Conditional Cube Attacks on Ascon-128 and Ascon-80pq in a Nonce-misuse Setting

Donghoon Chang[1,2], Deukjo Hong[1,3], and Jinkeon Kang[1]

[1] National Institute of Standards and Technology, Gaithersburg, Maryland, USA,
donghoon.chang@nist.gov, jinkeon.kang@nist.gov
[2] Department of Computer Science, Indraprastha Institute of Information Technology Delhi(IIIT-Delhi), Delhi, India
[3] Jeonbuk National University, Jeonju-si, Korea, deukjo.hong@jbnu.ac.kr

Abstract. Ascon-128 and Ascon-80pq use 12-round Ascon permutation for initialization and finalization phases and 6-round Ascon permutation for processing associate data and message. In a nonce-misuse setting, we present a new partial-state-recovery conditional-cube attack on Ascon-128 and Ascon-80pq, where 192 bits out of 320-bit state are recovered. For our partial state-recovery attack, its required data complexity, D, is about $2^{44.8}$ and *its required memory complexity, M, is negligible*. After a 192-bit partial state is recovered, in a nonce-misuse setting, we can further recover the full 320-bit state with time complexity, $T = 2^{128}$, and then we can recover the secret key with extra data complexity of $2^{31.5}$, extra time complexity of $2^{129.5}$, and memory complexity of $2^{31.5}$. A similar attack of recovering the partial state was independently developed by Baudrin et al. and published at the NIST fifth Lightweight Cryptography workshop. Note that our attack does not violate the NIST LWC security requirements on Ascon-128 and Ascon-80pq as well as the designers' claims.

Keywords: Ascon-128;Ascon-80pq; lightweight cryptography; state recovery; key recovery

1 Introduction

Ascon [1] is one of the finalists of the National Institute of Standards and Technology (NIST) lightweight cryptography standardization process. Ascon includes three AEAD variants: Ascon-128 (primary), Ascon-128a (second), and Ascon-80pq (third). The primary and second variants use 128-bit key and the third variant uses 160-bit key for its post-quantum security. Ascon-128 and Ascon-80pq are same except the initial value, IV, and the key size.

The designers of Ascon [1] claim that Ascon-128 and Ascon-80pq provide 128-bit security of privacy and authenticity when a repeated nonce is never used for the encryption under the same key. The maximum available data to the attacker is limited to 2^{64} 64-bit blocks per key. In nonce-misuse setting,

the designers claimed that Ascon-128 and Ascon-80pq still provide 128-bit security of privacy and authenticity even if nonces are reused a few times by accident as long as the combination of nonce and associated data stays unique. The designers said that they do not expect that key recovery attacks for Ascon-128 and Ascon-80pq can be found with complexity significantly below 2^{96} even after a secret state is recovered by an implementation attack, because both initialization and finalization are keyed additionally. The designers also claimed that since Ascon-128 and Ascon-80pq share the same building blocks and same parameters except the size of the key, the same security for Ascon-80pq against classical attacks as for Ascon-128 are expected.

1.1 Our Contribution

We present a new partial-state-recovery conditional-cube attack on Ascon-128 and Ascon-80pq. It recovers 192 bits of 320-bit state with data complexity of $2^{44.8}$ and negligible memory and time complexity $2^{44.8}$ (in terms of 64-bit XOR operation). Very recently, a similar attack of recovering the partial state was independently developed by Baudrin et al. and published at the NIST fifth Lightweight Cryptography workshop [2].

Then, for Ascon-80pq, we extend it to a full-state-recovery attack with time complexity of 2^{128}. Furthermore, we convert the full-state-recovery attack to the key-recovery attack with extra data complexity of $2^{31.5}$, extra time complexity of $2^{129.5}$, and memory complexity of $2^{31.5}$. Our attacks do not contradict Ascon designers' security claim. Nevertheless, they are meaningful in analyzing how secure Ascon-128 and Ascon-80pq are in the nonce-misuse setting.

In Sect. 4, we provide a new conditional cube attack with data complexity $2^{44.8}$ to recover 128-bit out of 256-bit secret capacity part of Ascon-128 and Ascon-80pq. Note that 64-bit rate part is already known to the attacker. Therefore, in total, 192-bit partial state out of 320-bit will be recovered by our attack. In Sect. 5, in case of Ascon-80pq, we continue to further recover the remaining 128-bit partial capacity part and then recover the secret key.

In Table 1, we summarize exiting attacks on Ascon-128 and Ascon-80pq including our attacks, where the numbers of rounds are represented as a 3-tuple representing the number of rounds during initialization, the number of rounds during associate data or message processing, and the number of rounds during finalization, respectively.

Comparison with Generic Attack in the Nonce-Misuse Setting. We also compare our attack with the generic attack in the nonce-misuse setting. The generic state-recovery attack on Sponge-based AEAD with c-bit secret capacity is as follows: In a nonce-misuse setting, we can recover a c-bit secret capacity part with a high probability through birthday attack by making $T = 2^t$ offline queries (time complexity) and $D = 2^d$ online queries (data complexity) where $c = t + d$ and memory $M = 2^{min(t,d)}$ (memory complexity) to detect a collision event. Consider Ascon-128 and Ascon-80pq which have $c = 256$. For example, when $T = 2^{128}$, then $D = 2^{128}$ and $M = 2^{128}$; when $M = 2^{50}$, we need either $D = 2^{206}$ or $T =$

Table 1. Summary of attacks on Ascon-128 and Ascon-80pq . Note that Ascon-128 and Ascon-80pq use (12,6,12) for the 3-tuple of rounds. Note that the size of a full sate is 320 bits.

Attack type	Method	Rounds [a]	Data	Time	Memory	Nonce misuse	Ref.
Key recovery	Conditional cube	6,⋆,⋆	2^{40}	2^{40}	-	No	[3]
	Cube	7,⋆,⋆	$2^{77.2}$	$2^{103.92}$	-	No	[3]
	Cube	7,⋆,⋆	2^{64}	2^{123}	-	No	[4]
	Cube	7,5,⋆	2^{33}	2^{97}	-	Yes	[5]
	Conditional cube (only Ascon-80pq)	⋆,6,⋆	$2^{44.8}$	$2^{129.94}$	$2^{31.5}$	Yes	Sect. 5
Forgery	Cube	⋆,⋆,5	2^{17}	2^{17}	-	Yes	[5]
	Cube	⋆,⋆,6	2^{33}	2^{33}	-	Yes	[5]
Full (320-bit) State-recovery	Cube	⋆,5,⋆	2^{18}	2^{66}	-	Yes	[5]
	Conditional cube (only Ascon-80pq)	⋆,6,⋆	$2^{44.8}$	2^{128}	-	-	Sect. 5
Partial (192-bit) State-recovery	Conditional cube	⋆,6,⋆	$2^{44.8}$	$2^{44.8}$ (XOR)	-	Yes	Sect. 4

[a] ⋆ indicates that the number of rounds can be arbitrary.

2^{206}. On the other hand, our attack requires negligible memory and $D = 2^{44.8}$ to recover 128 bits out of 256-bit secret capacity in a nonce-misuse setting. Then, we can recover the remaining 128-bit secret capacity with $T = 2^{128}$.

2 Preliminaries

The bit-rate is 64 bits and the capacity is 256 bits, The state size of Ascon-128 and Ascon-80pq is 320 bits. Given a bit string X, The 10^*-padding, $pad10^*_{64}$, is defined by $pad10^*_{64}(X) = X||1||0^s$ with the least non-negative integer s such that the bit-length of $pad10^*_{64}(X)$ is a multiple of 64.

2.1 Ascon-128 and Ascon-80pq

As shown in Fig. 1, the state size is 320 bits and the initial state is $IV||K||N$, where 64-bit $IV = 0x80800c0800000000$, 128-bit secret key K, and 128-bit nonce N are for Ascon-128, and 32-bit $IV = 0xa0400c06$, 160-bit secret key K, and 128-bit nonce N are for Ascon-80pq.

For initialization and finalization phases, 12-round Ascon permutation p^{12} is used, and for processing associate data or message/ciphertext, 6-round Ascon permutation p^6 is used. The input and output sizes of the permutations are 320 bits and the bit-rate is 64 bits and the capacity is 256 bits. In order to prevent the key recovery from a state-recovery, the key is XORed into the capacity after the initialization and before the finalization. Note that the 10* padding is always applied to the message $P = P_1||P_2||\cdots||P_t$ whereas it is applied to the associate data $A = A_1||A_2||\cdots||A_s$ only when A is not the empty.

