hypothesis, our monomial $\mathcal{M}$ results from a single trail over t rounds. It does mean that the only product leading to $\mathcal{M}$ in $c_{Y,X}$ is this particular choice of variables/coordinates pairs. The coefficient $a^t_{\mathcal{M},Y,X}$ is thus the product of the coefficients associated to those pairs. □

Proposition VIII.6. *Any straight-forward monomial only has straight-forward predecessors. The converse is not always true.*

Proof. If one of its predecessors is not straight-forward, then there are at least two choices of a trail and variables which lead to this predecessor. Ultimately, there are at least two choices of a trail and variables leading to our monomial.
On the other hand, a monomial can be obtained by two different sets of straight-forward predecessors and thus it would not be straight-forward itself. □

Example VIII.7. v_0v_{25} in $c_{0,0}$ after two rounds has two sets of straight-forward predecessors; as it has two sets of predecessors, is cannot be a straight-forward monomial.

Proposition VIII.8. *(Sufficient conditions on predecessors to be straight-forward) Let us consider a set of t_2-round predecessors of $\mathcal{M}$ following $\{c_{y_0,x_0}, \cdots, c_{y_{k-1},x_{k-1}}\} \xrightarrow{t_2} c_{Y,X} : ((c_{y_0,x_0}, \mathcal{M}_0), \cdots, (c_{y_{k-1},x_{k-1}}, \mathcal{M}_{k-1})$. Let us suppose that each monomial is present in the ANF of the respective coordinate after t_1 rounds. Moreover, let us suppose that:*

[8] As the first round is acting differently from the next ones (no growth of degree) and as it is very easy to keep track of the changes during this round, we often use the state after round 1 as our "initial" state.

- $\mathcal{M}_0, \cdots, \mathcal{M}_{k-1}$ *are pairwise coprime;*
- *this choice of a set of* t_2*-round predecessors and of a trail over* t_2 *rounds is the only choice leading to* $\mathcal{M} = \prod_{i=0}^{k-1} \mathcal{M}_i$ *in* $c_{Y,X}$.

Then $\mathcal{M}$ *is a straight-forward monomial of* $c_{Y,X}$ *after* $t_1 + t_2$ *rounds.*

Proof. It is exactly the definition of a straight-forward monomial. □

So the search for straight-forward monomials could be done round after round if we are able to discuss whether the second condition is respected or not. In order to do so, the algorithm presented above is for the moment our only option. More precisely, let us suppose that we found a family of $\mathcal{M}_i$ and a trail as in the previous proposition (the second bullet being not verified for the moment). In order to verify that this is the only choice leading to $\mathcal{M}$ in coordinate $c_{Y,X}$, we can use the knowledge of **all** the monomials present in all coordinates after t_1 rounds and all the trails leading to $c_{Y,X}$ over t_2 rounds to check whether no other product leads to $\mathcal{M}$. This is detailed in the following section.

VIII.2 An attempt at finding many more straight-forward monomials

In order to study and better understand straight-forward monomials, we need to be able to generate a lot of straight-forward monomials. To do so, we thought of going back and forth: compute products of straight-forward monomials one round before (see the necessary condition in Proposition VIII.6) in order to "climb up" one more round and then check if it is an actual straight-forward monomial by "going down" through all possible trails (see Figure 4.16).

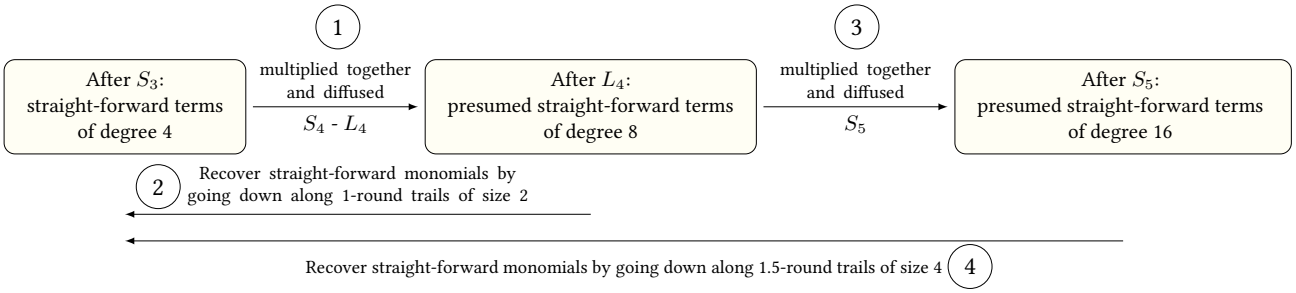


Figure 4.16: Second stage: going back and forth through the next rounds

There are multiple issues. Let us suppose that we are aware of all possible monomials appearing in all coordinates after t rounds. By going "down", we can filter every straight-forward monomial after t rounds. If we now want to go a step further we have at least two options, both with drawbacks:

