MILP-Based Cube Attack on the Reduced-Round WG-5 Lightweight Stream Cipher

Raghvendra Rohit(✉), Riham AlTawy, and Guang Gong

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
{rsrohit,raltawy,ggong}@uwaterloo.ca

Abstract. The cube attack is a powerful cryptanalytic tool for the analysis of stream ciphers, which until recently were investigated in a blackbox scenario with a minimal consideration to their internal and polynomial structures. In this paper, we analyze the lightweight stream cipher WG-5, which offers 80-bit security, using cube attacks in a non-blackbox polynomial setting employing the division property. WG-5 is a lightweight instantiation of the eSTREAM submission Welch-Gong stream cipher which provides mathematically proven random properties for its generated keystream. Our cube attack is automated using Mixed Integer Linear Programming models to theoretically bound the complexity of the superpoly recovery. The results of such an attack enable us to recover the secret key of WG-5 after 24 rounds of initialization utilizing $2^{6.32}$ keystream bits in $2^{76.81}$ time. Our attack on WG-5 has significantly lower data complexity than the algebraic attacks presented in the literature, albeit higher in computational complexity, it fits a more realistic scenario where large amount of data is hard to collect in lightweight constrained applications. Moreover, our attack is the first one to investigate the nonlinear feedback-based initialization phase of WG-5. Hence, such results are considered the best cryptanalytic ones in the case that the cipher runs a nonlinear key generation phase. Finally, our results are interesting in the sense that they enable us to argue how the design choices of WG-5 hinder the extension of cube attacks to more rounds in contrast to Grain 128a and Trivium, where such attacks can cover more than half of the number of initialization rounds.

Keywords: Welch-Gong stream cipher · Cube attacks · Division property · MILP · Lightweight stream ciphers

1 Introduction

The eSTREAM project [3] which was launched in 2004 is one of the first initiative that aimed to identify and recommend stream ciphers that fall under two profiles, (I) software oriented and (II) hardware efficient designs, for standardization.

M. O'Neill (Ed.): IMACC 2017, LNCS 10655, pp. 333–351, 2017.
https://doi.org/10.1007/978-3-319-71045-7_17

Profile II category received 25 submissions and the project was finalized after three phases of analysis by recommending the three stream ciphers Grain v.1 [14], Trivium [8], and Mickey 2.0 [7]. Following the finale of the eSTREAM project, the work on stream cipher design has slowed for a while, however, it is revived again by the current NIST lightweight standardization competition [17].

By investigating most of the recent stream cipher proposals such as Sprout [5], Fruit [27], Lizard [13], Plantlet [19], and Flip [18], one can easily spot that there is a noticeable class of them that follow a Grain-like structure where two Feedback Shift Registers (FSRs) are used to provide a guarantee on minimum periodicity. While some of them opt for utilizing one Linear Feedback Shift Register (LFSR) with a primitive polynomial to prove a minimum bound on the period of the keystream sequence, others employ one Non-Linear Feedback Shift Register (NLFSR) with known maximum periodicity. Nevertheless, all the previously mentioned proposals fail to provide guarantees for other important randomness criteria such as runs, t-tuple distribution, and ideal 2-level autocorrelation [12]. On the other hand, another class of cipher such as the eSTREAM profile II submission Welch-Gong (WG) cipher [20] follows a more rigorous approach to provide mathematically proven randomness properties which are not provided by other ciphers. More precisely, WG adopts only one LFSR that produces *m-sequences* followed by the Welch-Gong filtering transformation during keystream generation. Such a transformation is theoretically proven to generate a balanced keystream with long period, large and exact linear complexity, t-tuple distribution, and ideal 2-level autocorrelation. However, such desirable randomness properties which are provided by filtering *m-sequences* generated by the LFSR come with the price of the feasibility of a range of algebraic attacks [21,23], which are not applicable on other ciphers that employ NLFSRs during keystream generation. Nevertheless, in both classes of ciphers, NLFSRs are utilized during the state initialization phase and accordingly, the analysis of such phase provides better comparison to their resistance to attacks targeting their non-linear feedback-based state initialization.

In this paper, we investigate the security of the nonlinear initialization phase of WG-5 [4] which is a lightweight version of the eSTREAM submission WG [20]. WG-5 is a word oriented stream cipher that provides all the aforementioned randomness criteria. WG resists time-memory-data trade-off attacks by utilizing a state size that is double the size of the offered security, thus having a hardware footprint that ranges between 1229 and 1235 GEs for a throughput of 100 kbps. The best cryptanalytic result available for WG-5 is a univariate algebraic attack over the extension field $\mathbb{F}_{2^5}$ that recovers the secret key using around 2^{15} keystream bits in 2^{33} time [23]. Such attack [23] is applicable on WG-5 only when it runs a linear feedback keystream generation phase. The results of this paper are summarized as follows.

Our contributions. We analyze WG-5 with respect to non-blackbox polynomial-based cube attacks. More precisely, given the complicated structures of stream ciphers, conventional cube attacks always regard them as blackbox functions, and the attack was only proven feasible if its complexity

falls within the practical experimental range. In our analysis, we adopt the techniques from [25] which takes the polynomial structure of the analyzed stream cipher into consideration by tracing the propagation of a specific division property [24] through the initialization rounds. Accordingly, the propagation of the division property offers a theoretically proven bound on the number of key bits involved in the superpoly and the complexity of its recovery. Moreover, we further automate our attacks by proposing Mixed Integer Linear Programming (MILP) models for the division trails and then feed them to another MILP model for the whole attack. In what follows, we list our contributions.

- For the 24-round reduced initialization phase of WG-5, we model the division trail through the WG-5 permutation as an Sbox trail propagation which reduces the number of MILP inequalities and increases the solver chances in optimizing our model. We also provide the algorithmic description of all the proposed MILP models that we employ in our attack. The optimization of such models leads to a full key recovery when given $2^{6.32}$ keystream bits with $2^{76.81}$ time complexity.
- We present an argument which shows that the design choices in terms of feedback and filtering tap positions of WG-5 offer more security against cube attacks than Grain 128a and Trivium where such attacks break more than half the number of rounds of their initialization phases.