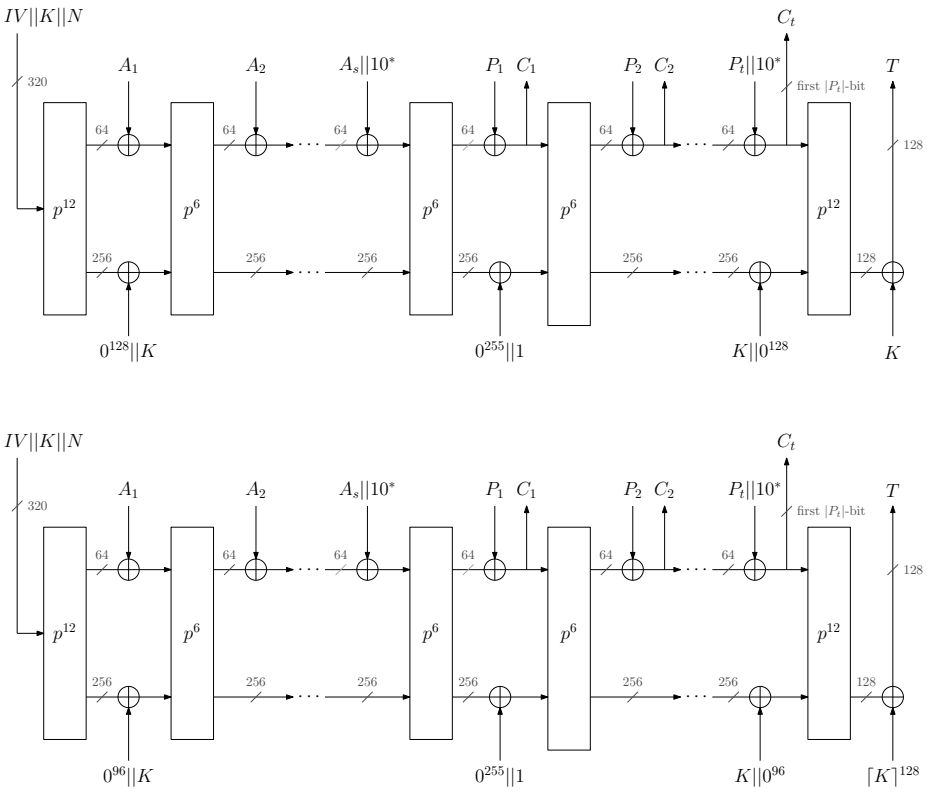

Fig. 1. Ascon-128 and Ascon-80pq (with 6-Round Encryption): in case that $|A| \neq 0$

2.2 Ascon Permutation

The round r of Ascon permutation is defined by $p = p_L \circ p_S \circ p_C$, where p_C is the constant addition with a 64-bit constant c_r at round r, p_S is the substitution layer, and p_L is the linear diffusion layer. Note that the round number r starts from zero. We denote $(S_r[0]||S_r[1]||S_r[2]||S_r[3]||S_r[4])$ as the input state of p_C or p_S at round r, and we denote $(S_{r+0.5}[0]||S_{r+0.5}[1]||S_{r+0.5}[2]||S_{r+0.5}[3]||S_{r+0.5}[4])$ as the input state of p_L at round r. And $(S_{r+1}[0]||S_{r+1}[1]||S_{r+1}[2]||S_{r+1}[3]||S_{r+1}[4])$ is denoted as the output state of p_L at round r.

Round Constant Addition p_C. Let c_r be a 64-bit constant at round r and each x_i for $0 \leq i \leq 4$ be 64 bits.

$$p_C : (x_0||x_1||x_2||x_3||x_4) \mapsto (x_0||x_1||(x_2 \oplus c_r)||x_3||x_4) \text{ for round } r$$

Each c_r at round r is defined as shown in Fig. 2. Note that each c_r is differently defined depending on the number of rounds of Ascon permutation.

Since the first 56 bits of each constant c_r are all zeros, the 320-bit state after the constant addition remains same except 8 bits as shown in Fig. 3.

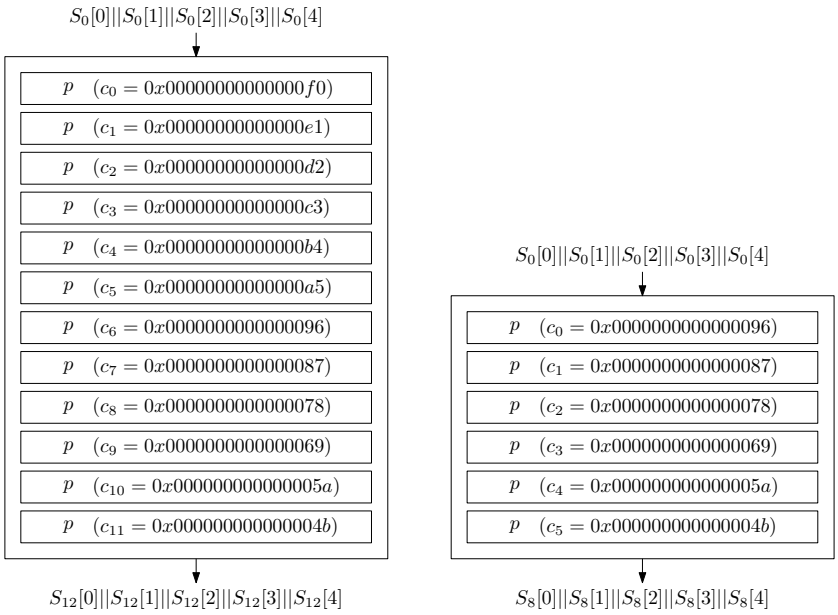

Fig. 2. 12-Round Ascon Permutation (Left) and 6-Round Ascon Permutation (Right)

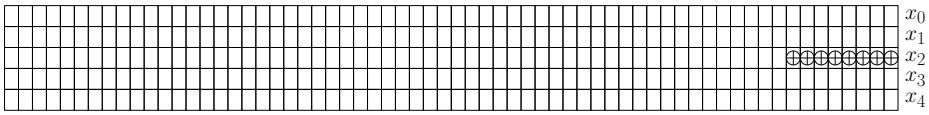

Fig. 3. Round Constant Addition p_C

Substitution Layer p_S with 5-bit S-box $\mathcal{S}(x)$. The same 5-bit S-box $\mathcal{S}(x)$ is applied to each column in parallel.

$$\mathcal{S} : (x_0||x_1||x_2||x_3||x_4) \mapsto (y_0||y_1||y_2||y_3||y_4)$$

where

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

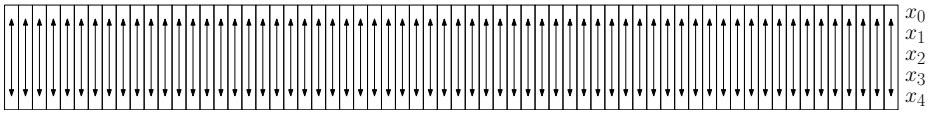

Fig. 4. Substitution Layer p_S

Linear Diffusion Layer p_L with 64-bit Diffusion Functions $\Sigma_i(x_i)$. x_i at row i is updated with $\Sigma_i(x_i)$.

$$\begin{aligned}
\Sigma_0(x_0) &= x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)\\
\Sigma_1(x_1) &= x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)\\
\Sigma_2(x_2) &= x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)\\
\Sigma_3(x_3) &= x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)\\
\Sigma_4(x_4) &= x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}$$

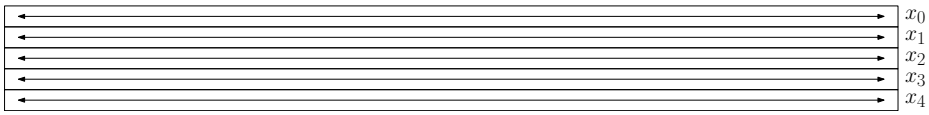

Fig. 5. Linear Diffusion Layer p_L

2.3 cube-sum

We follow the notations in [6]. Given a monomial t $(=x_1x_2\cdots x_k)$ with k variables, $C_t :=\{(0,...,0,0),(0,...,0,1),(0,...,1,0),(0,...,1,1),....,(1,...,1,1)\}$, which is the set of all binary vectors of the length k, is called the cube with k cube variables.

Given a polynomial $f(x_1,\cdots,x_k,y_1,\cdots,y_l)$ with $k+l$ variables, let us consider the situation that f is divided by $x_1x_2\cdots x_k$. f can described with the quotient $Q(y_1,\cdots,y_l)$ and the remainder $R(x_1,\cdots,x_k,y_1,\cdots,y_l)$ as follows:

$$f(x_1,\cdots,x_k,y_1,\cdots,y_l) = x_1x_2\cdots x_k \cdot Q(y_1,\cdots,y_l) + R(x_1,\cdots,x_k,y_1,\cdots,y_l)$$

Note that the quotient Q is depends only on y_i's.

