Comparison of cube attacks over different vector spaces

Richard Winter[1], Ana Salagean[1], and Raphael C.-W. Phan[2]

[1] Department of Computer Science, Loughborough University, Loughborough, UK
{R.Winter, A.M.Salagean}@lboro.ac.uk
[2] Faculty of Engineering, Multimedia University, Malaysia
raphael@mmu.edu.my

Abstract. We generalise the cube attack of Dinur and Shamir (and the similar AIDA attack of Vielhaber) to a more general higher order differentiation attack, by summing over an arbitrary subspace of the space of initialisation vectors. The Moebius transform can be used for efficiently examining all the subspaces of a big space, similar to the method used by Fouque and Vannet for the usual cube attack.
Secondly we propose replacing the Generalised Linearity Test proposed by Dinur and Shamir with a test based on higher order differentiation/ Moebius transform. We show that the proposed test provides all the information provided by the Generalised Linearity Test, at the same computational cost. In addition, for functions that do not pass the linearity test it also provides, at no extra cost, an estimate of the degree of the function. This is useful for guiding the heuristics for the cube/AIDA attacks.
Finally we implement our ideas and test them on the stream cipher Trivium.

Keywords: Cube/AIDA attack, Trivium, Linearity testing, Moebius transform, higher order differentiation

1 Introduction

The cube attack introduced by Dinur and Shamir [3] and the similar AIDA attack introduced by Vielhaber [11] have received much attention over the last few years. They can be viewed as higher order differential attacks (see [5] and [9]). The idea of higher order differentials was introduced in cryptography by Lai [10] and was used in many different attacks, most of them being statistical attacks, whereas the cube/AIDA attacks are primarily algebraic.

Several techniques were proposed in order to make the cube attack more efficient. Of particular interest to the present work are the Moebius transform used by Fouque and Vannet [7] and the Generalised Linearity Test introduced by Dinur and Shamir [4].

We propose generalising the cube attack by using using higher order differentiation in its general form. In other words, rather than summing over a "cube",

we sum over an arbitrary subspace of the space of public (tweakable) variables. The usual cube attack becomes then the particular case where the subspace is generated by vectors from the canonical basis. The Moebius transform can again be used to make computations more efficient, by reusing values to compute the summations over many subspaces at once.

Secondly we propose an alternative to the Generalised Linearity Test proposed by Dinur and Shamir [4]. Given a set of t linearly independent keys, our linearity test computes the higher order derivatives of order $2, 3, \ldots, t$ with respect to any subset of keys and then checks whether all the results are zero. If a function fails this linearity test, the lowest order of a non-zero derivative gives a lower bound for the degree of the function, so we obtain extra information at no extra cost. We show that the set of functions that pass the General Linearity Test in [4] is exactly the same as the set of functions that pass our proposed test. The extra information about the degree is useful for guiding the heuristics in the cube attack, as it gives us information as to whether we are close or not to obtaining a linear function. Also, in some implementations of the cube attack, quadratic equations are used if insufficient linear equations are found.

We implemented our ideas and tested the implementation on the stream cipher Trivium [1], which is a popular candidate for testing cube attacks. We looked at between 640 and 703 initialisation rounds. We tested several spaces of initialisation vectors of dimension 28 (including the space corresponding to a usual cube attack), and all their subspaces, using the Moebius transform. We estimated the degrees of the results using our proposed linearity test.

We found one particular vector space which, compared to the usual cube attack, produces significantly more linear equations, but at a slightly higher dimension of subspaces. However for most vector spaces the results are significantly worse than the usual cube attack, which leads us to believe that the success of the usual cube attack on Trivium is not only due to the relatively low degree of the polynomial, but also to the fact that the monomials of that polynomial are not uniformly distributed, as would be expected if it was a random polynomial. In other words, the polynomials in Trivium are "aligned" with the canonical basis rather than being in a generic position. We suggest therefore that preceding Trivium by a (secret) linear change of coordinates on the initialisation vectors would improve its resistance to cube attacks. We did some preliminary experimental testing of this idea, but a full exploration would be a topic of future work.

2 Preliminaries

2.1 Cube Attack

The Cube attack was originally proposed by Dinur and Shamir in [3] and is closely related to the AIDA attack introduced by Vielhaber in [11].

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function in variables $x_1 \ldots x_n$. Any Boolean function can be written in Algebraic Normal Form, i.e. as a polynomial function of degree at most one in each variable. Choosing a subset of indices $I =$

$\{i_1 \dots i_k\} \subseteq \{1, 2, \dots, n\}$, the "cube" C_I is defined by choosing the 2^k possible 0/1 combinations for the variables with indices in I, with the other variables left undetermined. Summing over all vectors in C_I we obtain a function

$$f_I = \sum_{\mathbf{v} \in C_I} f(\mathbf{v}).$$

which depends only on the variables which are not in I.

