Noname manuscript No.
(will be inserted by the editor)

SMT-based Cube Attack on Simeck32/64

Mojtaba Zaheri · Babak Sadeghiyan

Received: date / Accepted: date

Abstract Satisfiability modulo theories or SMT can be stated as a generalization of Boolean satisfiability problem or SAT. The core idea behind the introduction of SMT solvers is to reduce the complexity through providing more information about the problem environment.

In this paper, we take advantage of a similar idea and feed the SMT solver itself, by extra information provided through middle state Cube characteristics, to introduce a new method which we call ***SMT-based Cube Attack***, and apply it to improve the success of the solver in attacking reduced-round versions of the Simeck32/64 lightweight block cipher.

We first propose a new algorithm to find cubes with most number of middle state characteristics. Then, we apply these obtained cubes and their characteristics as extra information in the SMT definition of the cryptanalysis problem, to evaluate its effectiveness. Our cryptanalysis results in a full key recovery attack by 64 plaintext/ciphertext pairs on 12 rounds of the cipher in just 122.17 seconds. This is the first practical attack so far presented against the reduced-round versions of Simeck32/64.

We also conduct the cube attack on the Simeck32/64 to compare with the SMT-based cube attack. The results indicate that the proposed attack is more powerful than the cube attack.

Keywords SMT-based Attack · Cube Attack · Algebraic Attack · Simeck Lightweight Block Cipher

M. Zaheri
Department of Computer Engineering and IT
Amirkabir University of Technology (Tehran Polytechnic)
E-mail: mojtaba.zaheri@aut.ac.ir

B. Sadeghiyan
Department of Computer Engineering and IT
Amirkabir University of Technology (Tehran Polytechnic)
E-mail: basadegh@aut.ac.ir

1 Introduction

Since the introduction of Rijndael [8], winner of the AES[1] competition [16], attentions in the literature of cryptology were drawn to a specific type of attack, *i.e.* Algebraic Cryptanalysis. Novel algebraic primitives used in AES winner encouraged the researchers to focus on algebraic attacks to find if it is possible to take advantage of simple algebraic components of cipher algorithms in cryptanalysis. A typical algebraic attack involves three main steps: I) expressing the cipher operations as a system of equations, II) substituting in known data for some of the variables, and III) solving the system of equations for the key.

For the first step, attacker can choose which algebraic system to use; for example, (s)he might treat the text as a vector of bits and use Boolean algebra, or (s)he might choose to treat it as a vector of bytes and use arithmetic modulo 2^8.

For the second step, attacker usually uses some PCs[2] to substitute with the variables. Chosen plaintext and known plaintext attacks are suitable scenarios to provide these PCs.

Finally, there are different methods used for the last step of attack to solve the resulting system of equations; Gröbner basis algorithms and SAT solvers are more common in the literature, where SMT is a relatively new concept. In [18], the attack applies the SMT solvers to directly retrieve the secret key of Shuffle block cipher. However, in this paper, we apply the SMT-based attacks more elaborately to study if its direct application in cryptanalysis is successful.

What makes algebraic attacks hardly feasible is a combination of the size of the system of equations and nonlinearity in the relations involved. In any algebra, solving a system of linear equations is more-or-less straightforward provided there are more equations than variables. However, solving nonlinear systems of equations is far harder. Cipher designers, therefore, strive to make their ciphers highly nonlinear.

In one of the attempts to counter this issue in algebraic attacks, Dinur and Shamir proposed the Cube Attack [9]. This attack tries to reveal the secret key bits of the cipher in two steps; first, in a preprocessing phase, the attacker tries to find some linear algebraic equations of key variables by tweaking the relations between plaintexts and ciphertexts, while full control over plaintext, ciphertext, and key variables are available, so that (s)he might achieve a system of linearly independent equations. Next, in online phase, the attacker can easily solve this achieved system of linear equations by Gauss elimination to find the secret adjusted value of the key bits. In this paper, we take advantage of their proposed idea to improve SMT-based attacks, as well.

Recent advances and ubiquitous use of resource constraint devices tempted NSA to propose new cipher algorithms for these specific environments. SIMON and SPECK [4] are the NSA's new ciphers introduced for this purpose. However, similar to the case of DES[3] [17], NSA did not provide any security evaluation of the two ciphers and no analysis of their cryptographic strength is given, which makes them suspicious in the literature. So, the researchers tried to propose alternatives for them. As one of these tries, Yang, et al, proposed the Simeck family of

[1] Advanced Encryption Standard
[2] Plaintext-Ciphertext pairs
[3] Data Encryption Standard

lightweight block ciphers [11], which as they claim, combines the good design components of SIMON and SPECK ciphers. Simeck includes three versions; *Simeck32/64*, *Simeck48/96*, and *Simeck64/128*, where n and k in Simeckn/k are plaintext and key sizes, respectively.

A successful attack should have a complexity less than exhaustive search or brute force attack (*e.g.* less than 2^{128} in *Simeck64/128*). However, in practice, it is nearly impossible to verify such attacks on full-round ciphers with large input blocks. Hence, many researchers focus on reduced-round versions of ciphers with smaller input blocks, or even propose small scale variants of the ciphers [7], and try to attack more rounds of them in a reasonable time, as a proof of concept for their proposed improvements on their methods of cryptanalysis.

In this paper, we study our proposed attacks against *Simeck32/64* lightweight block cipher. We first introduce a new algorithm to find cubes with most number of middle state characteristics, including constant (zero / one) and linear equations. Next, we evaluate the success of the SMT-based attacks when these cubes and their equations are considered or not. Finally, the classic cube attack is done to compare the results. It is worth mentioning that we do not guess any key bits of the cipher and we only use structural properties of the cipher to break a higher number of rounds.

