A Closer Look at the S-box: Deeper Analysis of Round-Reduced ASCON-HASH

Xiaorui Yu[1], Fukang Liu[2], Gaoli Wang[1](✉), Siwei Sun[3], Willi Meier[4]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
51215902051@stu.ecnu.edu.cn,glwang@sei.ecnu.edu.cn
[2] Tokyo Institute of Technology, Tokyo, Japan
liufukangs@gmail.com
[3] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
siweisun.isaac@gmail.com
[4] FHNW, Windisch, Switzerland
willimeier48@gmail.com

Abstract. ASCON, a lightweight permutation-based primitive, has been selected as NIST's lightweight cryptography standard. ASCON-HASH is one of the hash functions provided by the cipher suite ASCON. At ToSC 2021, the collision attack on 2-round ASCON-HASH with time complexity 2^{103} was proposed. Due to its small rate, it is always required to utilize at least 2 message blocks to mount a collision attack because each message block is only of size 64 bits. This significantly increases the difficulty of the analysis because one almost needs to analyze equivalently at least 2ℓ rounds of ASCON in order to break ℓ rounds. In this paper, we make some critical observations on the round function of ASCON, especially a 2-round property. It is found that such properties can be exploited to reduce the time complexity of the 2-round collision attack to $2^{62.6}$. Although the number of attacked rounds is not improved, we believe our techniques shed more insight into the properties of the ASCON permutation and we expect they can be useful for the future research. Following the same analysis method and with SMT technique, we practically find some semi-free-start collision attacks for 4-round ASCON-HASH and ASCON-Xof with STP solver.

Keywords: ASCON · ASCON-HASH · Collision Attack · Algebraic Technique

1 Introduction

In 2013, NIST started the lightweight cryptography project. Later in 2016, NIST provided an overview of the project and decided to seek for some new algorithms as a lightweight cryptography standard. In 2019, NIST received 57 submissions and 56 of them became the first round candidates after the initial review. After the project proceeded into Round 2 [4], NIST selected 32 submissions as Round 2 candidates, including ASCON. After that, ASCON was selected to be one of the ten finalists of the lightweight cryptography standardization process. On February 7, 2023, NIST announced the selection of the ASCON family for the lightweight cryptography standardization.

ASCON [11] is a lightweight permutation-based primitive. It aims to provide efficient encryption and authentication functions while maintaining sufficiently high security.

Advantages. The main advantages of ASCON can be summarized as below:

- **Lightweight**: The design of ASCON is simple and suitable for hardware and software implementation. It is particularly suitable for resource constrained environments, such as IoT devices, embedded systems, and low-power devices.
- **High security**: ASCON provides high security and resists many different types of known attacks.
- **Adjustable**: ASCON supports different security levels and performance requirements. For example, ASCON-128 and ASCON-128a provide a 128-bit security level, suitable for high security requirements; ASCON-80pq provides an 80-bit security level, suitable for low-power and low-cost scenarios.
- **Authentication encryption**: ASCON can achieve both data encryption and integrity protection. It supports associated data and allows for verification of additional information during the encryption process, such as the identities of message senders and receivers.

History. ASCON was first published as a candidate in Round 1 [7] of the CEASER competition [1]. This original design (version v1) specified the permutation as well as the mode for authenticated encryption with two recommended family members: The primary recommendation Ascon-128 as well as a variant Ascon-96 with 96-bit key. For the subsequent version V1.1 [8] and V1.2 [9], minor functional tweaks were applied, including a reordering of the round constants and the modification of the secondary recommendation to the current Ascon-128a. Then, V1.2 [9] and the status update file [6] were submitted to the NIST Lightweight Cryptography project. The submission to NIST includes not only the authenticated cipher family, but also introduces modes of operation for hashing: ASCON-HASH and ASCON-XOF, as well as a third parameterization for authenticated encryption: Ascon-80pq. For ASCON-HASH and ASCON-XOF, they support 256-bit and arbitrary-length hash values, respectively.

On the collision resistance of ASCON-HASH. Due to the used sponge structure, the generic time complexity to find a collision of ASCON-HASH is 2^{128} and the memory complexity is negligible with Floyd's cycle finding algorithm [12]. Due to its small rate, it is quite challenging to find collisions for a large number of rounds. In [22], the first 2-round collision attack on ASCON-HASH was presented with time complexity 2^{125}. However, it is shown that such an attack is invalid because the used 2-round differential characteristic is invalid according to [14]. Later, at ToSC 2021 [13], a new and valid 2-round differential characteristic with an optimal differential probability was found. Based on the same attack strategy as in [22], they gave a 2-round collision with time complexity of 2^{103} in [13]. Very recently, Qin et al. presented collision attacks on 3 and 4 rounds of ASCON-HASH by turning preimages for ASCON-XOF into collisions for ASCON-HASH [20]. However, it can be found that both the time complexity and memory complexity of the 3/4-round collision attacks are very high, i.e. larger than 2^{120}. From a practical view, it seems that these attacks may be slower than the generic attack. In any case, all the collision attacks are far from being practical, even for 2 rounds.

Table 1: Summary of collision attacks on ASCON-HASH

Attack Type	Rounds	Time complexity	Memory Complexity	Reference
collision attack	2	2^{125*}	negligible	[22]
	2	2^{103}	negligible	[13]
	2	$2^{62.6}$	negligible	Sect. 4
	3	$2^{121.85}$	2^{121}	[20]
	4	$2^{126.77}$	2^{126}	[20]

* The characteristic used is invalid.

Our contributions. We aim to significantly improve the time complexity of the 2-round collision attack in [13] such that it can be much closer to a practical attack. Our contributions are summarized below:

1. We found that the 2-round collision attack in [13] is quite straightforward, i.e., the authors found a better characteristic but did not optimize the attack strategy. Hence, we are motivated to take a closer look at the used 2-round differential characteristic and aim to improve the attack by using some algebraic properties of the S-box as in the recent algebraic attack on LowMC [15,18], i.e., we are interested in the relations between the difference transitions and value transitions.
2. Based on our findings of the properties of the S-box, we propose to use a better attack framework and advanced algebraic techniques to improve the 2-round collision attack. As a result, the time complexity is reduced from 2^{103} to $2^{62.6}$, as shown in Table 1.
3. Although in [10], the authors gave a semi-free-start collision attack on 4-round ASCON-HASH and ASCON-XOF, they didn't publish the detailed process. So we take use of the differential characteristic of 4-round ASCON-HASH and ASCON-XOF given in [10] and successfully find more semi-free-start collision attacks on 4-round ASCON-HASH and ASCON-XOF, different from the one given in [10]. We mainly use the SMT technique to solve the problem. We give our detailed procedure of finding the results using STP solver [5] in this paper.

[5] https://github.com/stp/stp

Organization of this paper. In section 2, we define some notations that will be used throughout the paper and briefly describe ASCON-HASH. In section 3, we describe the collision attack framework that will be used in the new attacks. In section 4, we show how to optimize the existing 2-round collision attack with advanced algebraic techniques. In section 5, by exploiting similar algebraic properties of the S-box, we describe how to use an SMT/SAT-based method to search for semi-free-start collisions for 4-round ASCON. Finally, the paper is concluded in section 6.

2 Preliminaries

2.1 Notations

The notations used in this paper are summarized in Table 2.

Table 2: Notations

r	the length of the rate part for ASCON-HASH, $r = 64$
c	the length of the capacity part for ASCON-HASH, $c = 256$
S_j^i	the input state of round i when absorbing the message block M_j
$S^i[j]$	the j-th word (64-bit) of S_i
$S^i[j][k]$	the k-th bit of $S^i[j]$, $k = 0$ means the least significant bit and k is within modulo 64
x_i	the i-th bit of a 5-bit value x, x_0 represents the most significant bit
M	message
M_i	the i-th block of the padded message
$\ggg$	right rotation (circular right shift)
$a\%b$	$a \mod b$
0^n	a string of n zeroes

2.2 Description of ASCON-HASH

The ASCON family offers 2 important hash functions: ASCON-HASH and ASCON-XOF. ASCON-HASH is a sponge-based hash function [2]. In its core, it is a 12-round permutation P^a over a state of 320 bits. The hashing mode is shown in Figure 1.

For ASCON-HASH, the state denoted by X is divided into five 64-bit words, i.e., $X = X_0||X_1||X_2||X_3||X_4$. The first 64-bit word X_0 will be loaded in the rate part while the remaining 4 words (X_1, X_2, X_3, X_4) are loaded in the capacity part. The round function $f = f_L \circ f_S \circ f_C$ is composed of 3 operations: f_C is the constant addition, f_S is the substitution layer, and f_L is the linear diffusion layer. For simplicity, the ℓ-round ASCON permutation is simply denoted by f^ℓ.

