2016 24th Iranian Conference on Electrical Engineering (ICEE)

Comparing Resistance Against Cube like Attacks

Mojtaba Zaheri Babak Sadeghian

Department of Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic)
Tehran, Iran
{mojtaba.zaheri,basadegh}@aut.ac.ir

Abstract— **In this paper, we propose a new algorithm to compare resistance of symmetric ciphers against cube like attacks. This new algorithm combines two criteria of constantness and low-degree, to achieve a new criterion in order to evaluate the resistance. To check the validity of our algorithm, we investigated many symmetric ciphers that are proposed after 2002, and applied our method on a selected set of them. Then, we compared the results of the algorithm with the cube attacks done on the ciphers. The results correspond well with the experimental results. Next, we evaluated the selected set of the ciphers, and concluded that, MIBS and KLEIN are more resistant than others.**

Keywords- algebraic cryptanalysis; cube attacks; cube tester;

I. INTRODUCTION

Cube attack have been proposed first by Shamir in an invited talk in CRYPTO 2008 conference and then published in [1]. Following this work, many researches have been put on the cube attack, its new variants, and improvements including [2-10]. Over these years cube attack's complexity was the only reliable method in order to compare symmetric ciphers resistance against the attack. Ideas like analyzing cipher's internal structure were not fully trusted. Therefore, the problem of evaluating ciphers resistance against cube like attacks without applying the attack itself, is an open problem. For this purpose, some testers have been proposed in [4], and in some cases, they were useful. In this paper, we propose a new algorithm to evaluate and compare ciphers resistance against cube like attacks, which is not restricted to a specific structure, and can be used in almost all types of symmetric ciphers.

In their first model of the attack, Dinur and Shamir used only cubes with degree one, in corresponding polynomials, to find useful characteristics on cipher's algebraic normal form (ANF) representation. But, further improvements showed that also higher degree polynomials are useful for a successful attack, including [2, 5, 9]. According to these observations, in addition to the increasing cube size, the larger degree of leading superpoly on key bits is also important. Hence, our proposed algorithm uses constantness and low degree cube testers offered in [4] and tests cube's corresponding polynomial's degree in a bounded range.

Organization of the paper. Section II, presents some preliminaries where we introduce cube attack and cube testers. Next, in section III, we describe our new algorithm. In section IV, some evaluation functions are defined to apply to the outputs of the proposed algorithm. Section V, first shows the results of running the algorithm on a set of selected symmetric ciphers, and then evaluates the results with the evaluation functions. Also, results of applying the classical cube attack on two lightweight block ciphers SIMECK and MIBS are shown to compare with the results of the proposed algorithm. Section VI concludes the paper.

II. PRELIMINARIES

A. Cube attack

In cube attack, given m bits of plaintext $V = \{v_1, ..., v_m\}$ and n bits of key $K = \{k_1, ..., k_n\}$, the focus of attack is on one bit of ciphertext. Assume $X = V \bigcup K$, I is a subset of plaintext V, the binary function f of the ciphertext bit can be viewed as the following form:

$$f(X) = t_I . p_{S(I)} + q_I(X) \quad (1)$$

where

- t_I is a monomial, multiple of I variables.
- $p_{s(I)}$ is named superpoly.
- And q_I is the set of remaining monomials which do not have t_I factor.

The purpose of the cube attack is to find ANF representations of this form, where $p_{s(I)}$ is a linear polynomial of the variables in K. Therefore the corresponding t_I is named maxterm of the variables in V. K and V are the sets of superpoly variables (SV) and cube variables (CV), respectively.

The main reason that leads to this form is the following theorem:

978-1-4673-8789-7/16/$31.00 ©2016 IEEE

Theorem 1. $p_{S(I)} \equiv \sum_{v \in c_I} p$ modulo 2, where C_I is the set of all the possible values are given to the variables indexed by I.

A proof has been presented in [1]. If it is possible to find a linear representation for $p_{s(I)}$, then it can be used to get one bit of information.

B. Cube tester

In cube attack, the linear $p_{s(I)}$ is attained statistically from the t_I. In addition to linearity, also other properties of the t_I can be evaluated in the cube tester. For this purpose these main steps are involved:

I) A subset of plaintext V is chosen.

II) All conditions of this subset are set.

III) All computed outputs are summed.