Give a polynomial f, Xuejia Lai [7] proved the following relation between cube $C_{x_1x_2\cdots x_k}$ and quotient $Q(y_1,\cdots,y_l)$.

$$\sum_{(x_1,\cdots,x_k)\in C_{x_1x_2\cdots x_k}} f(x_1,\cdots,x_k,y_1,\cdots,y_l) = Q(y_1,\cdots,y_l)$$

Let us say that given f, the left side is the *cube-sum* with cube $C_{x_1x_2\cdots x_k}$. The relation shows that the cube-sum is equal to the quotient $Q(y_1,\cdots,y_l)$. If

$Q(y_1, \cdots, y_l)$ is zero, then the cube-sum should be zero. For example, given f of degree 63, the cube-sum with 64 cube variables should be zero. For another example, given f of degree 64, the cube-sum with 64 cube variables is either zero or one depending on whether there is a nominal of 64 cube variables or not.

2.4 Conditional Cube Attacks

Huang et al. [8] introduced the concept of *conditional cube attacks*. They considered two sets of cube variables: $\{v_1, ..., v_p\}$ and $\{u_1, ..., u_q\}$, where p and q are non-negative integers. Note that the starting round number is 0 and each cube variable indicates a bit of an input state of the underlying permutation or function. If there are *conditions* such that the following three requirements are satisfied *only when all the conditions are true*, then $\{v_1, ..., v_p\}$ is called a set of *conditional* cube variables and $\{u_1, ..., u_q\}$ is called a set of *ordinary* cube variables. In other words, if one of the conditions is not true, then one of three requirements will not be satisfied. Huang et al. considered a case that the degree of each round function is 2, which is same for Ascon-128a.

- *Requirement 1*: After round 0, there is no multiplication of two different variables in $\{v_1, ..., v_p, u_1, ..., u_q\}$. In other words, after round 0, there is no $v_i v_j$ $(i \neq j)$, no $u_i u_j$ $(i \neq j)$, and no $v_i u_j$.
- *Requirement 2*: After round 1, there is no $v_i v_j$ $(i \neq j)$.
- *Requirement 3*: After round 1, there is no $v_i u_j$.

In [8], Huang et al. proved Theorem 1.

Theorem 1. *[8] If there are p conditional cube variables v_1,...,v_p and q ordinary cube variables u_1,...,u_q, and all the related conditions are true to make the three above requirements satisfied, where*

1. $p, q \geq 1$, $n \geq 0$, *and* $q = 2^{n+1} - 2p + 1$ *or*
2. $q = 0$, $n \geq 0$, *and* $p = 2^n + 1$,

then the term $v_1 \cdots v_p u_1 \cdots u_q$ of degree p+q will not appear and terms of at most degree $p + q - 1$ can appear in the output polynomials of (n+2)-round function, where the degree of each round is 2. (End of Theorem 1)

In order to use Theorem 1 for key or state recovery attacks, we need another assumption that when any of the related conditions is not true, the term $v_1 \cdots v_p u_1 \cdots u_q$ appear with high probability in the output polynomials of $(n+2)$-round function. With this assumption we can identify whether all the conditions are true or not by checking if all the cube-sums corresponding output bits with given cube variables are zero after $n+2$ rounds. For example, in case of Ascon-128 or Ascon-pq80, we know the first 64-bit of 6-round Ascon-permutation during the encryption by XORing 64-bit plaintext block P_i and ciphertext block C_i as shown Fig. 7. In our conditional attack with $p = 1$ and $q = 31$, we assume that the probability that all the cube-sums corresponding 64-bit output bits with given cube variables are zero is negligible when one of conditions are false.

3 Pattern-A$^{\ggg i}$

3.1 Cube Variables

There are 40 cube variables: $v_0, v_1, \cdots, v_{27}$ and $w_1, w_2, \cdots, w_{12}$. These cube variables are divided into two sets: (1) a set of conditional cube variables = $\{v_0\}$ and (2) a set of ordinary cube variables = $\{v_1, \cdots, v_{27}, w_1, \cdots, w_{12}\}$. We set $S_0[0][i] = v_0$ and ordinary cube variables are set as described in Table 2.

Table 2. The Setting of Ordinary Cube Variables of Pattern-A$^{\ggg i}$. We apply mod 64 for each column number.

Column	Setting	Column	Setting	Column	Setting
$1+i$	$S_0[0][1+i] = v_1$	$21+i$	$S_0[0][21+i] = w_5$	$43+i$	$S_0[0][43+i] = v_{18}$
$2+i$	$S_0[0][2+i] = w_1$	$23+i$	$S_0[0][23+i] = w_6$	$45+i$	$S_0[0][45+i] = w_{10}$
$4+i$	$S_0[0][4+i] = v_2$	$24+i$	$S_0[0][24+i] = v_{10}$	$47+i$	$S_0[0][47+i] = w_{11}$
$5+i$	$S_0[0][5+i] = v_3$	$26+i$	$S_0[0][26+i] = v_{11}$	$48+i$	$S_0[0][48+i] = v_{19}$
$6+i$	$S_0[0][6+i] = v_4$	$27+i$	$S_0[0][27+i] = v_{12}$	$49+i$	$S_0[0][49+i] = v_{20}$
$7+i$	$S_0[0][7+i] = v_5$	$29+i$	$S_0[0][29+i] = w_7$	$50+i$	$S_0[0][50+i] = v_{21}$
$8+i$	$S_0[0][8+i] = v_6$	$30+i$	$S_0[0][30+i] = v_{13}$	$52+i$	$S_0[0][52+i] = v_{22}$
$9+i$	$S_0[0][9+i] = w_2$	$32+i$	$S_0[0][32+i] = w_8$	$56+i$	$S_0[0][56+i] = v_{23}$
$12+i$	$S_0[0][12+i] = w_3$	$34+i$	$S_0[0][34+i] = v_{14}$	$57+i$	$S_0[0][57+i] = w_{12}$
$14+i$	$S_0[0][14+i] = v_7$	$36+i$	$S_0[0][36+i] = w_9$	$58+i$	$S_0[0][58+i] = v_{24}$
$15+i$	$S_0[0][15+i] = v_8$	$37+i$	$S_0[0][37+i] = v_{15}$	$59+i$	$S_0[0][59+i] = v_{25}$
$16+i$	$S_0[0][16+i] = v_9$	$38+i$	$S_0[0][38+i] = v_{16}$	$60+i$	$S_0[0][60+i] = v_{26}$
$18+i$	$S_0[0][18+i] = w_4$	$41+i$	$S_0[0][41+i] = v_{17}$	$63+i$	$S_0[0][63+i] = v_{27}$

3.2 Conditions

As shown in Table 3, we define the 13 conditions of Pattern-A$^{\ggg i}$ required for the requirements 1-3 in Sect. 2.4.

Table 3. 13 Conditions of Pattern-A$^{\ggg i}$. We apply mod 64 for each column number.)

Column	Conditions	Count	Column	Conditions	Count
i	$S_0[1][i] = 0$	1	$29+i$	$S_0[3][29+i] = S_0[4][29+i] + 1$	1
$2+i$	$S_0[3][2+i] = S_0[4][2+i] + 1$	1	$32+i$	$S_0[1][32+i] = 0$	1
$9+i$	$S_0[3][9+i] = S_0[4][9+i] + 1$	1	$36+i$	$S_0[1][36+i] = 1$	1
$12+i$	$S_0[1][12+i] = 0$	1	$45+i$	$S_0[1][45+i] = 1$	1
$18+i$	$S_0[3][18+i] = S_0[4][18+i] + 1$	1	$47+i$	$S_0[3][47+i] = S_0[4][47+i] + 1$	1
$21+i$	$S_0[1][21+i] = 0$	1	$57+i$	$S_0[1][57+i] = 0$	1
$23+i$	$S_0[1][23+i] = 0$	1			

3.3 Propagation of Cube Variables

It is easy to check that the requirements 1 and 2 are always satisfied regardless of whether all the conditions are true or not, because *(i)* in the beginning of round 0, each column uses only one cube variable and nonlinear S-box is applied column-wise in parallel and *(ii)* there is only one conditional cube variable. And, through Lemma 1, we are going to show that the requirement 3 is satisfied only when all the conditions are true.

Lemma 1 shows that that if any of the column-i conditions ($0 \leq i \leq 63$) is not true, then there is a multiplication either between v_0 and v_j for some j ($1 \leq j \leq 27$) or between v_0 and w_j for some j ($1 \leq j \leq 12$) after 2 rounds.

