

Article

A Cube Attack on a Reduced-Round Sycon

Minjeong Cho , Hyejin Eom , Erzhen Tcydenova  and Changhoon Lee * 

Department of Computer Science and Engineering, Seoul National University of Science and Technology,
Seoul 01811, Korea

* Correspondence: chlee@seoultech.ac.kr

Abstract: The cube attack was proposed at the 2009 Eurocrypt. The attack derives linear polynomials for specific output bits of a BlackBox cipher. Cube attacks target recovery keys or secret states. In this paper, we present a cube attack on a 5-round Sycon permutation and a 6-round Sycon permutation with a 320-bit state, whose rate occupies 96 bits, and whose capacity is 224 bits. We found cube variables related to a superpoly with a secret state. Within the cube variables, we recovered 32 bits of the secret state. The target algorithm was Sycon with 5-round and 6-round versions of permutation. For the 5-round Sycon, we found a cube variable and recovered a state with a total of 2^{192} Sycon computations and 2^{37} bits of memory. For the 6-round Sycon, we found cube variables and recovered a state with a total of 2^{192} Sycon computations and 2^{70} bits of memory. When using brute force in a 5-round attack, 2^{224} operations were required, but the cube attack proposed in this paper had 2^{48} offline operations, and 2^{32} operations were required. When using brute force in a 6-round attack, 2^{224} operations were required, but the cube attack proposed in this paper required 2^{95} offline operations, and 2^{63} operations were required. For both attacks, offline could be used continuously after performing only once. To the best of our knowledge, this is the first cube attack on Sycon.

Keywords: sycon; cube attack; state recovery



Citation: Cho, M.; Eom, H.;

Tcydenova, E.; Lee, C. A Cube Attack on a Reduced-Round Sycon.

Electronics **2022**, *11*, 3605. <https://doi.org/10.3390/electronics11213605>

Academic Editor: Rameez Asif

Received: 24 August 2022

Accepted: 1 November 2022

Published: 4 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, wireless communication technology supports the high-speed communication of various devices, such as cellular phones, lightweight devices, and industrial sensors. In addition, wireless communication makes smart factories, smart cities, and self-driving cars possible and provides many conveniences for human beings. However, the importance of information security must be emphasized, because there are risks of being exposed to cyber security threat, manipulation or leakage of data and invasion of privacy during the transmission of data by wireless communication [1–3]. In wireless communication, information security may be provided through cryptographic algorithms [4,5]. However, since the security strength of cryptographic algorithms does not provide immutability, it must be continuously re-evaluated and reviewed for proper use.

A cube attack is the first type of attack to utilize existing linear, logarithmic, and correlation at the same time [6]. A cube attack can apply to block ciphers, stream ciphers, and MACs. With a cube attack, key recovery is possible. A cube attack creates a polynomial in GF(2) using a set of variables defined as a cube. A cube is the set of all cases with the given variables. The polynomial is associated with a cryptographic algorithm's output treated as a black box, expressed with a quotient and a remainder. The goal is to find the coefficient of the quotient. Then, the secret data are recovered using the obtained coefficient of the quotient.

At Eurocrypt 2009, a cube attack against Trivium, a block cipher-based stream cipher suitable for wireless communication environments, was announced [6]. Trivium is a stream cipher using an 80-bit key and 1152 initializations [7]. Since then, it has been proven through several papers that a cube attack is possible for several cryptographic algorithms such as SIMON-64/96, Ascon, ACORN, MROUS, GILMI, and Keyak [8–13].

This paper proposes a cube attack on Sycon. Sycon is an AEAD cipher sponge construction with a 128-bit key. Sycon was submitted to the NIST Lightweight Cryptography Competition [14] and was selected among the ciphers for the first round of this project.

In this paper, Sycon using a rate of 96 was reduced to five rounds and six rounds. Based on the low algebraic degree of Sycon, we were able to construct a cube attack with a complexity of 2^{192} for the 5-round Sycon permutation, described in Section 4. A total 2^{48} operations and 2^{37} bits of memory were required to recover the 32-bit state in the 5-round Sycon. In Section 6, we describe the use of similar algebraic properties to construct a cube attack to obtain a state recovery attack for the 6-round Sycon permutation. We recovered the same 32-bit state on the 6-round Sycon with 2^{95} operations and 2^{70} bits of memory. To the best of our knowledge, this paper describes the first cube attack on Sycon. The main contributions of this paper are summarized as follows:

- The cube attack against Sycon shows the potential threat applicable to wireless communication. Sycon could be considered in wireless communication for confidentiality and integrity, since Sycon is a lightweight AEAD algorithm.
- This is the first known state recovery attack against Sycon. The time complexity to recover the state of the 5-round and 6-round Sycon was 2^{192} , faster than brute force. This paper shows the possibility of transmission data tampering or sniffing by an attacker.

This paper proceeds as follows. Section 3 introduces cube attacks and the Sycon cipher. Section 4 describes a cube attack on Sycon. The Results and Discussion section describes the complexity of the attacks. Conclusions are provided in Section 6 with a summary of the proposed attack and the results.

2. Related Work

A cube attack was proposed on Trivium with 672 initialization rounds with 2^{19} bits operations, 735 initialization rounds with 2^{30} bits operations, and 767 initialization rounds with 2^{45} bits operations. Since then, an improved attack with an MILP model on Trivium with 675/735/840/841/842 initialization rounds was proposed in 2021, and an attack on Trivium with 843 initialization rounds was proposed in 2022 [15,16].