IV) Finally, cube tester uses property testers and exploits the sums to evaluate whether the result is distinguishable or not.

In contrast to cube attack that extracts key bits, cube testers detect non-random behaviors. They are totally based on property testing algorithms. In [4] , it is claimed that cube testers are among first explicit applications of property testing in cryptanalysis. They have introduced several cube testers including; Balance, Constantness, Low Degree, Presence of Linear Variables, and Presence of Neutral Variables:

Balance. A random function is expected to contain as many zeroes as ones in its truth table. Superpolys that have a strongly unbalanced truth table can thus be distinguished from random polynomials, by testing whether it evaluates as often to one as to zero, either deterministically (by evaluating the superpoly for each possible input), or probabilistically (over some random subset of the SV). For example, if CV are $(x_1,...,x_C)$ and SV are $(x_{C+1},...,x_n)$, the deterministic balance test is

1. $c \leftarrow 0$
2. for all values of $(x_{C+1},...,x_n)$
3. compute

$$\mathrm{p}(x_{C+1},...,x_n) = \sum_{(x_1,...,x_C)} f(x_{C+1},...,x_n) \in \{0,1\}$$

4. $c \leftarrow c + p(x_{C+1},...,x_n)$
5. **return** $D(c) \in \{0,1\}$

where D is some decision rule. A probabilistic version of the test makes $N < 2^S$ iterations, for random distinct values of $(x_{C+1},...,x_n)$. Complexity is respectively 2^n and $N \cdot 2^S$.

Constantness. A particular case of balance test considers the "constantness" property, i.e. whether the superpoly defines a constant function; that is, it detects either that f has maximal degree strictly less than C (null superpoly), or that f has maximal degree exactly C (superpoly equals the constant 1), or that f has degree strictly greater than C (non-constant superpoly). This is equivalent to the maximal degree monomial test used in [11], used to detect nonrandomness in 736-round Trivium stream cipher.

Low Degree. A random superpoly has degree at least $(S-1)$ with high probability. Cryptographic functions that rely on a low-degree function, however, are likely to have superpolys of low degree. Because it closely relates to probabilistically checkable proofs and to error-correcting codes, low-degree testing has been well studied; the most relevant results to our concerns are the tests for Boolean functions in [12, 13]. The test by Alon et al. [13], uses a randomized algorithm to see if a function $f:(0,1)^n \to (0,1)$ is a polynomial of degree at most k without free term, for a given integer parameter k (With some modifications, this algorithm can be adopted to test polynomials of degree at most k , without this limitation.). The algorithm accepts functions with polynomials of degree at most k , and with probability at least 2/3, rejects functions that are far from any such polynomials. It has a distance parameter ε, and rejects (with probability at least 2/3) any function whose value should be modified on more than a ε-fraction to become a degree k polynomial f satisfying $f(0,...,0)=0$. It repeats the following check $\theta(\frac{1}{2k\varepsilon} + k2^k)$ times: it selects, uniformly and at random, $k+1$ vectors $y_1,...,y_{k+1} \in (0,1)^n$, and then evaluates f on every non-empty subset of the selected inputs, and checks that sum of these evaluations is zero. If all checks succeed, then it accepts, otherwise it rejects. This tester generalizes the BLR linearity test that has been introduced in [14]. According to the theorem that has been proved in [13], complexity of the tester is $\theta(\frac{1}{\varepsilon} + k2^{2k})$. Contrary to the method of ANF reconstruction (exponential in S), the complexity of this algorithm is independent of the number of variables. Hence, cube testers based on this low-degree test have complexity which is independent of the number of SV's.

Presence of Linear Variables. This is a particular case of the low-degree test, for degree $d=1$ and a single variable. Indeed, the ANF of a random function contains a given variable in at least one monomial of degree at least two with probability close to 1. One can thus test whether a given superpoly variable appears only linearly in the superpoly, e.g. for x_1 using the following test similar to that introduced in [14]:

1. pick random $(x_2,...,x_S)$

2. **if** $p(0,x_2,...,x_S) = p(1,x_2,...,x_S)$

3. **return** nonlinear

4. repeat steps 1 to $3N$ times

5. **return** linear

This test answers correctly with probability about $1-2^{-N}$, and computes $N \cdot 2^{C+1}$ times the function f. If, say, a stream cipher is shown to have an IV bit linear with respect to a set of CV in the IV, independently of the choice of the key, then it directly gives a distinguisher.