Lemma 1. *[Requirement 3 is satisfied only when all the conditions are true.] When all the conditions in Table 3 are true, there is no multiplication between v_0 and v_j for any j ($1 \leq j \leq 27$), and between v_0 and w_j for any j ($1 \leq j \leq 12$) after 2 rounds. When at least one condition in Table 3 is false, then there is a multiplication either between v_0 and v_j for some j ($1 \leq j \leq 27$) or between v_0 and w_j for some j ($1 \leq j \leq 12$) after 2 rounds.*

Proof. Without loss of generality, for example, in Fig. 6, in case of $\texttt{Pattern-A}^{\ggg i}$ with $i = 0$ we can see the propagation of cube variables at round 0. As shown in columns #0, #10, #17, #19, #28, #39, #61 of the state S_1 of Fig. 6, there will be no multiplication between v_0 and v_j for any j ($1 \leq j \leq 27$), and between v_0 and w_j for any j ($1 \leq j \leq 12$) after 2 rounds. Let us consider the next case when at least one condition in Table 3 is false. The next case can be divided into two sub-cases as follows:

- **Subcase 1:** when one of the column-0 conditions is not true, there is a multiplication between v_0 and v_j for some j ($1 \leq j \leq 27$) at round 1.
- **Subcase 2:** When all of the column-0 conditions are true, if any of the column-j conditions ($1 \leq j \leq 63$) is not true, then there is a multiplication between v_0 and w_l for some l ($1 \leq l \leq 12$) at round 1.

For these two subcases, please see Appendix A for its proof. □

4 Partial State Recovery Attack

Our attack is in nonce-misuse setting, where IV can be repeated for the encryption. Once nonce IV and associate data A are fixed, as shown in Fig. 7, the 256-bit secret state ($S_0[1]$,$S_0[2]$,$S_0[3]$,$S_0[4]$), which is the capacity part in the beginning of the encryption phase, will be automatically determined. Our first attack target is to recover 128-bit information ($S_0[1]$,$S_0[3] + S_0[4]$) of the 256-bit secret state in the beginning of the encryption phase by using $\texttt{Pattern-A}^{\ggg i}$'s. And, in Sect. 5, our second target is to recover the remaining 128-bit secret information and the key K.

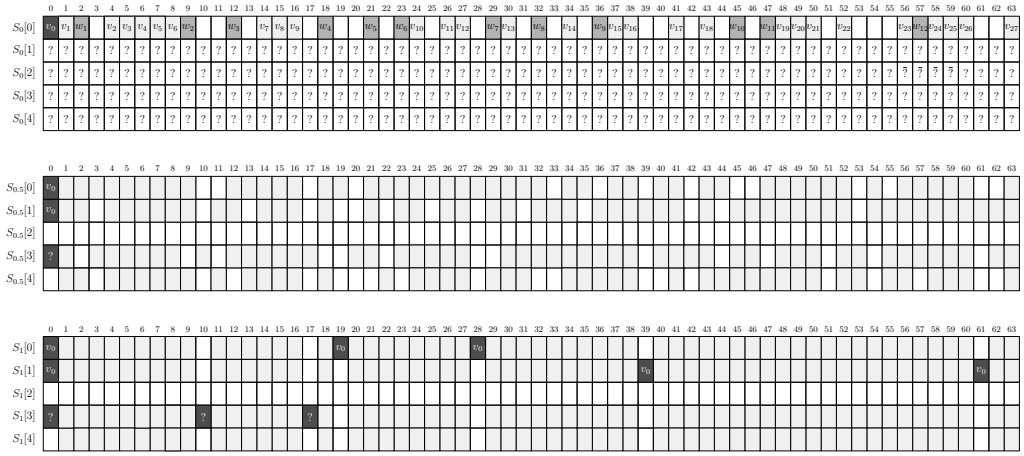

Fig. 6. `Pattern-A`$^{\ggg i}$ with $i = 0$ (a Propagation of Cube Variables in Round 0): Bits influenced by the conditional cube variable, v_0, are highlighted with block color, and bits influenced by ordinary cube variables, v_l's and w_j's ($1 \leq l \leq 27$ and $1 \leq j \leq 12$), are highlighted with gray color.

Step 1. Let us call `Pattern-A`$^{\ggg i}$ with $i = 0$ as `Pattern-A`. We start with `Pattern-A`. We let P_1 :=(v_0, v_1, w_1, 0, v_2, v_3, v_4, v_5, v_6, w_2, 0, 0, w_3, 0, v_7, v_8, v_9, 0, w_4, 0, 0, w_5, 0, w_6, v_{10}, 0, v_{11}, v_{12}, 0, w_7, v_{13}, 0, w_8, 0, v_{14}, 0, w_9, v_{15}, v_{16}, 0, 0, v_{17}, 0, v_{18}, 0, w_{10}, 0, w_{11}, v_{19}, v_{20}, v_{21}, 0, v_{22}, 0, 0, 0, v_{23}, w_{12}, v_{24}, v_{25}, v_{26}, 0, 0, v_{27}).

In order to use Theorem 1, our attack uses 1 conditional cube variable, v_0, and 31 ordinary cube variables by selecting all 27 v_i's ($1 \leq i \leq 27$) and w_1, w_2, w_3, and w_4.

Then, we choose any nonce N and any associate data A and we initialize 64-bit plaintext block P_1 as zeros except the bit-positions corresponding 32 selected cube variables, where we can see N, A, and P_1 in Fig. 7. We will perform the online 128-bit cubesum on $P_2 \oplus C_2$ by making 2^{32} encryption-queries (N, A, P_1, P_2), where P_1 is defined according to the 32 selected cube variables, P_2 is a fixed 128-bit, and N and A are fixed for all the 2^{32} encryption-queries.

If 64-bit online cubesum are not all zeros, we have to continue to search different nonce N and associate data A such that the 64-bit online cubesum are all zeros. The expected online data complexity is 2^{37} to find such an all-zero cubesum, because there are 32 cube variables and five conditions related to v_0, w_1, w_2, w_3, and w_4.

After we find a nonce N, associate data A such that the 64-bit online cubesum with 32 cube variables, v_i's ($0 \leq i \leq 27$) and w_1, w_2, w_3, and w_4, are all zeros, we move to Step 2. After completing Step 1, we know that 5 conditions are true, which mean that we recovered 5-bit information out of 256-bit secret capacity part ($S_0[1]$,$S_0[2]$,$S_0[3]$,$S_0[4]$) in Fig. 7.

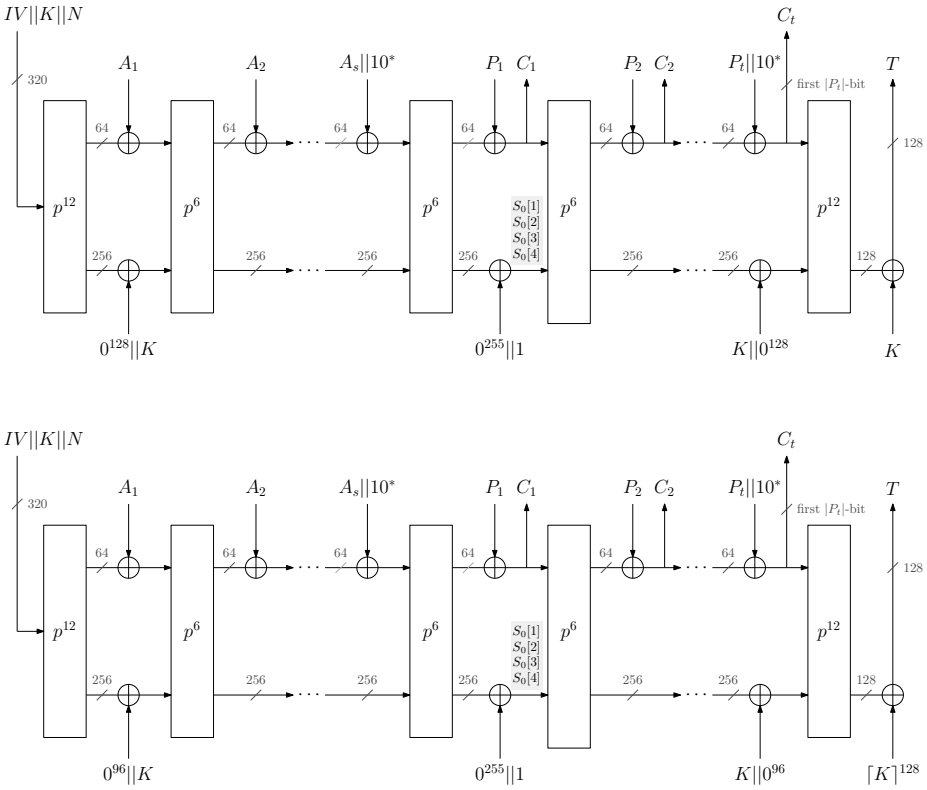

Fig. 7. Attack Strategy on Ascon-128 and Ascon-80pq : in case that $|A| \neq 0$

Step 2. We will use the same pattern, `Pattern-A`, used in Step 1. From Step 1, assume that with (N, A) we found 5 true conditions corresponding to v_0, w_1, w_2, w_3, and w_4. As shown in Table 3, there are 13 conditions of `Pattern-A`, where $i = 0$. So, in Step 2, we continue to check whether each of the remaining 8 conditions is true or not one by one with help of Algorithm 1. The data complexity of Step 2 is 2^{35}, because we need to perform the cubesum 8 times with 32 cube variables.