The rest of the paper is organized as follows. In Sect. 2, we recall the principals of the cube attack, division property, and how to model division trails using MILP. The specification of the WG-5 stream cipher is given in Sect. 3. In Sect. 4, we explain the details of the attack on the initialization phase of the WG-5 stream cipher, and how we model the WG-5 permutation as an Sbox to further reduce the number of MILP variables. Moreover, we give an algorithmic description of all MILP models used in our analysis and list the cube attack results and complexities. Furthermore, we compare our results on WG-5 to other cryptanalytic results available in the literature. In Sect. 5, we give an argument on the relation between the design parameters of WG-5 and the applicability of the cube attack, and further contrast such parameters to those of Grain and Trivium where cube attacks cover more than half the number of rounds of their initialization phases. Finally, the paper is concluded in Sect. 6.

2 Cube Attacks and the Division Property

In [25], Todo *et al.* proposed a method to apply cube attacks on stream ciphers employing the propagation of specific division trails. The consequence of their technique is that the application of the cube attack does not have to consider the analyzed cipher as a blackbox in order to recover its superpoly (the most difficult step in cube attacks) because the utilization of some specific division trails exploit the polynomial structure of the stream cipher. More precisely, since the cube attack is a kind of higher-order differential attack [16] and the division property is a technique to find higher-order differential trails, then the division property

can be used to analyze the Algebraic Normal Form (ANF) of the superpoly by investigating multiple division trails corresponding to a given cube. In order to better understand how we utilize this method in our analysis of the initialization phase of WG-5, in what follows, we recall the concepts and definitions related to the cube attack and division property.

2.1 Cube Attack

The cube attack [9] is based on higher-order differential cryptanalysis to recover the secret key of the investigated primitive by analyzing the ANF of the summation of a set of its outputs corresponding to a set of inputs. Unlike block ciphers, stream ciphers are easily evaluated in the forward direction to compute their output keystream and very hard to invert them. Accordingly, the cube attack has been extensively used in the analysis of stream ciphers [6,10,11,25] because the attacker has to manipulate the input and analyze the output without evaluating the cipher in the backward direction. More formally, let the analyzed stream cipher take an n-bit secret key $k = (k_0, k_1, \cdots, k_{n-1})$ and an m-bit $IV = (v_0, v_1, \cdots, v_{m-1})$, then, the first keystream bit is given by the polynomial $f(k, v)$ which operates on $n + m$ bits to output 1 bit. After sufficiently enough initialization rounds, the polynomial $f(k, v)$ becomes very complicated, thus the role of the cube attack is to simplify it by computing the higher-order differential of this polynomial which results in what is called the *superpoly*, that is the result of summing a set of polynomials $\bigoplus f(k, v)$ corresponding to a cube. Such a cube is a set of different public input variables taking all possible values and is denoted by C_I. If the structure of the superpoly is simple enough (e.g., linear or quadratic), then its ANF can be analyzed and secret variables can be recovered. Formally, let the set of public indices $I = \{i_1, i_2, \cdots, i_{|I|}\} \subset \{0, 1, \cdots, m-1\}$ denote the cube indices, then the polynomial $f(k, v)$ can be represented as:

$$f(k, v) = t_I \cdot p(k, v) + q(k, v),$$

where $t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$, $p(k, v)$ is a polynomial that does not contain any of the cube indices variables $(v_{i_1}, v_{i_2}, \cdots, v_{i_{|I|}})$, and $q(k, v)$ is independent of at least one variable from $(v_{i_1}, v_{i_2}, \cdots, v_{i|I|})$.

Let the cube C_I denote the set of all the possible $2^{|I|}$ values of $(v_{i_1}, v_{i_2}, \cdots, v_{i|I|})$, and the remaining input $n + m - |I|$ variables are set to some constant values, then the summation of $f(k, v)$ over all values of the cube C_I is given by

$$\bigoplus_{C_I} f(k, v) = \bigoplus_{C_I} t_I \cdot p(k, v) + \bigoplus_{C_I} q(k, v).$$

Since such summation reduces t_I to 1 because the set C_I has only one possibility where all the $|I|$ variables are equal to 1, and $q(k, v)$ vanishes because it misses at least one variable from the cube variables, then the above equation denotes the *superpoly* which is given by

$$superpoly : \bigoplus_{C_I} f(k, v) = p(k, v).$$

If the ANF of the superpoly is simple enough, then an attacker can query the encryption oracle with the chosen cube C_I. Hence, the returned first keystream bits are summed to evaluate the right-hand side of the superpoly and accordingly, secret variables can be recovered by solving a system of equations.

2.2 Division Property

The division property [24] is a generalization of the integral attacks [15] and a method to find higher-order differential trails. Moreover, a more refined bit-based division property is proposed in [26] and is defined as follows

Definition 1 (Bit-based division property [26]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{W}$ be a set whose elements take an n-dimensional vector of binary elements. The multiset $\mathbb{X}$ has the division property $\mathcal{D}^{1,n}_{\mathbb{W}}$ if it fulfills the following conditions*[1]*:*

$$\bigoplus_{x\in\mathbb{X}} \pi_u(x) = \begin{cases} unknown & if\ there\ exists\ w \in \mathbb{W}\ s.t\ u \succeq w, \\ 0 & otherwise, \end{cases}$$

where $u, w, x \in \mathbb{F}_2^n$, $\pi_u(x) = \prod_{i=0}^{n-1} x_i^{u_i}$ and $u \succeq w$ if $u_i \geq w_i$ for all i.

An attacker selects a set of chosen messages with a specific division property and traces its propagation until it reaches a round from where onwards the division property can not propagate. Accordingly, in the case of a cube attack, one prepares a set of $2^{|I|}$ chosen IVs where the variables $(v_{i_1}, v_{i_2}, \cdots, v_{i_{|I|}})$ take all the possible values. The division property of such a chosen set is $\mathcal{D}^{1,n}_{v}$, where $v_i = 1$ if $i \in \{i_1, i_2, \cdots, i_{|I|}\}$ and $v_i = 0$ for all remaining indices. Then one evaluates the propagation of this division property $\mathcal{D}^{1,n}_{v}$ for r rounds. We denote by $\{v\} \overset{def}{=} \mathbb{W}_0 \to \mathbb{W}_1 \to \cdots \to \mathbb{W}_r$ a r round division property propagation where $\mathbb{W}_i \subseteq \mathbb{F}_2^n$ for $0 \leq i \leq r$. Furthermore, we call $(w_0, w_1, \ldots, w_r) \in \mathbb{W}_0 \times \mathbb{W}_1 \times \ldots \times \mathbb{W}_r$ a r round division trail if w_{i-1} can propagate to w_i by division property propagation rules for all $i \in \{1, 2, \ldots, r\}$ [26,28]. The i-th bit at round r is balanced if $\mathbb{W}_r$ does not contain a unit vector whose i-th element is 1.

