Extensions of the Cube Attack based on Low Degree Annihilators

Aileen Zhang[1], Chu-Wee Lim[1], Khoongming Khoo[1], Wei Lei[2], and Josef Pieprzyk[3]

[1] DSO National Laboratories
20 Science Park Drive, Singapore 118230
{zyinghui,lchuwee,kkhoongm}@dso.org.sg
[2] School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
wl@pmail.ntu.edu.sg
[3] Department of Computing, Macquarie University, Australia
josef@ics.mq.edu.au

Abstract. At Crypto 2008, Shamir introduced a new algebraic attack called the cube attack, which allows us to solve black-box polynomials if we are able to tweak the inputs by varying an initialization vector. In a stream cipher setting where the filter function is known, we can extend it to the cube attack with annihilators: By applying the cube attack to Boolean functions for which we can find low-degree multiples (equivalently annihilators), the attack complexity can be improved. When the size of the filter function is smaller than the LFSR, we can improve the attack complexity further by considering a sliding window version of the cube attack with annihilators. Finally, we extend the cube attack to vectorial Boolean functions by finding implicit relations with low-degree polynomials.

Keywords Cube Attack, Algebraic Attack, Low-Degree Annihilators.

1 Introduction

In the history of cryptography, algebraic cryptanalysis is a rather recent trend. The underlying idea behind this attack is rather simple: in trying to attack a cryptosystem, write the problem as a set of polynomial equations with coefficients and unknowns in some common finite field K, most probably of characteristic 2. One then employs whatever means at one's disposal to solve this system of polynomial equations.

It has been long known that the general problem of solving such a system is NP-complete, even if the system comprises of only quadratic equations over $\mathbb{F}_2$ (see [13]). Nevertheless, many cryptographic systems appear susceptible to attacks via this approach. Indeed, a large arsenal of attacks have been designed with the algebraic approach in mind, including (but not restricted to) linearization, relinearization [9], eXtended Linearization [4], Gröbner basis [7, 8] and the fast algebraic attack [5].

In Aug 2008, during the Crypto conference, Adi Shamir [12] presented a new approach to algebraic attacks in an invited lecture. Termed *cube attack*, his method requires the attacker to launch an active attack (e.g. chosen-IV or chosen-PT) in order to extract useful information from the bits obtained. Roughly speaking, by skillfully choosing the bits in a systematic manner, the attacker may lower the degree of the polynomial quickly.

In Section 2, we shall give a description of Shamir's cube attack. Then, we offer several variations to the basic cube attack. In Section 3, we extend the cube attack to polynomials f for which we can

find a low degree g such that fg is also of low degree, and we apply this to the Toyocrypt cipher as an example. We call this the cube attack with low degree annihilator and show that it has better attack complexity than the basic cube attack and algebraic attack.

In Section 3.2, we refine the cube attack with low degree annihilators to the special case where the size of the filter function is smaller than the LFSR. We call this refinement the sliding window cube attack. We demonstrate several scenarios where the sliding window cube attack has better attack complexity than the cube attack with annihilators. We also compare our attack with the re-synchronization attack of Daemon et. al. [10] since it is also applicable in this case, and conclude that our attack gives better attack complexity under suitable conditions.

In Section 4, we consider the cube attack when applied to vectorial filter functions. We show that there always exist equations describing the vectorial functions, which has lower degree than low degree multiples of single-bit output Boolean functions. Thus this shows that theoretically, we have better attack complexity when we apply the vectorial cube attack rather than attacking single bit output of the S-boxes by the cube attack with low degree annihilators. Finally in Section 5, we summarize our findings and propose some further research directions.

2 Preliminaries: Cube Attack

First let us give a brief overview of the cube attack [12]. Throughout this article, all polynomials have coefficients in $\mathbb{F}_2$, and $\mathbf{x}$ (*resp.* $\mathbf{v}$) denotes the vector $(x_0, x_1, \ldots x_{n-1})$ (*resp.* $(v_0, v_1, \ldots, v_{m-1})$).

The primary idea behind this attack lies in the following theorem:

Theorem 1 *Let $f(\mathbf{x})$ be a polynomial in n variables of degree d. Suppose $0 < k \leq d$ and t is the monomial $x_0 x_1 \ldots x_{k-1}$. Write f in the form:*

$$f(\mathbf{x}) = t \cdot P_t(\mathbf{x}) + Q_t(\mathbf{x}),$$

where none of the terms in $Q_t(\mathbf{x})$ is divisible by t. Note that $\deg(P_t) \leq d - k$.

Then the sum of f over all $(x_0, \ldots, x_{k-1}) \in \mathbb{F}_2^k$, considered as a polynomial in $x_k, x_{k+1}, \ldots$, equals

$$P_t(\overbrace{1, \ldots, 1}^{k}, x_k, x_{k+1}, \ldots, x_{n-1})$$

and hence is a polynomial of degree at most $d - k$.

Proof. Consider the equality $f = t \cdot P_t + Q_t$. Split the sum into $\sum_{(x_0,\ldots,x_{k-1})} t \cdot P_t$ and $\sum_{(x_0,\ldots,x_{k-1})} Q_t$. In the first sum, $t = 0$ unless $x_0 = x_1 = \cdots = x_{k-1} = 1$ in which case

$$\sum_{(x_0,\ldots,x_{k-1}) \in \mathbb{F}_2^k} t \cdot P_t = P_t(\overbrace{1, \ldots, 1}^{k}, x_k, x_{k+1}, \ldots, x_{n-1}).$$

On the other hand Q_t is a sum of monomials, each of which is not divisible by t. Let m be any one of these monomials. Since m is not divisible by t, it excludes x_i for some $0 \leq i \leq k-1$. If it excludes (say) x_0, then the sum across all $(x_0, \ldots, x_{k-1}) \in \mathbb{F}_2^k$ can be further split into two sums: the sum for $x_0 = 0$ and for $x_0 = 1$. These two sums are equal since x_0 does not appear in m. Hence

$$\sum_{(x_0,\ldots,x_{k-1}) \in \mathbb{F}_2^k} m = 0 \quad \Longrightarrow \quad \sum_{(x_0,\ldots,x_{k-1}) \in \mathbb{F}_2^k} Q_t = 0.$$

This completes our proof of the theorem.