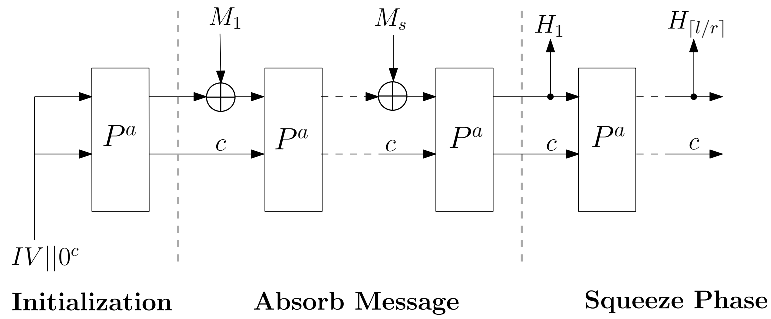

Fig. 1: The mode of ASCON-HASH

On the internal states. When absorbing the message block M_j, denote the 320-bit input state at round i $(0 \le i \le 11)$ by S_j^i and the state transitions are described below.

$$S_j^i \xrightarrow{f_C} S_j^{i,a} \xrightarrow{f_S} S_j^{i,s} \xrightarrow{f_L} S_j^{i+1}.$$

Note that if we only consider one message block, we simply omit j as below:

$$S^i \xrightarrow{f_C} S^{i,a} \xrightarrow{f_S} S^{i,s} \xrightarrow{f_L} S^{i+1}.$$

The corresponding graphic explanations can be referred to Fig. 2 and Fig. 3, respectively.

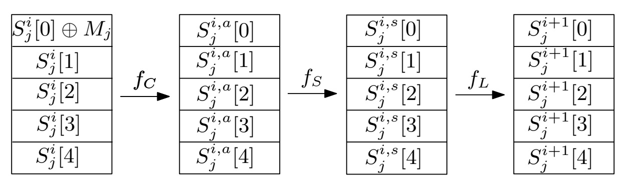

Fig. 2: The 1-round state transition when absorbing M_j

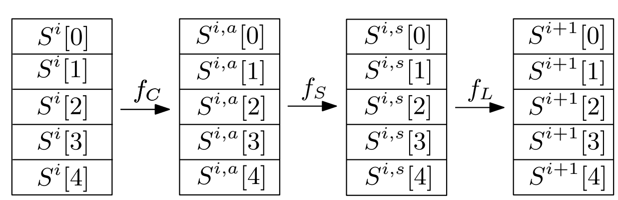

Fig. 3: The 1-round state transition

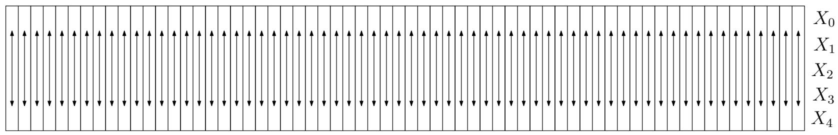

Fig. 4: The substitution layer

Fig. 5: The linear diffusion layer

Constant addition f_C. For this operation, an 8-bit round constant c_i is added to the word X_2, i.e., $X_2 \leftarrow X_2 \oplus c_i$. The round constants $(c_i)_{0 \leq i \leq 11}$ for 12-round ASCON-HASH are shown in Table 3.

Table 3: The round constants c_i

i	0	1	2	3	4	5	6	7	8	9	10	11
c_i	`0xf0`	`0xe1`	`0xd2`	`0xc3`	`0xb5`	`0xa5`	`0x96`	`0x87`	`0x78`	`0x69`	`0x5a`	`0x4b`

Substitution layer f_S. At this operation, the state will be updated by 64 parallel applications of a 5-bit S-box. The S-box $(y_0, \ldots, y_4) = \texttt{SB}(x_0, \ldots, x_4)$ is defined as follows:

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

As shown in Fig. 4, the input $(x_0, \ldots, x_4)$ and output $(y_0, \ldots, y_4)$ correspond to one column of the state.

Linear diffusion layer f_L. This operation is used to diffuse each 64-bit word X_i, as shown in Fig. 5. Specifically, X_i is updated by the function $\sum_i$ where $0 \leq i \leq 4$, as specified below:

$$\begin{cases} X_0 \leftarrow \Sigma_0(X_0) = X_0 \oplus (X_0 \ggg 19) \oplus (X_0 \ggg 28), \\ X_1 \leftarrow \Sigma_1(X_1) = X_1 \oplus (X_1 \ggg 61) \oplus (X_1 \ggg 39), \\ X_2 \leftarrow \Sigma_2(X_2) = X_2 \oplus (X_2 \ggg 1) \oplus (X_2 \ggg 6), \\ X_3 \leftarrow \Sigma_3(X_3) = X_3 \oplus (X_3 \ggg 10) \oplus (X_3 \ggg 17), \\ X_4 \leftarrow \Sigma_4(X_4) = X_4 \oplus (X_4 \ggg 7) \oplus (X_4 \ggg 41). \end{cases}$$

On the initial value and state. The hash function initializes the 320-bit state using a constant $IV = \texttt{0x00400c0000000000}$. Then, the 12-round ASCON permutation is applied and we obtain an initial state $S_1^0 = f^{12}(IV||0^{256})$, as specified below:

$$S_1^0 \leftarrow \begin{array}{l} \texttt{0xee9398aadb67f03d} \\ \texttt{0x8bb21831c60f1002} \\ \texttt{0xb48a92db98d5da62} \\ \texttt{0x43189921b8f8e3e8} \\ \texttt{0x348fa5c9d525e140} \end{array}$$

The padding rule of ASCON-HASH is as follows: it appends a single 1 and the smallest number of zeroes to M such that the size of padded message in bits is a multiple of $r = 64$. The complete description of the hashing function is given in Algorithm 1 in Appendix B.

3 The Attack Frameworks

For differential-based collision attacks on a sponge-based hash function, one essential step is to find a collision-generating differential characteristic. The second step is to find conforming message pairs satisfying this differential characteristic.

With the development of automatic tools, there are many possible methods to search for a desired differential characteristic. However, when it comes to the second step, i.e., satisfying the conditions of the differential characteristic, it always involves dedicated efforts and sometimes requires nontrivial techniques. For example, the linearization techniques for the KECCAK round function have been widely used to speed up the differential-based collision attack on KECCAK, e.g., the 1/2/3-round connectors [5, 19, 21]. As can be seen from the current record of the Keccak crunchy crypto collision contest[6], it is quite challenging to analyze sponge-based hash functions with a small rate, which is exactly the case of ASCON. It is thus not surprising to see that the best differential-based collision attack on ASCON could only reach up to 2 rounds.

For a sponge-based hash function with a small rate, one main obstacle exists in the available degrees of freedom in each message block. For ASCON, each message block only provides at most 64 free bits. However, for a differential characteristic used for collision attacks, there may exist more than 128 bit conditions, which directly makes it mandatory to utilize at least 3 message blocks.

Let us consider a general case and suppose that we have an ℓ-round collision-generating differential characteristic. Furthermore, suppose we will use k message blocks $(M_1, \ldots, M_k)$ to fulfill the conditions, i.e., we aim to find $(M_1, \ldots, M_k)$ and $(M_1, \ldots, M_{k-1}, M'_k)$ such that

$$S_{j+1}^0 = f^\ell \left(S_j^0 \oplus (M_j||0^{256}) \right) \text{ where } 1 \leq j \leq k-1,$$

[6] `https://keccak.team/crunchy_contest.html`

$$\star||0^{256} = f^{\ell}\left(S_k^0 \oplus (M_k||0^{256})\right) \oplus f^{\ell}\left(S_k^0 \oplus \mathtt{SB}(M_k'||0^{256})\right),$$

where $M_k \neq M_k'$ and $\star$ is an arbitrary r-bit value.

From the differential characteristic, suppose that there are n_c bit conditions on the capacity part of S_k^0 and the remaining conditions hold with probability 2^{-n_k}. Then, a straightforward method to find conforming message pairs is as follows:

Step 1: Find a solution of $(M_1, \ldots, M_{k-1})$ such that the n_c bit conditions on the capacity part of S_k^0 can hold.

Step 2: Exhaust M_k and check whether remaining n_k bit conditions can hold. If there is a solution, a collision is found. Otherwise, return to Step 1.

For convenience, we call the above procedure *the general 2-step attack framework*. Note that this has been widely used and it is really not a new idea.

For a sponge with rate r, we need to perform Step 2 for about $2^{n_k - r}$ times and hence we need to perform Step 1 for $2^{n_k - r}$ times. Suppose the time complexity to find a solution of $(M_1, \ldots, M_{k-1})$ and M_k is $T_{\mathtt{pre}}$ and $T_{\mathtt{k}}$, respectively. In this way, the total time complexity $T_{\mathtt{total}}$ is estimated as

$$T_{\mathtt{total}} = (k-1) \cdot 2^{n_k - r} \cdot T_{\mathtt{pre}} + 2^{n_k - r} \cdot T_{\mathtt{k}}. \tag{2}$$

If $T_{\mathtt{k}}$ and $T_{\mathtt{pre}}$ are simply treated as 2^r and 2^{n_c}, respectively, i.e., only the naive exhaustive search is performed, then

$$T_{\mathtt{total}} = (k-1) \cdot 2^{n_k + n_c - r} + 2^{n_k}.$$

In other words, the total time complexity is directly related to the probability of the differential characteristic, i.e., $2^{-n_c - n_k}$.