Ascon was selected as a finalist in the NIST Lightweight AEAD Cryptography Contest in 2017, and the strongest attack at that time was a cube attack, which had the time complexity of 2^{97} under the nonce misuse condition in the 7-round Ascon initialization step [17]. However, for Ascon in 2022, an improved cube attack was also conducted under the condition of nonce misuse in the initialization phase [18]. The complexity of the key recovery for a full-round Ascon was 2^{130} .

In [19], Dinur et al. presented a cube-like attack against Keccak hash function-based message authentication codes, authenticated encryption, and stream cipher. The key recovery attack was performed for up to seven rounds. Key recovery and forgery attacks were proposed for AE based on the Keccak hash function. In the case of the key recovery, the attacks were performed up to six rounds under the nonce respected condition, and the attacks were performed up to seven rounds under the nonce reused condition. A key recovery attack and keystream prediction attack were proposed for the Keccak hash function-based stream cipher, and it was shown that six rounds of key recovery and key stream prediction could perform attacks for up to nine rounds.

In [20], Salam et al. proposed a cube attack against the authenticated encryption stream cipher ACORN. This attack recovered a 128-bit key with a complexity of 2^{35} in 477 initialization rounds in ACORN, a proposed candidate for NIST CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness). In addition, full-round ACORN showed that a state recovery attack could be performed with a complexity of $2^{72.8}$ using a linear equation associated with the initial state. In [12], Yang et al. proposed a method of measuring the algebraic order and numerical mapping in NFSR-based ciphers and a method of finding a cube based on a greedy algorithm. It was shown that a key could

be recovered with a complexity of $2^{127.46}$ with cube variables using 123 variables for the 772 reduced-round ACORN.

In [21], Huang et al. introduced an efficient key recovery attack for the Keccak hash function-based MAC or Keccak hash function-based AE algorithm Keyak using a conditional cube attack. In [19], a MAC-based 7-round Keccak hash function was proposed. With 2^8 times more data, the time complexity could be decreased to 2^{72} . An attack against Keyak was feasible with the time complexity of 2^{74} and a data complexity of 2^{74} for eight rounds.

3. Preliminaries

In this section, we briefly introduce the necessary background for this paper. Firstly, we provide the notations used in this paper. Then, we provide a brief description of Sycon and the concept of a cube attack.

3.1. Abbreviations and Notations

The abbreviations used in this paper are listed in Table 1.

Table 1. Abbreviations.

Abbreviation	Full Word
AEAD	Authenticated Encryption with Associated Data
LSB	Least Significant Bit
MSB	Most Significant Bit
LFSR	Linear Feedback Shift Register
SB	S-box layer
SD	Subblock Diffusion
RC	Add Round Constant Layer

The symbol notations used in this paper are listed in Table 2.

Table 2. Notations.

Symbol	Meaning
$x \oplus y$	Bitwise XOR of x and y
$(x \lll n)$	Left circular shift by n -bits
rc_i	Round constant at round i
Π^ρ	An iterated permutation with ρ rounds over $(0, 1)^{320}$
$X Y$	Concatenate data X and Y
Si_0	Most Significant 32 bit of si
Si_1	Least Significant 32 bit of si
s^i	The i -th bit of s
0^n	n bit sequences of 0
$X \circ Y$	$Y(X(\text{data}))$ where the data are input, and X and Y are functions.
Q	Quotient of a polynomial
R	Remainder of a polynomial

3.2. Sycon Authenticated Encryption with the Associated Data Algorithm Specification

Sycon is an authenticated encryption with an associated data (AEAD) cipher [22]. AEAD is an encryption algorithm with a built-in integrity process using a secret key [23]. AEAD usually performs better than using two separate cryptographic processes with two different secret keys. Sycon provides two authenticated encryption algorithms with associated data and one hash algorithm in a sponge structure. In this section, we specify the Sycon whose rate is 96.

Sycon consists of initialization, related data processing, encryption/decryption, and finalization. The initialization phase loads 128-bit keys, a 128-bit nonce, and a 64-bit initialization vector into the 320-bit state variable. Then, it conducts two permutation calls, truncating the key by 64 bits and XORing it. The relevant data processing is applied after the

initialization phase, if the related data are not empty. Relevant data processing performs the permutation with the associated data (AD) and the current state as input. The relevant data processing will not perform if the relevant data are empty. In encryption/decryption, the encryption algorithm generates the ciphertext with the same length as the input plaintext. In this case, the size of the plaintext is a multiple of 96, and padding is performed if it is less than 96 bits. Then, we conduct the permutation to update the state. This process is repeated until all 96 bits of plaintext are processed. The finalization absorbs the key back into the state via a ratio of two permutation calls, and a 128-bit tag is output. A tag is a value that concatenates the contents of S2 and S3 among the state variables.

The state is XORed with a key or plaintext after permutation as shown in Figure 1. The LSB 224 bits of the state are XORed with a domain separator. The domain separator of Sycon is as follows: 0^{224} for initialization, $100 \parallel 0^{221}$ for AD processing, $010 \parallel 0^{221}$ for message, and $001 \parallel 0^{221}$ for tag generations. If the additional data are empty, $001 \parallel 0^{221}$ is replaced by $010 \parallel 0^{221}$.