Let us apply this theorem to cryptanalyze a stream cipher. Write the cipher in the form:

$$z = f(\mathbf{x}, \mathbf{v}),$$

which takes in an n-bit key $\mathbf{x}$ and an m-bit IV $\mathbf{v}$, and outputs the first bit of the keystream. Suppose $d = \deg f \leq m$. We describe the cube attack for the term $t = v_0 v_1 \cdots v_{d-2}$.

Fix the IV bits $v_{d-1}, v_d, v_{d+1}, \cdots \in \mathbb{F}_2$ and write C for the set of $\mathbf{v}$ with these values of $v_{d-1}, v_d, \ldots$. Thus $|C| = 2^{d-1}$. Sum $f(\mathbf{x}, \mathbf{v})$ over $\mathbf{v} \in C$. By applying Theorem 1 to t, this sum is linear in $\mathbf{x}$:

$$\sum_{\mathbf{v} \in C} f(\mathbf{x}, \mathbf{v}) = L(\mathbf{x}). \tag{1}$$

If $L(\mathbf{x}) \neq 0$, we call t a **maxterm** in accordance with [12], and obtain one linear relation in the key bits. To obtain $n-1$ more such relations, we can do the following.

- Use the same f, but use a different maxterm t.
- Use a different f, e.g. by using the second bit of the keystream.

With n linearly independent relations of the key bits, we can easily find them via Gaussian elimination.

Hence, given access to such a function f, *Cube Attack* proceeds to find the unknown vector $\mathbf{x}$ according to the following stages:

1. *First: the preprocessing stage.* This involves finding the coefficients of L for n such L. Each L has $n+1$ coefficients including the constant term; to find them, we need to compute the sum (1) for $n+1$ keys:

   $$\mathbf{x} = \mathbf{0}, \mathbf{e_0}, \mathbf{e_1}, \ldots, \mathbf{e_{n-1}},$$

   where $\mathbf{e_i}$ is the vector where the i-th component is 1 and the rest are 0. The amount of work required is $n(n+1)2^{d-1}$ evaluations of f.
   We also compute the inverse of the matrix of linear relations. This requires n^3 operations at most so the amount of work is upper-bounded by

   $$n(n+1)2^{d-1} + n^3.$$

2. *Second: the online stage.* Now we apply a chosen-IV attack on the cipher. Compute the sum (1) for n linear relations L. Each sum requires 2^{d-1} evaluations of f, so we need $n2^{d-1}$ evaluations of f in all. Since we already have the inverse of the L-matrix, we only need to perform matrix multiplication which takes n^2 operations. Hence, the amount of work is upper-bounded by

   $$n2^{d-1} + n^2.$$

Notice that the attack only assumes $\deg f \leq d$, and that we can evaluate f. No knowledge of the coefficients of f is required.

Remark 1. For a given maxterm t, in the case where $\deg(t) < n-1$, we may be able to derive multiple equations, since each maxterm gives an equation that may have monomials containing terms in the IV as well as in the key. Hence, substituting in different values for the terms in the IV that are not in the maxterm may produce different equations.

3 Cube Attack with Annihilators

In 2003, Courtois and Meier[5] observed that for some polynomials f, we can find a low degree g such that $h := fg$ is also of low degree. We shall apply this observation to derive an enhanced version of the cube attack.

As before, let $z = f(\mathbf{x}, \mathbf{v})$ represent the first output bit, where $\mathbf{x}$ is the key and $\mathbf{v}$ is the IV. Let $g(\mathbf{x}, \mathbf{v})$ be a polynomial such that:

- $g(\mathbf{x}, \mathbf{v})$ is of low degree e;
- $h(\mathbf{x}, \mathbf{v}) := f(\mathbf{x}, \mathbf{v})g(\mathbf{x}, \mathbf{v})$ is of degree $d \leq \deg(f)$ and $d > e$.

Our attack works as follows: suppose we pick the maxterm $v_0 v_1 \cdots v_{d-e-1}$. Fix the IV-bits $v_{d-e}, v_{d-e+1}, \cdots \in \mathbb{F}_2$ and let C be the set of $\mathbf{v}$ which has these values of $v_{d-e}, v_{d-e+1} \ldots$. Consider the sum:

$$\sum_{\mathbf{v} \in C} h(\mathbf{x}, \mathbf{v}) = \sum_{\mathbf{v} \in C} f(\mathbf{x}, \mathbf{v}) g(\mathbf{x}, \mathbf{v}).$$

By Theorem 1, on the left, we get a polynomial in $\mathbf{x}$ of degree at most $d - (d - e) = e$. On the right, note that $f(\mathbf{x}, \mathbf{v})$ is known since it is a keystream bit, so we get a polynomial of degree $\leq e$. Now we can solve for the secret bits by applying a range of techniques, such as linearization [4] or Gröbner basis techniques [7, 8].