In many cases, the attackers can optimize $T_{\mathtt{k}}$ by using some advanced techniques to satisfy partial conditions implied in the differential characteristic, i.e., $T_{\mathtt{k}}$ can be smaller than 2^r. For example, the target difference algorithm proposed in [5] is one of such techniques. However, to optimize $T_{\mathtt{pre}}$, one has to solve a problem similar to the $\ell-$round preimage finding problem. In most cases, this is not optimized due to the increasing difficulty and it is simply treated as $T_{\mathtt{pre}} = 2^{n_c}$.

3.1 The Literature and Our New Strategy

It is found that neither $T_{\mathtt{k}}$ nor $T_{\mathtt{pre}}$ has been optimized for the existing 2-round collision attacks on ASCON-HASH [13, 22] and they exactly follow the above attack framework. In the collision attack on 6-round GIMLI-HASH [14], the attackers optimized both $T_{\mathtt{k}}$ and $T_{\mathtt{pre}}$ where $k = 2$.

As can be noted in our new attacks on ASCON-HASH, optimizing $T_{\mathtt{k}}$ is indeed quite straightforward after a little deeper analysis of the round function and its 5-bit S-box. However, optimizing $T_{\mathtt{pre}}$ looks infeasible at the first glance. Indeed even if $T_{\texttt{k}}$ is optimized to 1, the improved factor is still quite small. Therefore, to achieve significant improvements, it is necessary to optimize $T_{\texttt{pre}}$.

Our idea to achieve this purpose is to further convert the n_c conditions on the capacity part of S_k^0 into some n_c^1 conditions on the capacity part of S_{k-1}^0, as Fig. 6 shows. In this way, our attack is stated as follows:

Step 1: Find a solution of $(M_1, \ldots, M_{k-2})$ such that the n_c^1 bit conditions on the capacity part of S_{k-1}^0 can hold.

Step 2: Enumerate all the solutions of M_{k-1} such that the conditions on the capacity part of S_k^0 can hold.

Step 3: Exhaust M_k and check whether remaining n_k bit conditions can hold. If there is a solution, a collision is found. Otherwise, return to Step 1.

To distinguish this from *the general 2-step attack framework*, we call the above procedure *the general 3-step attack framework*.

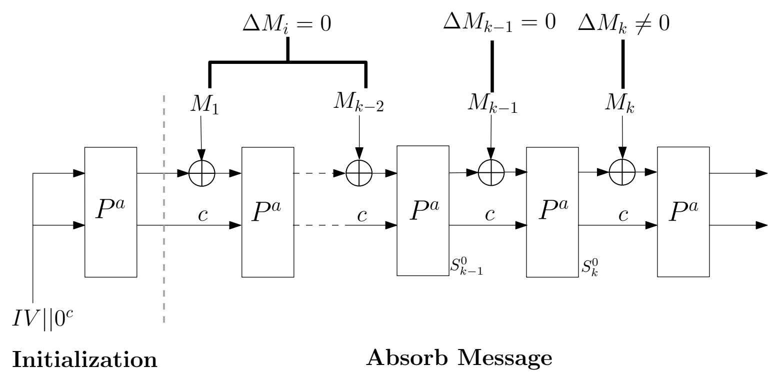

Fig. 6: The general 3-step attack framework

Analysis of the time complexity. For convenience, the time complexity of Step 1, 2 and 3 is denoted by $T_{\texttt{pre1}}$, $T_{\texttt{k-1}}$ and $T_{\texttt{k}}$, respectively. In this way, the total time complexity becomes

$$T_{\texttt{total}} = (k-2) \cdot 2^{n_k+n_c-2r} \cdot T_{\texttt{pre1}} + 2^{n_k+n_c-2r} \cdot T_{\texttt{k-1}} + 2^{n_k-r} \cdot T_{\texttt{k}}. \qquad (3)$$

Specifically, we need on average 2^{n_k-r} different valid solutions of $(M_1, \ldots, M_{k-1})$. In this sense, we need about $2^{n_k+n_c-2r}$ different valid solutions of $(M_1, \ldots, M_{k-2})$ because for each valid $(M_1, \ldots, M_{k-2})$, we expect to have 2^{r-n_c} valid solutions of M_{k-1}.

Based on Equation 3, if $n_c < r$ holds, we have

$$2^{n_k+n_c-2r} < 2^{n_k-r}.$$

Compared with Equation 2, this case has indicated the possibility to optimize the attack if $T_{\texttt{k-1}}$ can be significantly optimized and $T_{\texttt{pre1}}$ is relatively small, i.e., we know $T_{\texttt{pre1}} \leq 2^{n_c'}$.

On the purpose to convert conditions. As stated above, we have to optimize $T_{\texttt{pre1}}$. This is related to the original purpose to introduce conditions on the capacity part of S^0_{k-1}. Specifically, we expect that after adding these conditions, we can efficiently enumerate the solutions of M_{k-1} to satisfy the n_c conditions on the capacity part of S^0_k. In other words, without these conditions, we still can only perform the naive exhaustive search over M_{k-1} and no improvement can be obtained, i.e., the time complexity is

$$\begin{aligned} &(k-2)\cdot 2^{n_k+n_c-2r} + 2^{n_k+n_c-2r}\cdot 2^r + 2^{n_k-r}\cdot T_{\texttt{k}} \\ =&\,(k-2)\cdot 2^{n_k+n_c-2r} + 2^{n_k+n_c-r} + 2^{n_k-r}\cdot T_{\texttt{k}}. \end{aligned}$$

The big picture of our new attacks. In our attacks, we do not make more efforts to convert the n_c' conditions on S^0_{k-1} into conditions on the previous input states due to the increasing difficulty. Hence, in our setting, we will make

$$T_{\texttt{pre1}} = 2^{n_c'}.$$

In this way, the total time complexity is estimated as

$$T_{\texttt{total}} = (k-2)\cdot 2^{n_k+n_c+n_c'-2r} + 2^{n_k+n_c-2r}\cdot T_{\texttt{k-1}} + 2^{n_k-r}\cdot T_{\texttt{k}}. \tag{4}$$

In the following, we will describe how to significantly optimize $T_{\texttt{k-1}}$ and $T_{\texttt{k}}$ based on an existing 2-round differential characteristic of ASCON.

4 Collision Attacks on 2-Round ASCON-HASH

The collision attack in this paper is based on the 2-round differential characteristic proposed in [13], as shown in Table 4. Note that the first collision attack on 2-round ASCON-HASH was proposed in [22] but the differential characteristic is shown to be invalid in [14]. We have verified with the technique in [14] that the 2-round differential characteristic in [13] is correct.

Table 4: The 2-round differential characteristic in [13]

ΔS^0 (2^{-54})	ΔS^1 (2^{-102})	ΔS^2
0xbb450325d90b1581	0x2201080000011080	0xbaf571d85e1153d7
0x0	0x2adf0c201225338a	0x0
0x0	0x0	0x0
0x0	0x0000000100408000	0x0
0x0	0x2adf0c211265b38a	0x0

According to [13], there are 27 and 28 active S-boxes in the first and second round, respectively. Specifically, there are 54 bit conditions on the capacity part of the input S^0 and 102 bit conditions on the input state S^1 of the second round. With our notations, there are

$$n_c = 54, \quad n_k = 102.$$

With this differential characteristic, they used the technique in [22] to mount the collision attack with $k = 3$ message blocks and its time complexity is 2^{102}. It follows the general 2-step attack framework described above without optimization on $T_{\texttt{pre}}$ and $T_{\texttt{k}}$, i.e.,

$$T_{\texttt{pre}} = 2^{54}, \quad T_{\texttt{k}} = 2^{64}.$$

In this way, the total time complexity can be computed based on Equation 2, i.e.,

$$T_{\texttt{total}} = 2 \times 2^{102-64} \times 2^{54} + 2^{102-64} \times 2^{64} = 2^{93} + 2^{102} \approx 2^{102}. \tag{5}$$

It should be noted that in [13], the authors simply checked whether M_3 and $M_3 \oplus \Delta S_3^0$ can follow the 2-round differential characteristic by exhausting M_3 and hence the time complexity in [13] is estimated as $2 \times 2^{102} = 2^{103}$. In other words, they do not take the specific conditions into account, while in the above, we only check whether the conditions on the S^1 hold for each M_3.

4.1 Optimizing $T_{\texttt{k}}$ Using Simple Linear Algebra

Indeed, it is quite straightforward to optimize $T_{\texttt{k}}$. However, even if it is reduced to 1, the time complexity is still high, i.e., 2^{92} according to Equation 5. Let us elaborate on how to significantly optimize $T_{\texttt{k}}$ in this section. First, we need to study some properties of the S-box.

Studying the active S-boxes in the first round. First, we describe why there are 54 bit conditions on the capacity part of S^0.

Property 1 *[22] For an input difference $(\Delta_0, \ldots, \Delta_4)$ satisfying $\Delta x_1 = \Delta x_2 = \Delta x_3 = \Delta x_4 = 0$ and $\Delta x_0 = 1$, the following constraints hold:*

- *For the output difference:*

$$\begin{cases} \Delta y_0 \oplus \Delta y_4 = 1, \\ \Delta y_1 = \Delta x_0, \\ \Delta y_2 = 0. \end{cases} \tag{6}$$

- *For the input value:*

$$\begin{cases} x_1 = \Delta y_0 \oplus 1, \\ x_3 \oplus x_4 = \Delta y_3 \oplus 1. \end{cases} \tag{7}$$

Based on Property 1 and the 2-round differential characteristic in Table 4, we can derive $27 + 27 = 54$ bit conditions on the capacity part of S^0, i.e., 27 bit conditions on $S^0[1]$ and 27 bit conditions on $S^0[3] \oplus S^0[4]$. This also explains why $n_c = 54$.