MILP models for division property. The propagation of the division property becomes infeasible when the input block size increases because the size of the corresponding $\mathbb{W}_i$ increases too. Particularly, in order to determine if the i-th bit at round r is balanced, one has to try all possible division trails with a given input division property and prove that there is no division trail that leads to a division property at round r with a unit vector where the i-th bit equals 1. However, in [28], a MILP-based method was proposed that allowed the efficient propagation of the division property for larger input spaces. More precisely, a MILP solver [1] is used to efficiently evaluate the feasibility of all division trails that cover the analyzed r rounds which are modeled by specific MILP models, and a higher-order differential trail is found if the solver determines that there

[1] "unknown" in Definition 1 means the xor sum can be 0 or 1 with probability $p \neq 1$.

is no division trail. MILP models that describe the propagation of the division property through different ciphers utilize the following three models [25,28]. Note that we refer to '+' as integer addition in all the MILP models.

- *MILP model for Copy.* Let the division trail through a copy be denoted by $a \rightarrow (b_1, b_2, \ldots, b_m)$, then the following inequalities are used to model such propagation:
$$M.var \leftarrow a, b_1, b_2, \ldots, b_m \text{ as binary.}$$
$$M.con \leftarrow a = b_1 + b_2 + \ldots + b_m.$$
- *MILP model for XOR.* Let $(a_1, a_2, \cdots, a_m) \rightarrow b$ denote the division trail of XOR, then the following inequalities are sufficient to describe the propagation of the division property:
$$M.var \leftarrow a_1, a_2, \cdots a_m, b \text{ as binary.}$$
$$M.con \leftarrow a_1 + a_2 + \cdots + a_m = b.$$
- *MILP model for AND.* Let $(a_1, a_2, \cdots, a_m) \rightarrow b$ denote the division trail for AND, then the following inequalities are used to describe the propagation:
$$M.var \leftarrow a_1, a_2, \cdots a_m, b \text{ as binary.}$$
$$M.con \leftarrow b \geq a_i \text{ for } i = 1, 2, \cdots, m.$$

In what follows, we give the description of the WG-5 stream cipher and how we use the division property to launch a cube attack on its initialization phase.

3 Specification of the WG-5 Stream Cipher

WG-5 [4] is a lightweight instantiated version of the eSTREAM submission word oriented WG stream cipher. It utilizes an 80-bit secret key, an 80-bit initialization vector and a 32-stage LFSR defined over the extension field $\mathbb{F}_{2^5}$. As depicted in Fig. 1, the LFSR is defined using the primitive polynomial $x^{32} + x^7 + x^6 + x^4 + x^3 + x^2 + \gamma$, where the polynomial belongs to $\mathbb{F}_{2^5}[x]$, $\gamma = \alpha^4+\alpha^3+\alpha^2+\alpha+1$, and α is a root of $x^5+x^4+x^2+x+1$ with its polynomial $\in \mathbb{F}_2[x]$. We denote the state of WG-5 at i-th round by $S^i = S^i[0]||S^i[1]||\ldots||S^i[31]$, where $S^i[j] = (s^i_{5j}, s^i_{5j+1}, s^i_{5j+2}, s^i_{5j+3}, s^i_{5j+4})$ for $0 \leq j \leq 31$. The 80-bit secret key $(k_0, k_1, \ldots, k_{79})$ and 80-bit initialization vector $(v_0, v_1, \ldots, v_{79})$ are denoted by $K[0]||K[1]||\ldots||K[15]$ and $IV[0]||IV[1]||\ldots||IV[15]$, respectively. The cipher runs in two phases: initialization and keystream generation (KSG) phase. The initialization phase runs for 64 rounds with the output of WG-permutation (WGP) feedback into the state, whereas the non-linear feedback is not used during the KSG phase. We now formally describe the WG-5 cipher. Initially, the state is loaded with K and IV as follows:

$$S^0[j] = \begin{cases} K[j \bmod 2], & \text{if } j \equiv 0 \bmod 2 \\ IV[j \bmod 2], & \text{if } j \not\equiv 0 \bmod 2 \end{cases}$$

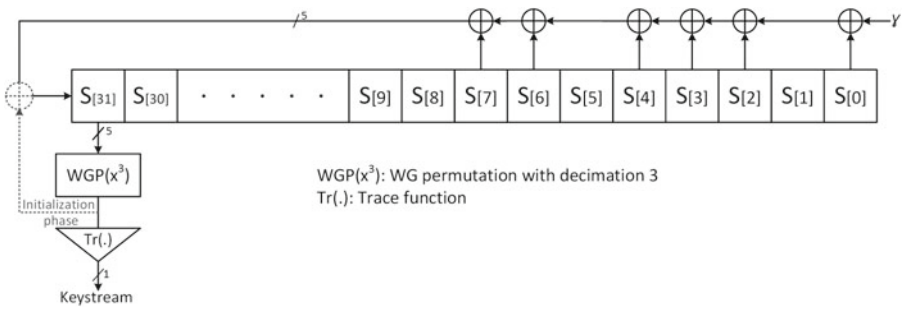

Fig. 1. Structure of WG-5

We use WG-5 with decimation 3 in our analysis[2]. The state update function is given by $S^{i+1}[j] = S^i[j+1], 0 \leq j \leq 30$ and $S^{i+1}[31] = \gamma S^i[0] \oplus S^i[2] \oplus S^i[3] \oplus S^i[4] \oplus S^i[6] \oplus S^i[7] \oplus \mathrm{WGP}((S^i[31])^3)$. During the KSG phase, the keystream bit is given by $z_{i-64} = Tr(\mathrm{WGP}(S^i[31])^3)$, where $Tr : \mathbb{F}_{2^5} \rightarrow \mathbb{F}_2$ denotes the Trace function. The corresponding boolean representation of keystream bit is given by $z_{i-64} = s^i_{155} + s^i_{156} + s^i_{157} + s^i_{158} + s^i_{159} + s^i_{155}s^i_{156} + s^i_{155}s^i_{157} + s^i_{155}s^i_{159} + s^i_{156}s^i_{158} + s^i_{156}s^i_{159} + s^i_{155}s^i_{156}s^i_{157} + s^i_{155}s^i_{157}s^i_{158} + s^i_{155}s^i_{157}s^i_{159} + s^i_{155}s^i_{158}s^i_{159} + s^i_{156}s^i_{157}s^i_{158} + s^i_{156}s^i_{158}s^i_{159}$. The state is then updated as follows:

$$S^{i+1}[j] = S^i[j+1], 0 \leq j \leq 30 \text{ and}$$

$$S^{i+1}[31] = \gamma S^i[0] \oplus S^i[2] \oplus S^i[3] \oplus S^i[4] \oplus S^i[6] \oplus S^i[7], \text{ for } i \geq 64.$$

In the following section, we describe our attack on the initialization phase of WG-5, and explain all the proposed MILP models used in our analysis. More detailed explanation is provided in the full paper [22].