We shall term this method *Cube Attack with Annihilators*[1]. Given the filter function f, assume we have a low degree multiple g of degree e such that $h = fg$ has low degree d. There are many efficient algorithms in literature to find f and g. See [1] and [11] for example. The attack proceeds as follows:

1. *First, the preprocessing stage.* We need to compute the polynomial

   $$P(\mathbf{x}) := \sum_{\mathbf{v} \in C} h(\mathbf{x}, \mathbf{v})$$

   which is of degree $\leq e$. Since this is a linear combination of $\sum_{i=0}^{e} \binom{n}{i}$ monomials, we need to evaluate this sum $\sum_{i=0}^{e} \binom{n}{i}$ times (by pumping in different $\mathbf{x}$'s) to find the coefficients. This requires $2^{d-e} \sum_{i=0}^{e} \binom{n}{i}$ evaluations of h to compute the coefficients of a single P. For linearization to work, we need $\sum_{i=0}^{e} \binom{n}{i}$ such polynomials, so the total amount of operations is:

   $$2^{d-e} \left(\sum_{i=0}^{e} \binom{n}{i} \right)^2$$

   evaluations of h.
2. *Second: the online phase.* For each of the $\sum_{i=0}^{e} \binom{n}{i}$ maxterms, we must compute $\sum_{\mathbf{v} \in C} f(\mathbf{x}, \mathbf{v}) g(\mathbf{x}, \mathbf{v})$. The polynomial $g(\mathbf{x}, \mathbf{v})$, for a fixed $\mathbf{v} \in C$ has typically $\sum_{i=0}^{e} \binom{n}{i}$ terms. Hence, the computation of the term $f(\mathbf{x}, \mathbf{v})g(\mathbf{x}, \mathbf{v})$ requires $2^{d-e} \sum_{i=0}^{e} \binom{n}{i}$ computations. For $\sum_{i=0}^{e} \binom{n}{i}$ such maxterms, we require $2^{d-e} \left(\sum_{i=0}^{e} \binom{n}{i} \right)^2$ computations. Finally linearization of

   $$\sum_{\mathbf{v} \in C} f(\mathbf{x}, \mathbf{v}) g(\mathbf{x}, \mathbf{v}) = \sum_{\mathbf{v} \in C} h(\mathbf{x}, \mathbf{v}) = P(\mathbf{x})$$

[1] We have used the term annihilators in naming our attack because from [1], the existence of low degree multiples is equivalent to the existence of low degree annihilators.

gives a system of $\sum_{i=0}^{e} \binom{n}{i}$ linear equations which requires $\left(\sum_{i=0}^{e} \binom{n}{i}\right)^3$ operations to solve. Hence, the total amount of operations is about:

$$2^{d-e}\left(\sum_{i=0}^{e}\binom{n}{i}\right)^2 + \left(\sum_{i=0}^{e}\binom{n}{i}\right)^3.$$

We note that both the basic cube attack and the cube attack with annihilators are chosen IV resynchronization attacks. However, some of their differences are as follows:

1. This variation of the cube attack requires us to compute $h = fg$ for an appropriate polynomial g. To find such a g, we most likely need to express f in algebraic normal form.
2. Here, we cannot perform the matrix inversion during the preprocessing stage, because the entries of the matrix depends on the keystream output.
3. Each polynomial evaluation (of g or h) requires $\sum_{i=0}^{e} \binom{n}{i}$ computations if we express the polynomials in algebraic normal form.

In the next subsection, we shall provide a concrete example of this variant of cube attack.

3.1 Application to the Toyocrypt Cipher with Re-synchronization

The main Toyocrypt cipher [15] comprises of a 128-bit MLFSR (modular linear feedback shift register), filtered through a nonlinear function f of degree 63. This f is given by:

$$f(s_0,\ldots,s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} + s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i,$$

where α_i, $0 \le i \le 62$, is a permutation of the set $\{63,\ldots,125\}$. The output of the filter function gives a keystream bit. Upon the next clocking, the MLFSR clocks once and passes through the filter function to give the next keystream bit. For simplicity of explanation, we can treat the MLFSR as a LFSR because as shown in [15], there is a one-to-one linear transformation between the states of the MLFSR and an LFSR.

In [3], Courtois described an algebraic attack on Toyocrypt. He observed that f can be approximated by a degree-4 polynomial g by ignoring the two terms of degree 17 and 63 respectively. The error rate in this approximation is given by 2^{-17} which is good enough for practical purposes. Later, in [5], Courtois and Meier found an even better attack by noting that the polynomials

$$f \cdot (s_{23} + 1) \quad \text{and} \quad f \cdot (s_{42} + 1)$$

are cubic since the variables s_{23} and s_{42} occur in all terms of f of degree at least 4.

The above observations will come in handy when we apply the two variants of cube attack on Toyocrypt. *We shall assume a simplified variant, where during initialization, an n-bit key and m-bit IV are linearly mixed to fill up the LFSR.*

Let us replace f with a quartic polynomial g as mentioned above. We may then write the first bit of the keystream as a quartic polynomial in the key (x_i) and the IV (v_j). In applying cube attack, we require a preprocessing work factor of $n^3 + 8n(n+1)$ and an online work factor of $n^2 + 8n$.

The attack fails if $f \neq g$ for one of the evaluations. We may safely assume that this does not occur during preprocessing (since checks can easily circumvent that); hence, the probability of success is

$$(1 - 2^{-17})^{8n}.$$

Even in the extreme case of $n = 128$, this is greater than 99%.

Cube Attack with Annihilators We can find a degree-1 polynomial g such that $fg = h$ is cubic. Hence the cube attack with annihilators requires only $2^{3-1}n^2 = 4n^2$ evaluations of a linear function during the preprocessing stage. During the online phase, the amount of work is $8n + n^3$.

A Comparison In Table 1, we compare the above variants of the cube attack (quartic approximation, low degree annihilators) with the basic cube attack on the filter function of degree 63 and the algebraic attack of [6, 14] using cubic equations on the Toyocrypt cipher with $n = 128$. We see that our cube attack variant has lower complexities and requires much fewer keystream bits.

