Extensions of the Cube Attack

Aileen Zhang, Chu-Wee Lim and Khoongming Khoo

DSO National Laboratories
20 Science Park Drive, Singapore 118230
Email: zyinghui,lchuwee,kkhoongm@dso.org.sg

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. We offer a few extensions of this attack by applying it to Boolean functions for which we can find low-degree multiples. We then extend this to vectorial Boolean functions by finding relations with low-degree polynomials.

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

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 [12]). 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 [8], eXtended Linearization [4], Gröbner basis [6, 7] and the fast algebraic attack [5].

In Aug 2008, during the Crypto conference, Adi Shamir [11] 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 also consider the special case where the filter function takes few inputs, and has linear initialization and linear feedback. In Section 4, we consider the cube attack applied to vectorial filter functions, and also the special case where the filter function takes few inputs. Finally we give our conclusion on these variations on the cube attack.

2 Preliminaries: Cube Attack

First let us give a brief overview of the cube attack [11]. 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 [11], 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, *Cube Attack* proceeds 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 [6, 7].

We shall term this method *Cube Attack with Annihilators*. It proceeds as follows:

1. *First, find g and h.* There are many efficient algorithms in literature. See [1] and [10] for example.
2. *Next, 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 $\binom{n}{e}$ monomials, we need to evaluate this sum $\binom{n}{e}$ times (by pumping in different $\mathbf{x}$'s) to find the coefficients. This requires $2^{d-e}\binom{n}{e}$ evaluations of h to compute the coefficients of a single P. For linearization to work, we need $\binom{n}{e}$ such polynomials, so the total amount of work is:

$$2^{d-e}\binom{n}{e}^2$$

evaluations of h.

3. *Finally, the online phase.* We must compute $\binom{n}{e}$ sums of $\sum_{\mathbf{v} \in C} f(\mathbf{x}, \mathbf{v})g(\mathbf{x}, \mathbf{v})$. The polynomial $g(\mathbf{x}, \mathbf{v})$, for a fixed $\mathbf{v} \in C$ has typically $\binom{n}{e}$ terms. Hence, the computation of the above term requires $\binom{n}{e}2^{d-e}$ computations. For $\binom{n}{e}$ such maxterms, we require $\binom{n}{e}^2 2^{d-e}$ computations. Finally linearization of

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

gives a system of $\binom{n}{e} \times \binom{n}{e}$ linear equations which requires $\binom{n}{e}^3$ operations to solve. Hence, the total amount of work is about:

$$\binom{n}{e}^2 2^{d-e} + \binom{n}{e}^3.$$

Here are some differences between the basic cube attack and this version.

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 $\binom{n}{e}$ 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 [14] 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:

$$\begin{aligned} f(s_0, \ldots, s_{127}) =& s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} + \\ & s_1s_2s_9s_{12}s_{18}s_{20}s_{23}s_{25}s_{26}s_{28}s_{33}s_{38}s_{41}s_{42}s_{51}s_{53}s_{59} + \prod_{i=0}^{62} s_i, \end{aligned}$$

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 [14], 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 [5, 13] 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 [5, 13]	Basic Cube Attack [11]	Basic Cube Attack with Quartic Approximation	Cube Attack with Annihilators (new)
Keystream Bits	2^{18}	2^{62}	2^{3}	2^{2}
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.

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 [9, 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 [9].

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.

The attack works as follows:

1. *Find g and h.* We use known algorithms from [1, 10] to find suitable g and h.
2. *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 $\binom{n}{e}$ 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 $\binom{n}{e}$, so h_t gets evaluated $2^{d-e}\binom{n}{e}$ times.
Hence the total complexity of this stage is

$$\lceil N/n \rceil \binom{n}{e}^2 2^{d-e}$$

3. *The online phase.* Each value of t has $\binom{n}{e}$ 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 $\binom{n}{e}2^{d-e}$. 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 $\binom{n}{e}$ maxterms for each t, we get $\binom{n}{e}$ equations in $\mathbf{y}_t$ of degree at most e, and we can solve for $\mathbf{y}_t$ by linearization. This has complexity $\binom{n}{e}^3$. 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

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

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.

Comparison with the Re-synchronization Attack of [9] Consider a stream cipher where a 128-bit LFSR is linearly initialized by a 128-bit key and IV, and filtered by an n-bit Boolean function. The re-synchronization attack on this stream cipher [9] has complexity $\lceil 128/n \rceil \times 2^n$. In Table 2, we tabulate the smallest n where our sliding window cube attack using equations of degrees e and d performs better than the re-synchronization attack of [9], i.e. when

$$\lceil 128/n \rceil \left(\binom{n}{e}^2 2^{d-e} + \binom{n}{e}^3 \right) + 128^3 < \lceil 128/n \rceil \times 2^n,$$

Therefore, we can use the re-synchronization attack of [9] when n is small and use the sliding window cube attack when n is bigger.