4 Cube Attack on WG-5

We adopt the techniques presented in [25,28] to propose the cube attack on WG-5. The attack procedure consists of two phases: *offline phase* and *online phase*.

1. **Offline phase.** The goal of this phase is to recover a superpoly that is almost balanced[3] for a given cube C_I. It consists of three steps:

[2] We use decimation 3 as the degree of each of the component functions for WGP is 4, whereas it is 3 for decimation 1.

[3] $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is almost balanced if $f = 0$ for $\approx 2^{n-1}$ values and $f = 1$ for the remaining values.

Step 1.1: Create a MILP model M for WG-5 whose initialization is reduced to R rounds. The model encodes the division property propagation for R rounds to check the feasibility of all R-round division trails.

Step 1.2: Choose a cube C_I by flipping bits in $I = \{i_1, i_2, \ldots, i_{|I|}\}$ and then evaluate the secret variables involved in the superpoly. Let $J = \{k_{j_1}, k_{j_2}, \ldots, k_{j_{|J|}}\}$ denotes the set of involved secret variables[4].

Step 1.3: Choose a value in the constant part of IV and compute $\oplus_{C_I} f(k, v) = p(\bar{k}, \bar{v})$, where $\bar{k} = \{k_{j_1}, k_{j_2}, \ldots, k_{j_{|J|}}\}$, $\bar{v} = \{(v_0, v_1, \ldots, v_{79}) - (v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}})\}$ and all the possible combinations of $k_{j_1}, k_{j_2}, \ldots, k_{j_{|J|}}$ are tried out, then $p(\bar{k}, \bar{v})$ is recovered and stored in a list for all values of $\bar{k}$. Assuming the best case that we can recover the balanced superpoly in a single trial, the time complexity of this phase is bounded by $2^{|I|+|J|}$. However, if N cubes are used, the time complexity is given by $N2^{|I|+|J|}$.

2. **Online phase.** The goal of this phase is to recover the entire secret key. This phase is further divided into two steps.

Step 2.1: Use the balanced superpoly recovered in the *offline phase* and query the cube C_I to the encryption oracle to obtain the value of $p(\bar{k}, \bar{v})$ which is then compared to the previously stored values. Then one bit is recovered from J as $p = 0$ for $2^{|J|-1}$ values and $p = 1$ for the remaining half values. To recover more than 1 bit we use multiple cubes.

Step 2.2: Guess the remaining secret key values.

In what follows, we describe all the steps of the attack.

4.1 Automating the Cube Attack on WG-5 Using MILP

We start by modelling the division property propagation for each of the functions used in WG-5. We use COPY, XOR and AND operations described in Sect. 2 to model all the functions in the initialization and key generation phases.

MILP model for the WG-permutation (WGP). To model the WG-permutation, we can use its boolean representation which is given in Sect. 5. However, this approach results in large number of MILP variables and constraints due to its high non-linearity and involvement of terms of up to degree 4 in each of the component function. Hence, we use an alternative approach, we treat WGP as a 5-bit Sbox. Let $(x_0, x_1, x_2, x_3, x_4)$ and $(y_0, y_1, y_2, y_3, y_4)$ be the input and output of the WGP Sbox, respectively. We use the *inequality_generator()* function in Sage [2] and Algorithms 1 and 2 in [28], and consequently find that only 12 inequalities are sufficient to model the division property propagation through the WGP Sbox. The inequalities are given by:

[4] Step 1.2 is computationally feasible because of MILP.

$$\begin{cases} 2x_0 + 2x_1 + 2x_2 + 2x_3 + 6x_4 - 3y_0 - 3y_1 - 3y_2 - 3y_3 - 3y_4 \geq -1 \\ 4x_3 - y_0 - y_1 - y_2 - y_3 - y_4 \geq -1 \\ 4x_0 - y_0 - y_1 - y_2 - y_3 - y_4 \geq -1 \\ -x_0 - x_2 - x_3 - y_0 + 4y_1 - y_2 - y_3 - 2y_4 \geq -4 \\ -6x_0 - 3x_1 - 6x_3 - 6x_4 + 2y_0 - 4y_1 + 3y_2 - y_3 + 2y_4 \geq -19 \\ -3x_0 - x_1 - x_2 - 3x_3 - 2x_4 + 9y_0 + 7y_1 + 8y_2 + 9y_3 + 9y_4 \geq 0 \\ x_0 + x_1 + x_2 + x_3 + x_4 - 3y_0 - 3y_1 - 3y_2 - 3y_3 + 5y_4 \geq -2 \\ -x_0 - 3x_2 - 3x_3 - 2x_4 + y_0 + y_2 + y_3 - 2y_4 \geq -8 \\ -x_0 - x_1 + 2x_2 - x_3 - x_4 - y_0 - 2y_1 - 2y_2 + 3y_3 - y_4 \geq -5 \\ -x_0 - 2x_1 - 2x_2 - 2x_3 - x_4 - 2y_0 - y_1 - y_2 - y_3 + 5y_4 \geq -8 \\ -2x_0 - x_1 - 2x_2 - 2x_4 + y_0 + y_1 - y_2 + y_4 \geq -6 \\ -x_0 - x_2 - x_3 + y_0 - y_4 \geq -3. \end{cases}$$

Algorithm 2 describes the MILP model for the WG-permutation.

MILP model for the feedback function (FBK). The function FBK in Algorithm 3 generates the MILP variables and constraints for the feedback function $\gamma S^i[0] \oplus S^i[2] \oplus S^i[3] \oplus S^i[4] \oplus S^i[6] \oplus S^i[7]$. Since $\gamma = (1, 1, 1, 1, 1)$, we model $\gamma S^i[0]$ as $S^i[0]$.