Factoring out the term $t_I = x_{i_1} \cdots x_{i_k}$, we can write f as

$$f(x_1, \dots, x_n) = t_I f_{S(I)} + r(x_1, \dots, x_n).$$

where $f_{S(I)}$ is a polynomial that shares no common variables with t_I, whereas r is a polynomial in which each term misses at least one variable in t_I. The main results on which the cube attack is based are:

Theorem 1. *([3, Theorem 1]) For any polynomial f and subset of variables I, $f_I \equiv f_{S(I)} \pmod 2$.*

Corollary 1. *If $\deg(f) = d$ and I contains $d - 1$ elements, then f_I has degree at most one.*

When mounting an actual attack, we have two types of variables, the secret variables $x_1, \dots, x_n$ and the public, or "tweakable" variables $v_1, \dots, v_m$, which the attacker can control. The cipher consists of a "black box" function $g : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2$. The attacker chooses a set I of indices of the public variables, sets the other public variables to constant values (usually zero) and computes g_I, which will now only depend on the secret variables. In the preprocessing phase, the attacker studies the cipher, so they can evaluate g_I for any chosen values of the secret variables. It is hoped that, for suitable choices of I (particularly the ones of cardinality approaching $\deg(g) - 1$, assuming $\deg(g) \leq m$), g_I is linear (but not constant) in the secret variables. Linearity tests are discussed in the next section.

If the preprocessing phase found a large number of sets I for which g_I is linear and non-constant (ideally n linearly independent g_I), one can then use this information in the online phase. Now the secret variables are unknown, but the attacker can still control the public variables. The attacker computes g_I for the values of I identified in the preprocessing phase, and then they can determine the secret variables by solving a system of linear equations.

2.2 Generalised Linearity Test

Consider a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. We want to decide whether f is an affine function, i.e. it is a polynomial of degree one or less. We assume n is large and evaluations of f are costly, so we cannot evaluate f for all its inputs. We are looking therefore for a probabilistic test.

In the original cube attack paper [3], Dinur and Shamir used the BLR test, i.e. the textbook definition of linearity: test whether $f(\mathbf{a}) + f(\mathbf{b}) = f(\mathbf{a} + \mathbf{b}) + f(\mathbf{0})$.

If f fails this test, then it is not an affine function. If it passes the test for "sufficiently many" pairs $\mathbf{a}, \mathbf{b}$, then we conclude that f is probably affine.

Since the test above needs 3 evaluations of f for each test (assuming we store and reuse $f(\mathbf{0})$), in [4, Section 4], Dinur and Shamir proposed the following Generalised Linearity Test, which has the advantage that it reuses many evaluations of f so it is overall much more computationally efficient.

Consider a set $\{\mathbf{b}_1, \ldots, \mathbf{b}_t\} \subseteq \mathbb{F}_2^n$ of linearly independent elements. The Generalised Linearity Test consists of the following set of $2^t - t - 1$ equations:

$$\left\{ f(\sum_{i=0}^{t} c_i \mathbf{b}_i) + \sum_{i=0}^{t} c_i f(\mathbf{b}_i) + ((\mathrm{w}(\mathbf{c}) - 1)) \bmod 2) f(\mathbf{0}) = 0 | \mathbf{c} \in \mathbb{F}_2^n, \mathrm{w}(c) \geq 2 \right\} \tag{1}$$

where w( ) denotes the Hamming weight. Again, if there are equations which are not satisfied by f, then f is not affine, otherwise we conclude that f is probably affine (assuming t is "large enough"). Note that here we need 2^t evaluations of f for $2^t - t - 1$ tests, so an amortised cost of just over one evaluation per test, compared to 3 evaluations for the previous test.

Remark 1. In the original description of this test in [4] there is a mistake, in that the term $(\mathrm{w}(\mathbf{c}) - 1) \bmod 2$ is missing. This would make the test incorrect whenever the weight is odd, so affine functions with non-zero constant term would wrongly fail the test.