This paper is organized as follows: Section 2, first reviews design specifications of the *Simeck32/64* lightweight block cipher. Next, it explains the Cube Attack and Cube Tester, and finally examines the SMT solver and its applications in cryptanalysis. In section 3, a new search algorithm for cube search is proposed and is applied to find top cubes and their characteristics for *Simeck32/64*. In section 4, first, the SMT-based attacks are performed on the cipher. Next, the ***SMT-based Cube Attack*** is proposed and applied to conduct a successful key recovery attack on **12**-round *Simeck32/64*. Section 5 describes classic Cube Attack on *Simeck32/64* to compare it with SMT-based attacks. Finally, Section 6 concludes the paper, and draws some roadmaps on future works in this area of research.

2 Preliminaries

In this section, a brief review of the basic concepts required for the paper is presented. First, structure, design components, and specifications of the Simeck32/64 lightweight block cipher are reviewed. Next, cube attacks and cube testers are explained. Finally, the SMT solvers and their applications in cryptanalysis are described.

2.1 Simeck32/64 Lightweight Block Cipher

In 2015, Yang et. al. combined selected design components of the two NSA ciphers SIMON and SPECK [4] to reach more compactness and efficiency in a new family of lightweight block ciphers named Simeck [11]. In fact, Simeck's round function is a slight modification of the round function used in SIMON, and it also uses this structure for key scheduling, just like the SPECK algorithm. In this paper, we focus on *Simeck32/64* which is the smallest member of the family. These notations are used to describe the cipher:

$x \odot y$: bitwise AND of x and y.
$x \oplus y$: bitwise XOR of x and y.
$x \lll c$: cyclic shift of x to the left by c bits

The *Simeck32/64* accepts 32-bit plaintexts and 64-bit master keys as input, and generates 32-bit ciphertexts as output. Round function and key scheduler of the cipher have a Feistel structure. The algorithm first breaks the 32-bit input plaintext into two 16-bit blocks l_0 and r_0, so that l_0 contains high-order bits, and r_0 contains low-order bits. Then, the round function is applied to them 32 times, to reach in l_{32} and r_{32}.

Round Function: $i^{\ th}$ round function is defined in the following:

$$R_{k_i}(l_i, r_i) = (r_i \oplus f(l_i) \oplus k_i, l_i) \tag{1}$$

where l_i and r_i are middle state words of *Simeck32/64* algorithm and k_i is the round key. Also, the function f has the following definition:

$$f(x) = (x \odot (x \lll 5)) \oplus (x \lll 1) \tag{2}$$

Key Scheduler: First, the 64-bit master key K is divided into four 16-bit blocks (t_2, t_1, t_0, k_0), and it's loaded into LFSR [4] as an initial state. k_0 and t_2 contain 16 low-order bits and high-order bits of K, respectively. Next, the structure of the round function as well as $C \oplus (z_0)_i$ constant values are used to update the registers and generate the round keys. The update operation is as follows:

$$\begin{cases} k_{i+1} = t_i \\ t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_0)_i \end{cases} \tag{3}$$

so that $0 \leq i \leq 31$, and $C = 2^{16} - 1$. The z_0 is a maximal sequence [5] with the period of 31, that is generated by initial polynomial $X^5 + X^2 + 1$ and initial state $(1, 1, 1, 1, 1)$. See [11] for more information about the Simeck family of lightweight block ciphers and its specifications.

2.2 Cube Attack

During the steps of a classic cube attack, introduced in [9], the target cipher is considered as a black box polynomial p over $GF(2)$ with n bits of input variables $(x_1, ..., x_n)$ and one single target output bit. It is assumed that the polynomial is of the ANF[6] form. The $(x_1, ..., x_n)$ are plaintext and master key variables. Since the power operation has no effect over $GF(2)$ ($x_i^2 = x_i \ mod\ 2$), each monomial t_I can be shown by the subset $I \subseteq \{1, ..n\}$ of multiplied variables. Although the explicit representation of the cipher's polynomial is assumed to be unknown to the attacker, the general representation is as follows:

[4] Linear Feedback Shift Register
[5] A **maximum length sequence (MLS)** is a type of pseudorandom binary sequence
[6] Algebraic Normal Form

$$p(x_1, ..., x_n) = (t_I \odot p_{S(I)}) \oplus q(x_1, ...x_n) \tag{4}$$

t_I is a common factor in the master polynomial p, and $p_{S(I)}$ is called a *superpoly* for I in p. For every p and every I this *superpoly* is a polynomial that does not have any common variable with t_I. Each $q(x_1, ...x_n)$ lacks at least one variable from I. Also, a *maxterm* of p is defined as a t_I wherein $degree(p_{S(I)}) = 1$, meaning that the resulting *superpoly* of I in p is a non-constant and linear polynomial. Each subset of I with size k, defines a k-dimension *cube* with 2^k vectors named C_I. During the attack, a subset I of plaintext variables are selected, all the possible 0/1 values are assigned to the variables indexed by I to generate the C_I vectors and the resulting assignments are encrypted, to get values of the target bit of ciphertexts. Next, summing the output bit values of all these vectors results in the new polynomial p_I. The following theorem from [9] explains this polynomial, and the main observation that the cube attack benefits from, to reach out a lower degree representation of the cipher polynomial.

Theorem 1 [9]. For any polynomial p and subset of input variables I, $p_I \equiv p_{S(I)}$ *modulo* 2.