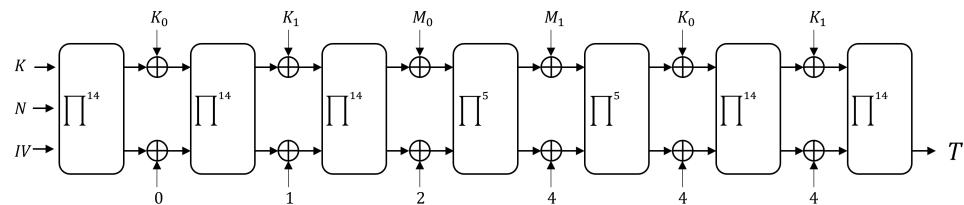


Figure 1. Sycon-AEAD-96 (when the length of the associated data is 0).

Sycon permutation is an iterative computation in a round function. In the round function, Sycon uses a 320-bit state. In the state, the first 64/96 bits are user message bits along the rate. The round function (R) of the Sycon permutation consists of a sequence of three distinct transformations: SBox (SB), SubBlockDiffusion (SD), and AddRoundConstant (RC), i.e., $R = RC \circ SD \circ SB$. The ρ -round permutation, denoted by Π^ρ , is constructed as $\Pi^\rho = R \circ \dots \circ R$.

The first layer is a nonlinear computation. Sycon's round function is SPN. Thus, for nonlinear computation, Sycon uses 64 S-boxes. The process of the S-boxes in the equation is as follows:

$$\begin{aligned} y_0 &= x_0 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_4 \\ y_1 &= x_0x_1 \oplus x_0x_3 \oplus x_0 \oplus x_1x_3 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\ y_2 &= x_0x_2 \oplus x_1 \oplus x_2x_4 \oplus x_2 \oplus x_3 \\ y_3 &= x_0 \oplus x_2x_3 \oplus x_2 \oplus x_3x_4 \oplus x_3 \oplus 1 \\ y_4 &= x_0x_4 \oplus x_0 \oplus x_1 \oplus x_3 \end{aligned} \quad (1)$$

The second layer is a diffusion layer that performs linear transformation on five 64-bit sub-blocks. The diffusion layer uses the following linear transformation:

$$\begin{aligned} S_0 &\leftarrow (S_0 \oplus (S_0 \lll 59) \oplus (S_0 \lll 54)) \lll 40 \\ S_1 &\leftarrow (S_1 \oplus (S_1 \lll 55) \oplus (S_1 \lll 46)) \lll 32 \\ S_2 &\leftarrow (S_2 \oplus (S_2 \lll 33) \oplus (S_2 \lll 02)) \lll 16 \\ S_3 &\leftarrow (S_3 \oplus (S_3 \lll 21) \oplus (S_3 \lll 42)) \lll 56 \\ S_4 &\leftarrow (S_4 \oplus (S_4 \lll 13) \oplus (S_4 \lll 26)) \end{aligned} \quad (2)$$

The third layer is the add round constant layer. Round constants use a four-bit LFSR defined by the polynomial $x^4 + x + 1$ over $\mathbb{F}_2$. The LFSR status is expressed as $rc = (rc_{i+3}, rc_{i+2}, rc_{i+1}, rc_i)$, where $rc_{i+4} = rc_i \oplus rc_{i+1}$. Starting from the initial state $rc = (0, 1, 0, 1)$, we generate a $\rho = 12$ round constant, where each state of the LFSR is given as a unique constant. The four-bit LFSR with status (rc_3, rc_2, rc_1, rc_0) is converted to a byte equal to $(0, 0, 0, 0, rc_{i+3}, rc_{i+2}, rc_{i+1}, rc_i)$. The round constants are given in Table 3.

Table 3. The round constants rc_i .

Round	Constants	Round	Constants
0	0xaaaaaaaaaaaa15	4	0xaaaaaaaaaaaa17
1	0xaaaaaaaaaaaa1a	5	0xaaaaaaaaaaaa1b
2	0xaaaaaaaaaaaa1d	6	0xaaaaaaaaaaaa0d
3	0xaaaaaaaaaaaa0e	7	0xaaaaaaaaaaaa06

3.3. Cube Attack

Let the cryptography algorithms be expressed with a polynomial f . The input of the cryptographic algorithm (e.g., plaintext, initial vector, nonce, associated authentication data) will be f 's input parameter, and the output of the cryptography algorithm (e.g., ciphertext, tag) will be the value of f 's computed result. If the block cipher has input as plaintext P , initial vector IV , and key K , and the output is ciphertext C , we can express the block cipher $f(P, IV, K) = c$ with $P = p_0p_1p_2p_3 \dots p_n$, $IV = iv_0iv_1iv_2iv_3 \dots iv_n$, $K = k_0k_1k_2k_3 \dots k_m$, and $C = c_0c_1c_2c_3 \dots c_n$, where p_i , k_i , and c_i is a bit representation, respectively.