2.3 Moebius Transform

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The Moebius transform of f is a function $f^M : \mathbb{F}_2^n \to \mathbb{F}_2$ defined as $f^M(\mathbf{y}) = \sum_{\mathbf{x} \preceq \mathbf{y}} f(\mathbf{x})$ where $\mathbf{x} = (x_1, \ldots, x_n)$ and the partial order relation $\preceq$ is defined as $(x_1, \ldots, x_n) \preceq (y_1, \ldots, y_n)$ iff $x_i \leq y_i$ for all $i = 1, \ldots, n$. It is well known that the Moebius transform has the property that for any $\mathbf{a} \in \mathbb{F}_2^n$, $f^M(\mathbf{a})$ equals the coefficient of the term $x_1^{a_1} \ldots x_n^{a_n}$ in f. Further details about the Moebius transform can be found, for example, in [8]. An efficient algorithm which, given the truth table of f computes the truth table of f^M in-place in $n2^{n-1}$ operations is also given in [8].

In connection with the cube attack, note that when choosing a set of variable indices $I = \{i_1, \ldots, i_k\}$, if we define $\mathbf{a}$ as having ones in the positions in I and zeroes elsewhere, we have $f_I(\mathbf{0}) = f^M(\mathbf{a})$. Hence the algorithm for computing the Moebius transform can also be used for efficiently computing f_J for all the subsets J of a large set I.

The idea of making the cube attack more efficient by reusing computations for cubes which are all subcubes of a very large cube was sketched by Dinur and Shamir [4]. Fouque and Vannet [6] fully developed this powerful technique via Moebius transforms, thus obtaining results for Trivium for a larger number of initialisation rounds than previous cube attacks.

2.4 Higher order differentiation

The notion of higher order derivative (or higher order differentiation) was introduced in the cryptographic context by Lai [10].

Definition 1. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function in n variables $x_1, \ldots, x_n$. Let $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}$. The* differentiation operator *(or finite difference operator) along a vector $\mathbf{a}$ associates to each function f the function $\Delta_{\mathbf{a}} f$ (the derivative of f) defined as*

$$\Delta_{\mathbf{a}} f(x_1, \ldots, x_n) = f(x_1 + a_1, \ldots, x_n + a_n) + f(x_1, \ldots, x_n).$$

Denoting $\mathbf{x} = (x_1, \ldots, x_n)$ we can also write $\Delta_{\mathbf{a}} f(\mathbf{x}) = f(\mathbf{x} + \mathbf{a}) + f(\mathbf{x})$.

Higher order differentiation (higher order derivative) refers to repeated application of this operator and will be denoted as:

$$\Delta^{(k)}_{\mathbf{a_1}, \ldots, \mathbf{a_k}} f = \Delta_{\mathbf{a_1}} \Delta_{\mathbf{a_2}} \ldots \Delta_{\mathbf{a_k}} f$$

where $\mathbf{a_1}, \ldots, \mathbf{a_k} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}$ are linearly independent. An explicit expression for computing higher order derivatives follows directly from the definition:

$$\Delta^{(k)}_{\mathbf{a_1}, \ldots, \mathbf{a_k}} f = \sum_{(c_1, \ldots, c_k) \in \{0,1\}^k} f(\mathbf{x} + c_1 \mathbf{a_1} + \ldots + c_k \mathbf{a_k}) \tag{2}$$

Differentiation decreases the degree of polynomials:

Theorem 2. *[10] Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and $\mathbf{a} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}$. Then* $\deg(\Delta_{\mathbf{a}} f) \leq \deg(f) - 1$.

The main construction of the cube attack can be reformulated in terms of higher order differentiation, see [5] and [9]. Namely for a set of indices $I = \{i_1, \ldots, i_k\}$

$$f_I = \Delta^{(k)}_{\mathbf{e_{i_1}}, \ldots, \mathbf{e_{i_k}}} f$$

where $\mathbf{e_i}$ are the vectors of the canonical basis, i.e. they have a one in position i and zeroes elsewhere.

3 General differentiation attack

As in Subsection 2.1 we assume the cipher consists of a "black box" function $g(\mathbf{x}, \mathbf{v})$ with $g : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2$ and $\mathbf{x}$ denoting secret variables and $\mathbf{v}$ denoting public variables. We generalise the cube/AIDA attacks by choosing an arbitrary subspace $V \subseteq \mathbb{F}_2^m$ of the space of public variables and defining a function g_V as

$$g_V(\mathbf{x}) = \sum_{\mathbf{v} \in V} g(\mathbf{x}, \mathbf{v}).$$

Denote by k the dimension of V and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ be a basis for V. Using equation (2) we can give an equivalent formula for g_V using higher order differentiation, namely $g_V(\mathbf{x}) = (\Delta^{(k)}_{\mathbf{v}_1, \ldots, \mathbf{v}_k} g)(\mathbf{x}, \mathbf{0})$. Note that the usual cube attack

becomes a particular case of this attack, for $V = \langle \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_k} \rangle$, where $\mathbf{e}_i$ are the vectors of the canonical basis, and $I = \{i_1, \ldots, i_k\}$ are the positions chosen for the usual cube attack.