The attack has two main phases: **Preprocessing**, and **Online**. In preprocessing phase, attacker has access to plaintext, ciphertext, and key variables, and tries to find linear equations on key bits. For this purpose, a random walk search algorithm is used to select a size k, and a subset I of plaintext variables so that $|I| = k$. Then, the attacker uses BLR linearity test introduced in [5] with different pairs of master keys $x, y \in \{0, 1\}^{|K|}$ for a sufficient number of times (at least 100 times as in [9]) to find if the specified t_I is a *maxterm*, so it results in a linear *superply*. The BLR test evaluates following equality as a characteristic of linear polynomials to check the linearity of the input polynomials:

$$p_{S(I)[0]} \oplus p_{S(I)[x]} \oplus p_{S(I)[y]} = p_{S(I)[x \oplus y]} \tag{5}$$

If the search algorithm finds linear *superpolys*, each of these *superpolys* give one bit information about key bits. If the subset I is smaller than required, it results in higher order *superpolys*, the linearity test fails, and we have to add one another variable to the subset I of indexes. If the subset I is larger than required, the resulting *superpoly* is a constant, and we need to decrease the number of variables on the subset I. Proper I should be found in the boundary between these two states, and if it is not detected, the attacker should start the process with another initial subset. When a *maxterm* is found, the attacker pursues the procedure explained in the following theorem from [9] to find the algebraic representation of the *superpolys* resulted from *maxterms*.

Theorem 2 [9]. Let t_I be a *maxterm* in a black box polynomial p. Then:

- **1.** The free term in $p_{S(I)}$ can be computed by summing modulo 2 the values of p over all the inputs of n variables which are zero everywhere except on the variables indexed by I in the summation cube C_I.
- **2.** The coefficient of key bit x_j in the linear expression $p_{S(I)}$ can be computed by summing modulo 2 all the values of p for input vectors which are zero everywhere except on the summation cube C_I and all the values of p for input vectors which are zero everywhere except on the summation cube and at x_j which is set to 1.

We refer an interested reader to [9] for proof of the theorems **1** and **2**.

In online phase, key variables have unknown values, and the attacker uses the linear *superpolys* found in the preprocessing phase, to get value of the secret key bits.

2.3 Cube Tester

Cube tester was introduced in [1]. In contrast to cube attack that extracts key bits, cube testers detect non-random behaviors. Similar to cube attacks, subset of I with size k, defines a k-dimension *cube* with 2^k vectors named C_I. During the test, a subset I of plaintext variables are selected, all the possible 0/1 values are assigned to the variables indexed by I to generate the C_I vectors and the resulting assignments are encrypted, to get values of the target bit of ciphertexts. Next, summing the output bit values of all these vectors results in the new polynomial p_I, which is target of the tests. These main steps are involved during the cube testing:

i a subset I of plaintext variables are selected,
ii all the possible 0/1 values are assigned to the variables indexed by I to generate the C_I vectors,
iii the resulting assignments are encrypted, to get values of the target bit of ciphertexts,
iv all the computed values of the ciphertext bit are summed to find the output value of the cube, and
v some *property-testers* are applied to evaluate whether the result is distinguishable from a random oracle or not.

Cube tester is totally based on *property-testers*. As paper [1] claims, it is one of the first explicit applications of property testing in cryptanalysis. They have introduced several cube testers such as; balance, constantness, low degree, and so on. As mentioned in Section 1, in this paper, we take advantage of *constantness* and *linearity* properties in the proposed cube search algorithm.

2.4 SMT-based Attack

SMT[7] is someway a generalization of SAT[8] problems, so that in addition to the Boolean variables used in SAT problems, also other non-binary variables, interpreted and uninterpreted functions, and predicates are allowed, and the type of these variables and predicates depends on the theories. The general idea behind the SMT Solvers is to provide the solver with more information about the problem environment through the theories, to help it to solve the problems more easily. Several different theories have been implemented by the SMT solvers so far, including *Bit-Vectors*, *Linear Integer/Real Arithmetics*, *Arrays*, *Tuples*, and others.

Most of SMT solvers follow the SMT-LIB version 2 format [3], while the problems should be defined in this standard format in order to be evaluated by SMT

[7] Satisfiability Modulo Theories
[8] Boolean Satisfiability

solvers. Among the defined SMT theories, *Bit-Vectors* is suitable in analysis of cipher algorithms. It supports fixed size bit vector variables, functions for extracting specific bits of a vector, addition, rotation, logical bit-wise operators, comparison, and others.

In [2], a framework was developed based on SMT solvers to automatise the analysis of differential propagation in the NORX authenticated encryption scheme. Similarly, [13] implements a framework based on SMT solvers to find the provably best differential and linear characteristics for various instantiations of SIMON lightweight block cipher.

Moreover, a simple procedure can be used to define the cryptanalysis problem in the SMT format, and solve it directly by SMT solvers. The procedure has three main phases:

I Define the cipher in *Bit-Vectors* theory, with plaintext/ciphertext pairs as input variables, and master key bits as output variables.
II Set the plaintext/ciphertext variables defined in *Bit-Vectors* theory by some previously generated plaintext/ciphertext pairs (with unknown adjusted master key), and let the master key variables to remain unknown in the SMT definition of the problem.
III Provide an SMT solver with the generated SMT constraints and call it to solve the problem. Then, a true model of the problem is obtained, which reveals the assigned values for the unknown master key.