Table 1. Comparison of Improved Variants of Cube Attack with the Basic Cube Attack and Algebraic Attack on Toyocrypt with 128-bit State Function Linearly Initialized by 128-bit Key and IV.

	Algebraic Attack [6, 14]	Basic Cube Attack [12]	Basic Cube Attack with Quartic Approximation	Cube Attack with Annihilators (new)
Keystream Bits	2^{18}	$2^{62} \times 128$	$2^3 \times 128$	$2^2 \times 128$
Pre-Computation	2^{30}	2^{76}	2^{21}	2^{16}
Online Complexity	2^{20}	2^{69}	2^{14}	2^{21}

Note that the keystream bits for algebraic attack can be obtained from one keystream while those of the other attacks have to be obtained across different keystreams from re-synchronizations. E.g., in the 4^{th} column, we need $2^2 \times 128 = 2^9$ keystream bits from 4 re-synchronizations, each having 128 bits.

Implementation We implemented both variants of the cube attack (quartic approximation and low-degree annihilators). In both versions, the Toyocrypt cipher can be broken in a few milliseconds on an ordinary PC. Although both variants seem to have comparable pre-computation + online attack time from Table 1, the cube attack with annihilators runs about twice as fast as the basic cube attack using quartic approximation. It also has the slight advantage of being 100% reliable and uses fewer re-synchronizations.

In a Nutshell The example of Toyocrypt is used to illustrate our cube attack variant. It demonstrates its effectiveness against ciphers in which multiplying f with a low-degree polynomial g dramatically lowers its degree.

3.2 Sliding Window Cube Attack on Filter Function Taking Few Inputs

Consider the case where our key is of size N and our filter function f takes only $n < N$ inputs from the state. Suppose we have the following two conditions:

– Linear initialization
– Linear feedback

There is a known re-synchronization attack on such a cipher with complexity $\lceil N/n \rceil \times 2^n$, see [10, Section 3]. We shall describe an extension of the cube attack with annihilators on this cipher where the complexity is generally better than the re-synchronization attack of [10].

Because of the linear initialization, we can write the inputs from the state at time t as $l_t(\mathbf{x}, \mathbf{v})$, where l_t is linear, and consider the filter function as a function of the inputs (as opposed to the entire state). Now its output at time t is

$$z_t = f(l_t(\mathbf{x}, \mathbf{v})) = f(\mathbf{y}_t + l_t(\mathbf{0}, \mathbf{v}))$$

where $\mathbf{y}_t = l_t(\mathbf{x}, \mathbf{0})$.

As before, suppose we have g and h of low degrees e and d respectively such that $fg = h$ and $e < d \leq \deg(f)$. Let us write $f_t(\mathbf{y}_t, \mathbf{v}) = f(\mathbf{y}_t + l_t(\mathbf{0}, \mathbf{v}))$, and define g_t and h_t similarly. We can apply the cube attack with annihilators to $f_t g_t = h_t$ to find $\mathbf{y}_t$ for any t. We choose $\lceil N/n \rceil$ values of t such that the corresponding $\mathbf{y}_t$ give us N linearly independent equations in the (x_i), and solve for the $\mathbf{y}_t$. We can then solve the N linear equations in (x_i) by Gaussian elimination.

Suppose we have found low degree g and h such that $h = fg$. The attack works as follows:

1. *First: The preprocessing stage.* We pick $\lceil N/n \rceil$ values of t such that the $\mathbf{y}_t$ give us N linearly independent equations in the (x_i). For each value of t, we pick $\sum_{i=0}^{e} \binom{n}{i}$ maxterms. For a given maxterm, we denote C to be the cube of 2^{d-e} vectors which have all possible combinations of values for the terms in the maxterm, and have all other terms fixed in some configuration.
   For each maxterm, we compute $\sum_C h_t(\mathbf{y}_t, \mathbf{v})$ by finding the coefficient of every $\mathbf{y}_t$-monomial, of which there are $\sum_{i=0}^{e} \binom{n}{i}$, so h_t gets evaluated $2^{d-e} \sum_{i=0}^{e} \binom{n}{i}$ times.
   Hence the total complexity of this stage is

$$\lceil N/n \rceil 2^{d-e} \sum_{i=0}^{e} \binom{n}{i}$$

2. *Second: The online phase.* Each value of t has $\sum_{i=0}^{e} \binom{n}{i}$ corresponding maxterms, and for each maxterm we can compute $\sum f_t(\mathbf{y}_t, \mathbf{v}) g_t(\mathbf{y}_t, \mathbf{v})$ since we know the keystream bits $f_t(\mathbf{y}_t, \mathbf{v})$. This is a polynomial in $\mathbf{y}_t$, and we find the coefficient of every $\mathbf{y}_t$-monomial as before. This has complexity $2^{d-e} \sum_{i=0}^{e} \binom{n}{i}$. We then equate it to $\sum h_t(\mathbf{y}_t, \mathbf{v})$ to obtain an equation in $\mathbf{y}_t$ of degree at most e. Since there are $\sum_{i=0}^{e} \binom{n}{i}$ maxterms for each t, we get $\sum_{i=0}^{e} \binom{n}{i}$ equations in $\mathbf{y}_t$ of degree at most e, and we can solve for $\mathbf{y}_t$ by linearization. This has complexity $(\sum_{i=0}^{e} \binom{n}{i})^3$.
   After solving for all $\lceil N/n \rceil$ of the $\mathbf{y}_t$, which are linear combinations of (x_i), we get N linear equations in (x_i), and can then solve for $\mathbf{x}$ using Gaussian elimination with complexity N^3.
   The total complexity of this stage is

$$\lceil N/n \rceil \left(2^{d-e} \Big(\sum_{i=0}^{e} \binom{n}{i} \Big)^2 + \Big(\sum_{i=0}^{e} \binom{n}{i} \Big)^3 \right) + N^3$$

Remark 2. This argument also applies in the more general case where the filter function can be written as a function of $\alpha_t(\mathbf{x})$ and $\beta_t(\mathbf{v})$ where α_t, β_t are not necessarily linear, and the α_t are of degree at most c for some small c. In this case, we need to solve for $\binom{n}{c}$ of the $\alpha_t(\mathbf{x})$, and then solve for $\mathbf{x}$ by linearization.