Using Theorem 2 we have that $\deg(g_V) \leq \deg(g) - \dim(V)$. Hence, as in the cube attack (see Corollary 1), if the dimension of V is $k = \deg(g) - 1$ we are guaranteed that g_V is linear or constant.

We can therefore search for spaces V such that the resulting g_V is a linear function in the secret variables. Linearity tests can detect whether the result is linear, like in the usual cube attack. This search space is a superset of the search space of the usual cube attack.

Moebius transform can be used here again for improved efficiency. Namely we start with a large vector space $V = \langle \mathbf{v}_1, \ldots, \mathbf{v}_k \rangle$ For any fixed value $\mathbf{x}$ of the secret variables we compute the truth table of the function $h(y_1, \ldots, y_k) = g(\mathbf{x}, y_1\mathbf{v}_1 + \ldots + y_k\mathbf{v}_k)$. We then apply the Moebius transform to h. For any subspace $V' = \langle \mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_j} \rangle$ of V let $\mathbf{a}$ be the vector with ones in exactly the positions $i_1, \ldots, i_j$ and zeroes elsewhere. We have $h^M(\mathbf{a}) = \sum_{(c_1,\ldots,c_j)\in\{0,1\}^j} g(\mathbf{x}, c_1\mathbf{v}_{i_1} + \ldots + c_j\mathbf{v}_{i_j}) = \sum_{\mathbf{v}'\in V'} g(\mathbf{x}, \mathbf{v}') = g_{V'}(\mathbf{x}, \mathbf{0})$. Hence, again, the Moebius transform h^M computes simultaneously all the $g_{V'}(\mathbf{x}, \mathbf{0})$ for all subspaces V' of V.

Remark 2. In [3] Dinur and Shamir also consider the possibility of setting some of the non-cube public variables to 1 rather than zero. That is not the same as our approach, as the set to sum over in that case is no longer a vector space. We can include that generalisation in our approach as follows. Let $\mathbf{c}$ be a fixed vector of public variables and V a vector space. Instead of computing f_V as before, we can compute instead the sum

$$\sum_{\mathbf{v}\in V} g(\mathbf{x}, \mathbf{c} + \mathbf{v})$$

which, using equation (2), can be proved to equal $(\Delta^{(k)}_{\mathbf{v}_1,\ldots,\mathbf{v}_k} g)(\mathbf{x}, \mathbf{c})$. The attack can work equally well in this scenario.

4 Proposed linearity test

We propose an alternative to the Generalised Linearity Test presented in Subsection 2.2. Again, let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and let $\{\mathbf{b}_1, \ldots, \mathbf{b}_t\} \subseteq \mathbb{F}_2^n$ be a set of t linearly independent elements. (The question of how to choose a suitable value for t is, as we shall see, exactly the same as in [3], and a further discussion of this choice is beyond the scope of this paper.) For each d with $1 \leq d \leq t$ consider the following set of equations:

$$L_d = \left\{ \sum_{\mathbf{u}\preceq\mathbf{c}} f(\sum_{i=0}^{t} u_i\mathbf{b}_i) = 0 | \mathbf{c} \in \mathbb{F}_2^t, \mathrm{w}(\mathbf{c}) = d \right\} \tag{3}$$

where $\mathbf{u} \preceq \mathbf{c}$ means $u_i \leq c_i$ for all $i = 1, \ldots, t$, as defined in Section 2.3. Each L_d has $\binom{t}{d}$ equations. Each equation can alternatively be written using higher

order derivatives:

$$L_d = \left\{ \Delta^{(d)}_{\mathbf{b}_{i_1},\ldots,\mathbf{b}_{i_d}} f(\mathbf{0}) = 0 | \{i_1,\ldots,i_d\} \subseteq \{1,\ldots,t\} \right\}.$$

Note that the equations in L_2 correspond to the usual (BLR) linearity tests. The equations in L_3 have been proposed for testing whether the function is quadratic in [4, Section 4]. The following result is quite straightforward but we prove it for completeness.

Proposition 1. *If f has degree d, then it satisfies all the equations in the sets $L_{d+1},\ldots,L_t$.*

Proof. By Theorem 2, $\deg(\Delta^{(j)}_{\mathbf{b}_{i_1},\ldots,\mathbf{b}_{i_j}} f) \leq \deg(f) - j = d - j$. Hence for all $j > d$ we have $\Delta^{(j)}_{\mathbf{b}_{i_1},\ldots,\mathbf{b}_{i_j}} f \equiv 0$, so the equations L_j are satisfied for all $j > d$.