MILP model for KSG. The function KSG in Algorithm 4 creates the MILP variables and constraints for the keystream bit $z = s^R_{155} + s^R_{156} + s^R_{157} + s^R_{158} + s^R_{159} + s^R_{155}s^R_{156} + s^R_{155}s^R_{157} + s^R_{155}s^R_{159} + s^R_{156}s^R_{158} + s^R_{156}s^R_{159} + s^R_{155}s^R_{156}s^R_{157} + s^R_{155}s^R_{157}s^R_{158} + s^R_{155}s^R_{157}s^R_{159} + s^R_{155}s^R_{158}s^R_{159} + s^R_{156}s^R_{157}s^R_{158} + s^R_{156}s^R_{158}s^R_{159}$. Furthermore, the bitwise AND and XOR operations are modeled using Algorithm 5.

We now present the MILP model for WG-5 in Algorithm 1. The function WG5EVAL evaluates all division trails for WG-5 whose initialization rounds are reduced to R. The number of MILP variables and constraints required in each function are given in Table 1.

Table 1. WG-5: MILP variables and constraints

Function	# of variables	# of constraints
WGP	15	17
FBK	65	35
KSG	79	63
R round of WG-5	160+159R + 5R	161 + 115R + 10R

4.2 Evaluating Involved Secret Variables and Superpoly Recovery

We prepare a cube C_I by flipping bits in $I = \{i_1, i_2, \ldots, i_{|I|}\}$ and then, we evaluate the involved secret variables in superpoly using the generic algorithm

Algorithm 1. MILP model for the initialization of WG-5

function WG5EVAL(R)
  Prepare empty MILP Model M
  $M.var \leftarrow S^0[j]$ for $0 \leq j \leq 31$
  for $i = 1$ to R **do**
    $(M, S', a) = \text{WGP}(S^{i-1})$
    $(M, S'', b) = \text{FBK}(S', [0, 2, 3, 4, 6, 7])$
    for $j = 0$ to 30 **do**
      $S^i[j] = S''[j + 1]$
    end for
    $M.con \leftarrow S''[0] = 0$
    $M.var \leftarrow S^i[31]$ as binary
    $M.con \leftarrow S^i[31] = a + b$
  end for
  $(M, S''', z) = \text{KSG}(S^R)$
  for $j = 0$ to 31 **do**
    $S'''[j] = 0$
  end for
  $M.con \leftarrow z = 1$
end function

proposed in [25]. We have given the description of the utilized algorithm (Algorithm 6) in Appendix B for the sake of completeness. The inputs to Algorithm 6 are the cube indices set I and the MILP model M for WG-5. The model M evaluates all the division trails for R rounds with input division property given by $v_i = 1$ for $i \in I$ and $v_i = 0$ for $i \in \{(0, 1, \ldots, 79) - I\}$. The reader is referred to [25] for the detailed explanation of Algorithm 6.

Searching cubes. We limit our search for the cubes to indices I such that $2^{|I|+|J|} < 2^{80}$. Table 2 lists the cubes we found that satisfies the above condition. Note that searching all $\binom{80}{|I|}$ cubes is infeasible and the cubes in Table 2 are the best so far for WG-5 according to our experimental results.

Recovering a balanced superpoly. We choose a value in the constant part of the IV and vary all $2^4 \times 2^{70}$ values to recover $p(k_5, k_6, \ldots, k_{74}, \bar{v})$ where $\bar{v} = (\{v_0, v_1, \ldots, v_{79}\} - \{v_j |\ j \in I_i\})$ for $1 \leq i \leq 5$ and $R = 24$. We also store 2^{70} values of $p(\bar{k}, \bar{v})$ as they will be used again in the online phase. We assume that we can recover a balanced superpoly in 1 trial for each of the cubes in Table 2. We expect that such an assumption holds with a high probability as there are 80-$|I_i| = 76$ values in the constant part of IV.

4.3 Key Recovery for 24 Rounds

We use the balanced superpolys recovered in offline phase for cubes I_1, I_2, I_3, I_4 and I_5 (see Table 2) in the online phase. We query the cube C_{I_i} to the encryption oracle and compute the sum $\oplus_{C_{I_i}} f(k, v)$. We then compare this sum with $\oplus_{C_{I_i}} f(k, v) = p(k_5, k_6, \ldots, k_{74}, \bar{v})$ stored in the offline phase for all possible

Table 2. Involved secret variables in superpoly for cube indices $I \in \{I_1, I_2, I_3, I_4, I_5\}$

Rounds	Involved secret variables J	Time complexity $log_2(.)$
15	$\{k_5, k_6, \ldots, k_{54}\}$	54
16	$\{k_5, k_6, \ldots, k_{54}\}$	54
17	$\{k_5, k_6, \ldots, k_{59}\}$	59
18	$\{k_5, k_6, \ldots, k_{59}\}$	59
19	$\{k_5, k_6, \ldots, k_{64}\}$	64
20	$\{k_5, k_6, \ldots, k_{64}\}$	64
21	$\{k_5, k_6, \ldots, k_{69}\}$	69
22	$\{k_5, k_6, \ldots, k_{69}\}$	69
23	$\{k_5, k_6, \ldots, k_{74}\}$	74
24	$\{k_5, k_6, \ldots, k_{74}\}$	74

$I_1 = \{0, 1, 2, 3\}, I_2 = \{0, 1, 2, 4\}, I_3 = \{0, 1, 3, 4\}, I_4 = \{0, 2, 3, 4\}, I_5 = \{1, 2, 3, 4\}$.
Here, time complexity means the complexity to recover the superpoly.

combinations of $\{k_5, k_6, \ldots, k_{74}\}$. We discard the values of $\{k_5, k_6, \ldots, k_{74}\}$ for which the sum is different. Since, we are using a balanced superpoly, $p(k_5, k_6, \ldots, k_{74}, \bar{v}) = 0$ for 2^{69} values and equals 1 for the remaining 2^{69} values. Thus, one bit of secret information can always be recovered. We use cubes I_1, I_2, I_3, I_4 and I_5 in our attack and hence can recover 5 secret variables. We then guess remaining 75 bits to recover the entire secret key. The attack time complexity for 24 rounds is then given by $5 \times 2^{74} + 2^{75} \approx 2^{76.81}$.