- **Degree** The dense polynomial f of degree d has $\sum_{i=0}^d \binom{2n+m}{i}$ possible polynomials over GF(2). To eliminate the nonlinear terms on the polynomials, the attack needs to eliminate $\sum_{i=2}^d \binom{2n+m}{i}$. Thus, when the degree becomes higher, it is hard to eliminate the nonlinear terms.
- **Cube Variables** To eliminate the nonlinear terms from the polynomial f , an attacker needs to divide the polynomials f by the other polynomial t , whose degree is $d - 1$. If $f(P, IV, K)$ divided by $t = p_0p_1 \dots p_{n-1}$, then f can be expressed as $f(P, IV, K) = tQ(p_n, IV, K) + R(P, IV, K)$. In this term, t is $n - 1$ bit cube variables.
- **Superpolys** We assume that dense polynomial f divided by $t = p_0p_1 \dots p_{n-1}$, $f(P, IV, K) = tQ(p_n, IV, K) + R(P, IV, K)$, as above. In this term, $Q(p_n, IV, K)$ is a superpoly with degree 1. In order to obtain a superpoly, the attacker can compute $\sum_{t \in C_t} f(P, IV, K) = Q(p_n, IV, K)$.

In a cube attack, finding cube variables is important, because when the degree of the polynomial becomes higher, an attacker needs more polynomials to use Gaussian elimination. When the cube variables are larger, the attacker breaks more rounds. Moreover, f can be divided by $m - 1$ variables, and the quotient will be a degree 1 polynomial. A cube attack should first formulate the polynomial f . If f is not a dense polynomial, superpoly Q 's degree will be changed along the chosen cube variables. For example, when we define cube variables as $t = p_0p_1 \dots p_{n-1}$, and if A is bits that multiplied with l , $l \leq n$ bits, then f could be $f(P, IV, K) = tQ(p_n, A) + R(P, IV, K)$. That is, the polynomial Q degree is $l - m$. Thus, in order to obtain Q , the attacker needs to use fewer cube variables $l - m - 1$.

To make m independent polynomials, the attacker needs to analyze where there are no multiplication values between the target bits and the input bits that the attacker can control. The attacker will select the control bits that do not have multiplication with the target bits. The attacker can know whether multiplication will be computed by analyzing the cryptographic algorithms' process. A typical example that has a multiplication step in a cryptography algorithm is an S-box. After the attacker finds the proper cube variables from the polynomials, a cube attack can decide the round that an attacker can use. Then, the cube attack is presented as follows:

1. **Offline Phase** The attacker computes and stores the $Q(p_n, IV, K)$. The targeted data bits can be expressed as linear polynomials $Q(p_n, A)$. The attacker computes a linear polynomial from the Q values. The attacker assigns 0 except for the cube variables. Then, the attacker sets bit by bit on x . From the data, the attacker computes each coefficient of x .
2. **Online Phase** Considering the oracle as given, the attacker derives the cube sum of the oracle query results $Q(p_n, A)$. From the polynomials saved in the offline phase, the attacker recovers target bits A . If A has $l - m - 1$ variables, the attacker needs an

$l - m - 1$ cube sum result. The attacker computes the Gaussian elimination to obtain the recovered target bits.

3. **Brute Force Phase** If the cube variables are not enough to obtain all rounds or all target bits, the attacker performs a brute force attack on the remaining bits. For example, if the recovered bits are l bits and the targeted bits are m bits, the attacker performs a 2^{m-l} exhaustive search.

4. State Recovery Attack on a Reduced-Round Sycon

In this section, we focus on state recovery attacks. First, we analyze the round-reduced Sycon. Then, in the later part of the section, the cube attacks on five-round Sycon and six-round Sycon are described.

4.1. Idea and Scenario

We propose an attack idea and scenario to recover the secret state of Sycon. Sycon has a 320-bit state variable, as described in Section 3. In this paper, the state variable $\mathbb{S}$ is expressed as a truncated form to 32-bit units as follows:

$$\mathbb{S} = S_{00} \| S_{01} \| S_{10} \| S_{11} \| S_{20} \| S_{21} \| S_{30} \| S_{31} \| S_{40} \| S_{41} \quad (3)$$

The polynomial of the output from the S-box has degree 2. After five rounds of the permutation are performed, the result $\mathbb{S}$ has a polynomial of degree 2^4 . Therefore, we require $2^4 - 1$ variables in the cube for an attack against the 5-round Sycon. In the same way, after six rounds, the degree of $\mathbb{S}$ is 2^5 . Thus, we need $2^5 - 1$ variables in the cube. We choose bits from the state variable that we have control over. We select S_{00} , S_{01} , and S_{10} as the cube variables. The cube attack is feasible if the variables are uniquely multiplied by S_{00} , S_{01} , and S_{10} . The variable with this characteristic in the S-box is S_{30} . S_{10} is only multiplied by S_{00} and S_{30} . Choosing S_{10} as the cube variable allows us to recover S_{30} . With the cube variables, we obtain the linear equations as follows:

$$L_i(S) = a_i^0 S_{10}^0 + a_i^1 S_{10}^1 + a_i^2 S_{10}^2 + \dots + a_i^{31} S_{10}^{31} + c_i, i = 0, 1, 2, \dots, 31 \quad (4)$$

To construct L_i , we obtain the result of the permutation by $M \oplus C$, and we choose the message bits that construct the cube variables. Figure 2 shows the attack scenario.