Studying the active S-boxes in the second round. As the next step, we further study the 28 active S-boxes in the second round. We observe that from ΔS^1 to $\Delta S^{1,s}$, there are only 3 different possible difference transitions $(\Delta x_0, \ldots, \Delta x_4) \rightarrow (\Delta y_0, \ldots, \Delta y_4)$ through the S-box, as shown below:

$$(1,1,0,0,1) \rightarrow (1,0,0,0,0),$$
$$(0,0,0,1,1) \rightarrow (1,0,0,0,0),$$
$$(0,1,0,0,1) \rightarrow (1,0,0,0,0).$$

Similar to the algebraic attacks on LowMC [15, 18], we study and exploit the properties of the $(x_0, \ldots, x_4)$ such that

$$\texttt{SB}(x_0, \ldots, x_4) \oplus \texttt{SB}(x_0 \oplus \Delta x_0, \ldots, x_4 \oplus \Delta x_4) = (\Delta y_0, \ldots, \Delta y_4) = (1,0,0,0,0)$$

where

$$(\Delta x_0, \ldots, \Delta x_4) \in \{(1,1,0,0,1), (0,0,0,1,1), (0,1,0,0,1)\}.$$

It is found that

- for $(\Delta x_0, \ldots, \Delta x_4) = (1,1,0,0,1)$, all possible $(x_0, \ldots, x_4)$ form an affine subspace of dimension 2, as shown below:

$$x_0 \oplus x_4 = 0, \quad x_1 = 1, \quad x_3 = 0; \tag{8}$$

- for $(\Delta x_0, \ldots, \Delta x_4) = (0,0,0,1,1)$, all possible $(x_0, \ldots, x_4)$ form an affine subspace of dimension 2, as shown below:

$$x_1 = 0, \quad x_2 = 0, \quad x_3 \oplus x_4 = 0; \tag{9}$$

- for $(\Delta x_0, \ldots, \Delta x_4) = (0,1,0,0,1)$, all possible $(x_0, \ldots, x_4)$ form an affine subspace of dimension 1, as shown below:

$$x_0 = 0, \quad x_1 \oplus x_4 = 1, \quad x_2 = 0, \quad x_3 = 0. \tag{10}$$

As a result, the difference transitions in the second round, i.e., the 28 active S-boxes, directly impose 102 *linear conditions* on S^1. Note that it is unclear whether the probability 2^{-102} is directly computed according to the differential distribution table (DDT) of the 5-bit S-box in [13]. At least, we do not see any such related claims in [13] that the probability 2^{-102} is caused by 102 linear conditions on S^1, i.e., the conditions may be *nonlinear* if we do not carefully study the relations between the difference transitions and values transitions. Indeed, we can simply generalize the above observations for any degree-2 S-box, as shown in Appendix C, i.e. all the conditions on the input bits must be linear for each valid diffeerence transition of a degree-2 S-box.

More nonlinear conditions on the capacity part of S^0. As can be noted from Equation 9 and Equation 10, there will be conditions on $S^1[2]$, i.e., the conditions on x_2 in Equation 9 and Equation 10. However, according to the definition of the S-box, we know that

$$y_2 = x_4 x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1.$$

Hence, after the capacity part of S_3^0 is fixed, $S^1[2]$ is irrelevant to $S^0[0]$. As a result, apart from the 54 linear conditions on the capacity part of S^0, there are also 21 nonlinear (quadratic) conditions on the capacity part of S^0. In other words, at the first glance, although there are 102 linear conditions on S^1, there are indeed only $102 - 21 = 81$ linear conditions on S^1 depending on $S^0[0]$ after the capacity part of S^0 is known. Hence, we can equivalently say that

$$n_c = 54 + 21 = 75, \quad n_k = 81.$$

With the general 2-step attack framework, the total time complexity is not affected as $n_c + n_k$ remains the same, i.e., it is still 2^{102}.

Optimizing $T_{\mathtt{k}}$. After knowing that there are 81 linear conditions on S^1 depending on $S^0[0]$ after the capacity part of S^0 is known, optimizing $T_{\mathtt{k}}$ is quite straightforward. Recall the general 2-step attack framework described previously. Specifically, by using 3 message blocks (M_1, M_2, M_3), we first generate valid (M_1, M_2) such that the 75 bit conditions on the capacity part of S_3^0 can hold. Then, since M_3 is only added to $S_3^0[0]$, S_3^1 directly becomes linear in M_3 and we know there are 81 linear conditions on S_3^1. Therefore, we can construct 81 linear equations in M_3, i.e., 64 variables. Similar to the idea in [16], solving this linear equation system is equivalent to exhausting all possible values of M_3 and hence $T_{\mathtt{k}}$ is reduced to the time complexity to solve 81 linear equations in 64 variables that requires $81 \times 81 \times 64 \approx 2^{19}$ bit operations. As explained before, only optimizing $T_{\mathtt{k}}$ is insufficient to significantly improve the attack and we need to further optimize $T_{\mathtt{pre}}$.

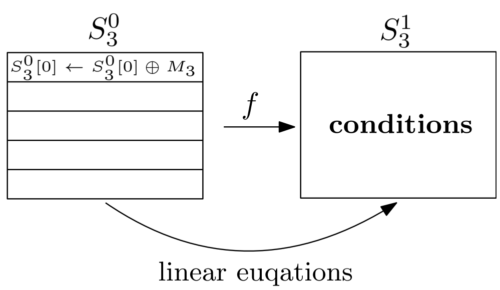

Fig. 7: Exhaust M_3 by solving linear equations

4.2 Finding Valid (M_1, M_2) with Advanced Techniques

To find valid (M_1, M_2), we are now only simply looping over (M_1, M_2) and checking whether the 75 bit conditions on the capacity part can hold. To improve the attack, we have to avoid such a naive loop. In what follows, we describe how to use the general 3-step attack framework stated above to overcome this obstacle.

The core idea is to utilize a 2-round property of ASCON. Let us explain it step by step.

Property 2 *For $(y_0, \ldots, y_4) = \mathit{SB}(x_0, \ldots, x_4)$, if $x_3 \oplus x_4 = 1$, y_3 will be independent to x_0.*

Proof. We can rewrite y_3 as follows:

$$y_3 = (x_4 \oplus x_3 \oplus 1)x_0 \oplus (x_4 \oplus x_3 \oplus x_2 \oplus x_1).$$

Hence, if $x_3 \oplus x_4 = 1$, y_3 is irrelevant to x_0.

Property 3 *Let*

$$(S^1[0], \ldots, S^1[4]) = f(S^0[0], \ldots, S^0[4]), \quad (S^2[0], \ldots, S^2[4]) = f(S^1[0], \ldots, S^1[4]),$$

where $(S^0[1], S^0[2], S^0[3], S^0[4])$ are constants and $S^0[0]$ is the only variable. Then, it is always possible to make u bits of $S^2[1]$ linear in $S^0[0]$ by adding at most $9u$ bit conditions on $S^0[3] \oplus S^0[4]$.

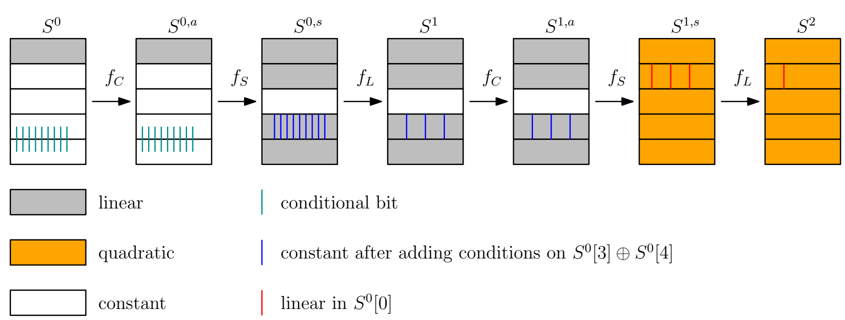

Fig. 8: Adding conditions on the capacity part to linearize $S^2[1]$

Proof. First, since $S^0[0]$ is the only variable, according to the definition of f, we know that $(S^1[0], S^1[1], S^1[3], S^1[4])$ are linear in $S^0[0]$ while $S^1[2]$ is still constant.

Each bit $S^2[1][i]$ can be expressed as

$$S^2[1][i] = S^{1,s}[1][i] \oplus S^{1,s}[1][i+61] \oplus S^{1,s}[1][i+39].$$

To make $S^2[1][i]$ linear in $S^0[0]$, we need to ensure

$$S^{1,s}[1][i] \oplus S^{1,s}[1][i+61] \oplus S^{1,s}[1][i+39]$$

is linear in $S^0[0]$. According to the definition of the S-box specified in Equation 1, the expression of y_1 is

$$y_1 = x_4 \oplus x_1x_3 \oplus x_3 \oplus x_2(x_3 \oplus x_1 \oplus 1) \oplus x_1 \oplus x_0.$$

Hence, if x_2 is constant, there is only one quadratic term x_1x_3 in the expression of y_1.