3.3 Applications of the Sliding Window Cube Attack

In this section, we give two examples of the sliding window cube attack on a filter function generator where the filter function has size $n = 128$ and the key has size $N = 256$ and 10000.

Example 1. Consider a filter function generator where a 256-bit key is linearly mixed with a 256-bit IV to fill up a 256 bit LFSR. Let the filter function be a Toyocrypt-like function defined by:

$$f(s_0, \ldots, s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_0 s_1 s_2 \ldots s_{31} + s_{32} s_{33} s_{34} \ldots s_{62},$$

where α_i, $0 \leq i \leq 62$, is a permutation of the set $\{63, \ldots, 125\}$. This function is balanced, has algebraic degree 32 and like the Toyocrypt filter function, near optimal nonlinearity $2^{127} - 2^{64}$ for protection against correlation attack. However, it is easy to see that we can multiply it by $(s_0 + 1)(s_{32} + 1)$ to get a degree 4 equation. Thus $e = 2$ and $d = 4$.

1. The complexity of the sliding window cube attack from Section 3.2 on this cipher is:
$$(256/128) \times \left(\Big(\sum_{i=0}^{2} \binom{128}{i} \Big)^2 \times 2^{4-2} + \Big(\sum_{i=0}^{2} \binom{128}{i} \Big)^3 \right) + 256^3 \approx 2^{40.03}$$
and it needs $\left(\sum_{i=0}^{2} \binom{128}{i} \right) \times (256/128) \approx 2^{14.01}$ keystream bits from each of $2^{4-2} = 4$ re-synchronizations.
2. The complexity of the cube attack with annihilator from Section 3 on this cipher is:
$$\Big(\sum_{i=0}^{2} \binom{256}{i} \Big)^2 \times 2^{4-2} + \Big(\sum_{i=0}^{2} \binom{256}{i} \Big)^3 \approx 2^{45.02}$$
and it needs $\left(\sum_{i=0}^{2} \binom{256}{i} \right) \approx 2^{15.00}$ keystream bits from each of $2^{4-2} = 4$ re-synchronizations.
3. The complexity of the direct cube attack [12] on this cipher by taking the filter function degree as 32 is:
$$256^2 \times 2^{32-1} + 256^3 \approx 2^{47.01}$$
and it needs 256 keystream bits from each of $2^{32-1} = 2^{31}$ re-synchronizations.
4. The complexity of the re-synchronization attack [10] on this cipher by taking the filter function input size as 128 bits is
$$(256/128) \times 2^{128} = 2^{129}$$
and it needs $(256/128) = 2$ keystream bits from each of 128 re-synchronizations.
5. The complexity of the fast algebraic attack [6] on this cipher (where we combine the pre-processing and online complexity) is:
$$\Big(\sum_{i=0}^{2} \binom{256}{i} \Big) \Big(\sum_{i=0}^{4} \binom{256}{i} \Big) + \Big(\sum_{i=0}^{2} \binom{256}{i} \Big)^3 + \Big(\sum_{i=0}^{4} \binom{256}{i} \Big) \log \Big(\sum_{i=0}^{4} \binom{256}{i} \Big) \approx 2^{45.23}$$
and it needs $\sum_{i=0}^{2} \binom{256}{i} \approx 2^{15.00}$ bits from one keystream.

Thus we see that the sliding window cube attack has better attack complexity than the other attacks when applied to on this filter function generator. □

When we increase the size of the LFSR as in the following example, we can see that the cube attack with annihilator may even perform worse than the direct cube attack [12] but the sliding window cube attack will still have better attack complexities than the other attacks.

Example 2. Consider a filter function generator where we use the same filter function as that in Example 1 but increase the size of the key and IV to 10000 bits and used that to initialize a 10000-bit LFSR. Then by replacing $N = 256$ with $N = 10000$ in Example 1, we have the following complexity for the various attacks:

1. Sliding Window Cube Attack: Attack Complexity = $2^{45.37}$ and it needs $2^{19.32}$ keystream bits from each of 4 re-synchronizations.
2. Cube Attack with Annihilators: Attack Complexity = $2^{76.73}$ and it needs $2^{25.58}$ keystream bits from each of 4 re-synchronizations.
3. Direct Cube Attack. Attack Complexity = $2^{57.58}$ and it needs $2^{13.29}$ keystream bits from each of 2^{31} re-synchronizations.
4. Resynchronization Attack. Attack Complexity = $2^{134.30}$ and it needs $2^{6.30}$ keystream bits from each of 128 re-synchronizations.
5. Fast Algebraic Attack. Attack Complexity = $2^{76.95}$ and it needs $2^{25.58}$ bits from one keystream.

Again, we see that the sliding window cube attack has better attack complexity than the other attacks when applied to on this filter function generator. □

4 Cube Attack on Vectorial Filter Function with low $(\mathbf{x}, \mathbf{v})$-Degree

4.1 Applying the Cube Attack to Vectorial Filter Functions

We now consider the case where the state function (e.g. LFSR) is filtered by a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^r$, $r > 1$. In 2005, Canteaut [2] introduced a method for finding implicit equations of the form $G(\mathbf{x}, \mathbf{v}, \mathbf{z}) = 0$ where $\mathbf{z} = F(\mathbf{x}, \mathbf{v})$ and $G(\mathbf{x}, \mathbf{v}, \mathbf{z})$ has low $(\mathbf{x}, \mathbf{v})$-degree and is of unrestricted degree in the output variable $\mathbf{z}$. Then this low degree equation can be solved by XL or linearization methods to recover the secret key.