4.4 Attack Comparison with Algebraic Attacks

The univariate algebraic attacks [23] exploits the fact that WG-5 is updated linearly during the keystream generation phase. Hence, using the trace representation of z_t, it is possible to find a multiple g (also known as annihilator) of filtering function f i.e. $fg = 0$ and g contains only one term of hamming weight 3. This lowers the data and time complexity of the conventional algebraic attack to 2^{15} and 2^{33}, respectively. The applicability of such attacks does not hold if the nonlinear WGP is feedback into the state during KSG phase because the concept of annihilator functions no longer exists. On the other hand, the attack proposed in this paper is not affected by the nonlinear feedback of WGP into state during KSG phase. Moreover, our attack requires significantly low data complexity which enables a more realistic attack scenario in constrained applications where the available online data that may be queried by an adversary under a given key is usually limited by the running protocol. In summary, we can attack 24 rounds of the initialization phase of WG-5 with data and time complexity of $2^{6.32}$ and $2^{76.81}$, respectively.

5 Comparison of the Initialization Phase of WG-5 with Those of Grain128a and Trivium

In this section, we present an argument to show how the initialization phase of WG-5 is more resistant to cube attacks than those of Grain128a and Trivium. We particularly choose Grain128a and Trivium because both are eSTREAM finalists and also they offer the same level of security as WG-5. We give a brief description of both stream ciphers in Appendix C.

We now look at the state update functions of both Grain128a and Trivium more carefully and deduce the following observations:

- *For Trivium, the degree of z is 3 after 81 rounds.* The algebraic degree of z can only be increased by AND terms $s_{90}s_{91}$, $s_{174}s_{175}$ and $s_{285}s_{286}$. Thus, the round at which the degree of z equals 3 is $\min(90, 174-93, 285-177) = 81$.
- *For Grain128a, the degree of z is 6 after 32 rounds.* The maximum index in h function is 95 (for b_{95} term). At round 32 (127-95) only the degree of b_{95} is 4 and the remaining terms are of degree 1. Hence, the degree of z is 6 because of $b_{12}b_{95}s_{94}$ term.

On the other hand, for WG-5 we find that the *degree of z is 6 in 1 round only.* The degree of each component of $S^1[31] = \gamma S^0[0] \oplus S^0[2] \oplus S^0[3] \oplus S^0[4] \oplus S^0[6] \oplus S^0[7] \oplus \mathrm{WGP}((S^0[31])^3) = (s^1_{155}, s^1_{156}, s^1_{157}, s^1_{158}, s^1_{159})$ equals 4. This can be deduced from the boolean representation of the component functions of the WG-permutation given below.

$$
\begin{aligned}
y_0 &= x_0x_1x_3x_4 + x_0x_1x_4 + x_0x_2x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_0x_4 + x_0 \\
&\quad + x_1x_2x_3 + x_1x_2 + x_1x_3 + x_3x_4 \\
y_1 &= x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_0x_1x_2 + x_0x_1x_3 + x_0x_1x_4 + x_0x_1 \\
&\quad + x_0x_2x_4 + x_0x_2 + x_0x_3x_4 + x_0x_4 + x_1x_2x_3x_4 + x_1x_4 + x_1 \\
&\quad + x_2x_4 + x_2 + x_3x_4 \\
y_2 &= x_0x_1x_2x_3 + x_0x_1x_4 + x_0x_1 + x_0x_2 + x_0x_3x_4 + x_1x_2x_3x_4 + x_1x_2 \\
&\quad + x_1x_4 + x_2x_3x_4 + x_2x_3 + x_2x_4 + x_2 + x_3x_4 + x_3 + x_4 \\
y_3 &= x_0x_1x_2x_3 + x_0x_1x_3 + x_0x_1 + x_0x_2x_3x_4 + x_0x_2x_3 + x_0x_2x_4 \\
&\quad + x_0x_3x_4 + x_0x_4 + x_1x_2x_4 + x_1x_3x_4 + x_1x_3 + x_1 \\
y_4 &= x_0x_1x_2x_4 + x_0x_1x_2 + x_0x_1x_3x_4 + x_0x_1 + x_0x_2x_3x_4 + x_0x_2 \\
&\quad + x_0x_3x_4 + x_0x_3 + x_0x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_2 + x_1x_3 \\
&\quad + x_1x_4 + x_1 + x_2x_3x_4 + x_2x_3 + x_2x_4 + x_4
\end{aligned}
$$

Since z at round 1 is given by $s^1_{155} + s^1_{156} + s^1_{157} + s^1_{158} + s^1_{159} + s^1_{155}s^1_{156} + s^i_{155}s^1_{157} + s^1_{155}s^1_{159} + s^1_{156}s^i_{158} + s^i_{156}s^i_{159} + s^1_{155}s^1_{156}s^1_{157} + s^1_{155}s^1_{157}s^1_{158} + s^1_{155}s^1_{157}s^1_{159} + s^1_{155}s^1_{158}s^1_{159} + s^1_{156}s^1_{157}s^1_{158} + s^1_{156}s^1_{158}s^1_{159}$, then the degree of z is 6.

Based on the degree comparison of 32 rounds of Grain128a and 81 rounds of Trivium with 1 round of WG-5, we see that degree in WG-5 grows much faster.

We also observe that all the 5 bits processed by WGP at the i-th round are used to generate the keystream bit at round $(i+1)$ along with $5 \times 6 = 30$ new bits from the feedback function. This is not the same case with Grain128a because the updated bits b_{127} and s_{127} in i-th round are used in keystream bit at $i+32$ and $i+33$, respectively. Similarly, for Trivium the values of t_1, t_2 and t_3 at i-th round are used in keystream bit at $i+90, i+81$ and $i+108$ rounds, respectively. Thus, cubes of higher dimension whose superpoly involves few secret variables exist for both Grain128a and Trivium. For example, Todo *et al.* [25] experimentally found 92 dimension cube for 183 rounds Grain128a whose superpoly involves 16 secret key bits. Also, for Trivium reduced to 832 rounds, they found a 72 dimension cube which has only 5 secret variables in its superpoly. We tried some cubes of higher dimension for WG-5 and found that all the 80 secret variables are involved in the superpoly. The best cubes we have found are listed in Table 2 and they can cover 24 rounds of WG-5. Thus, based on the above observations, we conclude that the initialization phase of WG-5 is more stronger than those of Grain128a and Trivium with respect to cube attacks.