One can give an alternative proof of this result using the Moebius transform.

Based on the result above we propose an alternative to the Generalised Linearity Test. Namely we test whether a function f has degree one or less by testing whether it satisfies the equations in the sets $L_2,\ldots,L_t$. An advantage of this test is that if a function fails the test (i.e. has degree more than one), we can get, at no additional cost, an indication of its degree. Namely, if d is the highest number for which some equations in L_d are not satisfied, then we know that $\deg(f) \geq d$. We will estimate the degree of f as being d.

The proposed test contains the same number of equations (namely $2^t - t - 1$) and needs the same number of evaluations of f (namely 2^t) as the Generalised Linearity Test. If the function f is affine (has degree at most one), then it passes both types of test, so there are no false negatives. If f has degree 2 or more it might still pass one of the types of tests (i.e. we can have false positives). We can ask ourselves whether some functions can pass our proposed test but fail the Generalised Linearity Test or vice-versa. We show that this is not possible, in other words the tests are equivalent. More precisely:

Proposition 2. *A function f satisfies the Generalised Linearity Test (1) iff it satisfies the sets of equations $L_2, L_3, \ldots L_t$.*

Proof. We rename $y_{\{i_1,\ldots,i_j\}} = f(\mathbf{b_{i_1}} + \ldots + \mathbf{b_{i_j}})$. Both the Generalised Linearity Test and the set of equations $L_2, L_3, \ldots L_t$ can be rewritten as homogeneous systems of $2^t - t - 1$ linear equations in the 2^t unknowns y_J for all $J \subseteq \{1,\ldots,t\}$. Both sets of equations are in triangular form, so both solution spaces have dimension $t+1$. To prove that the two solution spaces are equal, it suffices therefore to prove one inclusion. We show that a solution of the first set of equations is also a solution for the second. The first system gives immediately the solution

$$y_{\{i_1,\ldots,i_j\}} = \sum_{\ell=1}^{j} y_{\{i_\ell\}} + ((j-1) \bmod 2) y_\emptyset. \tag{4}$$

Consider now an equation from the second set corresponding to a vector $\mathbf{c}$ of weight $d \geq 2$. Let $I = \{i_1,\ldots,i_d\}$ be the positions of the non-zero entries of $\mathbf{c}$.

The equation becomes $\sum_{J \subseteq I} y_J = 0$. Substituting the solution (4) of the first system of equations in this equation we obtain an equation that only contains the variables $y_{\{i_1\}}, \ldots, y_{\{i_d\}}$ and $y_\emptyset$. We count how many times each of these variables appears: $y_{\{i_1\}}$ will appear a number of times equal to the number of subsets J of I that contain $y_{\{i_1\}}$; this is half of the subsets, i.e. 2^{d-1} times. The variable $y_\emptyset$ appears once for each subset $J \subseteq I$ of even cardinality, i.e. $\sum_{\ell=0}^{\lfloor s/2 \rfloor} \binom{s}{2\ell} = 2^{d-1}$ times. Hence the left hand side of the equation becomes $2^{d-1}(y_{\{i_1\}} + \ldots + y_{\{i_1\}} + y_\emptyset)$. Since $d \geq 2$ and we are in $\mathbb{F}_2$, we have $2^{d-1} = 0$ so the equation is satisfied.

Finally note that the Moebius transform can again be used for efficiency. Namely putting $h(y_1, \ldots, y_t) = f(y_1 \mathbf{b_1} + \ldots + y_t \mathbf{b_t})$ and computing the Moebius transform h^M of h, the set of equations L_d are precisely the equations $h^M(y_1, \ldots, y_t) = 0$ for all $(y_1, \ldots, y_t)$ of weight d. Moreover we obtain automatically the sets of equations L_1 and L_0. If f satisfies L_1 then f is a constant, which is another test needed in the cube attack.

5 Implementation

We implemented our ideas and tested them on the stream cipher Trivium [1]. The public variables are in this case the initialisation vector and the secret variables are the key.

To test the cube attack over different vector spaces, as described in Sec. 3 we generated a large vector space V of initialisation vectors of dimension 28 giving us $\binom{28}{k}$ subspaces of each dimension $k = 0, \ldots, 28$. Linearity testing is performed as described in Section 4 using a basis of 6 linearly independent keys, meaning we evaluate at 2^6 keys for a total of $2^6 - 6 - 1$ evaluations. This will allow us to detect results that are constant, of estimated degree 1 to 5 or degree 6 or more.