In a similar way, we can extend the cube attack with annihilators to vectorial filter functions. Let a vectorial filter function be denoted by

$$\mathbf{z} = F(\mathbf{x}, \mathbf{v})$$

where $\mathbf{x}$ is the key of size n, $\mathbf{v}$ is the IV of size m, and $\mathbf{z}$ is a vector of multiple output bits. We can find $G(\mathbf{x}, \mathbf{v}, \mathbf{z})$ of low $(\mathbf{x}, \mathbf{v})$-degree e such that $H(\mathbf{x}, \mathbf{v}) := G(\mathbf{x}, \mathbf{v}, F(\mathbf{x}, \mathbf{v}))$ also has low $(\mathbf{x}, \mathbf{v})$-degree d, with $e < d \leq \deg(F)$. Proposition 1 in Section 4.2 ensures that we can always find such functions $G(\mathbf{x}, \mathbf{v}, \mathbf{z})$ and $H(\mathbf{x}, \mathbf{v})$.

We can apply an adaptation of the attack on 1-bit filter functions to G and H. For a given maxterm, we denote C to be the cube of 2^{d-e} vectors which have all possible combinations of values for the terms in the maxterm, and have all other terms fixed in some configuration. For each $\mathbf{v} \in C$, $\mathbf{z} = F(\mathbf{x}, \mathbf{v})$ is known as it is a keystream bit, so by substituting these keystream bits into $\sum_C G(\mathbf{x}, \mathbf{v}, \mathbf{z}) = \sum_C H(\mathbf{x}, \mathbf{v})$, we get a polynomial of degree at most e. We do this for $\sum_{i=0}^{e} \binom{n}{i}$ maxterms to find $\sum_{i=0}^{e} \binom{n}{i}$ polynomials of degree at most e, and then solve for $\mathbf{x}$ by linearization.

The attack proceeds as follows:

1. *First: the preprocessing stage.* First, we pick $\sum_{i=0}^{e}\binom{n}{i}$ maxterms. For each maxterm, we compute $\sum_C H(\mathbf{x},\mathbf{v})$ by finding the coefficient of every $\mathbf{x}$-monomial, of which there are $\sum_{i=0}^{e}\binom{n}{i}$, so H gets evaluated $2^{d-e}\sum_{i=0}^{e}\binom{n}{i}$ times.
   The total complexity of this stage is
$$2^{d-e}\left(\sum_{i=0}^{e}\binom{n}{i}\right)^2$$
2. *Second: the online phase.* For each maxterm we can compute $\sum_C G(\mathbf{x},\mathbf{v},\mathbf{z})$ as a polynomial of $\mathbf{x}$, since we have the keystream bits $\mathbf{z}$. This has complexity $2^{d-e}\sum_{i=0}^{e}\binom{n}{i}$. We equate this to $\sum H(\mathbf{x},\mathbf{v})$ to obtain an equation in $\mathbf{x}$ of degree at most e. Since there are $\sum_{i=0}^{e}\binom{n}{i}$ maxterms, we get $\sum_{i=0}^{e}\binom{n}{i}$ equations in $\mathbf{x}$ of degree at most e, and we can solve for $\mathbf{x}$ by linearization. This has complexity $(\sum_{i=0}^{e}\binom{n}{i})^3$.
   The total complexity of this stage is
$$2^{d-e}\left(\sum_{i=0}^{e}\binom{n}{i}\right)^2+\left(\sum_{i=0}^{e}\binom{n}{i}\right)^3$$

Remark 3. Given a stream cipher filtered by the vectorial function

$$\mathbf{z}=(z_1,\ldots,z_r)=F(\mathbf{x},\mathbf{v}).$$

A straightforward attack would be to apply the cube attack on a linear combination of output bits, which we denote by $z=\sum_{i\in I}z_i=f(\mathbf{x},\mathbf{v})$. If the attacker is able to find a multiple $f(\mathbf{x},\mathbf{v})g(\mathbf{x},\mathbf{v})$ of low degree d where $zg(\mathbf{x},\mathbf{v})$ has low degree e, the attack complexity can be much reduced as in the attack on Toyocrypt in Section 3.1.

However, it is easy to see that low-degree equations $f(\mathbf{x},\mathbf{v})g(\mathbf{x},\mathbf{v})$ and $zg(\mathbf{x},\mathbf{v})$ are special cases of the equation $G(\mathbf{x},\mathbf{v},F(\mathbf{x},\mathbf{v}))$ of low $(\mathbf{x},\mathbf{v})$-degree d and $G(\mathbf{x},\mathbf{v},\mathbf{z})$ of low $(\mathbf{x},\mathbf{v})$-degree e, considered in Section 4. Therefore we expect the vectorial cube attack in Section 4 to utilize lower degree equations than the single-bit cube attack. This will translate into lower attack complexity when we linearize and solve the resulting system of equations for the secret keys.

4.2 Existence of Low Degree Equations for Vectorial Cube Attack

In contrast with Canteaut's method [2], we need not to have $H(\mathbf{x},\mathbf{v})=0$ for all $\mathbf{x},\mathbf{v}$, so the condition that $2^r\sum_{i=0}^{e}\binom{n+m}{i}>2^{n+m}$ is not necessary. Instead, we require a weaker condition stated as an existence result in Proposition 1 below. The proposition also implies that finding low degree annihilators for vectorial Boolean function case ($r>1$) is no harder than the single output bit case ($r=1$).