6 Conclusion

In this paper, we have investigated the lightweight stream cipher WG-5 with respect to non-blackbox cube attacks. Specifically, we have utilized the division property to find higher-order differential trails corresponding to a set of chosen initial values generated from specific cubes, and consequently the structure of the superpoly is recovered. Moreover, we have automated the process of the propagation of the division property by proposing MILP models for the WG-5 initialization and keystream generation phases. We have further modeled the WG permutation as an Sbox to reduce the number of variables and inequalities in the model which raises the chances of the MILP solver to find a feasible solution. The results of our cube attack reveals low data complexity requirements which when compared to the existing algebraic attacks, offer a more realistic attack scenario for lightweight constrained applications where the amount of data available to attacker under a given key is restricted by the running protocol. Also, unlike algebraic attacks, our attack is applicable on WG-5 whether it runs a linear or nonlinear keystream generation phase. Finally, the findings of our analysis enable us to argue that the WG-5 design parameters in terms of feedback and filtering tapping positions inhibit the extension of the cube attack to more rounds, in contrast to Grain128a and Trivium where such an attack covers more than half of the rounds of their initialization phases. Thus, we conclude that the initialization phase of WG-5 is more resistant to cube attacks than Grain's and Trivium's.

Acknowledgment. We would like to thank the reviewers of IMACC 2017 for their valuable comments that helped improve the quality of the paper. This work is supported by the National Institute of Standards and Technology (NIST) and Natural Sciences and Engineering Research Council of Canada (NSERC).

A MILP Models for WG-5 Components

Algorithm 2. MILP model for WGP

function WGP(S) ▷ $S = (s_0, s_1, \ldots, s_{159})$
  $M.var \leftarrow s'_{155+i}, x_i, y_i$ as binary for $0 \leq i \leq 4$
  $M.con \leftarrow s_{155+i} = s'_{155+i} + x_i$ for $0 \leq i \leq 4$
  Add constraints to M according to the WGP inequalities 4.1
  for $j = 0$ to 30 **do**
    $S'[j] = S[j]$ ▷ $S'[j] = (s'_{5j}, s'_{5j+1}, s'_{5j+2}, s'_{5j+3}, s'_{5j+4})$
  end for
  return $(M, S', [y_0, y_1, y_2, y_3, y_4])$
end function

Algorithm 3. MILP model for the FBK function in WG-5

function FBK(S, I)
  for $i \in I$ **do**
    $M.var \leftarrow s'_{5i+j}, x_{5i+j}$ as binary for $0 \leq j \leq 4$
  end for
  $M.var \leftarrow y_i$ as binary for $0 \leq i \leq 4$
  for $i \in I$ **do**
    $M.con \leftarrow s_{5i+j} = s'_{5i+j} + x_{5i+j}$ for $0 \leq j \leq 4$
  end for
  for $j = 0$ to 4 **do**
    $temp = 0$
    for $i \in I$ **do**
      $temp = temp + x_{5i+j}$
    end for
    $M.con \leftarrow y_j = temp$
  end for
  for $j \in \{(0, 1, \ldots, 31) - I\}$ **do**
    $S'[j] = S[j]$
  end for
  return $(M, S', [y_0, y_1, y_2, y_3, y_4])$
end function

Algorithm 4. MILP model for the KSG operation in WG-5

function KSG(S)
  $(M, S_1, a_1) = \text{AND}(S, [155, 156])$
  $(M, S_2, a_2) = \text{AND}(S_1, [155, 157])$
  $(M, S_3, a_3) = \text{AND}(S_2, [155, 159])$
  $(M, S_4, a_4) = \text{AND}(S_3, [156, 158])$
  $(M, S_5, a_5) = \text{AND}(S_4, [156, 159])$
  $(M, S_6, a_6) = \text{AND}(S_5, [155, 156, 157])$
  $(M, S_7, a_7) = \text{AND}(S_6, [155, 157, 158])$
  $(M, S_8, a_8) = \text{AND}(S_7, [155, 157, 159])$
  $(M, S_9, a_9) = \text{AND}(S_8, [155, 158, 159])$
  $(M, S_{10}, a_{10}) = \text{AND}(S_9, [156, 157, 158])$
  $(M, S_{11}, a_{11}) = \text{AND}(S_{10}, [156, 158, 159])$
  $(M, S_{12}, a_{12}) = \text{XOR}(S_{11}, [155, 156, 157, 158, 159])$
  $M.var \leftarrow z$ as binary
  $M.con \leftarrow z = \sum_{i=1}^{12} a_i$
  return (M, S_{12}, z)
end function

Algorithm 5. MILP model for AND and XOR operations in WG-5

function AND(S, I)
  $M.var \leftarrow s'_i, x_i$ as binary for i in I
  $M.var \leftarrow y$ as binary
  $M.con \leftarrow s_i = s'_i + x_i$ for i in I
  $M.con \leftarrow y \geq x_i$ for i in I
  for $i \in \{(0, 1, \ldots, 159) - I\}$ **do**
    $s'_i = s_i$
  end for
  return (M, S', y)
end function
function XOR(S, I)
  $M.var \leftarrow s'_i, x_i$ as binary for i in I
  $M.var \leftarrow y$ as binary
  $M.con \leftarrow s_i = s'_i + x_i$ for i in I
  $temp = 0$
  for $i \in I$ **do**
    $temp = temp + x_i$
  end for
  $M.con \leftarrow y = temp$
  for i in $\{(0, 1, \ldots, 159) - I\}$ **do**
    $s'_i = s_i$
  end for
  return (M, S', y)
end function

B A Generic Algorithm for the Evaluation of the Involved Secret Variables in a Superpoly [25]

Algorithm 6. MILP model to find involved secret variables in superpoly

function EXTRACTSECRETVARIABLES(MILP model M, Cube Indices I)
  $M.var \leftarrow k_i$ as binary for $0 \leq i \leq n-1$, ▷ $k_0, k_1, \ldots, k_{n-1}$ are secret variables
  $M.var \leftarrow v_i$ as binary for $0 \leq i \leq m-1$, ▷ $v_0, v_1, \ldots, v_{m-1}$ are public variables
  $M.con \leftarrow v_i = 1$ for $i \in I$
  $M.con \leftarrow v_i = 0$ for $i \in \{(0, 1, \ldots, m-1) - I\}$
  $M.con \leftarrow \sum_{i=0}^{n-1} k_i = 1$
  do
    solve MILP model M
    if M is feasible **then**
      pick $j \in \{0, 1, \ldots, n-1\}$ s.t $k_j = 1$
      $J = J \cup \{j\}$
      $M.con \leftarrow k_j = 0$
    end if
  while M is feasible
  return J
end function