As stated in Section 1, paper [3] applies this procedure to directly retrieve the secret keys of reduced-round versions of the Shuffle block cipher, which was introduced based on *swap-or-not shuffle* technique [12]. In the following sections, we illustrate a more efficient application of the described procedure, to improve the success of SMT-based attacks.

3 Cubes with Most Number of Characteristics

In classic cube attack, it is assumed that the attacker has access to a black-box polynomial of the target cipher. However, based on the Kerckhoffs's principle, "a cryptosystem should be secure even if everything about the system, except the key, is public knowledge". So, if we assume the attacker knows the structures of the target cipher, (s)he can exploit the information about its internal structure to simplify the attack. Middle state cube characteristics could be considered as an example of such information, so that the attacker can use this extra information when attacking a cryptosystem.

The paper [10] illustrates the fact that cube PCs can do the attacks more efficiently than randomly selected PCs. In fact, the authors chose correlated messages based on an algebraic high order differential. In addition, in paper [19], ElimLin attack on some cubes behaves better than other cubes, and this behavior is invariant to affine transformation. However, no criteria for cube selection was introduced.

In spite of previous works, we introduce an algorithm which can find the good cubes automatically while applying some specific criteria. The motivating idea behind proposing this new algorithm is that a cube with more number of middle state characteristics would have more useful information for attack than a random cube.

Algorithm 1 is a search algorithm which aims to find the cube with more number of middle state bits having cube characteristics, through testing all the middle state bits starting from the first round of the cipher. The included tests are *constantness* and *linearity* from [1], as they have relatively low time complexity, and are easily applicable to the attacks we'll introduce in next section.

Algorithm 1 Algorithm to find cubes with most number of characteristics.

1: **procedure** SEARCH(S, T, B, R)
2: $\quad \overrightarrow{CC_{BSF}} = \emptyset$
3: $\quad C_{BSF} = \emptyset$
4: $\quad$ **while** $time < T$ **do**
5: $\quad\quad \overrightarrow{CC_T} = \emptyset$
6: $\quad\quad C_T = ALG_{Rand}(S)$
7: $\quad\quad$ **for** $r \in R$ **do**
8: $\quad\quad\quad$ **for** $b \in B$ **do**
9: $\quad\quad\quad\quad \overrightarrow{CC_T} \xleftarrow{+} ALG_{Test}(C_T, r, b)$
10: $\quad\quad\quad$ **end for**
11: $\quad\quad$ **end for**
12: $\quad\quad$ **if** $|\overrightarrow{CC_{BSF}}| < |\overrightarrow{CC_T}|$ **then**
13: $\quad\quad\quad C_{BSF} = C_T$
14: $\quad\quad\quad \overrightarrow{CC_{BSF}} = \overrightarrow{CC_T}$
15: $\quad\quad$ **end if**
16: $\quad$ **end while**
17: $\quad$ **return** C_{BSF} **and** $\overrightarrow{CC_{BSF}}$
18: **end procedure**

The algorithm accepts the size of the cube, the time limit, the state block bit indexes and rounds of the cipher as input, and searches for the top cube of size S in a limited time T by testing all state bits B of all rounds R. In line 6, a temporary cube C_T of size S is being generated randomly by algorithm $ALG_{Rand}(S)$. In line 9, algorithm $ALG_{Test}(C_T, r, b)$ tests state bit b for round r for the temporary cube C_T to see if it finds *Constantness* or *Linearity* properties, and if successful, it adds the characteristic to a temporary list named $\overrightarrow{CC_T}$ (we denote $\xleftarrow{+}$ for add to list operation, for convenience). The $ALG_{Test}(C_T, r, b)$ is repeated on all state bits B for all rounds R. Then, in line 12, it checks if the temporary cube C_T reaches to a more number of characteristics than the best cube so far C_{BSF}. If yes, it updates the C_{BSF} and $\overrightarrow{CC_{BSF}}$ to this newly found top cube and its list of characteristics. The algorithm repeats this procedure until the time runs out. Finally, it returns C_{BSF} and $\overrightarrow{CC_{BSF}}$ as top found cube and its characteristics.

For the *Simeck32/64*, we ran the algorithm for cubes of size 1 to 10, in time limits of 1 hour to 1 week, varying regarding the size of the cube. Moreover, since these small cubes have very low probability to find characteristics in a higher number of rounds of *Simeck32/64*, we set the maximum number of rounds to 10, to speed up the running time of the algorithm for each instance. The experiments were performed on 64-bit Ubuntu 14.04.5 with i5-4460 (3.20GHz) processor and 16 GB memory. Moreover, the implemented version of algorithm does the cube computations in parallel to take advantage of all cores of the CPU.

Table 1 represents the found cubes of size 1 to 10 with the most number of characteristics. First three columns show the cube size and indexes, and the number of PC pairs respectively. Columns 4, 5, and 6 show number of zero constant,

Table 1 Cubes Found with Most Number of Characteristics in Simeck32/64

Size	Indexes	#PCs	Zeros	Ones	Linears	Total
1	$P_{\{19\}}$	2^1	112	8	4	124
2	$P_{\{6,23\}}$	2^2	172	0	0	172
3	$P_{\{4,19,21\}}$	2^3	196	0	0	196
4	$P_{\{3,5,20,22\}}$	2^4	212	0	0	212
5	$P_{\{11,13,15,16,30\}}$	2^5	220	0	0	220
6	$P_{\{3,5,18,20,22,26\}}$	2^6	226	0	0	226
7	$P_{\{1,3,5,18,20,22,27\}}$	2^7	230	0	0	230
8	$P_{\{2,3,5,6,19,20,22,23\}}$	2^8	234	0	0	234
9	$P_{\{3,4,5,6,7,21,22,23,24\}}$	2^9	244	0	0	244
10	$P_{\{3,4,5,7,11,18,21,22,24,28\}}$	2^{10}	244	0	0	244

one constant, and linear middle state bits with cube characteristics, respectively. The last column shows the sum of three previous columns, *i.e.* number of all found characteristics. As the table shows, the number of found characteristics increases by size from 124 to 244. Moreover, all the found characteristics are zero constants except for top cube of size 1.