We utilised a 64-bit parallelised implementation of Trivium in order to analyse 64 rounds simultaneously. The first bit of output represents round 640, with the 64th bit of output representing round 703. This allows us to compare our results with the results presented by Dinur and Shamir in their original cube paper [3] which found cubes of size 12 between 672 and 680 rounds. We also used parallelisation to implement the preprocessing phase of the cube attack. We utilised a multicore machine so that each core receives one of the 2^6 keys and runs Trivium for the given key and all the 2^{28} initialisation vectors in the large vector space. Each core then applies the Moebius transform on the data it computed. This significantly improved the efficiency of the preprocessing phase.

The vector space V is specified by a basis of 28 vectors, and it will be helpful to think of them as the rows of a 28×80 matrix A. The implementation can run the standard cube attack, by choosing 28 variable indices $i_1, \ldots, i_{28}$ and setting the entries of A so that columns $i_1, \ldots, i_{28}$ form a diagonal matrix and the remaining columns are all zero. When running the attack on an arbitrary vector space, we again choose 28 variable indices $i_1, \ldots, i_{28}$, set the columns $i_1, \ldots, i_{28}$ to form a diagonal matrix, but specify two probabilities p and q which define whether the entries in the remaining $80 - 28 = 52$ columns set to 0 or 1.

We define q as the probability that a column in matrix A will follow the probability p or be set to all zeroes. The probability p defines the probability that an entry will be set to 1 in matrix A. This means that when p is set to 0, q becomes irrelevant and when q is set to 0, p becomes irrelevant. Setting either p or q to 0 is the equivalent of running the standard cube attack.

We run the attack where $q = 1$ and $p = 0, 0.03, 0.5, 0.97, 1$, meaning that all columns in the basis which don't correspond to any of the $i_1, \ldots, i_{28}$ indices are chosen according to the probability p (for $p = 1$, all the remaining 52 columns are set to all ones in the basis). A further test is run where $q = 0.0625$ and $p = 0.5$ which generates a fairly sparse matrix A as the probability of a column of variables being chosen using probability p is low therefore most variables are set to 0. We kept the choice of $i_1, \ldots, i_{28}$ the same in all cases.

6 Discussion

Figs. 1 and 2 show how many subspaces of each dimension (as a percentage of all subspaces of that dimension) were found to return constant results where $q = 1$ and $p = 0$ or $p = 1$

Multiple lines show the results over different numbers of rounds, from 641 to 703. Predictably there are fewer constant results found at smaller dimensions as the number of rounds increases, indicating that the degree of the underlying polynomial is increasing.

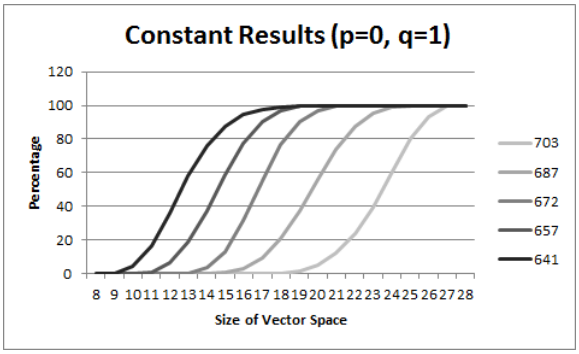

Fig. 1. Percentage of Constant Vector Spaces where $p = 0, q = 1$ for selected rounds

When comparing the two figures, it is clear that constant results are found at smaller dimensions when $p = 0$ compared to $p = 1$. There were no constant results found for 703 rounds when $p = 1$ whereas constant results were found in cubes as small as 19 when $p = 0$. This shows that changing the vector space can have a negative effect on an attacker's ability to find linear results.

This result is confirmed by Fig. 3 and Fig. 4, which are similar to Fig. 1 and Fig. 2 but show the percentages of subspaces that produce linear (rather than

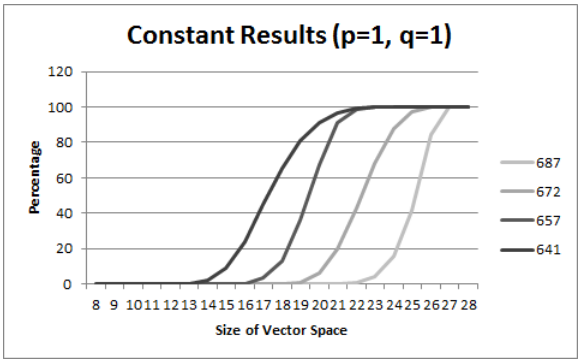

Fig. 2. Percentage of Constant Vector Spaces where $p = 1, q = 1$ for selected rounds