Presence of Neutral Variables. Dually to the above linearity test, one can test whether a SV is neutral in the superpoly, that is, whether it appears in at least one monomial. For example, the following algorithm tests the neutrality of x_1, for $N \le 2^{S-1}$:

1. pick random $(x_2,...,x_S)$

2. **if** $p(0,x_2,...,x_S) \neq p(1,x_2,...,x_S)$

3. **return** not neutral

4. repeat steps 1 to $3N$ times

5. **return** neutral

This test answers correctly with probability about $1-2^{-N}$ and runs in time $N \cdot 2^{C+1}$. For example, if x_1,x_2,x_3 are the CV and x_4,x_5,x_6 the SV, then x_6 is neutral with respect to $x_1x_2x_3$ if the superpoly $p(x_4,x_5,x_6)$ satisfies $p(x_4,x_5,0) = p(x_4,x_5,1)$ for all values of (x_4,x_5). A similar test was implicitly used in [15], via the computation of a neutrality measure.

As mentioned in the previous section, in some papers, cubes with a higher degree superpoly are also used for successful key recovery attack. Therefore, in addition to increasing cube size, the larger degree of leading superpoly on key bits is also important. Based on this observation, constantness and low-degree properties have been selected to compare resistance of symmetric ciphers against cube attack.

III. PROPOSED CUBE DEGREE TESTER ALGORITHM

As the cube computation mounts on the low degree test, the cube degree tester has an intractable complexity for higher degrees. Therefore, the number of the tests in the proposed low-degree tester algorithm is modified to 128 (where 128 is near to 100 which is the number of the tests have been done in the classical cube attack) to make the test practical. Therefore, complexity of the test changes to $2^7 \cdot (2^{k+1}-1) \cdot 2^l$ partial encryptions for cubes of size l. With this modification, test's accuracy decreases for degrees higher than 4. But, it is acceptable for just evaluating the cipher's algebraic representation.

Combined cube degree test algorithm. We have first combined our modified degree tester and the constantness tester, in order to generalize the test over all rounds of ciphers. Combined cube degree tester algorithm, which we call **CDT** algorithm is as follows:

Algorithm. Cube Degree Tester CDT

```
Input: {DegreeLimit,Cube}
Output: {Out}
rounds = 1; Out = {};
while(constantness_tester_output(rounds,Cube) = = true)
        rounds++; // go to upper rounds
end while
degree  = 1;
while(
     (modified_degree_tester_output(rounds,degree,Cube) = = false)
       and ( degree <= DegreeLimit))
        degree++; //test one higher degree
 end while
 add (rounds,degree) to Out;
 if(degree > DegreeLimit and is_first_try)
         rounds++; degree = 0;
         go to step 3. // one more test on higher rounds
 end if
 return Out
```

where

- **Rounds:** rounds of encryption.
- **Degree:** input parameter of degree test.
- **DegreeLimit:** whole algorithm's degree limit.

The algorithm starts with a constantness-test on one round of cipher. If the test succeeds, algorithm, increases the number of rounds, and repeats the test. If the test failed, the algorithm would run degree-test for higher degrees, and if it found a degree lower than DegreeLimit, it would return **degree** and **rounds** as a result. Otherwise the algorithm repeats this process just one more time to ensure if in higher rounds the degree of ANF decreases. For instance, one row of results as one run of **CDT** algorithm on GRAIN-128 stream cipher is as follows:

Detected degree: 6 Number of rounds: 139

Cube dimension: 8 Cube indexes: 65, 85,88,13,10,27,49,20

where cube indexes start from 0 as the least significant bit of the input.

Theorem 2. CDT algorithm's complexity in worst case is the following amount:

$$t_{worst} = O\left(\left(\frac{R(R+1)}{2}\right) + (2^{D+1}-1)\cdot(2R-1)\right)\cdot 2^7 \cdot 2^l \cdot pe\right) \quad (2)$$

where,

- R : is ciphers total number of rounds.

- D : is tester's degree limit.
- l **:** is the cube's size.
- pe **:** is one round partial encryption complexity.

