SCIENCE CHINA
Information Sciences

• LETTER •

January 2022, Vol. 65 119101:1–119101:3
https://doi.org/10.1007/s11432-019-2742-6

Cube attacks on round-reduced MORUS and GIMLI

Siwei CHEN, Zejun XIANG*, Xiangyong ZENG & Shasha ZHANG

Faculty of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan 430062, China

Received 2 August 2019/Revised 10 November 2019/Accepted 23 December 2019/Published online 7 September 2021

Citation Chen S W, Xiang Z J, Zeng X Y, et al. Cube attacks on round-reduced MORUS and GIMLI. Sci China Inf Sci, 2022, 65(1): 119101, https://doi.org/10.1007/s11432-019-2742-6

Dear editor,

Cube attack is a chosen plaintext/initial value (IV) key-recovery attack, which was proposed by Dinur and Shamir at EUROCRYPT 2009 [1]. As an important cryptanalysis technique, it is widely used in symmetric ciphers. The main ideal of cube attack is to look for cubes to acquire several linear polynomials (superpolys) with respect to secret variables. The secret variables can be recovered by solving these linear equations. However, thus far, no general method has been established to search for cubes.

Contribution. In this study, we propose a new method to automatically search for candidate cubes that can derive linear superpolys with a high probability. First, we utilize the same strategy as that of conditional cube attack [2] to divide an r-round cipher into two stages: the first r_1 rounds and the last r_2 rounds with $r_1 + r_2 = r$. Then, we analyze the algebraic normal form (ANF) of the first r_1-round output and attempt to make it linear by adding some conditions to IV, and we evaluate the algebraic degree of the last r_2-round output using the division property [3–5]. Moreover, we introduce Algorithm 1 to retrieve the high-degree monomials of the r_2-round output and Algorithm 2 to search candidate cubes. Note that the variables in each of the high-degree terms correspond to a set of positions of the r_2-round input that can be treated equivalently as r_1-round output. Thus, if we select cube variables from the ANFs of r_1-round output, the the resulting superpolys have a high probability to be linear.

Our method to search cubes. For a given r-round cipher, we divide it into two stages: $E_r = E_{r_2} \circ E_{r_1}$, i.e., $r_1 + r_2 = r$. We denote $\boldsymbol{s}^0$ and $\boldsymbol{s}^{r_1}$ as the input and output of E_{r_1} and $\boldsymbol{s}^{r_1+r_2}$ as the output of E_r. Moreover, $\boldsymbol{s}^0$ is the initial state that comes with secret and public variables. In this case, the r-round encryption can be represented as $\boldsymbol{s}^0 \xrightarrow{E_{r_1}} \boldsymbol{s}^{r_1} \xrightarrow{E_{r_2}} \boldsymbol{s}^{r_1+r_2}$. Therefore, if we add some conditions to the first stage to make E_{r_1} linear, the degree of E_r will decrease and be equivalent to the degree of E_{r_2}, and the cube attacks can cover more rounds.

Stage 1. We set all public variables (except one) involved in each nonlinear term of $\boldsymbol{s}^{r_1}$ to constant values for all nonlinear terms to be removed. Note that the public variables set to constant values at this stage can no longer be selected as cube variables. This requires that the number of rounds r_1 is chosen such that the degree of ANF of $\boldsymbol{s}^{r_1}$ is low. Otherwise, too many public variables may be set to constant values, and the remaining free public variables may be inadequate to devise a cube attack.

Stage 2. Each coordinate of $\boldsymbol{s}^{r_1+r_2}$ is a polynomial over $\boldsymbol{s}^{r_1}$, and we use a bit-based division property to estimate the degree of E_{r_2}. Then, we use Algorithm 1 for a fixed bit of $\boldsymbol{s}^{r_1+r_2}$ to retrieve all high-degree terms that may be involved in its ANF. Note that the algebraic degree of E_{r_2} has to be of practical size because cube attack needs to experimentally test the linearity of the superpolys, and the cube size is roughly equal to the degree.