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

Table 2. Size of Boolean function n where the Sliding Window Cube Attack is Faster than the Re-synchronization Attack of [9], Key Size is $N = 128$.

Degree of $zg(\mathbf{x}, \mathbf{z})$	Degree of $f(\mathbf{x}, \mathbf{z})g(\mathbf{x}, \mathbf{z})$	Size of Boolean Function
$e = 2$	$2 \leq d \leq 8$	$n \geq 25$
$e = 2$	$9 \leq d \leq 11$	$n \geq 26$
$e = 3$	$3 \leq d \leq 13$	$n \geq 40$
$e = 4$	$4 \leq d \leq 21$	$n \geq 56$
$e = 5$	$5 \leq d \leq 24$	$n \geq 71$

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 $e < d \leq \deg(F)$.

In contrast with Canteaut's method, we need not have $H(\mathbf{x}, \mathbf{v}) = 0$ for all $\mathbf{x}, \mathbf{v}$, and so we do not need the condition that $2^r\binom{n+m}{e} > 2^{n+m}$. Instead, we require that

$$2^r\binom{n+m}{e} + \binom{n+m}{d} > 2^{n+m}.$$

See the Appendix part A for details.

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 $\binom{n}{e}$ maxterms to find $\binom{n}{e}$ polynomials of degree at most e, and then solve for $\mathbf{x}$ by linearization.

The attack is as follows:

1. *Find G and H.* See the Appendix part A for a method to find G and H.
2. *The preprocessing stage.* First, we pick $\binom{n}{e}$ 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 $\binom{n}{e}$, so H gets evaluated $\binom{n}{e}2^{d-e}$ times.
   The total complexity of this stage is

$$\binom{n}{e}^2 2^{d-e}$$

3. *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 $\binom{n}{e}2^{d-e}$. 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 $\binom{n}{e}$ maxterms, we get $\binom{n}{e}$ equations in $\mathbf{x}$ of degree at most e, and we can solve for $\mathbf{x}$ by linearization. This has complexity $\binom{n}{e}^3$. The total complexity of this stage is
$$\binom{n}{e}^2 2^{d-e} + \binom{n}{e}^3$$

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 Sliding Window Cube attack on Vectorial Filter Function Taking Few Inputs

As in the 1-bit case above, we can consider the case where our key is of size N and our vectorial filter function F takes only $n < N$ inputs from the state, and has r outputs. Suppose we have the following two conditions:

– Linear initialization
– Linear feedback for the state function, e.g. LFSR

We can then 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
$$\mathbf{z}_t = F(l_t(\mathbf{x}, \mathbf{v})) = F(\mathbf{y}_t + l_t(\mathbf{0}, \mathbf{v}))$$
where $\mathbf{y}_t = l_t(\mathbf{x}, \mathbf{0})$.

As in the attack on the vectorial filter function, suppose we have $G(\mathbf{y}, \mathbf{w}, \mathbf{z})$ of low $(\mathbf{y}, \mathbf{w})$-degree e such that $H(\mathbf{y}, \mathbf{w}) := G(\mathbf{y}, \mathbf{w}, F(\mathbf{y} + \mathbf{w}))$ has low $(\mathbf{y}, \mathbf{w})$-degree d, and $e < d \leq \deg(F)$. Let us write $G_t(\mathbf{y}_t, \mathbf{v}, \mathbf{z}) = G(\mathbf{y}_t + l_t(\mathbf{0}, \mathbf{v}), \mathbf{z}_t))$ and $H_t(\mathbf{y}_t, \mathbf{v}) = H(\mathbf{y}_t + l_t(\mathbf{0}, \mathbf{v}))$. We can apply the attack on the vectorial filter function to $G_t = H_t$ to find $\mathbf{y}_t$ for any t. This gives us n linear equations in the (x_i), so we can apply the attack to $\lceil N/n \rceil$ values of t such that the corresponding $\mathbf{y}_t$ give us N linearly independent equations in the (x_i). We can then solve these equations by Gaussian elimination.