Proposition 1 *(Existence of Low Degree Equations) Let a vectorial Boolean function $F:\{0,1\}^n\times\{0,1\}^m\to\{0,1\}^r$ be denoted by $\mathbf{z}=F(\mathbf{x},\mathbf{v})$ where $\mathbf{x}$ is the key of size n, $\mathbf{v}$ is the IV of size m, and $\mathbf{z}$ is a vector of multiple output bits. For $0<e<d\le deg(F)$, if*

$$(2^r-1)\sum_{i=0}^{e}\binom{n+m}{i}+\sum_{i=0}^{d}\binom{n+m}{i}>2^{n+m},$$

then there exists $G(\mathbf{x},\mathbf{v},\mathbf{z})$ of $(\mathbf{x},\mathbf{v})$-degree e such that $H(\mathbf{x},\mathbf{v}):=G(\mathbf{x},\mathbf{v},F(\mathbf{x},\mathbf{v}))$ is of degree d.

Proof. Construct a matrix M with each row indexed by a value of $(\mathbf{x}, \mathbf{v})$, there are 2^{n+m} rows. Let the columns range over all $(\mathbf{x}, \mathbf{v}, \mathbf{z})$-monomials with $(\mathbf{x}, \mathbf{v})$-degree at most e, $\mathbf{z}$-degree unrestricted except that $\mathbf{z} = \mathbf{0}$, as well as all the $(\mathbf{x}, \mathbf{v})$-monomials with degree at most d. The number of columns $n_C := (2^r - 1)\sum_{i=0}^{e}\binom{n+m}{i} + \sum_{i=0}^{d}\binom{n+m}{i}$.

Define the (i, j)-th entry of M to be the value of the monomial corresponding to the j-th column evaluated with the value of $(\mathbf{x}, \mathbf{v})$ at the i-th row, with $\mathbf{z} = \mathbf{F}(\mathbf{x}, \mathbf{v})$.

If $n_C > 2^{n+m}$, we can find a column vector $\mathbf{y} \in \mathcal{F}_2^{n_C}$ by Gaussian elimination such that $M\mathbf{y} = \mathbf{0}$. Then for every value of $(\mathbf{x}, \mathbf{v})$, corresponding to row i, we have

$$\sum_j y_j M_{ij} = 0.$$

Namely, each solution for $\mathbf{y}$ corresponds to a linear combination of some of the column index monomials, the sum of which evaluates to 0 for every value of $(\mathbf{x}, \mathbf{v})$. Let G be the sum of all $(\mathbf{x}, \mathbf{v}, \mathbf{z})$-monomials and H be the sum of all $(\mathbf{x}, \mathbf{v})$-monomials. The proof is done. □

Remark 4. For a general single-bit output Boolean function, it may have algebraic immunity $n/2$, in which case, the best we can do is $d = e = n/2$ for the annihilator cube attack. But for the vectorial case, as shown above, we always get low degree equations when the existence condition holds and in many cases, d and e are lower than $n/2$. Thus theoretically the vectorial cube attack is better than the single-bit output cube attack with annihilator.

4.3 Results from Searching Implicit Low Degree Equations for Vectorial Boolean Functions

We have implemented the algorithm in the proof of Proposition 1 for a few well-known vectorial Boolean functions. Experimental results obtained seem to be even better than the above stated theoretical bound, namely, even if the condition for existence does not hold, as long as the number of columns exceeds the number of non-zero rows in the Reduced Row Echelon Form (RREF) of the matrix, we are still able to find low degree annihilators for $F(\mathbf{x}, \mathbf{v})$. The results are presented in Appendix A.

The algorithm is of at least exponential space complexity to $n + m$. However, it may provide a good motivation and starting point to find efficient algorithms to search for such low degree implicit equations for vectorial cube attack.

5 Conclusion

We have proposed several variants of the cube attack, which makes use of low degree equations. First, the cube attack with annihilators combines the low degree multiples used in algebraic attack with cube attack. The complexity of this combined attack is better than just a direct application of the cube attack or the algebraic attack by itself. This is demonstrated in the attack on Toyocrypt where the attack complexities are lower and the keystream needed is greatly reduced as shown in Table 1.

Second, when the size of the filter function is smaller than the LFSR, we proposed the sliding window cube attack with annihilators. As shown in Examples 1 and 2, it has better complexity than the cube attack with annihilators and the related resynchronization attack of Daemen et. al. [10].

Finally, the vectorial cube attack works on multi-output stream ciphers and it combines the cube attack with a new form of low degree implicit equations. The existence of such equations can

be ensured by the rank computation of certain "monomial" matrices. Because the upper bound of the degree of the implicit equations in the vectorial cube attack is less than that of the degree of annihilators in the single-bit case, we see that theoretically, the vectorial cube attack has better attack complexity than applying the cube attack with annihilators to single-bit output of the vectorial function. We also did some experiments to find low degree implicit equations for the vectorial cube attack and got some results which are better than that expected by theory. These findings may serve as a motivation to find an efficient algorithm to find the low degree vectorial equations $G(\mathbf{x}, \mathbf{v}, F(\mathbf{x}, \mathbf{v}))$ and $G(\mathbf{x}, \mathbf{v}, \mathbf{z})$ for the vectorial cube attack of Section 4.