According to the above analysis, $S^1[2]$ is always constant. Hence, we have

$$\begin{aligned} & S^{1,s}[1][i] \oplus S^{1,s}[1][i+61] \oplus S^{1,s}[1][i+39] \\ = & S^1[1][i]S^1[3][i] \oplus S^1[1][i+61]S^1[3][i+61] \oplus S^1[1][i+39]S^1[3][i+39] \\ & \oplus L_i(S^1[0], \ldots, S^1[4]) \end{aligned} \tag{11}$$

where L_i is a linear function.

Furthermore, according to Property 2, we can make $S^{0,s}[3][i]$ $(0 \leq i \leq 63)$ irrelevant to $S^0[0]$ by adding 1 bit condition on $S^0[3] \oplus S^0[4]$. In this way, we can add at most 9 bit conditions on $S^0[3] \oplus S^0[4]$ to make $(S^1[3][i], S^1[3][i+61], S^1[3][i+39])$ irrelevant to $S^0[0]$ since each bit of $S^1[3]$ is linear in 3 bits of $S^{0,s}[3]$. Once $(S^1[3][i], S^1[3][i+61], S^1[3][i+39])$ is irrelevant to $S^0[0]$, $S^{1,s}[1][i] \oplus S^{1,s}[1][i+61] \oplus S^{1,s}[1][i+39]$ becomes linear in $S^0[0]$ according to Equation 11. Hence, to make u bits of $S^2[1]$ linear in $S^0[0]$, we need to add at most $9u$ bit conditions on $S^0[3] \oplus S^0[4]$.

A graphical explanation for Property 3 can be seen from Fig. 8.

Property 4 *Let*

$$(S^1[0], \ldots, S^1[4]) = f(S^0[0], \ldots, S^0[4]), \quad (S^2[0], \ldots, S^2[4]) = f(S^1[0], \ldots, S^1[4]),$$

where $(S^0[1], S^0[2], S^0[3], S^0[4])$ *are constants and* $S^0[0]$ *is the only variable. Then, it is always possible to make* u *bits of* $S^2[1]$ *linear in* $S^0[0]$ *by guessing* $3u$ *linear equations in* $S^0[0]$.

Proof. Similar to the proof of Property 3, we have

$$\begin{aligned} & S^{1,s}[1][i] \oplus S^{1,s}[1][i+61] \oplus S^{1,s}[1][i+39] \\ = & S^1[1][i]S^1[3][i] \oplus S^1[1][i+61]S^1[3][i+61] \oplus S^1[1][i+39]S^1[3][i+39] \\ & \oplus L_i(S^1[0], \ldots, S^1[4]) \end{aligned}$$

where L_i is a linear function and $(S^1[0], S^1[1], S^1[2], S^1[3], S^1[4])$ are linear in $S^0[0]$. Hence, if we guess $(S^1[3][i], S^1[3][i+61], S^1[3][i+39])$, $S^2[1][i]$ will be linear in $S^0[0]$. In other words, by guessing 3 linear equations in $S^0[0]$, $S^2[1][i]$ can be linear in $S^0[0]$.

A graphical explanation for Property 4 can be seen from **??**.

Improving the attack. Based on the above discussions, it is now possible to further improve the 2-round collision attack. We utilize the general 3-step attack framework where $k = 3$, i.e., we use message blocks (M_1, M_2, M_3). From previous analysis, there are 54 linear conditions on the capacity part of S_3^0 and among them, 27 bit conditions are on $S_3^0[1]$ (or $S_2^2[1]$). Based on Property 3 and Property 4, it is possible to satisfy these 54 linear conditions more efficiently with advanced algebraic techniques, i.e., we can improve $T_{\mathtt{k-1}}$. We emphasize that there are additional 21 quadratic conditions on the capacity part of S_3^0, but we will not consider them to speed up the exhaustive search over M_2 due to the increasing difficulty, i.e., it is required to solve degree-4 Boolean equations.

Specifically, based on Property 3, we can add $9u_1$ conditions on the capacity part of S_2^0 such that u_1 bits of $S_3^0[1]$ can be linear in M_2 after the capacity part of S_2^0 is known. Moreover, based on Property 4, after the capacity part of S_2^0 is known, we can guess $3u_2$ linear equations in M_2 such that u_2 bits of $S_3^0[1]$ can be linear in M_2. In total, we set up $u_1 + 4u_2$ linear equations in 64 variables to satisfy $u_1 + u_2$ out of 27 bit conditions. Then, we perform the Gaussian elimination on these $u_1 + 4u_2$ linear equations and obtain

$$u_3 = 64 - u_1 - 4u_2$$

free variables.

Note that the first round is always freely linearized and the remaining $54 - u_1 - u_2$ linear conditions on S_3^0 can be expressed as quadratic equations in these u_3 free variables. In a word, to efficiently exhaust M_2 such that the 54 conditions on S_3^0 can hold, we can perform the following procedure:

Step 1: Guess $3u_2 = 42$ bits of M_2 and construct $4u_2 + u_1$ linear equations.
Step 2: Apply the Gaussian elimination to the system and obtain $u_3 = 64 - u_1 - 4u_2$ free variables.
Step 3: Construct $54 - u_1 - u_2$ quadratic equations in these u_3 variables and solve the equations.
Step 4: Check whether the remaining 21 quadratic conditions on the capacity part of S_3^0 can hold for each obtained solution.

We use a similar method in [3, 17] to estimate the time complexity to solve a quadratic equation system. After some calculations, the optimal choice of (u_1, u_2, u_3) is as follows:

$$u_1 = 3, \quad u_2 = 13 \quad u_3 = 9.$$

In other words, we need to perform the Gaussian elimination on 55 linear equations in 64 variables for $2^{3u_2} = 2^{39}$ times. Then, we need to solve 38 quadratic equations in 9 variables for 2^{39} times. The total time complexity is estimated as

$$2^{39} \times (55^2 \times 64 + 38^2 \times 45) \approx 2^{56.6}$$

bit operations. The cost of Step 4 is negligible since it is expected to perform such a check for about $2^{64-54} = 2^{10}$ times.

Time complexity evaluation. Based on the previous general 3-step attack framework using 3 message blocks (M_1, M_2, M_3), we have $9u_1 = 27$ conditions on S_2^0 and we need $2^{81+75-128} = 2^{28}$ different valid M_1. The cost of this step can be estimated as $2^{28+27} = 2^{55}$ calls to the 2-round ASCON permutation. Then, for each valid M_1, i.e., each valid S_2^0, we can exhaust M_1 with $2^{56.6}$ bit operations. At last, for each valid (M_1, M_2), we can exhaust M_3 with about 2^{19} bit operations. Assume that one round of the ASCON permutation takes about $15 \times 64 \approx 2^{10}$ bit operations, the total time complexity can be estimated as

$$T_{\mathtt{total}} = 2^{28} \times 2^{27} + 2^{28} \times 2^{56.6-11} + 2^{17} \times 2^{19-11} \approx 2^{73.6}$$

calls to the 2-round ASCON permutation.

4.3 Further Optimizing the Guessing Strategy

In the above improved 2-round collision attack, we mainly exploit Property 3 and Property 4 to make some conditional bits of $S_2^2[1]$ linear in M_2. Specifically, the core problem is to make

$$(S_2^1[3][i], S_2^1[3][i+61], S_2^1[3][i+39])$$

constant by either guessing their values according to Property 4 or adding conditions on $S_2^0[3] \oplus S_2^0[4]$ according to Property 3. However, the two strategies are independently used for different bits of $S_2^2[1]$. It can be noted that for one specific conditional bit of $S_2^2[1]$, i.e., $S_2^2[1][i]$, we can guess g out of 3 bits of $(S_2^1[3][i], S_2^1[3][i+61], S_2^1[3][i+39])$ and add $3 \times (3-g)$ conditions on $S_2^0[3] \oplus S_2^0[4]$ to achieve the same goal. In other words, for the same conditional bit, we can use a hybrid guessing strategy.

As the next step, we aim to optimize the guessing strategy such that we can obtain a sufficient number of linear equations by guessing a smaller number of linear equations or adding a smaller number of extra conditions on $S_2^0[3] \oplus S_2^0[4]$. For example, for the above naive guess strategy, we need to add $9u_1 = 27$ bit conditions on $S_2^0[3] \oplus S_2^0[4]$ and we need to further guess $3u_2 = 39$ linear equations in order to get $u_1 + 4u_2 = 3 + 52 = 55$ linear equations in M_2. Can we guess fewer bits to achieve better results?