- we can use the quadratic part of the S-box (and the linear layer) in order to push further our straight-forward monomials (let us keep in mind that we are for the moment only interested in monomials of highest degree) without taking into account the coefficients of the monomials (we are indeed trying to guess them by only knowing the coefficients after round 1). The problem with this method is that we cannot distinguish when a XOR deletes a monomial from the coordinate or transforms it to a non-straight-forward monomial. It results in a list of monomials whose predecessors are surely straight-forward (necessary condition respected) but which can be non-straight-forward and even absent.

- we can also use the ANF of one round and filter each monomial which is involved in a XOR operation with itself. The resulting list would not take into account absent monomials and it would also delete some unwanted non-straight-forward monomials. However this technique also deletes some straight-forward monomials: for example when a monomial is XORed three times with itself, twice with the same coefficient and once with a different one, the resulting monomial could be considered as straight-forward[9] but is deleted by this filtering.

Another issue appears on the way down. The main reason we are focusing of this special kind of terms (highest-degree, straight-forward) is because we cannot store the complete ANF of the permutation. However, according to what was detailed above, we need to keep the lists of **all** monomials one round before in order to guarantee that a monomial is straight-forward round after round. This seems problematic, especially after five rounds. We can however use a "one-step-forward-two-step-backward" technique in which we verify if a monomial of round t is straight-forward using the knowledge of the monomials of round $t-2$ and 2-round trails, as shown on Figure 4.16 with 1.5-round trails. We managed this way to generate non-exhaustive lists of straight-forward degree-16 terms after the fifth S-box layer. All of these is still being investigated at the moment of writing. While having the feeling of grasping something here, a lot of work is still needed to deeply understand what we are looking at.

Finally, a last comment on the degree of the coefficients in front of the highest-degree terms. By looking at the coefficients after the second S-box layer in this nonce-misuse scenario, we can see that only a single row contains terms of degree 4 whose coefficients are also of degree 4. This prevents the highest possible growth of degree for coefficients (the maximal degree of coefficients in front of highest-degree terms is thus 7 after the next round). With an avalanche effect similar to the one studied in Section VI, the maximal degree of the coefficients is thus 28 after round 6. So, after round 6, there is always a probability higher than 2^{-28} for the coefficient of a straight-forward monomial to be equal to 1. Indeed, they are composed as the product of at most 28 affine combinations of one bit of information about the key. In average, it will even be higher than that: only about 10% of the coefficients are of highest degree possible after round 3. This seems to indicate that a straight-forward monomial of degree 32 will never give more information than 28 bits (for a cost of 2^{32} plaintexts). On the other hand, we will have more chance to randomly find a coefficient equal to 1 (and which actually gives 28 bits of information, contrary to a coefficient equal to 0) than if it was actually a product of 32 affine combinations. In other words, by forcing the coefficients of our chosen straight-forward monomials to have low degrees, we will manage to get less information, but more often. This part also needs to be looked at more precisely.

[9] According to the definition given above, it is not a straight-forward monomial. However, its coefficient can still be recovered directly and easily as the product of several previous coefficients. This could be another interesting way of defining straight-forward monomials.

5 Conclusion and perspectives

At the end of this first experience in a research environment within Inria COSMIQ team, it is now time to look at all what was done during those six months with the benefit of hindsight.

As in Ascon, in the research world everything is not linear. Whether through the reading of an article or while trying to get a new grasp at an already well-studied encryption method, it is very striking to see how thoughts come and go: one second all seems to be brighter and clearer, one second later, doubt is coming back. It is one thing to understand that it takes some time to make progress; it is another thing to experience it.

On the other hand, it is also noteworthy to look at the bigger picture of these six months. I tried in this document to build links between the very first ideas we had and where we are at the time of writing. Even unmeaning pists can later give some more meaningful insights. This was clearly not obvious for me when I started.

Research temporality is disconcerting, and acclimatizing completely to this work seems to take some time as well. It was also one of the goals of this internship and I have to admit that I came to really appreciate it.

Regarding our accomplished work, even though I cannot present a finished product, I am confident that it will lead to some other oddities at some point. It however raised a lot of interesting questions.