Let's examine the found middle state cube characteristics for the top cube of size 1 as an example. Figure 1 illustrates this information inside the internal structure of the *Simeck32/64*. L_i, R_i and K_i in the figure represent the left side and right side state blocks and round key for i^{th} round of the cipher and f represents the f-function. The green, blue and pink squares show zero constant, one constant, and linear middle state bits with cube characteristics, found by Algorithm 1. Also, the red square shows the cube bit index.

As it is shown in Figure 1, sum over two cube PCs of $I = \{19\}$ gives us 112 zero, 8 one, and 4 linear characteristics. Moreover, We have observed two important facts in the results:

- the left side characteristics are always substituted to the right side of the next round, which is direct effect of the Feistel network.
- the R_1 middle state cube characteristics are just cube summations over the input plaintexts, as they directly come from L_0 without any key variables or f-function being involved.

So, these middle state cube characteristics are redundant and have no valuable information usable for an SMT-based attack. We'll benefit from these observations in next section to avoid unneeded extra cost constraints in the SMT-based cube attack.

4 SMT-based Attacks on Simeck32/64

In this section, first, the basic procedure stated in subsection 2.4 is followed to reveal secret key bits of *Simeck32/64* by substituting the cipher's input variables in the SMT definition by 2^2 to 2^9 number of randomly generated plaintexts and their equivalent ciphertexts, to see performance of the basic attack. Since the 2^1 number of PC pairs do not have enough information to identify a unique master key, and 2^{10} number of PC pairs result in a big SMT problem with high number

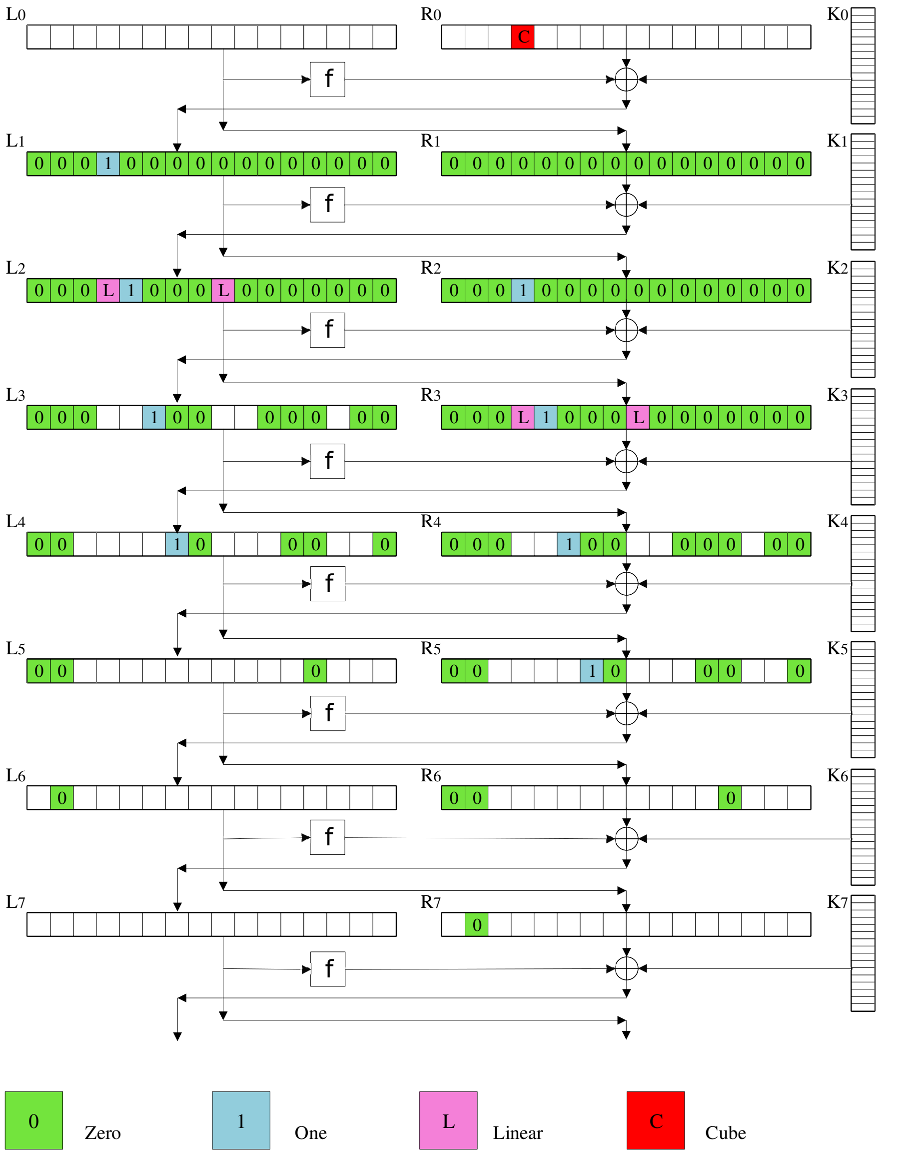

Fig. 1 Middle State Cube Characteristics Found on *Simeck32/64* by Algorithm 1 ($I = \{19\}$).

of unknown variables, they were excluded from the tests. Next, the success of the attack is evaluated by PC pairs of random cubes and also top found cubes from the new search algorithm. Finally, the middle state cube characteristics are added to the SMT definition of the problem to evaluate its effect on the success of SMT-based attacks.