Step 3. In Step 3, we want to show how to recover more secret bits by using `Pattern-A`$^{\ggg i}$'s. Note that nonce N and associate data A are already fixed in Step 1. In order to use `Pattern-A`$^{\ggg i}$, we need that $S_0[1][i] = 0$, which is corresponding to the conditional cube variable v_0, is true. Since an attacker does not know the actual value of $S_0[1][i]$ and the value of $S_0[1][i]$ is assumed to be randomly selected, we expect that on average 32 patterns out of 64 `Pattern-A`$^{\ggg i}$'s can be used. From Step 1 and Step 2, we already know $S_0[1][12] = 0$ so we can use `Pattern-A`$^{\ggg 12}$. Additionally, from Step 1 and Step 2, we know the values of $S_0[1][i]$'s for $i = 12, 21, 23, 32, 36, 45, 57$. We expect that on average at least 3 out

Algorithm 1 Step 2.

Input: `Pattern-A`, (N, A)

Output: The correctness of the condition corresponding to each w_j, where $j \in \{5, \cdots, 12\}$

1: Initialize 64-bit plaintext block P_1 as explained in Step 1 and P_2 as any 64-bit value.
2: **for** $j = 5$ to 12 **do**
3: With (N, A, P_1, P_2) perform 64-bit online cubesum with all 28 v_i's and w_1, w_2, w_3, and w_j.
4: If the 64-bit cubesum are all zeros, then the condition corresponding to w_j is true, otherwise false.
5: **end for**

of 7 values are zero. We can use $\texttt{Pattern-A}^{\ggg i}$ to recover more secret bits whenever $S_0[1][i] = 0$. As an example, let me explain how we can use $\texttt{Pattern-A}^{\ggg i_1}$, where $S_0[1][i_1] = 0$, to recover more secret bits in a similar way of Step 1 and Step 2. We let P_1 be the i_1-bit right-rotation of (v_0, v_1, w_1, 0, v_2, v_3, v_4, v_5, v_6, w_2, 0, 0, w_3, 0, v_7, v_8, v_9, 0, w_4, 0, 0, w_5, 0, w_6, v_{10}, 0, v_{11}, v_{12}, 0, w_7, v_{13}, 0, w_8, 0, v_{14}, 0, w_9, v_{15}, v_{16}, 0, 0, v_{17}, 0, v_{18}, 0, w_{10}, 0, w_{11}, v_{19}, v_{20}, v_{21}, 0, v_{22}, 0, 0, 0, v_{23}, w_{12}, v_{24}, v_{25}, v_{26}, 0, 0, v_{27}).

In order to use Theorem 1, our attack uses 1 conditional cube variable, v_0, and 31 ordinary cube variables by selecting all 27 v_i's ($1 \le i \le 27$ and $i \neq i_1$) and selecting w_{j_1}, w_{j_2}, w_{j_3}, and w_{j_4} from $\{w_1, \cdots, w_{12}\}$, where $1 \le j_1, j_2, j_3, j_4 \le 12$.

Then, we initialize 64-bit plaintext block P_1 as zeros except the bit-positions corresponding 32 selected cube variables. We will perform the online 64-bit cubesum on $P_2 \oplus C_2$ by making 2^{32} encryption-queries (N, A, P_1, P_2), where P_1 is defined according to the 32 selected cube variables, P_2 is a fixed 128-bit, and N and A are already determined from Step 1 and 2. With at most 2^{36} data complexity we will continue to search different j_1, j_2, j_3, and j_4, such that the 64 bits of the cubesum are all zeros, which means that all 4 conditions corresponding to selected 4 w_j's, denoted by $w_{j_1^*}$, $w_{j_2^*}$, $w_{j_3^*}$, and $w_{j_4^*}$, are considered to be true.

Step 4. As shown in Table 3, there are 13 conditions of $\texttt{Pattern-A}^{\ggg i_1}$, where i_1 is defined from Step 3. And from Step 3, we know that 5 out of 13 conditions are true. So, in Step 4, we continue to check whether each of the remaining 8 conditions is true or not one by one. Algorithm 2 shows the detailed procedure of Step 4. The data complexity of Step 4 is 2^{35}, because we need to perform the online cubesum 8 times with 32 cube variables.

Total Data Complexity. We can repeat Step 3 and Step 4 with different $\texttt{Pattern-A}^{\ggg i}$'s, where $S_0[1][i] = 0$, in order to recover more secret bits. By software simulation, we found that on average 37 patterns are required to recover 128-bit information $(S_0[1], S_0[3] + S_0[4])$. However, as explained earlier, on average 32 patterns can be utilized due to the condition assigned to the column corresponding to the conditional cube variable v_0. When we repeat Step

Algorithm 2 Step 4.

Input: $\texttt{Pattern-A}^{\ggg i_1}$, $\{j_1^*, j_2^*, j_3^*, j_4^*\}$, (N, A)
Output: The correctness of the condition corresponding to each w_j, where $j \in \{1, \cdots, 12\} \setminus \{j_1^*, j_2^*, j_3^*, j_4^*\}$

1: Initialize 64-bit plaintext block P_1 as explained in Step 3 and P_2 as any 64-bit value.
2: Let $J := \{j_1, j_2, \cdots, j_8\} = \{1, \cdots, 12\} \setminus \{j_1^*, j_2^*, j_3^*, j_4^*\}$
3: **for** $l = 1$ to 8 **do**
4: With (N, A, P_1, P_2) perform 64-bit online cubesum with all 28 v_i's and $w_{i_1^*}$, $w_{i_2^*}$, $w_{i_3^*}$, and w_{j_l}.
5: If the 64-bit cubesum are all zeros, then the condition corresponding to w_{j_l} is true, otherwise false.
6: **end for**

3 and Step 4 for 31 times, according to our simulation, the average number of bits that can be recovered with 32 patterns is about 126 bits out of 128-bit $(S_0[1], S_0[3] + S_0[4])$, and about the chance of 12% we can fully recover 128-bit $(S_0[1], S_0[3] + S_0[4])$. Therefore, after we repeat Step 1 $\sim$ Step 4 for at most nine times, we can recover 128-bit $(S_0[1], S_0[3] + S_0[4])$.

In order to recover 128-bit $(S_0[1], S_0[3] + S_0[4])$, Step 1 is repeated 9 times, Step 2 is repeated 9 times, Step 3 is repeated 9×31 times, and Step 4 is repeated 9×31 times. Since Step 1 requires 2^{37} data, Step 2 requires 2^{35} data, Step 3 requires 2^{36} data, Step 4 requires 2^{35} data, the total data complexity is $2^{44.78}$ $(=9 \times (2^{37} + 2^{35} + 31 \times (2^{36} + 2^{35})))$.

5 Full-State and Key Recovery Attacks on Ascon-80pq

In the previous section, we recovered 128-bit secret information ($S_0[1]$,$S_0[3] + S_0[4]$) out of 256-bit secret capacity ($S_0[1]$,$S_0[2]$,$S_0[3]$,$S_0[4]$) in Fig. 7. In order to recover the remaining secret 128-bit, we can exhaustively search 2^{128} cases which is not effective for Ascon-128, because Ascon-128 uses 128-bit secret key. On the other hand, for Ascon-80pq we can perform the exhaustive search on 2^{128} cases to recover the full-state of Ascon-80pq. Note that Ascon-80pq uses 160-bit secret key. Though the designers claims the security of Ascon-80pq is 128-bit, it is still interesting how much Ascon-80pq can resist against nonce-misuse attacks.

Next, we are going to show how to perform a key-recovery attack on Ascon-80pq after a full-state is recovered. The key-recovery attack consists of two phases, online and offline. Note that the attack works in a nonce-misuse setting.

Online Phase. See Fig. 8. After a state is recovered, it is easy to compute the following states up to the input state of the finalization phase. The first 63 bits of the input-state can be fixed by choosing the last block of message. Note that 10^*-padding is always applied to the last block of message, so at most 63 bits can be easily controlled by the attacker. The remaining 257 bits are

out of control. So, the attacker can get $2^{31.5}$ input-states, $\{(X, Y_i, Z)\}_{1 \le i \le 2^{31.5}}$, of the finalization with $2^{128.5}$ time complexity, where X and Z are arbitrary fixed 64-bit and 96-bit values, respectively, and each Y_i can be any value of 160 bits. And then through $2^{31.5}$ online queries, for each i the attacker can obtain T_i corresponding to (X, Y_i, Z) and store $\{(X, Y_i, Z, T_i)\}_{1 \le i \le 2^{31.5}}$ with $2^{31.5}$ memory complexity.