Proof. Whole tester's complexity can be defined as this formula:

$$((1+2+....+r_1)+(2^1+2^2+...+2^{d_1})\cdot r_1 \\ +\varepsilon\cdot((r_1+1+r_1+2+....+r_2)+(2^1+2^2+...+2^{d_2})\cdot r_2))\cdot 2^7\cdot 2^l\cdot pe \quad (3)$$

where, r1 and r2 are last rounds in first and second tries, and d1 and d2 are the degrees, which are detected in the first and second tries. Since the prediction about degree detection in the algorithm is not possible, definition of ε is considered. If degree overflow happens, its value will be 1, otherwise 0. As a result, the following formula is obtained:

$$((\frac{r_1(r_1+1)}{2})+(2^{d_1+2}-(d_1+3))\cdot r_1 \\ +\varepsilon\cdot((\frac{r_2(r_2+1)}{2}-\frac{r_1(r_1+1)}{2})+(2^{d_2+2}-(d_2+3))\cdot r_2))\cdot 2^7\cdot 2^l\cdot pe \quad (4)$$

In worst case **first try** finishes in round R-1, and **second try** finishes in round R of the cipher, without degree detection:

$$((\frac{R(R+1)}{2})+(2^{D+1}-1)\cdot(R-1)+(2^{D+1}-1)\cdot R)\cdot 2^7\cdot 2^l\cdot pe \quad (5)$$

And finally the formula 6 is obtained:

$$((\frac{R(R+1)}{2})+(2^{D+1}-1)\cdot(2R-1))\cdot 2^7\cdot 2^l\cdot pe \quad (6)$$

The proof is done.

IV. PROPOSED EVALUATION FUNCTIONS

In this section, some evaluation functions are proposed which can be applied to aggregated tables of outputs of the **CDT** algorithm. These functions not only represent valuable information about the ciphers algebraic characteristics but also help to compare their resistance against cube like attacks. The aggregated tables are achieved from outputs of the algorithm. For example, the **Table** I aggregates the outputs for 20 first rounds of GRAIN-128 stream cipher. Every entry in this table corresponds to the number of detected degrees in whole test.

Our first evaluation function is based on the observation that only degree-one superpolys are useful in the basic cube attack:

$$F_1=\frac{\sum_{r=1}^{R} r\cdot a_{r,1}}{R\cdot T} \quad (7)$$

where, R and T are the cipher's total number of rounds, and the total number of output records, respectively. Also $a_{r,1}$ is the number of rows in the test results that have linear superpolys in r rounds of the cipher.

TABLE I. AGGREGATED TABLE OF RESULTS FOR GRAIN-128

Rounds	Detected degree									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**-**
1	18	1	7	1	0	0	0	0	0	0
2	14	1	3	3	0	0	0	0	0	0
3	4	0	1	1	1	0	0	0	0	0
4	3	1	2	0	0	0	0	0	0	0
5	2	1	1	0	0	0	0	0	0	0
6	2	0	1	0	0	0	0	0	0	0
7	5	0	0	1	0	0	0	0	0	0
8	3	0	3	0	0	0	0	0	0	0
9	4	0	0	0	0	0	0	0	0	0
10	4	0	0	1	0	0	0	0	0	0
11	2	2	0	0	0	0	0	0	0	0
12	3	0	0	0	0	0	0	0	0	0
13	1	0	0	3	0	0	0	0	0	0
14	1	1	3	4	0	0	0	0	0	0
15	4	0	1	0	0	0	0	0	0	0
16	2	2	1	1	0	0	0	0	0	0
17	0	3	0	0	0	0	0	0	0	0
18	2	0	1	0	0	0	0	0	0	0
19	2	1	1	2	0	0	0	0	0	0
20	2	1	1	2	0	0	0	0	0	0

As previously mentioned in this paper, some new variants of cube attacks, like quadratic and cubic cubes employ up to degree 3 superpolys. Also, there are some methods that use algebraic techniques in combination with cube attack. For this reason, we have generalized the previously introduced function, and the following function have obtained:

$$F_2 = \frac{\sum_{r=1}^{R}\sum_{d=1}^{D} r \cdot 2^{D-d+1} \cdot a_{r,k}}{2^D \cdot R \cdot T} \quad (8)$$

where, R , T and $a_{r,k}$ are same as above, and D is the algorithm's **DegreeLimit**.