Running time for each instance of the attack was limited to 1000 seconds, 50 instances were run, and the average time was calculated for the instances solved

Table 2 SMT-based Attacks with Random PCs on Simeck32/64

Number of PCs	7 *rounds*	8 *rounds*	9 *rounds*
2^2	3.58	240.71 {12}	$\perp$
2^3	0.78	73.22	137.67 {1}
2^4	0.42	24.60	451.29 {2}
2^5	0.45	199.14	$\perp$
2^6	0.67	331.08	500.44 {2}
2^7	1.75	184.50	571.50 {4}
2^8	2.50	64.86	304.95 {14}
2^9	5.99	59.65	363.48 {19}

in the limited time. For implementation, the Z3py[9] was used to generate problem constraints, and the Microsoft Z3 SMT solver [15] version 4.4.2 was employed to solve the resulting constraints. All the attacks were performed on the same computer as Section 3, though the SMT solver was not run in parallel.

4.1 SMT-based Attack with Random PCs

In this subsection, we evaluate the performance of the basic SMT-based attack. In the first step, we defined the components of the cipher in *Bit-Vectors* theory of the SMT format by using Z3py API. Next, we provided the Z3 SMT solver with 2^2 to 2^9 number of random PC pairs which were generated by *Simeck32/64* with the random adjusted master keys, and then queried the solver to solve the resulting problem. This procedure was repeated 50 times and the average spent time for solved instances was calculated.

Table 2 illustrates the results. First column shows number of used PC pairs and three next columns illustrate the spent time in seconds for the solver to return a true model of master key variables, for 7, 8, and 9-round *Simeck32/64*. Also, the number inside the braces shows the number of solved instances out of 50. As can be seen in the table, the classic SMT-based attack can solve at most 19 instances out of 50, for 9-round *Simeck32/64* with random PC pairs of size 2^9. This is not acceptable as a successful attack. Therefore, the attack on 8 rounds of the cipher which solves all the 50 instances by 2^4 PC pairs in 24.60 seconds, is considered as the best result of the attack.

4.2 SMT-based Attack with Random Cube PCs

This subsection evaluates success of SMT-based attacks with random cube PCs instead of random PCs. To do so, 50 instances of randomly selected cube plaintexts for each size 2^2 to 2^9 were provided, and the equivalent ciphertexts were generated by the *Simeck32/64* with random master keys. Then, we substituted these PC pairs for the input variables of the SMT definition, to see its effect on the attack.

Table 3 shows the results. Based on the results, the attack can reveal secret keys of 12-round cipher at most in 5 instances for random cubes of size 7. However,

[9] The Z3 API in Python.

Table 3 SMT-based Attacks with Random Cube PCs on Simeck32/64

Number of PCs	10 *rounds*	11 *rounds*	12 *rounds*
2^2	399.26 {21}	⊥	⊥
2^3	68.58 {48}	110.93 {15}	⊥
2^4	7.26 {47}	193.06 {19}	869.94 {2}
2^5	5.72 {49}	185.52 {36}	510.17 {4}
2^6	2.45	148.37 {31}	359.49 {1}
2^7	3.14	200.58 {41}	426.48 {5}
2^8	604.94 {4}	⊥	⊥
2^9	⊥	⊥	⊥

Table 4 SMT-based Attacks with Chosen Cube PCs on Simeck32/64.

Number of PCs	10 *rounds*	11 *rounds*	12 *rounds*
2^2	11.45	553.80 {9}	⊥
2^3	4.44	22.27 {48}	⊥
2^4	0.31	1.64	504.97 {18}
2^5	0.52	2.88	451.15 {31}
2^6	0.90	7.80	626.13 {32}
2^7	1.57	6.18	477.64 {46}
2^8	3.23 {46}	10.41 {1}	⊥
2^9	643.73 {2}	–	⊥

an 11-round attack is successful in 41 instances out of 50 with the same size of cube, and consequently is considered as the best attack.

4.3 SMT-based Attack with Chosen Cube PCs

Now, the cubes found by Algorithm 1 are used for the attack. As explained in Section 3, we ran the algorithm to find the top cube of each size, based on the number of *Constantness* and *Linearity* characteristics. For each of the top found cubes in Table 1, 50 SMT instances of the *Simeck32/64* cryptanalysis problem with random unknown master keys were generated, and the instances were run by the solver for the attack. Table 4 shows the results.

This attack was successful on 11-round Simeck32/64 for all 50 instances of top found cube of size 4 with the cube indexes $I = \{3, 5, 20, 22\}$ in just 1.64 seconds, which is better than the attack with random cubes. Moreover, the top cube of size 7 with cube indexes $I = \{1, 3, 5, 18, 20, 22, 27\}$ results in a successful attack on 12 rounds, in 46 instances in 477.64 seconds, so is considered as best known result with this type of attack.

4.4 SMT-based Attack with Chosen Cube PCs and Characteristics

In this subsection, the success of the attack was considered where in addition to PC pairs of the top cube, also its found characteristics were fed to the SMT definition of the problem, which we name ***SMT-based Cube Attack***. For this purpose, we first identified the useful characteristics. As mentioned in Section 3,

Table 5 SMT-based Attack with Chosen Cube PCs and Characteristics on Simeck32/64.