Note that there are 27 conditional bits in $S_2^2[1]$. For completeness, we denote the set of i such that $S_2^2[1][i]$ is conditional by $\mathcal{I}$ and we have

$$\begin{aligned}\mathcal{I} = \{&0, 7, 8, 10, 12, 16, 17, 19, 24, 27, 28, 30, 31, 32, 34, 37,\\ &40, 41, 48, 50, 54, 56, 57, 59, 60, 61, 63\}.\end{aligned}$$

For each $i \in \mathcal{I}$, let

$$\mathcal{P}_i = \{i, (i+61)\%64, (i+39)\%64\}.$$

Further, let

$$\mathcal{P}_i = \mathcal{P}_{i,g} \cup \mathcal{P}_{i,a}, \quad \mathcal{P}_{i,g} \cap \mathcal{P}_{i,a} = \emptyset.$$

In other words, to linearize $S_2^2[1][i]$, we guess $S_2^1[3][j_0]$ where $j_0 \in \mathcal{P}_{i,g}$ and make $S_2^1[3][j_1]$ constant where $j_1 \in \mathcal{P}_{i,a}$ by adding 3 conditions on

$$S_2^0[3][j_1]\oplus S_2^0[4][j_1], \quad S_2^0[3][j_1+10]\oplus S_2^0[4][(j_1+10)], \quad S_2^0[3][j_1+17]\oplus S_2^0[4][(j_1+17)],$$

We can build a simple MILP model to determine the optimal choice of a subset $\mathcal{I}' \subseteq \mathcal{I}$ and the corresponding $\mathcal{P}_{i,g}$ and $\mathcal{P}_{i,a}$ where $i \in \mathcal{I}'$ such that the total time complexity of the attack is optimal. Specifically, assuming that after adding u_4 conditions on $S_2^0[3] \oplus S_2^0[4]$ and guessing u_5 bits of $S_2^1[3]$, we can set up u_6 linear equations for u_6 conditional bits of $S_2^2[1]$. In this way, we have in total $u_5 + u_6$ linear equations and after the Gaussian elimination, we can set up $54 - u_6$ quadratic equations in $u_7 = 64 - u_5 - u_6$ free variables. After some configurations, we propose to choose

$$u_4 = 31, \quad u_5 = 28, \quad u_6 = 27$$

as the optimal parameters. In other words, we can make all the 27 conditional bits of $S_2^2[1]$ linear in M_2 by guessing 28 linear equations in $S_2^1[3]$ and adding 31 bit conditions on $S_2^0[3] \oplus S_2^0[4]$. In this way, we need to perform the Gaussian elimination to $u_5 + u_6 = 55$ linear equations in 64 variables that requires about $2^{17.6}$ bit operations and then solve 27 quadratic equations in $u_7 = 64 - 55 = 9$ variables. Based on the method [3,17] to estimate the time complexity to solve such an overdefined quadratic equation system, it takes about $27^2 \times 45 + 2^3 \times 12^2 \times 6 \approx 2^{15.3}$ bit operations. Hence, the new total time complexity is

$$T_{\texttt{total}} = 2^{28} \times 2^{31} + 2^{28} \times 2^{28} \times (2^{17.6} + 2^{15.3}) \times 2^{-11} + 2^{17} \times 2^{19-11} \approx 2^{62.6}.$$

In conclusion, with the optimal guess strategy and advanced algebraic techniques, we can improve the best collision attack on 2-round ASCON-HASH by a factor of about $2^{40.4}$. For completeness, the required 28 guessed bits of $S_2^1[3]$ and the 31 condition bits of $S_2^0[3] \oplus S_2^0[4]$ are shown in Table 5.

Table 5: The optimal guessing strategy

$\bigcup_{i\in\mathcal{I}} \mathcal{P}_{i,g}$
$\{0, 3, 4, 7, 8, 10, 14, 15, 17, 21, 24, 25, 27, 28, 31, 32, 34, 35, 37, 38, 41, 45, 48, 51, 54, 55, 58, 61\}$
$\bigcup_{i\in\mathcal{I}} \mathcal{P}_{i,a}$
$\{2, 5, 6, 9, 12, 13, 16, 19, 23, 29, 30, 36, 39, 40, 46, 47, 49, 50, 53, 56, 57, 59, 60, 63\}$
$\{j, (j + 10)\%64, (j + 17)\%64 \mid j \in \bigcup_{i\in\mathcal{I}} \mathcal{P}_{i,a}\}$
$\{0, 2, 3, 5, 6, 9, 10, 12, 13, 15, 16, 19, 22, 23, 26, 29, 30, 33, 36, 39, 40, 46, 47, 49, 50, 53, 56, 57, 59, 60, 63\}$

5 Semi-Free-Start Collision Attack on 4-round ASCON

As can be observed in the above improved 2-round collision attack, one critical step is to study the relations between the difference transitions and value transitions through the S-box, i.e., we can derive linear conditions on some internal state bits to make the difference transitions hold. In the following, we will explore similar such properties for the 4-round differential characteristic proposed by the designers in [10], as shown in Table 6.

5.1 Deriving Implicit Linear Conditions

For each state difference from ΔS^i to ΔS^{i+1} where $i \geq 0$, we can know the corresponding input difference $(\Delta x_0, \ldots, \Delta x_4)$ and output difference $(\Delta y_0, \ldots, \Delta y_4)$ for each active S-box. Our goal is to find linear conditions on $(x_0, \ldots, x_4)$ such that

$$\texttt{SB}(x_0, \ldots, x_4) \oplus \texttt{SB}(x_0 \oplus \Delta x_0, \ldots, x_4 \oplus \Delta x_4) = (\Delta y_0, \ldots, \Delta y_4)$$

always holds for each possible $(\Delta x_0, \ldots, \Delta x_4) \rightarrow (\Delta y_0, \ldots, \Delta y_4)$ in round i. We specify them round by round, as shown below:

- Round $i = 0$: there are only 3 active S-boxes in round 0 and we can derive 6 linear conditions:
$$S^{0,a}[1][j] = 0, \quad S^{0,a}[3][j] \oplus S^{0,a}[4][j] = 1, \text{ for } j \in \{18, 60, 62\}.$$
- Round $i = \{1, 2, 3\}$: please refer to Table 8, Table 9, Table 10 in Appendix A for explanations.

In summary, there are totally 6, 27, 125 and 137 linear conditions on S^0, S^1, S^2 and S^3, respectively. Specifically, as f_C is applied before f_S, S^i and $S^{i,a}$ are linearly related. In the Table 8, Table 9, Table 10, we record the conditions on S^i. Note that because $S^{i,a}$ is the input to the substitution layer, we convert them into linear conditions on S^i where $1 \leq i \leq 3$.

Performing the search with STP. With these information, we can use a SAT/SMT-based method to efficiently find an input satisfying all these conditions. We tried in total 2 different strategies. The first strategy is shown below:

Step 1: Start from S^2 where we load the corresponding 125 constraints.
Step 2: Load the constraints on S^3 and model the relation between S^2 and S^3.
Step 3: Solve the model with the STP solver[7] and for each solution, check the conditions on (S^0, S^1).

According to 100 experiments, the solver outputs a solution of (S^2, S^3) in about 1.5s and the valid solution of (S^0, S^1, S^2, S^3) can be found in about 106 seconds.

The second strategy is simple. We directly load all the constraints and expect that the solver directly outputs a valid solution of (S^0, S^1, S^2, S^3). For this strategy, we obtain a solution in about 1483 seconds. Some examples are given in Table 7.

In a word, we demonstrate that it is indeed quite efficient to find a conforming message pair for the 4-round differential characteristic. Since no details are provided in [10], we expect that this work can fill this blank.

[7] https://github.com/stp/stp

Table 6: Semi-free-start collision for 4-round ASCON-HASH and ASCON-XOF in [10]

round	state (S)	difference (ΔS)
0	0x177537760b6a7b4b	0x9000000000040000
	0x6e7a0bba2ed9e436	0x0
	0x9aff10e403752f21	0x0
	0x7ac1d330cf9ee9c2	0x0
	0x88fc524dd1092975	0x0
1	0x8e2919a34aa78b4f	0x1040120900040000
	0xf8ec50f5193e17ff	0x1000080001040004
	0xf8c88d0910726467	0x0
	0x5c7453f66c0f3efd	0x0
	0x03f613581bb25cb9	0x0
2	0x14e7b8acbbf085f1	0x904088490145a084
	0x6a25ac7c557f0f4e	0x10428a4101248000
	0x9984d786381625f7	0x08400c2001821006
	0x1e230875a0079fa9	0x114602278c44c186
	0xe0c29f3a0dff9d81	0x10e0902102082008
3	0x5e994e62eba7e010	0xc1824ac20aa400cb
	0xc502f6422ec4b3d7	0x14831e8a81a4814e
	0xc96362c46ea40408	0x14831e0281a48183
	0xbf0c9307b5efe0b1	0xe30040611a1b4881
	0x2afe991b302b65a3	0x320000ab913b484c
4	0xccaa3e2b2adb8f9b	0xb5463ce488575401
	0x2648f9ab9dc8f4e0	0x0
	0x8d17e35ce6ae9626	0x0
	0xf92955837cd0e419	0x0
	0xb78b0c1137cdc72d	0x0