In addition, the number of runs in which the algorithm did not detect superpoly's degree, are also important. These are considered in the following function:

$$F_3 = \frac{\sum_{r=1}^{R} (R - r + 1) \cdot b_r}{R \cdot T} \quad (9)$$

where, b_r is number of rows in the test results that have no determined degree in r rounds of the cipher. Higher value in this function corresponds to stronger cipher against cube attacks and is meant to be more difficult for cryptanalysis.

V. TEST AND EVALUATION

A. Test

In order to check the validity of the algorithm, 20 stream and block ciphers which are proposed after 2002 are investigated and the algorithm is applied on a selected set of them. Some of these selected ciphers such as GRAIN-128, KATAN, and SIMON have been attacked with cube technique previously. Moreover, other selected ciphers such as SIMECK, SPECK, RECTANGLE, KLEIN, and MIBS have not been attacked with the cube technique. Detailed descriptions of these ciphers are provided in [16-22].

CDT algorithm parameters are set to these values, to make each sample's running time convenient in a single PC:

- $D = 9$: The algorithm's **DegreeLimit**.
- R : total number of rounds in the cipher.
- l : **CDT** algorithm carried out for all cubes of size 1 to 8, 128 times.

where the R parameter varies in different ciphers. For example, RECTANGLE and GRAIN-128 have 25 and 256 rounds respectively. With the specified assumptions, total complexity is lower than the following amount according to formula (2):

$$t_{worst} = \mathrm{O}\left(2^{35} \cdot R \cdot pe\right) \quad (10)$$

For instance, in RECTANGLE lightweight block cipher, the test suggests a complexity less than 2^{40} partial encryptions. In other words, it is less than 2^{35} complete RECTANGLE encryptions. Moreover, the experimental results show that, the actual running time is much less than this value. For instance, cube degree tests on RECTANLE rarely exceed 6 rounds of the cipher's partial encryptions. According to reference [20] the whole analysis of RECTANGLE cipher can be computed in a single PC.

After the execution of the **CDT** algorithm on the selected set of the ciphers, the evaluation functions are applied to the outputs. The results are shown in **Table II**.

TABLE II. **CDT** ALGORITHM'S RESULTS ON CIPHERS.

Cipher	Cube attack	*F1*	*F2*	*F3*
KATAN32-80	[23, 24]	0.031957	0.058815	0
GRAIN96-128	[3]	0.031894	0.050383	0.153959
SIMON32-64	[24]	0.004209	0.015344	0.362568
SIMECK32-64	-	0.004039	0.013083	0.407198
SPECK32-64	-	0.00073	0.002075	0.688589
RECTANGLE64-80	-	0.000298	0.000378	0.812778
KLEIN64-80	-	0.000068	0.001099	0.748205
MIBS64-80	-	0	0.000734	0.798052

B. Evaluation

The results of the **Table II** are illustrated in Figures **1** and **2**. **Figure 1** shows degree detected superpolys. As showed in **Figure 1**, it can be concluded that ciphers including KATAN, GRAIN-128, SIMON, and SIMECK have more low-degree superpolys and are more likely to have successful cube attacks. Also it indicates that, MIBS, KLEIN, RECTANGLE, and SPECK do not have good low-degree superpolys for cube attack. On another view, in most cases, cipher's internal structure confirms the results. For instance, SIMON and SPECK are both almost ARX construction, meaning that they rely on Addition, word Rotation and XOR as said in [25]. Also for better performance in hardware, SIMON uses AND gates instead of Additions. This difference has impacts on cipher's algebraic representation and can be seen in the **Figure 1**. For another instance, although the recently proposed cipher SIMECK has the same structure as SIMON, the key scheduler in SIMECK differs from SIMON. The key scheduler in SIMECK uses a degree 2 function, while SIMON uses linear function. Impact of this difference can be seen in the **Figure 1** as well.

Figure 2 illustrates the inability of determining cube's degree more carefully. Based on the **Figure 2** it can be found out why two ciphers KATAN and GRAIN-128 have been selected for the cube attack previously. The authors of [9] claimed that, cube attack works better on stream ciphers, than block ciphers. Since GRAIN-128 is a stream cipher, and also KATAN uses TRIVIUM stream cipher's structure

introduced in [26], it can be concluded that **CDT** algorithm's results confirm this claim.