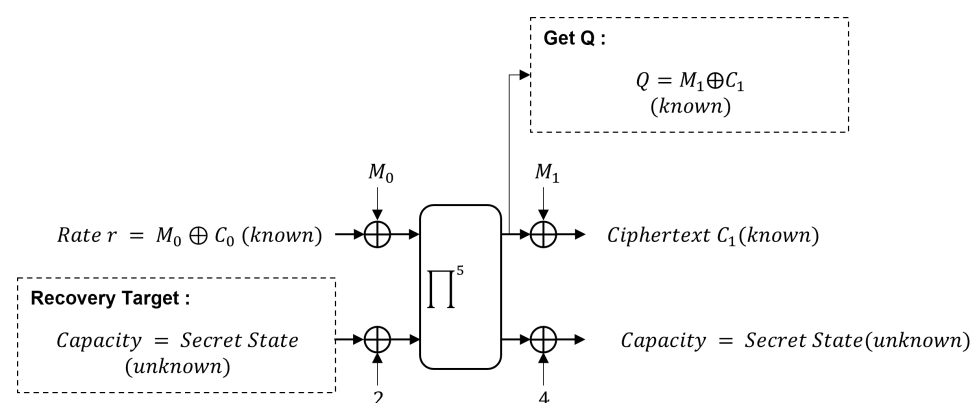


Figure 2. State recovery attack scenario on the reduced-round Sycon.

4.2. Attack on the Five-Round Sycon

4.2.1. Offline Phase

Sycon encrypts plaintext by truncating it in units of 96 bits. We chose 96 bits of plaintext. The state variable S was concatenated as S_{00} , S_{01} , S_{10} , S_{11} , S_{20} , S_{21} , S_{30} , and S_{31} , where each size was 32 bits.

$$\mathbb{S} = S_{00} \| S_{01} \| S_{10} \| S_{11} \| S_{20} \| S_{21} \| S_{30} \| S_{31} \| S_{40} \| S_{41} \quad (5)$$

P_0 , P_1 , and P_2 were the plaintext values chosen. Ciphertext C_0 , C_1 , and C_2 were computed by performing XOR of the current state variables MSB 96 bits S_{0_0} , S_{0_1} , and S_{1_0} and the plaintext P_0 , P_1 , and P_2 .

$$\begin{aligned}C_0 &= S_{0_0} \oplus P_0 \\C_1 &= S_{0_1} \oplus P_1 \\C_2 &= S_{1_0} \oplus P_2\end{aligned}\quad (6)$$

Since the ciphertext and the plaintext were known, we computed the XORing of the ciphertext and the plaintext to obtain S_{0_0} , S_{0_1} , S_{1_0} .

$$\begin{aligned}S_{0_0} &= C_0 \oplus P_0 \\S_{0_1} &= C_1 \oplus P_1 \\S_{1_0} &= C_2 \oplus P_2\end{aligned}\quad (7)$$

We set the cube variables in S_{0_1} . Since the degree of the S-box was 2, the result after five rounds had a polynomial of degree 16. The attacker chose a 15-bit cube in the 5-round Sycon state variable to obtain a polynomial of degree 1 and the rest of the values as constants. We recovered 16 bits of MSB and 16 bits of LSB of S_{3_0} , respectively. Therefore, two cube variables were required, and each cube variable set had 2^{15} elements. The cube variable could be $C_{S_{1_0}} := \{0x0, 0x1, \dots, 0xffff\}$ and $C_{S_{1_0}} := \{0x00000, 0x10000, \dots, 0xffff0000\}$. We assigned all the other variables to 0 except for the cube variable S_{0_1} and used S_{0_0} in the attack. S_{1_0} was used in the attack as it was multiplied only by S_{0_0} and S_{3_0} , as described in Section 2.

Since S_{3_0} depends only on S_{0_0} , after five rounds, the polynomial could be expressed as a polynomial for the quotient of S_{0_0} and S_{3_0} .

$$\begin{aligned}\mathbb{S} &= S_{3_0}Q(S_{0_0}, S_{3_1}) + R(S_0, S_1, S_2, S_3, S_4) \\Q(S) &= \sum_{S_{1_0} \in C_{S_{1_0}}} f(\mathbb{S}) \\&= a^0 S_1^0 + a^1 S_1^1 \dots + a^{31} S_1^{31} + c \text{ where } a^i \in \{0, 1\}\end{aligned}\quad (8)$$

We used the process shown in Table 4 to find the variable coefficients of a polynomial for $Q(S)$. As a result, we obtained the following values. The results obtained were stored in the memory and used in the online phase.

$$\begin{aligned}\sum_{(v_1, \dots, v_{15}) \in C_{S_{1_0}}} f(S_{0_0}, S_{0_1}, S_{1_0}, S_{1_1}, S_{2_0}, S_{2_1}, S_{3_0}, S_{3_1}, S_{4_0}, S_{4_1}) &= Q(S_{0_0}, S_{3_0}), \\ \text{with } S_{0_0} &= 0x00000000, 0x00000001, 0x00000002, \dots, 0x0000ffff, \\ S_{3_0} &= 0x00000000, 0x00000001, 0x00000002, \dots, 0x0000ffff\end{aligned}\quad (9)$$

$$\begin{aligned}\sum_{(v_{16}, \dots, v_{31}) \in C_{S_{1_0}}} f(S_{0_0}, S_{0_1}, S_{1_0}, S_{1_1}, S_{2_0}, S_{2_1}, S_{3_0}, S_{3_1}, S_{4_0}, S_{4_1}) &= Q(S_{0_0}, S_{3_0}), \\ \text{with } S_{0_0} &= 0x00000000, 0x00010000, 0x00020000, \dots, 0xffff0000, \\ S_{3_0} &= 0x00000000, 0x00010000, 0x00020000, \dots, 0xffff0000\end{aligned}\quad (10)$$

Table 4. Attack on the five-round Sycon: Offline phase process.