References

1. F. Armknecht, C. Carlet, P. Gaborit, S. Knzli, W. Meier and O. Ruatta, "Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks", LNCS 4004, *Eurocrypt 2006*, pp. 147-164, Springer-Verlag, 2006.
2. A. Canteaut, "Open Problems Related to Algebraic Attacks on Stream Ciphers", LNCS 3969, *WCC 2005*, pp. 120-134, Springer-Verlag, 2006.
3. N. Courtois, "Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt", LNCS 2587, *ICISC 2002*, pp. 182-199, Springer-Verlag, 2002.
4. N. Courtois, A. Klimov, J. Patarin and A. Shamir, "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations", LNCS 1807, *Eurocrypt 2000*, pp. 392-407, Springer-Verlag, 2000.
5. N. Courtois and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback", LNCS 2656, *Eurocrypt 2003*, pp. 345-359, Springer-Verlag, 2003.
6. N. Courtois and W. Meier, "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback", LNCS 2729, *Crypto 2003*, pp. 176-194, Springer-Verlag, 2003.
7. J.-C. Faugère, "A New Efficient Algorithm for Computing Gröbner Bases (F_4)", *Journal of Pure and Applied Algebra* 139 (1), pp. 61-88, Elsevier Science, 1999.
8. J.-C. Faugère, ""A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F_5)", *Proceedings of the 2002 international symposium on Symbolic and algebraic computation (ISSAC)*, pp. 75-83, ACM Press, 2002.
9. A. Kipnis and A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization", LNCS 1666, *Crypto 1999*, Springer-Verlag, 1999.
10. J. Daemen, R. Govaerts and J. Vandewalle, "Resynchronization Weaknesses in Synchronous Stream Ciphers", LNCS 765, *Eurocrypt'93*, pp. 159-167, Springer-Verlag, 1994.
11. F. Didier and J.-P. Tillich, "Computing the Algebraic Immunity Efficiently", LNCS 4047, *FSE 2006*, pp. 359-374, Springer-Verlag, 2006.
12. I. Dinur and A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials", LNCS 5479, *Eurocrypt 2009*, pp. 278-299, Springer-Verlag, 2009.
13. M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman and Company, 1979.
14. P. Hawkes and G. Rose, "Rewriting Variables: The Complexity of Fast Algebraic Attacks on Stream Ciphers", LNCS 3152, *Crypto 2004*, pp. 390-406, Springer-Verlag, 2004.
15. M. Mihaljevic and H. Imai, "Cryptanalysis of Toyocrypt-HS1 Stream Cipher", IEICE Transactions on Fundamentals, vol. E85-A, pp. 66-73, Jan. 2002.

Appendix

A Implicit Low Degree Equations for Vectorial Boolean Functions

Let a configuration be $(n + m, r, e, d)$, we restrict the output of $F(\mathbf{x}, \mathbf{v})$ to the first r bits. Let n_R be the number of rows in the matrix M, n_C the number of columns, n_{RREF} be the row rank of M,

namely, the number of non-zero rows when M is reduced to the reduced row echelon form (RREF). Let n_S be the number of low degree equations obtained. When working in $GF(2^n)$, the irreducible polynomial is denoted $m(x)$.

$n+m$	r	e	d	n_R	n_C	n_{RREF}	n_S
8	2	1	2	2^8	64	64	0
8	2	1	3	2^8	120	120	0
8	2	1	4	2^8	190	190	0
8	2	2	3	2^8	204	203	1
8	2	2	4	2^8	274	248	26
8	2	3	4	2^8	442	256	186
8	3	1	2	2^8	100	100	0
8	3	1	3	2^8	156	156	0
8	3	1	4	2^8	226	224	2
8	3	2	3	2^8	352	256	96
8	3	2	4	2^8	422	256	166
8	3	2	4	2^8	422	256	166
8	3	3	4	2^8	814	256	558
8	4	1	2	2^8	172	172	0
8	4	1	3	2^8	228	225	3
8	4	1	4	2^8	298	256	42
8	4	2	3	2^8	648	256	392
8	4	2	4	2^8	718	256	462
8	4	3	4	2^8	1558	256	1295

Table 2. $F : \{0,1\}^8 \to \{0,1\}^8$, F is the S-Box of AES.

$n+m$	r	e	d	n_R	n_C	n_{RREF}	n_S
9	2	1	2	2^9	76	76	0
9	2	1	3	2^9	160	160	0
9	2	1	4	2^9	286	286	0
9	2	2	3	2^9	268	268	0
9	2	2	4	2^9	394	384	10
9	2	3	4	2^9	646	511	135
9	3	1	2	2^9	116	116	0
9	3	1	3	2^9	200	200	0
9	3	1	4	2^9	326	326	0
9	3	2	3	2^9	452	441	11
9	3	2	4	2^9	578	510	68
9	3	3	4	2^9	1166	512	654
9	4	1	2	2^9	196	196	0
9	4	1	3	2^9	280	280	0
9	4	1	4	2^9	406	403	3
9	4	2	3	2^9	820	512	308
9	4	2	4	2^9	946	512	434
9	4	3	4	2^9	2206	512	1694

Table 3. $F : GF(2^9) \rightarrow GF(2^9)$, the inverse function, $m(x) = x^9 + x^4 + 1$

$n+m$	r	e	d	n_R	n_C	n_{RREF}	n_S
10	2	1	2	2^{10}	89	89	0
10	2	1	3	2^{10}	209	209	0
10	2	1	4	2^{10}	419	419	0
10	2	2	3	2^{10}	344	344	0
10	2	2	4	2^{10}	554	554	0
10	2	3	4	2^{10}	914	873	41
10	3	1	2	2^{10}	133	133	0
10	3	1	3	2^{10}	253	253	0
10	3	1	4	2^{10}	463	463	0
10	3	2	3	2^{10}	568	568	0
10	3	2	4	2^{10}	778	751	27
10	3	3	4	2^{10}	1618	1024	594
10	4	1	2	2^{10}	221	221	0
10	4	1	3	2^{10}	341	341	0
10	4	1	4	2^{10}	551	551	0
10	4	2	3	2^{10}	1016	993	23
10	4	2	4	2^{10}	1226	1024	202
10	4	3	4	2^{10}	3026	1024	2002

Table 4. $F : GF(2^{10}) \rightarrow GF(2^{10})$, the inverse function, $m(x) = x^{10} + x^3 + 1$