Algorithm 1 Return the monomials of degree d of the ith bit position

```
Input: The position bit $i$, degree $d$ and an MILP model $\mathcal{M}$1)
Output: A set $\mathcal{N}$ of all the monomials of degree $d$.
1: $\mathcal{N} = \emptyset$;
2: $\mathcal{M}$.con $\leftarrow \sum_{j=0}^{m-1} a_j^0 = d$;
3: while $\mathcal{M}$ is feasible do
4:     $\mathbb{N} = \emptyset$;
5:     $\boldsymbol{x} = (x_0, \ldots, x_{m-1}) = (0, \ldots, 0)$;
6:     $\mathcal{M}$.optimize();
7:     for each $j \in \{0, \ldots, m-1\}$ do
8:         var = $\mathcal{M}$.getVarByName($a_j^0$);
9:         if var.getAttr('$x$')= 1 then
10:            $\mathbb{N} = \mathbb{N} \bigcup \{j\}$;
11:            $x_j = 1$;
12:        end if
13:    end for
14:    Add $\mathbb{N}$ into $\mathcal{N}$;
15:    $\mathcal{M}$.con $\leftarrow \sum_{j=0}^{m-1} (-1)^{x_j+1} \cdot a_j^0 \leqslant d - 1$;
16:    $\mathcal{M}$.update();
17: end while
18: return $\mathcal{N}$.
```

Let us focus on the ith bit of $\boldsymbol{s}^{r_1+r_2}$. Assume that the

* Corresponding author (email: xiangzejun@hubu.edu.cn)

1) This model is developed based on the model of searching integral distinguishers in [5]. In $\mathcal{M}$, the output division vector is fixed to a unit vector, and the initial division vector is unknown. However, the weight of the initial division vector will be fixed in Algorithm 1.

Table 1 Results on MORUS and GIMLI

Cipher	Attack type	Method	Steps or rounds	Time	Source
MORUS-640-128	Key-recovery	Cube.A Div.p[a)]	3/16	2^{117}	Ref. [6]
	Key-recovery	Cube.A Div.p	4/16	2^{119}	Ref. [6]
	Key-recovery	Cube attack	4.4/16	$2^{27.91}$	Our result
	Distinguishing	Linear cryptanalysis	16/16	2^{76}	Ref. [7]
GIMLI	Key-recovery	Cube attack	7/24	$2^{27.06}$	Our result

a) Cube attacks based on division property [8].

degree of $s_i^{r_1+r_2}$ is d and let the set $\mathbb{N}$ be the variable indices of a degree d term of $s_i^{r_1+r_2}$ returned by Algorithm 1. Then, the ANF of $s_i^{r_1+r_2}$ contains $\Pi_{j\in\mathbb{N}} s_j^{r_1}$. Once done with the linearization process in the first state, each $s_j^{r_1}$ ($j \in \mathbb{N}$) becomes a linear polynomial over public variables, and we can obtain a linear superpoly with a high probability if we select one public variable from each $s_j^{r_1}$ ($j \in \mathbb{N}$) as cube variable. Furthermore, the cube variables are required to be either independent or multiplied with a linear combination of secret variables. Then, we can apply Algorithm 2 to select cube variables for an $(r_1 + r_2)$-round cipher.