Offline Pseudo Code
Input : State $\mathbb{S} = S_{00}S_{01}S_{10}S_{11}S_{20}S_{21}S_{30}S_{31}S_{40}S_{41}$, Cube Set $C_{S_{10}}$ Output : CubeSum $CS[2^{16}] = \{0^{510}, 0^{510}, \dots, 0^{510}\}$ Algorithm : $\mathbb{S} := 0^{320}$ For i in 0 to $0x0000ffff$: $CS[i] := 0^{510}$ $S_{30} := i$ For j in 0 to $0x0000fffe$: $S_{00} := j$ For cube value in $C_{S_{10}}$: $\mathbb{S} = Sycon(S)$ $CStmp := CS[i] \oplus S_{10}$ $S := 0^{320}$ $CS[i] := CS[i] CStmp$ For i in 0 to $0xffff0000$: $CS[i] := 0^{510}$ $S_{30} := i$ For j in 0 to $0xfffe0000$: $S_{00} := j$ For cube value in $C_{S_{10}}$: $S = Sycon(S)$ $CStmp := CS[i] \oplus S_{10}$ $\mathbb{S} := 0^{320}$ $CS[i] := CS[i] CStmp$ return CS

4.2.2. Online Phase

In the online phase, we implemented an oracle. The oracle allowed the choice of arbitrary plaintext S_{01} and S_{10} , and the cube variables $C_{S_{10}}$ were defined internally as a fixed set. The oracle computed $Q(S_{00}, S_{30})$ for the plaintext chosen. We computed two times for the MSB 32 bits and LSB 32 bits of S_{30} . The oracle calculated the superpolys as follows:

$$\sum_{(v_1, \dots, v_{15}) \in C_{S_{10}}} f(S_{00}, S_{01}, S_{10}, S_{11}, S_{20}, S_{21}, S_{30}, S_{31}, S_{40}, S_{41}) = Q(S_{00}, S_{30}), \quad (11)$$

with $S_{00} = 0x00000000, 0x00000001, 0x00000002, \dots, 0x0000ffff$

$$\sum_{(v_{16}, \dots, v_{31}) \in C_{S_{10}}} f(S_{00}, S_{01}, S_{10}, S_{11}, S_{20}, S_{21}, S_{30}, S_{31}, S_{40}, S_{41}) = Q(S_{00}, S_{30}), \quad (12)$$

with $S_{00} = 0x00000000, 0x00010000, 0x00020000, \dots, 0xffff0000$.

We explored the superpoly $Q(S_{00}, S_{30})$ obtained by querying the oracle in the memory space stored during the offline phase. If the same value as $Q(S_{00}, S_{30})$ was in the memory, we determined S_{30} , because the memory stored $Q(S_{00}, S_{30})$ for every S_{30} . The $Q(S_{00}, S_{30})$ that we explored in the memory was as follows:

$$Q(0x00000000, S_{30}), Q(0x00000001, S_{30}), Q(0x00000002, S_{30}), \dots, Q(0x0000ffff, S_{30}) \quad (13)$$

$$Q(0x00000000, S_{30}), Q(0x00010000, S_{30}), Q(0x00020000, S_{30}), \dots, Q(0xffff0000, S_{30}) \quad (14)$$

4.2.3. Brute Force Phase

We recovered the remaining 192 bits of state by brute force. There was no memory required, but we needed 2^{192} computations.

4.3. Attack on the Six-Round Sycon

4.3.1. Offline Phase

We chose the 32-bit S_{01} . With the chosen plaintext, we assigned the LSB 31 bits of S_{10} as the cube variable $C_{S_{10}} := \{0^{32}, \dots, 0\text{xffffffff}\}$. The cube variable 2^{32} of S_{01} , S_{20} , S_{21} , S_{31} , S_{40} , and S_{41} was fixed as 0^{32} , because S_{10} was only multiplied with S_{00} and S_{30} . We used the process shown in Table 5 to find the variable coefficients of a polynomial for $Q(S)$. S_{30} could be rephrased as $Sycon(\mathbb{S}) = S_{30}Q(S_{00}, S_{31}) + R(S_{00}, S_{10}, S_{20}, S_{30}, S_{40}, S_{41})$. $S_{30}Q(S)$ was $\sum_{S_{10} \in C_{S_{10}}} Sycon(\mathbb{S})$. For each $j \in \{0, 1\}^{64}$, we computed the following:

$$\sum_{(v_1, \dots, v_{31}) \in C_{S_{10}}} f(S_{00}, S_{01}, S_{10}, S_{11}, S_{20}, S_{21}, S_{30}, S_{31}, S_{40}, S_{41}) = Q(S_{00}, S_{30}),$$

$$S_{00} = 0\text{x}00000000, 0\text{x}00000001, 0\text{x}00000002, \dots, 0\text{xffffffff},$$

$$S_{30} = 0\text{x}00000000, 0\text{x}00000001, 0\text{x}00000002, \dots, 0\text{xffffffff}.$$
(15)

Table 5. Attack on the six-round Sycon: Offline phase process.