Table 7: Semi-free-start collisions for 4-round ASCON-HASH and ASCON-XOF

round	Result 1 (State S)	Result 2 (State S)	Result 3 (State S)
	0x6f3efab5f501d116	0xebd726f16895041f	0xb6b5bca0cb598d0d
	0x082990745e92414e	0x4f54446e9cfb3cdb	0x2153d303a8734b8c
0	0xa429e93463313492	0xc5437bb993cb906a	0x97dc27a6259e7fb7
	0x070b11e3cc676a98	0x7ff22eaee45a8d76	0x31721017cb7d5c55
	0xde8143841bbb645e	0xa0c65761b906d440	0xa181b777f901823d
	0xb9bc6018fb5daf80	0xcead4b3dc3704f95	0x9d00e741960c551e
	0x26d5dc41b71ea587	0xf3ca38a1bad24c39	0x68ff7105e74dc84e
1	0x18edd9f4d661dbe5	0x74cd6b98ba89ae6f	0xde2ebc978a1324b6
	0xb85745c86657d06f	0xd2c202c858e57374	0x9c2b556b826ff4d4
	0x41caa8a0e03b3374	0x666e0760c8fba563	0xc98c3841097379e2
	0xe8effd0ccb54a995	0x64e655c5daa69bc5	0x50f774cd8a270180
	0x7b7aceec845b634f	0x387e6629b44b252f	0x4d74a6a9ad4be13f
2	0x7269c616b98651d3	0xdfcdafb2d99f2f8b	0x1cddbf32e00753cd
	0xc9386a755903debd	0xaf244ad6fc539dfd	0x3a3f4aca5d42588a
	0x5a73faba7c4fa5e0	0x5e7b5a0b3f7d9da0	0x82da6b932ef49f88
	0xb1005013ab32836a	0xdf9f610483aa4889	0xb4c48a81df87c2b9
	0xdd277c400ae0b085	0xc541fcc4efa095c9	0xc142cbc4ee248595
3	0xd16a69d62ac0213a	0xc93ce9c2efc0109b	0xc567ffd28e44309b
	0x72c9b376e56f80a1	0x305d8647e9f45432	0x7b256f7291759002
	0x360f8c3a00ab14cb	0x20b69d0d8170e3ad	0x22e3b5a8909101b9
	0xdc251abeec776446	0x55523c5c2000d0dd	0x2f6ea61dc4f7c9ea
	0x51687328dfd6d033	0x0cf9c3239fd9b120	0x5424198b0d0c4131
4	0x2bb046c8b78dba46	0x857c09c6c7ae4a75	0x297817bcd6ec9360
	0x7f3e74d6c8c1147c	0x6d31e7124ed0fc77	0xd766eac7fb8e586c
	0x46ad0168cb618e09	0x7bc89fe833a7bf9d	0x4be7f4600b8b5f09

6 Conclusion

By carefully studying the relations between the difference transitions and values transitions through the S-box, we show that the existing collision attacks on 2-round ASCON-HASH can be significantly improved with the aid of advanced algebraic techniques. Furthermore, based on similar relations, we also complement the designers' semi-free-start collision attack by using a dedicated SAT/SMT-based method to find semi-free-start colliding message pairs. It is found that such a 4-round semi-free-start collision can be found in less than 2 minutes. We expect our close look at the algebraic properties of the S-box can inspire more efficient attacks on ASCON-HASH or ASCON-XOF.

Acknowledgements This work is supported by the National Key Research and Development Program of China (No. 2022YFB2701900); the National Natural Science Foundation of China (Nos. 62072181, 62132005); and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

References

1. The CAESAR committee, CAESAR: competition for authenticated encryption: security, applicability, and robustness (2014), https://competitions.cr.yp.to/caesar-submissions.html
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: ECRYPT hash workshop. No. 9 (2007)
3. Bouillaguet, C., Delaplace, C., Trimoska, M.: A Simple Deterministic Algorithm for Systems of Quadratic Polynomials over $\mathbb{F}_2$. In: Bringmann, K., Chan, T. (eds.) 5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022, Virtual Conference, January 10-11, 2022. pp. 285–296. SIAM (2022). https://doi.org/10.1137/1.9781611977066.22, https://doi.org/10.1137/1.9781611977066.22
4. Bovy, E., Daemen, J., Mennink, B.: Comparison of the second round candidates of the NIST lightweight cryptography competition. Bachelor Thesis, Radboud University (2020)
5. Dinur, I., Dunkelman, O., Shamir, A.: New Attacks on Keccak-224 and Keccak-256. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 442–461. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_25, https://doi.org/10.1007/978-3-642-34047-5_25
6. Dobraunig, C., Eichlseder, M., Mendel, F., Schläer, M.: Status Update on ASCON v1. 2 (2020)
7. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1. Submission to Round 1 of the CAESAR competition (2014), https://competitions.cr.yp.to/round1/Asconv1.pdf
8. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1.1. Submission to Round 2 of the CAESAR competition (2015), https://competitions.cr.yp.to/round2/Asconv11.pdf

9. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to Round 1 of the NIST Lightweight Cryptography project (2019), https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Ascon-spec.pdf
10. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Preliminary Analysis of ASCON-XOF and ASCON-HASHJ (2019)
11. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1.2: Lightweight Authenticated Encryption and Hashing. J. Cryptol. **34**(3), 33 (2021). https://doi.org/10.1007/s00145-021-09398-9, https://doi.org/10.1007/s00145-021-09398-9
12. Floyd, R.W.: Nondeterministic algorithms. J. ACM **14**(4), 636–644 (1967). https://doi.org/10.1145/321420.321422, https://doi.org/10.1145/321420.321422
13. Gérault, D., Peyrin, T., Tan, Q.Q.: Exploring Differential-Based Distinguishers and Forgeries for ASCON. IACR Trans. Symmetric Cryptol. **2021**(3), 102–136 (2021). https://doi.org/10.46586/tosc.v2021.i3.102-136, https://doi.org/10.46586/tosc.v2021.i3.102-136
14. Liu, F., Isobe, T., Meier, W.: Automatic Verification of Differential Characteristics: Application to Reduced GIMILI. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 219–248. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_8, https://doi.org/10.1007/978-3-030-56877-1_8
15. Liu, F., Isobe, T., Meier, W.: Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 368–401. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_13, https://doi.org/10.1007/978-3-030-84252-9_13
16. Liu, F., Isobe, T., Meier, W., Yang, Z.: Algebraic Attacks on Round-Reduced Keccak. In: Baek, J., Ruj, S. (eds.) Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13083, pp. 91–110. Springer (2021). https://doi.org/10.1007/978-3-030-90567-5_5, https://doi.org/10.1007/978-3-030-90567-5_5
17. Liu, F., Meier, W., Sarkar, S., Isobe, T.: New Low-Memory Algebraic Attacks on LowMC in the Picnic Setting. IACR Trans. Symmetric Cryptol. **2022**(3), 102–122 (2022). https://doi.org/10.46586/tosc.v2022.i3.102-122, https://doi.org/10.46586/tosc.v2022.i3.102-122
18. Liu, F., Sarkar, S., Wang, G., Meier, W., Isobe, T.: Algebraic Meet-in-the-Middle Attack on LowMC. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 225–255. Springer (2022). https://doi.org/10.1007/978-3-031-22963-3_8, https://doi.org/10.1007/978-3-031-22963-3_8
19. Qiao, K., Song, L., Liu, M., Guo, J.: New Collision Attacks on Round-Reduced Keccak. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017,

Proceedings, Part III. Lecture Notes in Computer Science, vol. 10212, pp. 216–243 (2017). https://doi.org/10.1007/978-3-319-56617-7_8, https://doi.org/10.1007/978-3-319-56617-7_8
20. Qin, L., Zhao, B., Hua, J., Dong, X., Wang, X.: Weak-diffusion structure: Meet-in-the-middle attacks on sponge-based hashing revisited. Cryptology ePrint Archive, Paper 2023/518 (2023), https://eprint.iacr.org/2023/518, https://eprint.iacr.org/2023/518
21. Song, L., Liao, G., Guo, J.: Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 428–451. Springer (2017). https://doi.org/10.1007/978-3-319-63715-0_15, https://doi.org/10.1007/978-3-319-63715-0_15
22. Zong, R., Dong, X., Wang, X.: Collision Attacks on Round-Reduced GIMLI-HASH/ASCON-XOF/ASCON-HASH. Cryptology ePrint Archive, Paper 2019/1115 (2019), https://eprint.iacr.org/2019/1115, https://eprint.iacr.org/2019/1115

A Conditions on the Internal States

In this part, we present the conditions on the internal states for the 4-round differential characteristic, as shown in Table 8, Table 9 and Table 10. All the values in this section are binary.

Table 8: Conditions on $S^{1,a}$

j	Δx	Δy	conditions ($x_i = S^{1,a}[i][j]$)
2	01000	11110	$x_0 \oplus x_4 = 1, x_2 = 1, x_3 = 1$
18	11000	11110	$x_0 \oplus x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 0$
24	01000	11110	$x_0 \oplus x_4 = 1, x_2 = 0, x_3 = 0$
32	10000	01011	$x_1 = 1, x_3 \oplus x_4 = 0$
35	10000	11000	$x_1 = 0, x_3 \oplus x_4 = 1$
41	10000	11010	$x_1 = 0, x_3 \oplus x_4 = 0$
43	01000	10110	$x_0 \oplus x_4 = 1, x_2 = 1, x_3 = 0$
44,54	10000	01011	$x_1 = 1, x_3 \oplus x_4 = 0$
60	11000	10111	$x_0 \oplus x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0$

B The Algorithmic Description of ASCON-HASH

The The Algorithmic Description of ASCON-HASH is shown in Algorithm 1.