Algorithm 2 Return candidate cubes

```
Input: The $r_1$-round state $\boldsymbol{s}^{r_1} = (s_0^{r_1}, \ldots, s_{m-1}^{r_1})$ and the set
  $\mathcal{N}$ returned by Algorithm 1.
Output: A set $\mathcal{C}$ of candidate cubes.
1: $\mathcal{C} = \emptyset$;
2: for each set $\mathbb{N} \in \mathcal{N}$ do
3:     $\mathbb{C}' = \emptyset$;
4:     for each $i \in \mathbb{N}$ do
5:         Add an IPV $v_{I_i}$ of $s_i^{r_1}$ into $\mathbb{C}'$;
6:     end for
7:     $\mathbb{C}'' \leftarrow \mathbb{C}'$;
8:     for each $i \in \mathbb{N}$ do
9:         Replace IPV $v_{I_i}$ by KPV $v_{K_i}$ of $s_i^{r_1}$ in $\mathbb{C}''$;
10:        Add the cube set $\mathbb{C}''$ into set $\mathcal{C}$;
11:        $\mathbb{C}'' \leftarrow \mathbb{C}'$;
12:    end for
13: end for
14: return $\mathcal{C}$.
```

Definition 1. A public variable is called an independent public variable (IPV) if it appears in the ANF of $s_j^{r_1}$ and not multiplied with any other variable (public or secret). A public variable is called a keyed public variable (KPV) if it appears in the ANF of $s_j^{r_1}$ and is multiplied with a linear combination of secret variables.

In the following, we present a toy example to demonstrate the whole procedure briefly.

Example 1. Consider a 3-round cube attack on a 16-bit toy cipher E, where the initial state contains an 8-bit secret key $\boldsymbol{x}$ and an 8-bit IV $\boldsymbol{v}$. Meanwhile, let $\boldsymbol{s}^i = (s_0^i, \ldots, s_{15}^i)$ be the output of ith round encryption. We divide the 3-round encryption into $E = E_2 \circ E_1$, i.e., $r_1 = 1$ and $r_2 = 2$. We focus on s_0^3 and assume that its algebraic degree is 3 (regarding $\boldsymbol{s}^1$). Algorithm 1 returns a set of monomials $\mathcal{N} = \{\{0, 1, 2\}\}$ of Stage 2. This implies that $s_0^1 s_1^1 s_2^1$ is the only possible monomial of degree 3 that appears in the ANF of s_0^3. Suppose that the ANFs of s_0^1, s_1^1, and s_2^1 are given below:

$$\begin{cases} s_0^1 = v_0 + v_1 x_0 + v_6 v_7 + x_0 x_1, \\ s_1^1 = v_2 + v_3 x_3, \\ s_2^1 = v_4 + v_5 x_5 x_6. \end{cases}$$

The first step is to linearize s_0^1, s_1^1, and s_2^1 in Stage 1. It is clear that the algebraic degree of s_0^1 is equivalent to 2, and we have to set one or both v_6 and v_7 to constant values. Thus, there are three methods to linearize s_0^1 that will result in different cubes.

Case 1. Set one or both of v_6 and v_7 to 0, and the set of IPV is $\{v_0, v_2, v_4\}$. Then, the IPV v_0 in s_0^1 is replaced with the KPV v_1 of s_0^1, and this will result in a candidate cube $\{v_1, v_2, v_4\}$. Since the ANF of s_0^3 possibly contains $s_0^1 s_1^1 s_2^1$, $v_1 v_2 v_4 x_0$ will then appear in the ANF of s_0^3 with a high probability. In other words, this cube will have a superpolys of x_0 with a high probability. Similarly, we can get another candidate cube $\{v_0, v_3, v_4\}$ by substituting v_2 with the KPV v_3 of s_1^1. As for s_2^1, no KPV exists, and v_5 cannot be selected as a cube variable because it is multiplied with a nonlinear term $x_5 x_6$ that may lead to a nonlinear superpoly. Thus, in this case, we can find two candidate cubes using Algorithm 2: $\{v_1, v_2, v_4\}$ and $\{v_0, v_3, v_4\}$.

Case 2. Set v_7 to 1, and v_6 is a new IPV of s_0^1 apart from v_0 in this case. Thus, there are two sets of IPV: $\{v_0, v_2, v_4\}$ and $\{v_6, v_2, v_4\}$. For each set of IPV, similar procedure with Case 1 is performed to select candidate cubes. We can identify three candidate cubes: $\{v_1, v_2, v_4\}, \{v_0, v_3, v_4\}$, and $\{v_6, v_3, v_4\}$.