To evaluate the performance of the selected ciphers against cube attack, and to compare their results with the proposed algorithm's outputs, we have carried out the classical cube attack on the ciphers SIMECK and MIBS. In SIMECK with 64 bit key we have achieved successful key recovery attack up to 10 rounds of the cipher, with complexity of 2^{44}. Also in MIBS with 80 bit key we managed to do key recovery attack on 4 rounds of the cipher with complexity of 2^{72}. The difference between the attacked rounds of SIMECK and MIBS indicates the validity of the outputs of the proposed algorithm.

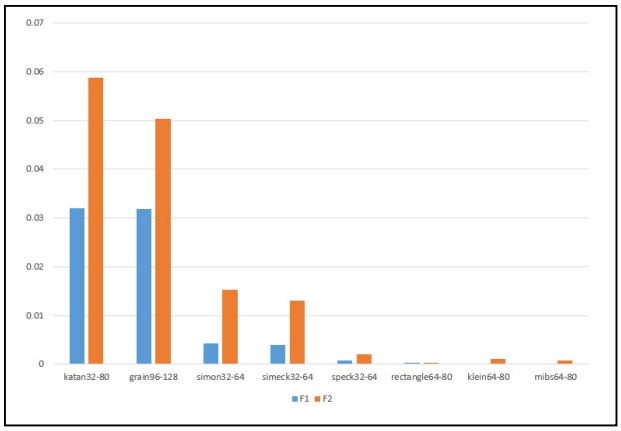

Figure 1. Degree detected superpolys. F1: only degree-one superpolys are considered. F2: All degree-detected superpolys are considered.

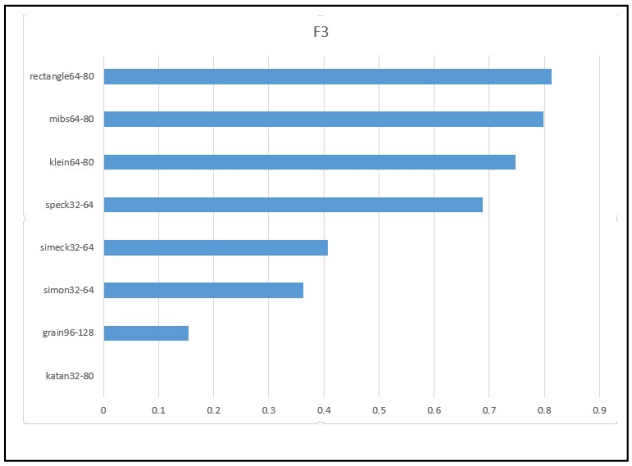

Figure 2. Non-detection rate.

VI. CONCLUSION

An algorithm which tests the resistance of ciphers against cube like attacks is proposed. As a question, we wanted to select the weakest cipher against cube attacks. For this purpose, we introduced a new algorithm that compares different ciphers with one single criterion, in a similar concern with the branch number criterion which is offered for block ciphers against differential and linear cryptanalysis in [27, 28]. The branch number is the minimum number of active S-boxes in two consecutive rounds of a non-trivial differential characteristic or non-trivial linear trail. In order to verify the validity of the proposed algorithm, a selected set of ciphers are evaluated. Also two lightweight block ciphers SIMECK and MIBS are attacked with the cube technique, which confirms the algorithms results.

An open problem is to find a direct relation between our results and the actual attacked rounds and complexity. For this purpose, one solution is to do a large number of tests with different symmetric ciphers to evaluate the relation. Also by these tests, more verdict on the validity of the algorithm may be obtained.

Moreover, some fitness functions are introduced, which show a criterion on ciphers. Another future plan is to find other fitness functions with more accurate outputs.

REFERENCES

[1] I. Dinur and A. Shamir, "Cube attacks on tweakable black box polynomials," in *Advances in Cryptology-EUROCRYPT 2009*, ed: Springer, 2009, pp. 278-299.

[2] F.-M. Quedenfeld and C. Wolf, "Algebraic Properties of the Cube Attack," *IACR Cryptology ePrint Archive,* vol. 2013, p. 800, 2013.

[3] I. Dinur and A. Shamir, "Breaking Grain-128 with dynamic cube attacks," in *Fast Software Encryption*, 2011, pp. 167-187.