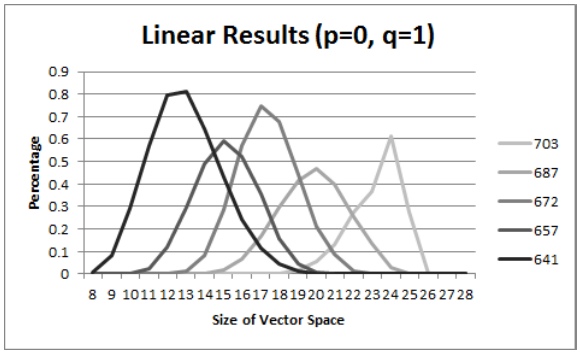

Fig. 3. Percentage of Linear Vector Spaces where $p = 0, q = 1$ for selected rounds

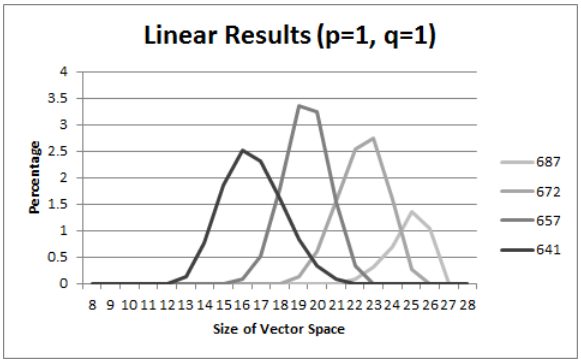

Fig. 4. Percentage of Linear Vector Spaces where $p = 1, q = 1$ for selected rounds

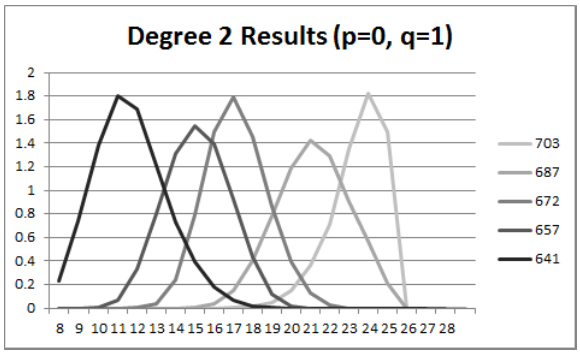

Fig. 5. Percentage of Degree 2 Vector Spaces where $p = 0, q = 1$ for selected rounds

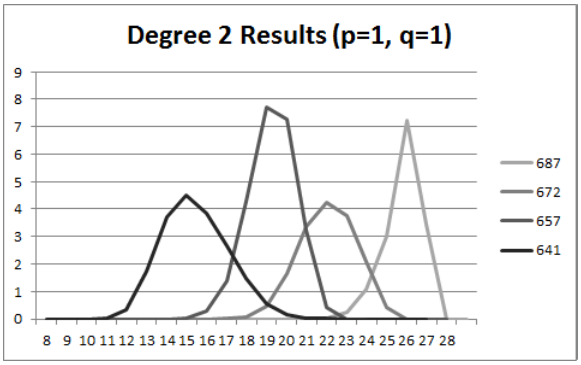

Fig. 6. Percentage of Degree 2 Vector Spaces where $p = 1, q = 1$ for selected rounds

constant) results. For each of the rounds presented, the peak dimension where linear results are most frequent is slightly larger (by 3 to 5 units) when $p = 1$ (Fig. 4) compared to where $p = 0$ (Fig. 3).

There are however some benefits to changing the vector space, as shown in Fig. 7. When analysed on the same scale, we can see that a higher percentage of subspaces produce linear results when we increase the search space by changing the vector space of the cube attack. Across all rounds, the test where $p = 1$ consistently showed a 3 to 4 times higher percentage of linear results being found as compared to where $p = 0$.

Furthermore, the percentage of subspaces found at dimension 14 where $p = 1$ in Fig. 7 is equivalent to the percentage of cubes found at cube size 12 and 13 where $p = 0$. While again reinforcing the result that the required dimension does increase, this shows that it is not always of significant detriment to the attacker.

The trend of increasing the required vector space dimension continues when we test using a small value for q. Fig. 7 shows a large increase in the percentage of linear cubes found when $p = 0.5, q = 0.0625$ compared to $p = 1, q = 1$ although these linear results are found at a significantly higher dimension.