Case 3. Set v_6 to 1, and v_7 is a new IPV of s_0^1 apart from v_0. Then, we determine three candidate cubes: $\{v_1, v_2, v_4\}, \{v_0, v_3, v_4\}$, and $\{v_7, v_3, v_4\}$.

Thus, we have found four candidate cubes using Algorithm 2: $\{v_1, v_2, v_4\}$, $\{v_0, v_3, v_4\}$, $\{v_6, v_3, v_4\}$, and $\{v_7, v_3, v_4\}$. However, we have to confirm that the other 13 bits, i.e., $s_3^1, s_4^1, \ldots, s_{15}^1$ can also be linearized for each candidate cube. If the candidate cube cannot linearize the other bits then this cube will be rejected. Once these 13 bits are done with the linearization procedures, all candidates that survive are sent to the linearity test procedure.

Applications to MORUS[2)] *and* GIMLI[3)]. We apply our attack model to round-reduced MORUS and GIMLI, where there exists no associated data and the nonce can be reused. For MORUS, we achieved a 4.4-steps cube attack with time complexity of $2^{27.91}$, whereas the cube attack in [6] following four steps has a time complexity of 2^{119}. For GIMLI, we also achieved a 7-rounds practical key-recovery attack with time complexity of $2^{27.06}$. The results are summarized in the Table 1. All the experiments were conducted on the following platform: Intel(R) Core(TM) i7-7700k CPU@3.6 GHz,

2) https://competitions.cr.yp.to/round3/morusv2.pdf.

3) https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/gimli-spec-round 2. pdf.

8.00 GB RAM, Windows 10, 64-bit operating system. The source codes are available at the website[4].

Note that all versions of full MORUS have been theoretically attacked in [7, 9] using linear cryptanalysis. But we would like to emphasize that the main advantage of our cube attacks is that it is practical.

Conclusion. This study introduces a new method to efficiently search for cubes in the preprocessing phase of cube attack based on division property. We observed that the high-degree monomials present in the second stage can help the attackers identify cube variables. If the cube variables are selected from the corresponding positions indicated by those high-degree monomials, there is a high probability to result in linear superpolys. For this method to be proven effective, we applied it to two authenticated encryptions, MORUS and GIMLI, and we reached the longest rounds under practical attack scenario for both ciphers.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61802119).

References

1 Dinur I, Shamir A. Cube attacks on tweakable black box polynomials. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, 2009. 278–299

2 Huang S Y, Wang X Y, Xu G W, et al. Conditional cube attack on reduced-round keccak sponge function. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Paris, 2017. 259–288

3 Todo Y, Morii M. Bit-based division property and application to simon family. In: Proceedings of Fast Software Encryption, Bochum, 2016. 357–377

4 Sun B, Hai X, Zhang W Y, et al. New observation on division property. Sci China Inf Sci, 2017, 60: 098102

5 Xiang Z J, Zhang W T, Bao Z Z, et al. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, 2016. 648–678

6 Li Y B, Wang M Q. Cryptanalysis of MORUS. Des Codes Cryptogr, 2019, 87: 1035–1058

7 Shi D P, Sun S W, Sasaki Y, et al. Correlation of quadratic boolean functions: cryptanalysis of all versions of full MORUS. In: Proceedings of International Cryptology Conference, Santa Barbara, 2019. 180–209

8 Todo Y, Isobe T, Hao Y L, et al. Cube attacks on non-blackbox polynomials based on division property. In: Proceedings of International Cryptology Conference, Santa Barbara, 2017. 250–279

9 Ashur T, Eichlseder M, Lauridsen M M, et al. Cryptanalysis of MORUS. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, 2018. 35–64

4) https://github.com/chensivvei/CubeAttack.git.