[4] J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir, "Cube testers and key recovery attacks on reduced-round MD6 and trivium," in *Fast Software Encryption*, 2009, pp. 1-22.

[5] S. F. Abdul-Latip, M. R. Reyhanitabar, W. Susilo, and J. Seberry, "Extended cubes: enhancing the cube attack by extracting low-degree non-linear equations," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 296-305.

[6] A. Zhang, C.-W. Lim, K. Khoo, L. Wei, and J. Pieprzyk, "Extensions of the cube attack based on low degree annihilators," in *Cryptology and Network Security*, ed: Springer, 2009, pp. 87-102.

[7] X.-j. Zhao, T. Wang, and S. Guo, "Improved Side Channel Cube Attacks on PRESENT," *IACR Cryptology ePrint Archive,* vol. 2011, p. 165, 2011.

[8] P.-A. Fouque and T. Vannet, "Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks," in *Fast Software Encryption*, 2014, pp. 502-517.

[9] L. Ding, Y. Wang, and Z. Li, "Linear Extension Cube Attack on Stream Ciphers," *IACR Cryptology ePrint Archive,* vol. 2014, p. 249, 2014.

[10] I. Dinur and A. Shamir, "Side Channel Cube Attacks on Block Ciphers," *IACR Cryptology ePrint Archive,* vol. 2009, p. 127, 2009.

[11] H. Englund, T. Johansson, and M. S. Turan, "A framework for chosen IV statistical analysis of stream ciphers," in *Progress in Cryptology–INDOCRYPT 2007*, ed: Springer, 2007, pp. 268-281.

[12] A. Samorodnitsky, "Low-degree tests at large distances," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 506-515.

[13] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron, "Testing low-degree polynomials over GF (2)," in *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, ed: Springer, 2003, pp. 188-199.

[14] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," in *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 1990, pp. 73-83.

[15] S. Fischer, S. Khazaei, and W. Meier, "Chosen IV statistical analysis for key recovery attacks on stream ciphers," in *Progress in Cryptology–AFRICACRYPT 2008*, ed: Springer, 2008, pp. 236-245.

[16] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing,* vol. 2, pp. 86-93, 2007.

[17] C. De Canniere, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers," in *Cryptographic Hardware and Embedded Systems-CHES 2009*, ed: Springer, 2009, pp. 272-288.

[18] Z. Gong, S. Nikova, and Y. W. Law, *KLEIN: a new family of lightweight block ciphers*: Springer, 2012.

[19] M. Izadi, B. Sadeghiyan, S. S. Sadeghian, and H. A. Khanooki, "MIBS: a new lightweight block cipher," in *Cryptology and Network Security*, ed: Springer, 2009, pp. 334-348.

[20] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "RECTANGLE: A Bit-slice Ultra-Lightweight Block Cipher Suitable for Multiple Platforms," *IACR Cryptology ePrint Archive,* vol. 2014, p. 84, 2014.

[21] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, "The simeck family of lightweight block ciphers," *Cryptographic Hardware and Embedded Systems-CHES,* 2015.

[22] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK Families of Lightweight Block Ciphers," *IACR Cryptology ePrint Archive,* vol. 2013, p. 404, 2013.

[23] G. V. Bard, N. T. Courtois, J. Nakahara Jr, P. Sepehrdad, and B. Zhang, "Algebraic, AIDA/cube and side channel analysis of KATAN family of block ciphers," in *Progress in Cryptology-INDOCRYPT 2010*, ed: Springer, 2010, pp. 176-196.

[24] Z. Ahmadian, S. Rasoolzadeh, M. Salmasizadeh, and M. R. Aref, "Automated Dynamic Cube Attack on Block Ciphers: Cryptanalysis of SIMON and KATAN."

[25] R.-P. Weinmann, "AXR-Crypto Made from Modular Additions, XORs and Word Rotations. Dagstuhl Seminar 09031, January 2009," ed.

[26] C. De Cannière, "Trivium: A stream cipher construction inspired by block cipher design principles," in *Information Security*, ed: Springer, 2006, pp. 171-186.

[27] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, "The cipher SHARK," in *Fast Software Encryption*, 1996, pp. 99-111.

[28] M. Kanda, "Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function," in *Selected Areas in Cryptography*, 2001, pp. 324-338.