Offline Pseudo Code
Input : State $\mathbb{S} = S_{00}S_{01}S_{10}S_{11}S_{20}S_{21}S_{30}S_{31}S_{40}S_{41}$, Cube Set $C_{S_{10}}$ Output : CubeSum $CS[2^{32}] = \{0^{510}, 0^{510}, \dots, 0^{510}\}$ Algorithm : $S := 0^{320}$ For i in 0 to 0xffffffff : $CS[i] := 0^{1020}$ $S_{30} := i$ For j in 0 to 0xffffffff : $s_{00} := j$ For cube value in $C_{S_{10}}$: $S = Sycon(S)$ $CStmp := CS[i] \oplus S_{10}$ $\mathbb{S} := 0^{320}$ $CS[i] := CS[i] CStmp$ return CS

4.3.2. Online Phase

We implemented an oracle, where the cube variables $C_{S_{00}}$ were defined internally as a fixed set. The oracle computed $Q(S_{00}, S_{30})$ for the plaintext chosen. The results calculated by the oracle were as follows for the online cube-sum:

$$\sum_{(v_1, \dots, v_{31}) \in C_{S_{10}}} f(S_{00}, S_{01}, S_{10}, S_{11}, S_{20}, S_{21}, S_{30}, S_{31}, S_{40}, S_{41}) = Q(S_{00}, S_{30}),$$

$$S_{00} = 0\text{x}00000000, 0\text{x}00000001, 0\text{x}00000002, \dots, 0\text{xffffffff}.$$
(16)

We explored the value $Q(S_{00}, S_{30})$ obtained through the oracle in the memory space stored during the offline phase. If the same value as $Q(S_{00}, S_{30})$ was in the memory, we determined S_{30} , because the memory stored $Q(S_{00}, S_{30})$ for every S_{30} . The $Q(S_{00}, S_{30})$ that we explored in the memory was as follows:

$$Q(0\text{x}00000000, S_{30}), Q(0\text{x}00000001, S_{30}), Q(0\text{x}00000002, S_{30}), \dots, Q(0\text{xffffffff}, S_{30}) \quad (17)$$

4.3.3. Brute Force Phase

We recovered the remaining 192-bit state bits by brute force.

5. Results and Discussion

First, we analyze the complexity in the offline phase. Computing the equations for just one case in the memory required 2^{15} 5-round Sycon and 16 bits of memory. We performed the same process 2^{32} times repeatedly for each $S3_0$ and $S0_0$. Therefore, in the offline phase, we required 2^{48} Sycon computations and 2^{37} bits of memory. In the online phase, we required 2^{32} computations to recover the target bits. The remaining 192 bits of state were recovered by brute force, so we required 2^{192} . In total, the attack needed a computational complexity of $2^{48} + 2^{32} + 2^{192} \approx 2^{192}$.

To recover the six-round Sycon permutation state, we needed an offline phase, an online phase, and a brute-force phase. In the offline phase, we precalculated the cube sum, and it needed a computation complexity of 2^{31} and 64 bits of memory. We performed this process 2^{64} times repeatedly for all cases in $S3_0$ and $S0_0$. In total, in the offline phase, 2^{95} 6-round Sycon computations and 2^{70} bits of memory were required. In the online phase, we only needed to perform 2^{64} of six-round Sycon permutations. The remaining 192 bits of state were recovered by brute force. In the brute-force phase, there was no memory required, but we needed 2^{192} computations. Thus, in total, the attack required a computational complexity of $2^{95} + 2^{70} + 2^{192} \approx 2^{192}$ (Table 6).

Table 6. Complexity of the 224-bit secret state recovery attack.

Round	Memory				Computation			
	This Paper			Brute Force *	This Paper			Brute Force *
	Offline	Online	Brute Force		Offline	Online	Brute Force	
5	2^{37}	-	-	-	2^{48}	2^{32}	2^{192}	2^{224}
6	2^{70}	-	-	-	2^{95}	2^{63}	2^{192}	2^{224}

* This was the first state recovery attack against Sycon. So the attack complexity based on brute force is the best result up to now.

The AEAD cipher encrypts the data to be transmitted and creates a tag for data integrity [24]. In order to give functions, Sycon has the following four phases: Initialization, associated data processing, encryption/decryption, and finalization. Each step should be analyzed in different ways [25]. In the future, we can extend the cube attack to the initialization or finalization phase of Sycon to recover the secret key.

6. Conclusions

In this paper, we proposed a state recovery attack against five-round and six-round Sycon. Sycon keeps 224 bits in secret and 96 bits as plaintext/ciphertext. With no information, we needed 2^{224} computations. From the attack we proposed, 32 bits of the state $S3_1$ could be recovered against the 5-round Sycon, with a time complexity of 2^{192} . We also proposed an attack against the six-round Sycon. The attack recovered the same 32 bits with the time complexity of 2^{95} and 2^{70} of memory in the offline phase. The time complexity was 2^{192} . This was faster than brute force over the 2^{224} possible states by a factor of about 2^{32} .

Author Contributions: Methodology, M.C.; Writing—original draft, M.C.; Formal analysis, H.E. and E.T.; Writing—review and editing, H.E. and E.T.; Supervision, C.L.; All authors read and agreed to the published version of the manuscript.