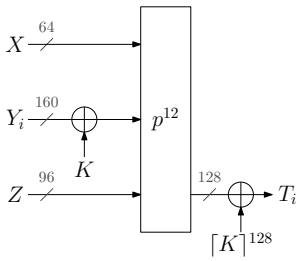

Fig. 8. Online

Offline Phase. See Fig. 9. Given X and Z which were computed from the online phase, the attacker randomly chooses 160-bit V and computes W and check if (1) there is i such that $\lceil Y_i \rceil^{128} \oplus T_i = \lceil V \rceil^{128} \oplus W$ and (2) a 160-bit guessed secret key $K = Y_i \oplus V$ is correct by checking input-output pair of Ascon-80pq. If there is no such i, then the attacker repeats to randomly choose V till the attacker finds i satisfying (1) and (2). We expect that the attacker can find i and the correct key K with $2^{128.5}$ time complexity without an additional memory.

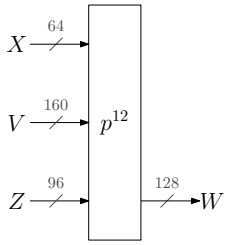

Fig. 9. Offline

Therefore, once a state is recovered, with a high probability the attacker can recover the key with $2^{31.5}$ data complexity, $2^{129.5}$ ($\approx 2^{128.5} + 2^{128.5}$) time complexity, and $2^{31.5}$ memory complexity.

6 Conclusion

In this paper, we described how a 128-bit partial secret state $(S_0[1], S_0[3]+S_0[4])$ of secret capacity can be recovered with a conditional cube attack. A future work would be to (i) efficiently recover the remaining 128-bit partial secret state and (ii) then perform the key recovery attack on Ascon-128 in a similar way explained in Sect. 5.

References

1. C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. Ascon. Submission to the NIST Lightweight Cryptography Standardization Process, 2021. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf.
2. Jules Baudrin, Anne Canteaut, and Léo Perrin. Practical cube-attack against nonce-misused Ascon. NIST Lightweight Cryptography Workshop 2022, May 2022.
3. Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional Cube Attack on Round-Reduced ASCON. *IACR Transactions on Symmetric Cryptology*, 2017(1):175–202, Mar. 2017.
4. Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-free key-recovery and distinguishing attacks on 7-round ascon. *IACR Trans. Symmetric Cryptol.*, 2021(1):130–155, 2021.
5. Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. Cryptanalysis of round-reduced ASCON. *Sci. China Inf. Sci.*, 60(3):38102, 2017.
6. 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, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 733–761. Springer, 2015.
7. Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard E. BlahutDaniel J. CostelloJr.Ueli MaurerThomas Mittelholzer, editor, *Communications and Cryptography. The Springer International Series in Engineering and Computer Science (Communications and Information Theory)*, volume 276, pages 227–233. Springer, 1994.
8. 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, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 259–288, 2017.

Appendix A. Proof of Lemma 1

We are going to show the cases with $i = 0$, because pattern-A$^{\ggg i}$ is rotation-invariant.

Subcase 1.

We need to prove that when one of the column-0 conditions is not true, there is a multiplication between v_0 and v_i for some i $(1 \leq i \leq 63)$ at round 1. There is one condition on the column-0: $S_0[1][0] = 0$.

Let us check what would happen if $S_0[1][0] = 0$ is not true. Note that $S_0[0][0] = v_0$. And the second and fifth outputs y_1 and y_4 of S-box are defined as follows:

$$y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0,$$
$$y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1.$$

Therefore, when we compute the column-0's second and fifth outputs y_1 and y_4 of S-box at round 0, $x_0 = v_0$ and $x_1 = 1$. So, y_1 and y_4 are always influenced by v_0. So, $S_{0.5}[1][0]$ and $S_{0.5}[4][0]$ are marked with v_0 in Fig. 10.

Note that $S_0[0][7] = v_5$. And the second output y_1 of S-box is defined as follows:

$$y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0$$

Therefore, when we compute the column-7's second output y_1 of S-box at round 0, $x_0 = v_5$ and y_1 is always influenced by v_5. So, $S_{0.5}[1][5]$ is marked with v_5 in Fig. 10. Similarly, $S_{0.5}[1][41]$ is marked with v_{17} in Fig. 10.

And then row-1 and row-4 are updated as follows:

$$\Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$\Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

By Σ_1, $S_1[1][0]$, $S_1[1][39]$, and $S_1[1][61]$ are always influenced by v_0. By Σ_4, $S_1[4][0]$, $S_1[4][7]$, and $S_1[4][41]$ are always influenced by v_0. By Σ_1, $S_1[1][7]$ is always influenced by v_5 and $S_1[1][41]$ is always influenced by v_{17}. As highlighted with a red circle, since $S_1[1][7]$ and $S_1[4][7]$, and $S_1[1][41]$ and $S_1[4][41]$ are multiplied via S-box at round 1, there will be v_0v_5 and v_0v_{17} after round 1.

Subcase 2.

As we analyzed for Subcase 1, Subcase 2 can be similarly analyzed as shown Fig. 11 $\sim$ Fig. 22.

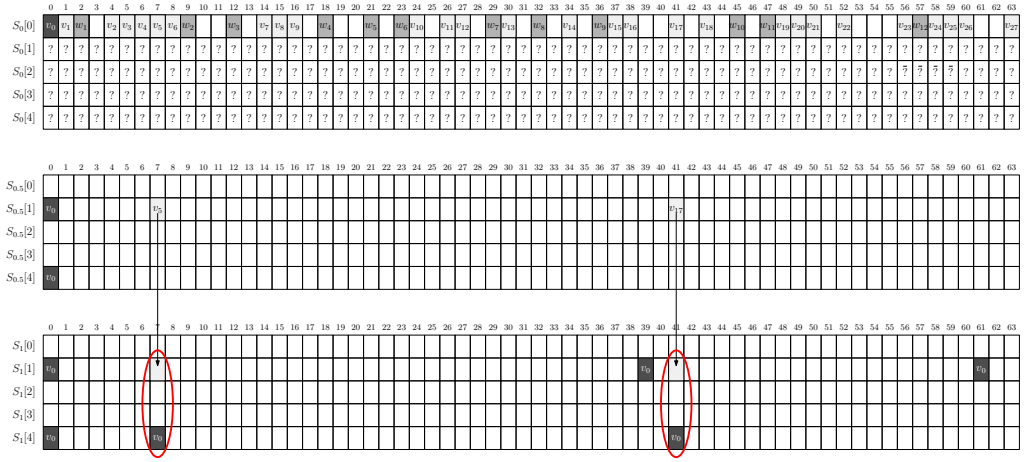

Fig. 10. Subcase 1. one of the column-0 conditions of S_0 is not true: $S_0[1][0] = 0$ is not true

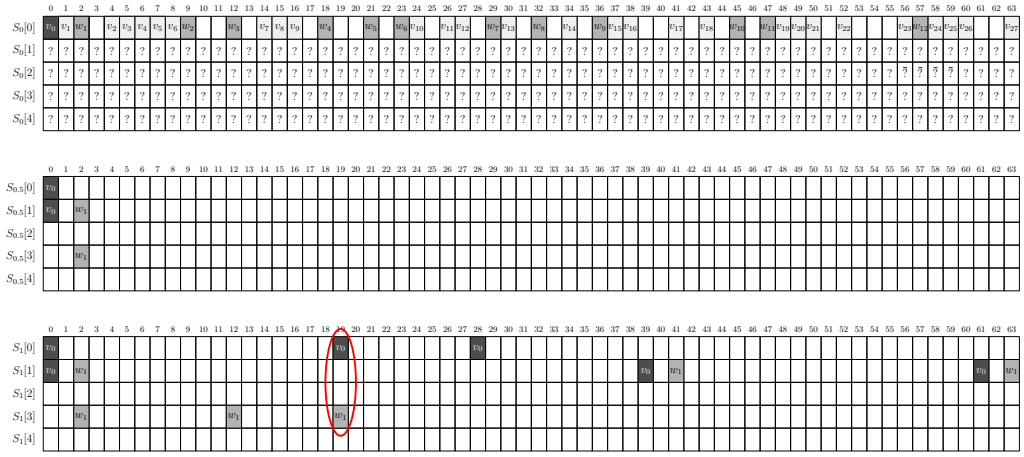

Fig. 11. Subcase 2. one of the column-2 conditions of S_0 is not true: $S_0[3][2] = S_0[4][2]+1$ is not true

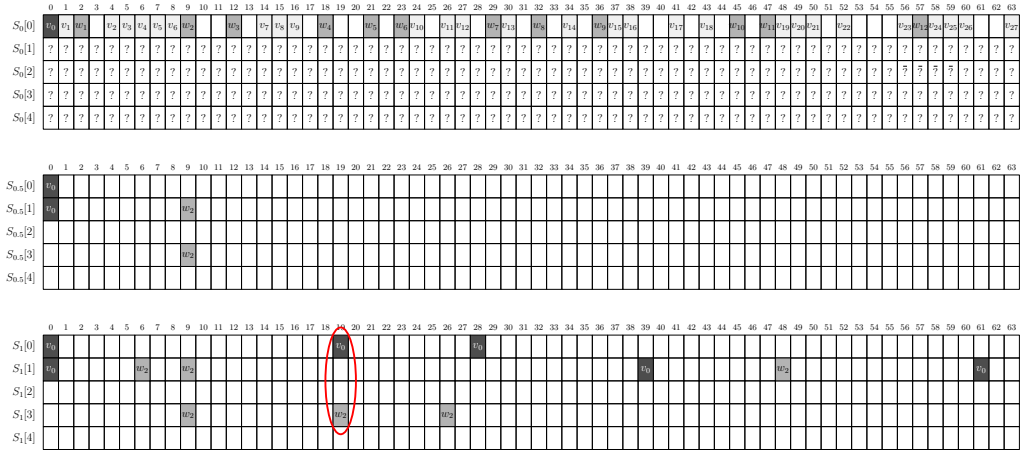

Fig. 12. Subcase 2. one of the column-9 conditions of S_0 is not true: $S_0[3][9] = S_0[4][9]+1$ is not true