C Description of Grain128a and Trivium

Grain128a is a NLFSR based stream cipher of Grain family with two 128-bit states represented by $(b_0, b_1, \ldots, b_{127})$ and $(s_0, s_1, \ldots, s_{127})$. The state is loaded with 128-bit key and 96-bit IV as follows $(b_0, b_1, \ldots, b_{127}) = (k_0, k_1, \ldots, k_{127})$ and $(s_0, s_1, \ldots, s_{127}) = (iv_0, iv_1, \ldots, iv_{95}, 1, \ldots, 1, 0)$. The initialization phase runs for 256 rounds with the state update function given by

$$
\begin{aligned}
g \leftarrow\ & b_0 + b_{26} + b_{56} + b_{91} + b_{96} + b_3 b_{67} + b_{11} b_{13} \\
& + b_{17} b_{18} + b_{27} b_{59} + b_{40} b_{48} + b_{61} b_{65} + b_{68} b_{84} \\
& + b_{88} b_{92} b_{93} b_{95} + b_{22} b_{24} b_{25} + b_{70} b_{78} b_{82} \\
f \leftarrow\ & s_0 + s_7 + s_{38} + s_{70} + s_{81} + s_{96} \\
h \leftarrow\ & b_{12} s_8 + s_{13} s_{20} + b_{95} s_{42} + s_{60} s_{79} + b_{12} b_{95} s_{94} \\
z \leftarrow\ & h + s_{93} + b_2 + b_{15} + b_{36} + b_{45} + b_{64} + b_{73} + b_{89} \\
(b_0, b_1, \ldots, b_{127}) \leftarrow\ & (b_1, b_2, \ldots, b_{127}, g + s_0 + z) \\
(s_0, s_1, \ldots, s_{127}) \leftarrow\ & (s_1, s_2, \ldots, s_{127}, f + z).
\end{aligned}
$$

During the KSG phase, z is not feedback to the state and directly used as the keystream bit.

Trivium is also an NLFSR based stream cipher with state size 288. The 80-bit key and 80-bit IV are loaded into the state as follows $(s_0, s_1, \ldots, s_{92}) = (k_0, k_1, \ldots, k_{79}, 0, \ldots, 0)$, $(s_{93}, s_{94}, \ldots, s_{176}) = (iv_0, iv_1, \ldots, iv_{79}, 0, \ldots, 0)$ and $(s_{177}, s_{178}, \ldots, s_{287}) = (0, 0, \ldots, 0, 1, 1, 1)$. The state update function of Trivium is given by

$$
\begin{aligned}
t_1 &\leftarrow s_{65} + s_{92} \\
t_2 &\leftarrow s_{161} + s_{176} \\
t_3 &\leftarrow s_{242} + s_{287} \\
z &\leftarrow t_1 + t_2 + t_3 \\
t_1 &\leftarrow t_1 + s_{90}s_{91} + s_{170} \\
t_2 &\leftarrow t_2 + s_{174}s_{175} + s_{263} \\
t_3 &\leftarrow t_3 + s_{285}s_{286} + s_{68} \\
(s_0, s_1, \ldots, s_{92}) &\leftarrow (t_3, s_0, \ldots, s_{91}) \\
(s_{93}, s_1, \ldots, s_{176}) &\leftarrow (t_1, s_{93}, \ldots, s_{175}) \\
(s_{177}, s_1, \ldots, s_{287}) &\leftarrow (t_2, s_{177}, \ldots, s_{286}).
\end{aligned}
$$

The initialization phase runs for 1152 rounds without producing an output while z is used as the keystream bit during KSG phase.

References

1. Gurobi: MILP optimizer. http://www.gurobi.com/
2. SageMath. http://www.sagemath.org/
3. eSTREAM: the ECRYPT stream cipher project (2008)
4. Aagaard, M.D., Gong, G., Mota, R.K.: Hardware implementations of the WG-5 cipher for passive rfid tags. In: 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 29–34 (2013)
5. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 451–470. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_22
6. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_1
7. Babbage, S., Dodd, M.: The MICKEY stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_15
8. De Cannière, C.: Trivium: a stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006). https://doi.org/10.1007/11836810_13
9. Dinur, I., Shamir, A.: Cube attacks on tweakable blackbox polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
10. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_10

11. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_26
12. Gong, G., Youssef, A.M.: Cryptographic properties of the Welch-Gong transformation sequence generators. IEEE Trans. Inf. Theor. **48**(11), 2837–2846 (2002)
13. Hamann, M., Krause, M., Meier, W.: Lizard: a lightweight stream cipher for power-constrained devices. IACR Trans. Symmetric Crypt. **2017**(1), 45–79 (2017)
14. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: IEEE International Symposium on Information Theory, pp. 1614–1618 (2006)
15. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9
16. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography 1994. LNCS, vol. 276, pp. 227–233. Springer, MA (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
17. McKay, K., Bassham, L., Sönmez Turan, M., Mouha, N.: Report on lightweight cryptography (NISTIR8114) (2017)
18. Méaux, P., Journault, A., Standaert, F.-X., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 311–343. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_13
19. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Trans. Symmetric Crypt. **2017**(2), 52–79 (2017)
20. Nawaz, Y., Gong, G.: Wg: a family of stream ciphers with designed randomness properties. Inf. Sci. **178**(7), 1903–1916 (2008)
21. Orumiehchiha, M.A., Pieprzyk, J., Steinfeld, R.: Cryptanalysis of WG-7: a lightweight stream cipher. Crypt. Commun. **4**(3–4), 277–285 (2012)
22. Rohit, R., AlTawy, R., Gong, G.: MILP-based cube attack on the reduced-round WG-5 lightweight stream sipher. The University of Waterloo CACR Archive, Technical report CACR 2017-06 (2017). http://cacr.uwaterloo.ca/techreports/2017/cacr2017-06.pdf
23. Rønjom, S.: Improving algebraic attacks on stream ciphers based on linear feedback shift register over $\mathbb{F}_{2^K}$. Des. Codes Crypt. **82**(1–2), 27–41 (2017)
24. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
25. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_9
26. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_18

27. Vahid Amin Ghafari, H.H., Chen, Y.: Fruit: Ultra-lightweight stream cipher with shorter internal state. Cryptology ePrint Archive, Report 2016/355 (2016). http://eprint.iacr.org/2016/355
28. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24