Number of PCs	10 *rounds*	11 *rounds*	12 *rounds*
2^2	5.50	447.66 {9}	$\perp$
2^3	1.07	17.73	$\perp$
2^4	0.42	1.53	409.44 {24}
2^5	1.11	3.21 {49}	169.53 {45}
2^6	1.09	7.51	122.17
2^7	1.96	5.50	341.46 {49}
2^8	500.24 {17}	$\perp$	$\perp$
2^9	501.41 {12}	$\perp$	$\perp$

two types of the characteristics are not beneficial enough to be used in the attack. So, we excluded the middle state cube characteristics for right side state block R_1 and left side state blocks L_i. Then we fed the remained characteristics to the SMT definition of the problem when the PC pairs of top cubes were set as input. Table 5 illustrates results of the attack with the explained settings.

Based on the results, the attack on 12-round Simeck32/64 is successful in all 50 instances for cube of size 6 with cube indexes $I = \{3, 5, 18, 20, 22, 26\}$ in just 122.17 seconds, which is the best known attack result, proposed in this paper.

4.5 Evaluations

First, the SMT-based attack with random PCs attacks up to 8 rounds of Simeck32/64 for all 50 instances in 24.60 seconds, on average. Next, when random cubes were used, the attack improves to 11 rounds for 41 out of 50 instances of the problem in 200.58 seconds, which improves the attack. However, when we use the top cubes found by the Algorithm 1, the SMT solver can reveal secret key bits for the 12-round Simeck32/64 for 46 out of 50 instances in 477.64 seconds, which outperforms the previous attack. Finally, when the middle state cube characteristics of the top found cubes are fed to the SMT definition of the problem, the solver is able to find the secret key values of the 12-round Simeck32/64 for all 50 instances, on average in just 122.17 seconds, that is the best result among the attacks.

So, the results confirm two claims of the paper: I) the attacks with top cubes found by the proposed algorithm outperform the random cubes, so our proposed algorithm and criteria are good indicators and beneficial methods for cube selection. and II) also the middle state cube characteristics are useful information for the solver, and improve the attacks.

Moreover, we observed some important points in these attacks:

- When size of the cube increases, it brings us with much more information about the problem environment, *e.g.* Algebraic equations. However, it increases the volume of inputs and unknown variables simultaneously, which makes it hard for the SMT solver to handle.
- In contrast, when size of the cube decreases, it needs fewer PC pairs, and handling of the related amount of data is easier in the SMT solver. Nevertheless, it provides us with much less information, which diminishes the success of the attacks.

5 Cube Attack on Simeck32/64

We mounted the classic cube attack described in 2.2 on *Simeck32/64* lightweight block cipher, to find if the cipher is vulnerable to it, and also compare the cube attack with SMT-based attacks. We implemented the classic cube search algorithm in parallel to take advantage of all cores of the CPU. Then we ran the search algorithm on the same computer as Sections 3 and 4 for one month to find linear cube characteristics on reduced-round versions of the cipher.

The results indicate that the attack can proceed up to 10 rounds of the cipher. During the preprocessing phase of the attack, we found multiple 10-round *maxterms* with cubes of size at least 25, and selected 19 smallest cubes to make the data complexity feasible (less than half of the possible data). Among them, two cubes are of size 25, four cubes are of size 26, and thirteen cubes are of size 27. Thus, the data and time complexities of the online phase of the attack are as follows:

- $C_{data} = 2 \times 2^{25} + 4 \times 2^{26} + 13 \times 2^{27} \approx 2^{30.954}$,
- $C_{time} = 2^{30.954} + 2^{64-19} \approx 2^{45}$.

In fact, we need around 2^{31} plaintext/ciphertext pairs (half of the entire pairs) of the *Simeck32/64* with the unknown adjusted master key to start the online phase of the attack. Then we compute the cubes to solve the equalities and get one bit information from each *maxterm* equation. This procedure reveals 19 out of 64 secret key bits. Next, we can find the remaining 45 key bits through exhaustive search. All in all, the cube attack decreases the time complexity in a brute force attack on 10-round *Simeck32/64* by a factor of around 2^{19}.

It seems that the classic cube attack is more successful than the basic SMT-based attack, though the cube attack has an enormous time and data complexity, and the SMT-based attack would predominate if it takes same amount of time. However, obviously we reach in better results on more rounds of the cipher, when the two methods were combined together, by using cube PC pairs and their characteristics in the SMT definition of the problem.

6 Conclusion & Future Works

In this paper, we first proposed a new search algorithm which is able to find top cubes with the most number of constant and linear middle state characteristics. Next, we took advantage of the top cubes and their characteristics found by this new algorithm to improve SMT-based attacks on *Simeck32/64* lightweight block cipher. Moreover, the classic cube attack was conducted on the same cipher to compare with the results. Table 6 summarizes these results.

The paper introduced following original contributions:

- proposing a novel algorithm to search for cubes with most number of middle state characteristics (constant or linear) on a cipher.
- presenting the first algebraic attacks on *Simeck32/64*. As far as we know, there is no other SMT-based, cube, or algebraic attacks against the Simeck family of lightweight block ciphers in the research literature, so far.