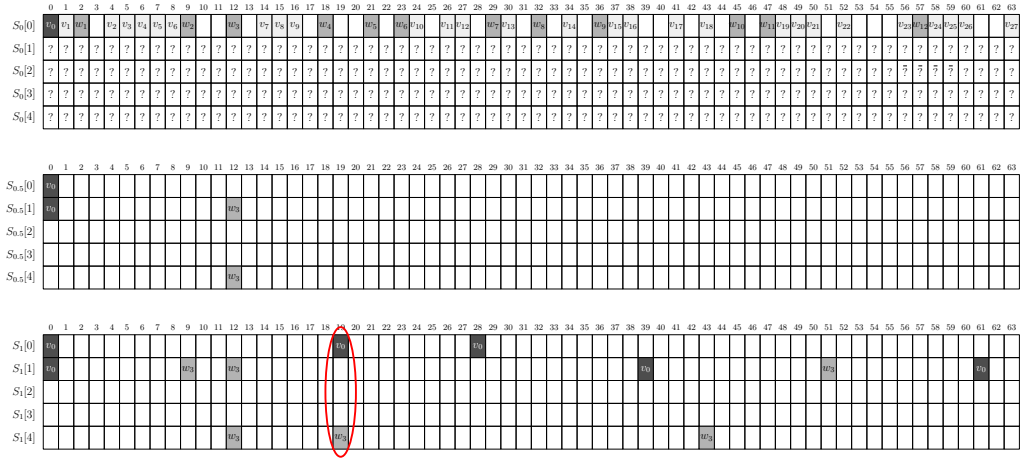

Fig. 13. Subcase 2. one of the column-12 conditions of S_0 is not true: $S_0[1][12] = 0$ is not true

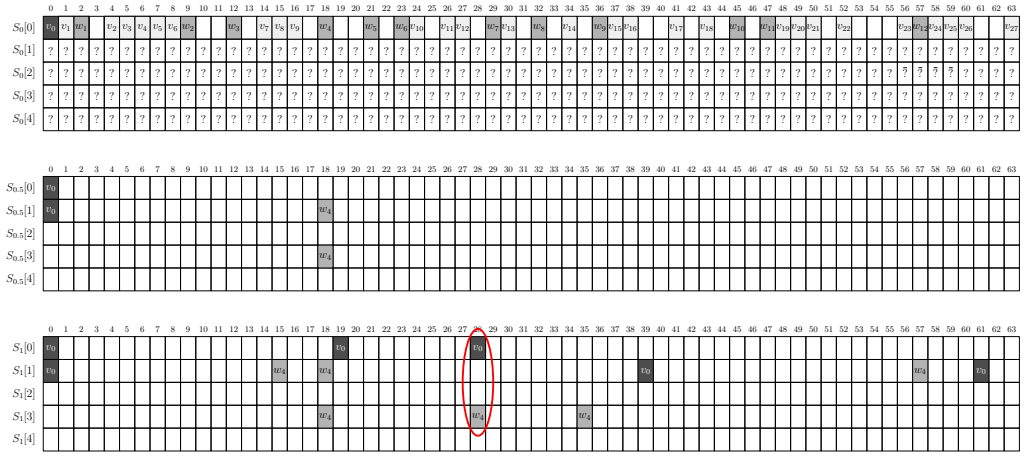

Fig. 14. Subcase 2. one of the column-18 conditions of S_0 is not true: $S_0[3][18] = S_0[4][18] + 1$ is not true

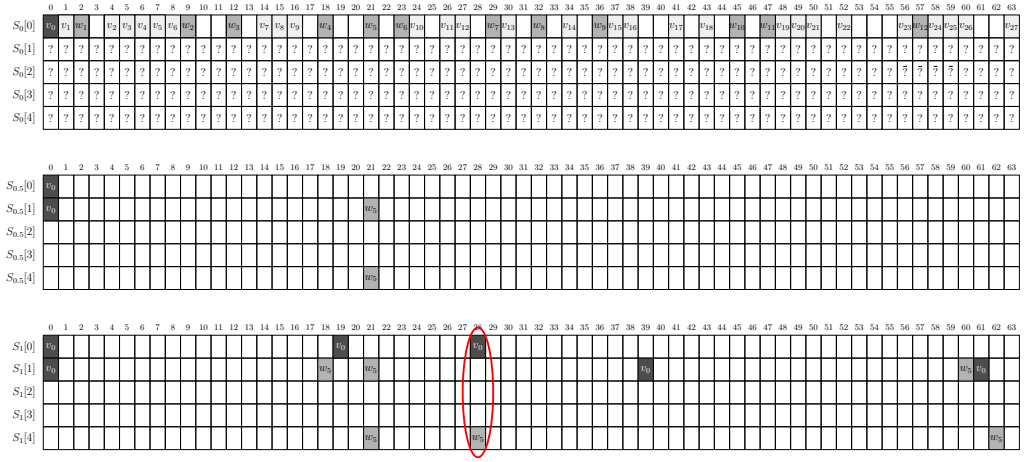

Fig. 15. Subcase 2. one of the column-21 conditions of S_0 is not true: $S_0[1][21] = 0$ is not true

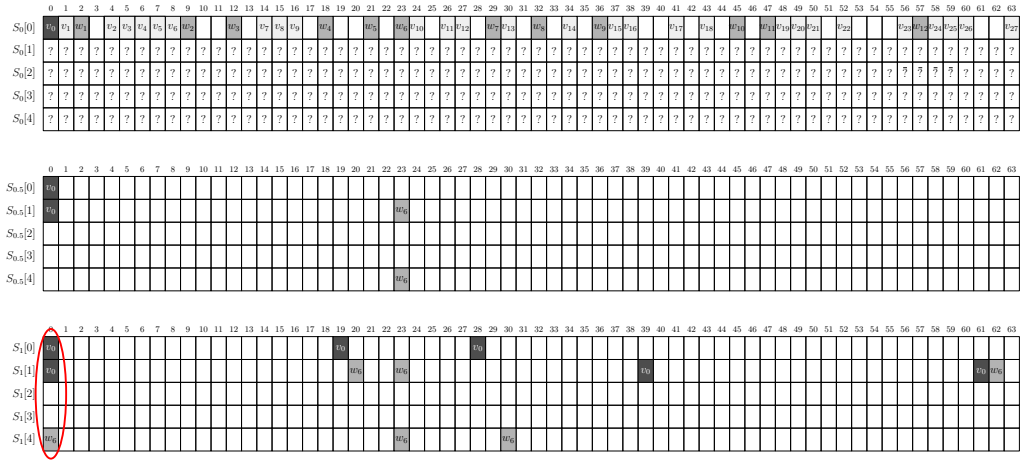

Fig. 16. Subcase 2. one of the column-23 conditions of S_0 is not true: $S_0[1][23] = 0$ is not true

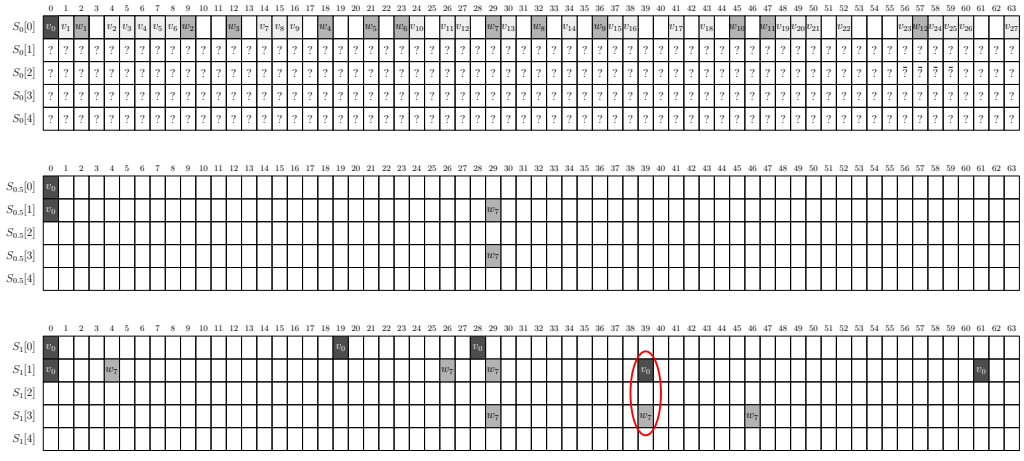

Fig. 17. Subcase 2. one of the column-29 conditions of S_0 is not true: $S_0[3][29] = S_0[4][29] + 1$ is not true