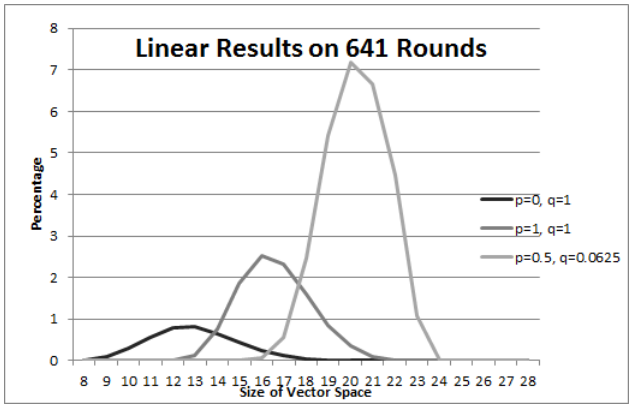

Fig. 7. Percentage of Linear Vector Spaces where $q = 1$ when $p = 0$ and $p = 1$, and $q = 0.0625$ when $p = 0.5$

When the value of p is set to a value other than 0 or 1 while $q = 1$, the attack becomes significantly less effective. We tested with both very dense set of basis vectors ($p = 0.97$) and very sparse ($p = 0.03$) as well as uniform ($p = 0.5$) and the results were nearly identical in all cases. Tab. 1 shows that in these cases there were no constant, linear or degree 2 results found, as well as an insignificant number of degree 3 results using 641 rounds. The similar behaviour between all values of p in this range when $q = 1$ could be due to the fact that although we controlled the density of the basis vectors, the rest of the vectors in the space will have quite high density irrespective of the density of the basis vectors due

to the high value of q. This is in contrast to the result presented in Fig. 7 which showed a significant number of linear results when $p = 0.5$ and q is small.

Table 1. Percentage of Vector Spaces of degree 0 to 6 when $0 < p < 1$ and $q = 1$

Degree	0	1	2	3	4	5	6
Percentage	0	0	0	$< 0.001\%$	0.78%	49.21%	50%

The fact that for Trivium the cube attack over arbitrary vector spaces V performs in general worse than the usual cube attack when q is large is, on one hand, disappointing from an attacker's point of view. However on the other hand it offers insights into the properties of Trivium. Namely it shows that cube attacks work for Trivium not only because the degree increases relatively slowly through the rounds, but also because for the degree that is achieved, the distribution of the monomials is not uniform, as would be expected for a random function. This phenomenon has been observed in other contexts, for example the density of terms of each degree is estimated by Fouque and Vannet [6]. Another manifestation of this phenomenon is that out of all the linear equations that were found by different cube attacks on Trivium reported in the literature, the vast majority contain only one or two secret key variables, instead of around 40 variables as would be expected.

An alternative way to look at this would be to create a new, enhanced "black box" for Trivium, which contains a multiplication of the vector of public IV variables by an arbitrary fixed invertible matrix. The result of the multiplication is fed into the usual Trivium black box function. It is expected that the usual cube attack on this enhanced black box would have a much reduced chance of success, as it would be, in effect, equivalent to running our generalised cube attack on the usual Trivium black box. Obviously, the matrix needs to be secret, as otherwise the attacker could undo its effect. Preliminary tests on small cubes seem to confirm this idea, but a full investigation will be the subject of future work.

Acknowledgements The authors would like to thank the referees for useful comments. One of the referees brought to our attention the recent paper [2] that we had not been aware of, and in which higher order derivatives with respect to an arbitrary vector space (as explored in Sec. 3) were used for statistical attacks on the NORX cipher.

References

1. Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, pages 171–186, 2006.
2. Sourav Das, Subhamoy Maitra, and Willi Meier. Higher order differential analysis of NORX. *IACR Cryptology ePrint Archive*, 2015:186, 2015.

3. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *EUROCRYPT*, pages 278–299, 2009.
4. Itai Dinur and Adi Shamir. Applying cube attacks to stream ciphers in realistic scenarios. *Cryptography and Communications*, 4(3-4):217–232, 2012.
5. Ming Duan and Xuejia Lai. Higher order differential cryptanalysis framework and its applications. In *International Conference on Information Science and Technology (ICIST)*, pages 291–297, 2011.
6. Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 502–517, 2013.
7. Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. *IACR Cryptology ePrint Archive*, 2015:312, 2015.
8. Antoine Joux. *Algorithmic Cryptanalysis*. Chapman & Hall/CRC, 1st edition, 2009.
9. S. Knellwolf and W. Meier. High order differential attacks on stream ciphers. *Cryptography and Communications*, 4(3-4):203–215, 2012.
10. Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard E. Blahut, Daniel J. Costello, Jr., Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography*, volume 276 of *The Springer International Series in Engineering and Computer Science*, pages 227–233. Springer Verlag, 1994.
11. M. Vielhaber. Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack. Cryptology ePrint Archive, Report 2007/413, 2007. urlhttp://eprint.iacr.org/.