Table 9: Conditions on $S^{2,a}$

j	Δx	Δy	conditions ($x_i = S^{2,a}[i][j]$)
1	00110	11101	$x_0 = 0, x_1 = 1, x_2 \oplus x_3 = 1, x_4 = 0$
2	10110	11100	$x_0 \oplus x_3 = 0, x_1 = 1, x_2 \oplus x_3 = 1, x_4 = 0$
3	00001	11010	$x_0 = 0, x_1 = 1, x_3 = 1$
7	10010	00101	$x_0 \oplus x_3 = 0, x_1 = 0, x_2 = 1, x_4 = 1$
8	00010	11101	$x_0 = 1, x_1 \oplus x_2 = 0, x_4 = 1$
12	00100	10110	$x_1 = 0, x_3 = 1$
13	10001	10111	$x_0 \oplus x_4 = 0, x_3 = 0$
14	00010	11011	$x_0 = 0, x_1 \oplus x_2 = 0, x_4 = 0$
15	11010	01011	$x_0 \oplus x_1 = 1, x_0 \oplus x_3 = 0, x_2 = 0, x_4 = 1$
16	10000	01011	$x_1 = 1, x_3 \oplus x_4 = 0$
17	00100	01110	$x_1 = 1, x_3 = 1$
18	11010	10001	$x_0 \oplus x_1 = 1, x_1 \oplus x_3 = 0, x_2 = 1, x_4 = 1$
19	00001	11110	$x_0 = 0, x_1 = 1, x_3 = 0$
21	01000	11111	$x_0 \oplus x_4 = 0, x_2 = 0, x_3 = 0$
22	10010	11110	$x_0 \oplus x_3 = 1, x_1 = 1, x_2 = 0, x_4 = 1$
23	00100	11110	$x_1 = 0, x_3 = 0$
24	11100	11001	$x_1 \oplus x_2 = 1, x_3 \oplus x_4 = 1, x_4 = x_1 \oplus x_0 \oplus 1,$
25	00001	01101	$x_0 = 1, x_1 = 0, x_3 = 0$
26	00010	10111	$x_0 = 0, x_1 \oplus x_2 = 1, x_4 = 1$
27	00010	10101	$x_0 = 1, x_1 \oplus x_2 = 1, x_4 = 1$
31	00010	11001	$x_0 = 1, x_1 \oplus x_2 = 0, x_4 = 0$
32	11011	10011	$x_0 \oplus x_1 = 0, x_1 \oplus x_3 = 1, x_3 \oplus x_4 = 1, x_2 = 0$
33	00010	10111	$x_0 = 0, x_1 \oplus x_2 = 1, x_4 = 1$
34	00010	11001	$x_0 = 1, x_1 \oplus x_2 = 0, x_4 = 0$
35	10000	01001	$x_1 = 1, x_3 \oplus x_4 = 1$
37	00111	01111	$x_1 = 1, x_2 \oplus x_3 = 1, x_3 \oplus x_4 = 0$
38	11000	01111	$x_0 \oplus x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$
41	01010	01001	$x_0 = 0, x_1 \oplus x_3 = 0, x_2 = 1, x_4 = 1$
42	00100	00110	$x_1 = 1, x_3 = 0$
43	11100	11011	$x_1 \oplus x_2 = 1, x_3 \oplus x_4 = 0, x_4 = x_1 \oplus x_0 \oplus 1,$
44	00001	01101	$x_0 = 1, x_1 = 0, x_3 = 0$
47	11001	10000	$x_0 \oplus x_4 = 0, x_2 = 1, x_3 = 0$
49	01010	00010	$x_0 = 1, x_1 \oplus x_3 = 1, x_2 = 0, x_4 = 1$
50	00010	11001	$x_0 = 1, x_1 \oplus x_2 = 0, x_4 = 0$
53	00001	11000	$x_0 = 1, x_1 = 1, x_3 = 1$
54	11111	10101	$x_0 \oplus x_4 = 0, x_1 \oplus x_2 = 0, x_3 \oplus x_4 = 1$
55	00001	01101	$x_0 = 1, x_1 = 0, x_3 = 0$
56	00010	10011	$x_0 = 0, x_1 \oplus x_2 = 1, x_4 = 0$
59	00100	01110	$x_1 = 1, x_3 = 1$
60	11011	10010	$x_0 \oplus x_4 = 1, x_1 \oplus x_4 = 0, x_2 = 1, x_3 \oplus x_4 = 1$
63	10000	11000	$x_1 = 0, x_3 \oplus x_4 = 1$

Table 10: Conditions on $S^{3,a}$ where $\Delta y = 10000$

j	Δx	conditions $((x_i = S^{3,a}[i][j]))$
63,62,56,46,38,27,25	10010	$x_0 \oplus x_3 = 1, x_1 = 1, x_2 = 1, x_4 = 0$
60,31,24	01101	$x_0 = 1, x_1 = x_2, x_3 = 1, x_1 = x_4$
58,48,44,42,15,8	01100	$x_1 \oplus x_2 = 1, x_0 \oplus x_4 = 1$
61,57,37,32,28,20,19,17,16,14,11	00011	$x_1 = 0, x_2 = 0, x_3 = x_4$
55,49,43,41,23,18	11100	$x_1 = x_2, x_3 \oplus x_4 = 1, x_4 = x_0 \oplus x_1$
1	11100	$x_1 \oplus x_2 = 1, x_3 \oplus x_4 = 1, x_4 = x_0 \oplus x_1$
39,3	11001	$x_0 = x_4, x_2 = 1, x_3 = 0$
6	11001	$x_0 = x_4, x_2 = 0, x_3 = 0$
35,2	01001	$x_0 = 0, x_2 = 0, x_3 = 0, x_1 \oplus x_4 = 1$
33,21	11101	$x_1 \oplus x_2 = 1, x_0 = x_4, x_3 = 1$
7,0	10110	$x_0 \oplus x_3 = 1, x_2 = x_3, x_1 = 1, x_4 = 1$

Algorithm 1: ASCON-HASH

Input: $M \in \{0,1\}^*$
Output: hash $H \in \{0,1\}^{256}$
Initialization:
$S_1^0 \leftarrow f^{12}(IV||0^c)$;

Absorbing:
$M_1, \ldots, M_s \leftarrow M||1||0^*$;
for $i = 1, \ldots, s$ **do**
  $S_{i+1}^0 \leftarrow f^{12}\left(S_i^0 \oplus (M_i||0^c)\right)$;
end

Squeezing:
$S^0 \leftarrow S_{s+1}^0$;
for $i = 1, \ldots, t = \lceil 256/r \rceil$ **do**
  $H_i \leftarrow S^0[0]$;
  $S^0 \leftarrow f^{12}(S^0)$;
end
return $\lfloor H_1|| \ldots ||H_t \rfloor_{256}$;

C On Degree-2 S-box

For an n-bit S-box whose algebraic degree is 2, we can show that for any valid pair of input and output difference, the inputs satisfying this difference transition must form an affine subspace.

Let $(x_0, \dots, x_{n-1}) \in \mathbb{F}_2^n$ and $(y_0, \dots, y_{n-1}) \in \mathbb{F}_2^n$ be the input and output of the S-box. Further, let

$$y_i = f_i(x_0, \dots, x_{n-1}),\ 0 \le i \le n-1,$$

where the algebraic degree of f_i is at most 2.

Given any valid input difference $(\Delta x_0, \dots, \Delta x_{n-1})$ and output difference $(\Delta y_0, \dots, \Delta y_{n-1})$, we aim to show that $(x_0, \dots, x_{n-1})$ satisfying the following n equations must form an affine subspace:

$$\begin{aligned} f_0(x_0, \dots, x_{n-1}) \oplus f_0(x_0 \oplus \Delta x_0, \dots, x_{n-1} \oplus \Delta x_{n-1}) &= \Delta y_0, \\ &\cdots \\ f_{n-1}(x_0, \dots, x_{n-1}) \oplus f_{n-1}(x_0 \oplus \Delta x_0, \dots, x_{n-1} \oplus \Delta x_{n-1}) &= \Delta y_{n-1}. \end{aligned}$$

First, since $(\Delta x_0, \dots, \Delta x_{n-1}) \to (\Delta y_0, \dots, \Delta y_{n-1})$ is a valid difference transition, there must exist solutions to the above n equations. We only need to show that all the n equations are indeed linear in $(x_0, \dots, x_{n-1})$ for each given $(\Delta x_0, \dots, \Delta x_{n-1}, \Delta y_0, \dots, \Delta y_{n-1})$ and then the proof is over. Note that the algebraic degree of f_i is at most 2. In this case,

$$f_i(x_0, \dots, x_{n-1}) \oplus f_i(x_0 \oplus \Delta x_0, \dots, x_{n-1} \oplus \Delta x_{n-1})$$

must be linear in $(x_0, \dots, x_{n-1})$, thus completing the proof.