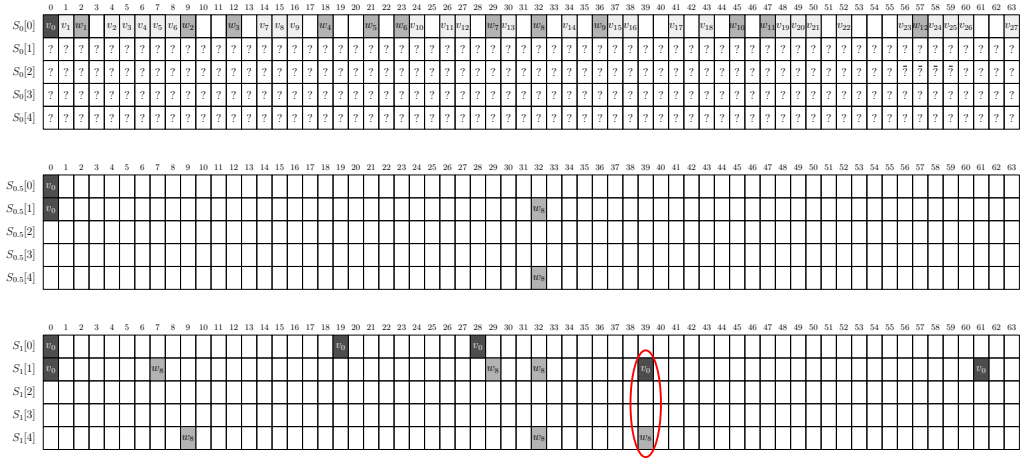

Fig. 18. Subcase 2. one of the column-32 conditions of S_0 is not true: $S_0[1][32] = 0$ is not true

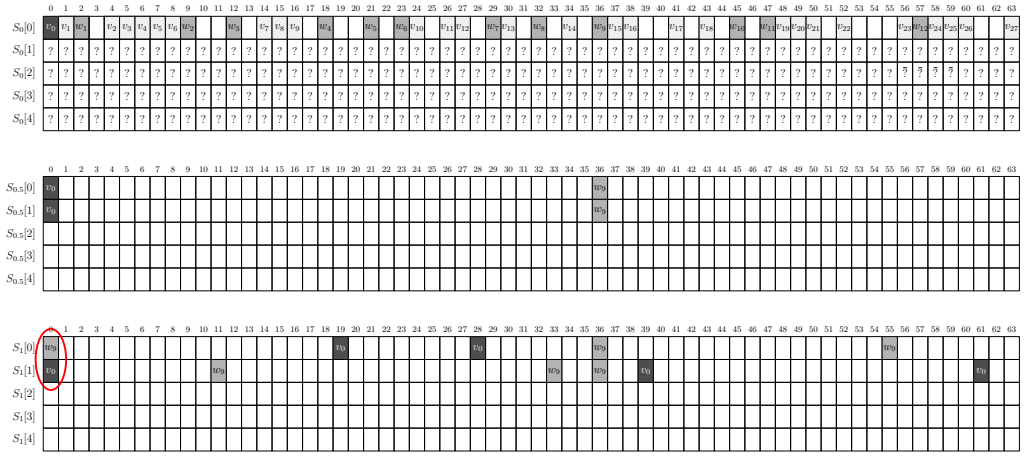

Fig. 19. Subcase 2. one of the column-36 conditions of S_0 is not true: $S_0[1][36] = 1$ is not true

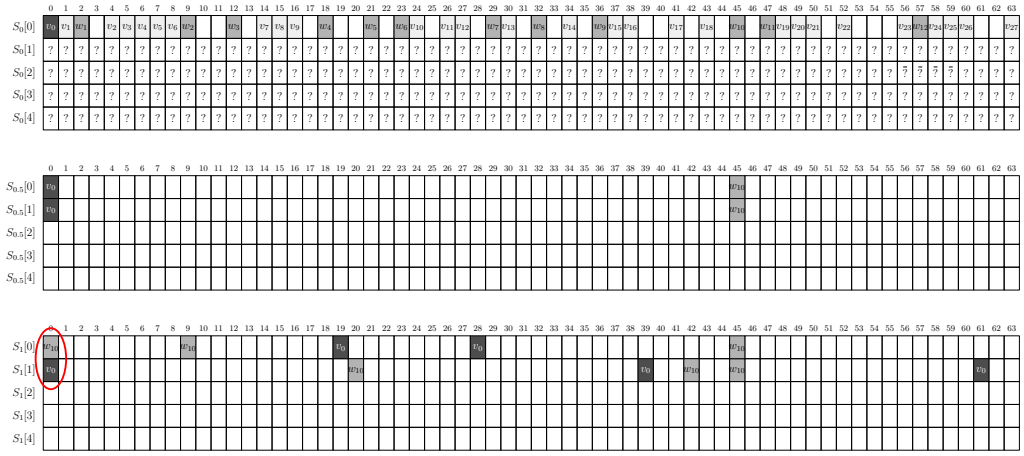

Fig. 20. Subcase 2. one of the column-45 conditions of S_0 is not true: $S_0[1][45] = 1$ is not true

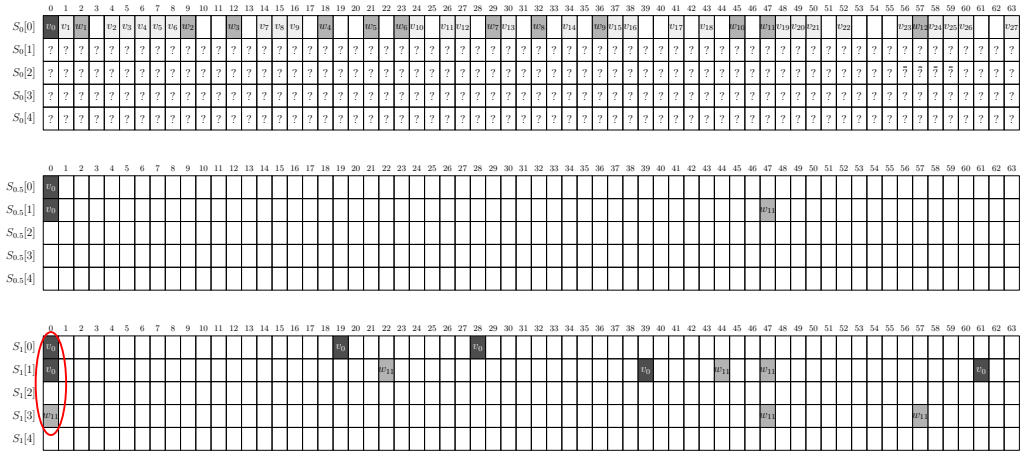

Fig. 21. Subcase 2. one of the column-47 conditions of S_0 is not true: $S_0[3][47] = S_0[4][47] + 1$ is not true

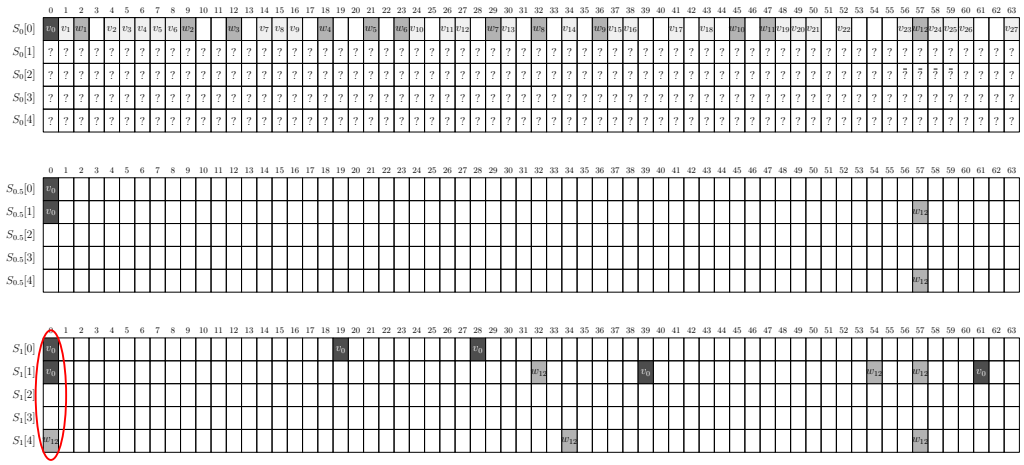

Fig. 22. Subcase 2. one of the column-57 conditions of S_0 is not true: $S_0[1][57] = 0$ is not true