Table 6 Summary of the proposed attacks on the *Simeck32/64*

Attack	Rounds	Data	Time
SMT-based Cube Attack	12	2^6	122.17 (s)
SMT-based Attack with Chosen Cube PCs	12	2^7	477.64 (s){46}
SMT-based Attack with Random Cube PCs	11	2^7	200.58 (s){41}
Cube Attack	10	$2^{30.954}$	2^{45}
SMT-based Attack with Random PCs	8	2^4	24.60 (s)

– improving the success of SMT-based attacks on more rounds of the cipher, through its combination with cube PCs and their middle state characteristics found using the new search algorithm, as extra information fed to the solver.
– in addition to these findings, in subsection 6.1, we make some recommendations for future research directions in this area of study.

Moreover, our implementation codes for the proposed cube search algorithm, classic cube search algorithm, SMT-based attacks, and linear equations for 10-round cube attack, as well as tests for some sample linear cube equations of 4-round to 10-round *Simeck32/64* are available on GitHub [20].

6.1 Future Works

There are some ideas that can be considered as future work, and probably result in enhancement of the proposed attacks:

- We used *Constantness* and *Linearity* properties in the proposed search algorithm. However, other combinations of cube properties like linear only, or low-degree middle state cube characteristics would also be helpful for the attack. Thus, as a future work, it would be considered to evaluate other combinations of these properties in the search algorithm, and measure their success in SMT-based attacks.

- We performed the SMT-based attacks on a single core of CPU. As a future work, the load of the attack can be shared among multiple instances of the solver in parallel, or an SMT solver with the support of parallelism can be used.

- We didn't use key guessing technique because our focus was only on evaluating the success of the top found cubes on SMT-based attacks. However, it would be a useful method to evaluate the proposed attacks on more rounds of the cipher, as it results in a much simpler SMT problem by reducing the number of unknown variables.

- We used Microsoft Z3 SMT solver for the SMT-based attacks. There are multiple other SMT solvers like SONOLAR [14] and BOOLECTOR [6], which have good performance on problems defined in *Bit-Vectors* theory. So, it would be good to compare the success of SMT-based Cube Attack in presence of different SMT solvers.

- In this paper, the focus was on analyzing the success of SMT solvers in cryptanalysis. However, the proposed ideas can also be used in other types of attacks like Gröbner basis algorithms or SAT solvers.

- Attacking multiple other cipher algorithms is helpful to make a more general verification of the proposed ideas. So it could be considered as a future work to test success of the proposed attacks on a list of recently introduced cipher algorithms.

- In Section 5 we selected top 19 cubes with smallest sizes to make the data complexity of cube attack feasible. However, there are plenty of data overlaps in cube computations. These overlaps could help us to decrease the data and time complexities. Thus, as another future work, in online phase, we can calculate the overlaps, save them in memory, and reuse them for computations of other cubes. This way, not only is the exact amount of data needed for the attack determined, but also it could reduce the overall time complexity of the attack by probably enabling us to use more cubes by the same amount of data.

References

1. Aumasson, J.P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Fast Software Encryption, pp. 1–22. Springer Berlin Heidelberg (2009)
2. Aumasson, J.P., Jovanovic, P., Neves, S.: Analysis of NORX: investigating differential and rotational properties. In: International Conference on Cryptology and Information Security in Latin America, pp. 306–324. Springer (2014)
3. Barrett, C., Stump, A., Tinelli, C.: The SMT-LIB Standard: Version 2.0. In: Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK) (2010)
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. In: Cryptology ePrint Archive, Report 2013/404. `http://eprint.iacr.org/` (2013)
5. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences **47**, 549–595 (1993)
6. Brummayer, R., Biere, A.: Boolector: An efficient SMT solver for bit-vectors and arrays. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 174–177 (2009)
7. Cid, C., Murphy, S., Robshaw, M.J.: Small Scale Variants of the AES. In: FSE, pp. 145–162. Springer (2005)
8. Daemen, J., Rijmen, V.: The design of Rijndael: AES-The advanced encryption standard. Springer Science & Business Media (2013)
9. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 278–299. Springer Berlin Heidelberg (2009)
10. Faugère, J.C., Perret, L.: Algebraic Cryptanalysis of Curry and Flurry Using Correlated Messages. In: Information Security and Cryptology, vol. 6151, pp. 266–277. Lecture Notes in Computer Science (2010)
11. Gangqiang, Y., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck family of lightweight block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 307–329. Springer Berlin Heidelberg (2015)
12. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Advances in Cryptology-CRYPTO 2012, pp. 1–13. Springer (2012)
13. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Annual Cryptology Conference, pp. 161–185. Springer (2015)
14. Lapschies, F., Peleska, J., Gorbachuk, E., Mangels, T.: SONOLAR SMT-Solver. System Desc. In: SMT-COMP12 (2012)
15. Moura, L.D., Bjørner, N.: Z3: An efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 337–340. Springer (2008)
16. Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., Roback, E.: Report on the development of the Advanced Encryption Standard (AES). Journal of Research of the National Institute of Standards and Technology 106, no. 3 p. 511 (2001)
17. Standard, D.E.: Federal information processing standards publication 46. National Bureau of Standards, US Department of Commerce (1977)

18. Stanek, M.: Experimenting with Shuffle Block Cipher and SMT Solvers. In: Cryptology ePrint Archive, Report 2014/919. `http://eprint.iacr.org/` (2014)
19. Sušil, P., Sepehrdad, P., Vaudenay, S., Courtois, N.: On selection of samples in algebraic attacks and a new technique to find hidden low degree equations. International Journal of Information Security pp. 51–65 (2016)
20. Zaheri, M., Sadeghiyan, B.: GitHub project for SMT-based Cube Attacks on Simeck32/64. `https://github.com/m0jt4b4/SimeckCubeSMT` (2017)