Funding: This work was supported as part of the Military Crypto Research Center (UD210027XD) funded by the Defense Acquisition Program Administration (DAPA) and the Agency for Defense Development (ADD).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abdalzaher, M.S.; Samy, L.; Muta, O. Non-zero-sum game-based trust model to enhance wireless sensor networks security for IoT applications. *IET Wirel. Sens. Syst.* **2019**, *9*, 218–226. [\[CrossRef\]](#)
2. Abdalzaher, M.S.; Muta, O. A game-theoretic approach for enhancing security and data trustworthiness in IoT applications. *IEEE Internet Things J.* **2020**, *7*, 11250–11261. [\[CrossRef\]](#)
3. Abdalzaher, M.S.; Seddik, K.; Muta, O. Using repeated game for maximizing high priority data trustworthiness in wireless sensor networks. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 552–557.
4. Phuc, T.S.D.; Xiong, N.N.; Lee, C. Cryptanalysis of the XO-64 Suitable for Wireless Systems. *Wirel. Pers. Commun.* **2017**, *93*, 589–600. [\[CrossRef\]](#)
5. Phuc, T.S.D.; Lee, C. Cryptanalysis on SDDO-Based BM123-64 Designs Suitable for Various IoT Application Targets. *Symmetry* **2018**, *10*, 353. [\[CrossRef\]](#)
6. Dinur, I.; Shamir, A. Cube attacks on tweakable black box polynomials. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April, 2009; pp. 278–299.
7. De Cannière, C.T.; Preneel, B. A stream cipher construction inspired by block cipher design principles. In *Information Security*; Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B., Eds.; Springer: Berlin/Heidelberg, Germany; pp. 171–186.
8. Mroczkowski, P.; Szmjdt, J. The cube attack on stream cipher Trivium and quadraticity tests. *Fundam. Informaticae* **2012**, *114*, 309–318. [\[CrossRef\]](#)
9. Li, Z.; Dong, X.; Wang, X. Conditional Cube Attack on Round-Reduced ASCON. Cryptology ePrint Archive 2017. Available online: <https://eprint.iacr.org/2017/160.pdf> (accessed on 24 July 2022).
10. Bi, W.; Li, Z.; Dong, X.; Li, L.; Wang, X. Conditional cube attack on round-reduced River Keyak. *Des. Codes Cryptogr.* **2018**, *86*, 1295–1310.
11. Chen, S.; Xiang, Z.; Zeng, X.; Zhang, S. Cube attacks on round-reduced MORUS and GIMLI. *Sci. China Inf. Sci.* **2022**, *65*, 1–3. [\[CrossRef\]](#)
12. Yang, J.; Liu, M.; Lin, D. Cube cryptanalysis of round-reduced ACORN. In Proceedings of the International Conference on Information Security, New York, NY, USA, 16–18 September 2019; pp. 44–64.
13. He, Y.; Wang, G.; Li, W.; Ren, Y. Improved cube attacks on some authenticated encryption ciphers and stream ciphers in the Internet of Things. *IEEE Access* **2020**, *8*, 20920–20930. [\[CrossRef\]](#)
14. Lightweight Cryptography Round 1. Available online: <https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates> (accessed on 22 August 2022).
15. Sun, Y. Cube Attack against 843-Round Trivium. Cryptology ePrint Archive 2021. Available online: <https://eprint.iacr.org/2021/547> (accessed on 24 July 2022).
16. Delaune, S.; Derbez, P.; Gontier, A.; Prud'Homme, C. A Simpler Model for Recovering Superpoly on Trivium. In *Proceedings of the International Conference on Selected Areas in Cryptography*; Springer: Cham, Switzerland, 2022; pp. 266–285.
17. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schl  ffer, M. Ascon v1. 2: Lightweight authenticated encryption and hashing. *J. Cryptol.* **2021**, *34*, 1–42. [\[CrossRef\]](#)
18. Chang, D.; Hong, D.; Kang, J. Conditional Cube Attacks on Ascon-128 and Ascon-80pq in a Nonce-Misuse Setting. Cryptology ePrint Archive 2022. Available online: <https://eprint.iacr.org/2022/544> (accessed on 24 July 2022).
19. Dinur, I.; Morawiecki, P.; Pieprzyk, J.; Srebrny, M.; Straus, M. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015; pp. 733–761.
20. Salam, M.I.; Bartlett, H.; Dawson, E.; Pieprzyk, J.; Simpson, L.; Wong, K.K.H. Investigating cube attacks on the authenticated encryption stream cipher ACORN. In Proceedings of the International Conference on Applications and Techniques in Information Security, Cairns, QLD, Australia, 26–28 October 2016; pp. 15–26.
21. Huang, S.; Wang, X.; Xu, G.; Wang, M.; Zhao, J. Conditional cube attack on reduced-round Keccak sponge function. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 30 April–4 May 2017; pp. 259–288.
22. Mandal, K.; Saha, D.; Sarkar, S.; Todo, Y. Sycon: A new milestone in designing ASCON-like permutations. *J. Cryptogr. Eng.* **2022**, *12*, 305–327. [\[CrossRef\]](#)
23. Frankel, S.E.; Kent, K.; Lewkowski, R.; Orebaugh, A.D.; Ritchey, R.W.; Sharma, S.R. SP 800-77. Guide to IPsec VPNs. 2005. Available online: <https://csrc.nist.gov/library/alt-SP800-77.pdf> (accessed on 24 July 2022).
24. Seok, B.; Sicato, J.C.S.; Erzhen, T.; Xuan, C.; Pan, Y.; Park, J.H. Secure D2D communication for 5G IoT network based on lightweight cryptography. *Appl. Sci.* **2019**, *10*, 217. [\[CrossRef\]](#)
25. Teng, W.L.; Salam, I.; Yau, W.C.; Pieprzyk, J.; Phan, R.C.W. Cube attacks on round-reduced TinyJAMBU. *Sci. Rep.* **2022**, *12*, 1–13.