The attack works as follows:

1. *Find G and H.* See the Appendix part A for a method to find G and H.
2. *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 $\binom{n}{e}$ 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 $\binom{n}{e}$, so H_t gets evaluated $\binom{n}{e}2^{d-e}$ times.
   The total complexity of this stage is
   $$\lceil N/n \rceil \binom{n}{e}^2 2^{d-e}$$
3. *The online phase.* Each value of t has $\binom{n}{e}$ corresponding maxterms, and 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 $\binom{n}{e}2^{d-e}$. We equate this to $\sum_C H_t(\mathbf{y}_t, \mathbf{v})$ to obtain an equation in $\mathbf{y}_t$ of degree at most e. Since there are $\binom{n}{e}$ maxterms for each t, we get $\binom{n}{e}$ equations in $\mathbf{y}_t$ of degree at most e, and we can solve for $\mathbf{y}_t$ by linearization. This has complexity $\binom{n}{e}^3$.
   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
   $$\lceil N/n \rceil \left(\binom{n}{e}^2 (2^{d-e} + \binom{n}{e}^3 \right) + N^3$$

Remark 4. As explained in Section 3.2, the vectorial sliding window cube attack is faster than the re-synchronization attack of [9] when n is not too small. Therefore it can be used in conjunction with the attack of [9] to tackle vectorial filter functions of different input size n.

5 Conclusion

We have proposed two variants of the cube attack, which makes use of low degree equations. First, the cube attack with annihilators works on single-bit output stream ciphers and it combines the low degree multiples used in algebraic attack with cube attack to get attack complexities that are much better than those of the individual attacks. This is demonstrated in the attack on Toyocrypt where the attack complexities are lower and the keystream needed is greatly reduced. Second, the vectorial cube attack works on multi-output stream ciphers and it combines the cube attack with a new form of low degree vectorial equations, which can be obtained from rank computation of certain "monomial" matrices. It improves on the low-degree cube attack on a linear combination of output bits.

We also described sliding window cube attacks on single and multi-output Boolean functions. These improve the cube attack complexities for filter function generators when the size of the Boolean function is smaller than the size of the LFSR. This can be used in conjunction with the linear re-synchronization attack of [9] for better attack complexities across different input size n.

An open question is 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 attack of Section 4.

6 Acknowledgement

We would like to thank Prof. Josef Pieprzyk for his careful reading of the paper and many invaluable suggestions.

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. 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.
7. 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.
8. A. Kipnis and A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization", LNCS 1666, *Crypto 1999*, Springer-Verlag, 1999.
9. J. Daemen, R. Govaerts and J. Vandewalle, "Resynchronization Weaknesses in Synchronous Stream Ciphers", LNCS 765, *Eurocrypt'93*, pp. 159-167, Springer-Verlag, 1994.
10. F. Didier and J.-P. Tillich, "Computing the Algebraic Immunity Efficiently", LNCS 4047, *FSE 2006*, pp. 359-374, Springer-Verlag, 2006.
11. I. Dinur and A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials", ...

12. M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman and Company, 1979.
13. 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.
14. 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 Method to find Implicit Low Degree Equations for Vectorial Boolean Functions

Here we present a method such that given $F(\mathbf{x},\mathbf{v})$, where $\mathbf{x}$ is of size n and $\mathbf{v}$ is of size m, we can find $G(\mathbf{x},\mathbf{v},\mathbf{z})$ and $H(\mathbf{x},\mathbf{v})$ so that

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

and G and H are of low degrees e and d respectively, with $e < d \leq n$.

We construct the matrix M where the rows range over all the possible values of $(\mathbf{x},\mathbf{v})$, so there are 2^{n+m} rows. We let its columns range over all $(\mathbf{x},\mathbf{v},\mathbf{z})$-monomials with $(\mathbf{x},\mathbf{v})$-degree at most e, and $\mathbf{z}$-degree unrestricted, as well as all the $(\mathbf{x},\mathbf{v})$-monomials with degree at most d. There are $2^r\binom{(n+m)}{e} + \binom{(n+m)}{d}$ such monomials.

We define the (i,j)th entry of M to be the value of the monomial corresponding to the jth column evaluated with the value of $(\mathbf{x},\mathbf{v})$ corresponding to the ith row, where $\mathbf{z} = F(\mathbf{x},\mathbf{v})$ in the monomial.

If this matrix has more columns than rows, i.e.

$$2^r\binom{(n+m)}{e} + \binom{(n+m)}{d} \geq 2^{n+m}$$

then we can find a vector $\mathbf{y} \in \mathbb{F}_2^{2^r\binom{(n+m)}{e}+\binom{(n+m)}{d}}$ such that $M\mathbf{y} = \mathbf{0}$. Then for every value of $(\mathbf{x},\mathbf{v})$, corresponding to row i, we have

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

Hence we have a linear equation of the monomials that is true for all values of $(\mathbf{x},\mathbf{v})$. Let G be equal to all the terms containing $\mathbf{z}$, and H be equal to the rest of the terms.