In particular, it seems possible to link most of the different studies presented here in some new fashions. The work on the straight-forward monomials in the nonce-misuse scenario is not finished yet and needs to be improved. It seems possible to study in the same way straight-forward monomials in a nonce-respecting scenario and thus coming back to Scenarios 1, 2 and 3 studied before. Is it possible and/or interesting to study them through other representations of Ascon? Could the cyclic behavior be used to be more effective while looking at straight-forward monomials? Could the notion of diffusion, which underlies our study of the shuffling and mixing of the highest-degree terms, show any weakness in Ascon? Could it guide the design of future encryption functions? Could we look at other NIST lightweight candidates in the way we are analyzing Ascon? How does this interact with existing works, especially on monomial trails?

In front of so many questions, it is very easy to jump from one subject to another. Yet, I am very lucky to be able to continue on this trail with guidance. Indeed, as of September 1st, I am starting a doctoral thesis co-supervised by Anne Canteaut & Léo Perrin. I will be working on the "security analysis of [some] lightweight symmetric primitives"; the remarks and questions presented above being the starting point of our studies. I am really pleased to be able to continue this exciting experience while being as well supervised as I was during this internship.

Bibliography

[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.

[BB02] Elad Barkan and Eli Biham. In how many ways can you write Rijndael? In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 160–175, Queenstown, New Zealand, December 1–5, 2002. 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.

[BC13] Christina Boura and Anne Canteaut. On the influence of the algebraic degree of f^{-1} on the algebraic degree of $G \circ F$. *IEEE Trans. Inf. Theory*, 59(1):691–702, 2013.

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 654–682, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BDLF10] Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque. Another look at complementation properties. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 347–364, Seoul, Korea, February 7–10, 2010. 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.

[BP17] Alex Biryukov and Leo Perrin. State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive, Report 2017/511, 2017. https://eprint.iacr.org/2017/511.

[BS01] Alex Biryukov and Adi Shamir. Structural cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 394–405, Innsbruck, Austria, May 6–10, 2001. 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.

[DBRP99] Carl D'Halluin, Gert Bijnens, Vincent Rijmen, and Bart Preneel. Attack on six rounds of Crypton. In Lars R. Knudsen, editor, *FSE'99*, volume 1636 of *LNCS*, pages 46–59, Rome, Italy, March 24–26, 1999. Springer, Heidelberg, Germany.

[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.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE'97*, volume 1267 of *LNCS*, pages 149–165, Haifa, Israel, January 20–22, 1997. Springer, Heidelberg, Germany.

[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.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[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.

[FKM08] Simon Fischer, Shahram Khazaei, and Willi Meier. Chosen IV statistical analysis for key recovery attacks on stream ciphers. In Serge Vaudenay, editor, *AFRICACRYPT 08*, volume 5023 of *LNCS*, pages 236–245, Casablanca, Morocco, June 11–14, 2008. Springer, Heidelberg, Germany.

[HLM+20] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 466–495, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[HSWW20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 446–476, Daejeon, South Korea, December 7–11, 2020. 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.

[KW02] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 112–127, Leuven, Belgium, February 4–6, 2002. Springer, Heidelberg, Germany.

[Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard E. Blahut, Daniel J. Costello, Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233. Springer US, Boston, MA, 1994.

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

[Luc02] Stefan Lucks. The saturation attack - a bait for Twofish. In Mitsuru Matsui, editor, *FSE 2001*, volume 2355 of *LNCS*, pages 1–15, Yokohama, Japan, April 2–4, 2002. Springer, Heidelberg, Germany.

[MSST21] Kalikinkar Mandal, Dhiman Saha, Sumanta Sarkar, and Yosuke Todo. Sycon: A new milestone in designing ascon-like permutations. Cryptology ePrint Archive, Report 2021/157, 2021.

[NIS15] Secure Hash Standard (SHS). Technical Report Federal Information Processing Standard (FIPS) 180-4, National Institute of Standards and Technology U.S. Department of Commerce, August 2015.

[NIS17] NIST Lightweight Cryptography competition, January 2017. `https://csrc.nist.gov/Projects/lightweight-cryptography`.

[RHSS21] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. *IACR Transactions on Symmetric Cryptology*, 2021(1):130–155, March 2021.

[Rio19] Sébastien Riou. DryGASCON. Technical report, National Institute of Standards and Technology, 2019. `https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates`.

[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.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[SMS19] Sumanta Sarkar, Kalikinkar Mandal, and Dhiman Saha. Sycon v1.0. Technical report, National Institute of Standards and Technology, 2019. `https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates`.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 648–678, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

List of Figures

List of Tables