The Cube Attack on Stream Cipher Trivium and Quadraticity Tests*

Piotr Mroczkowski and Janusz Szmidt

Military Communication Institute
ul. Warszawska 22A, 05-130 Zegrze, Poland
Military University of Technology
ul. Kaliskiego 2, 00-980 Warsaw, Poland

November 14, 2010

Abstract

In 2008 I. Dinur and A. Shamir presented a new type of algebraic attack on symmetric ciphers named cube attack. The method has been applied to reduced variants of stream ciphers Trivium and Grain-128, reduced variants of the block ciphers Serpent and CTC and to a reduced version of the keyed hash function MD6. Independently a very similar attack named AIDA was introduced by M. Vielhaber. In this paper we develop quadraticity tests within the cube attack and apply them to a variant of stream cipher Trivium reduced to 709 initialization rounds. Using this method we obtain the full 80-bit secret key. In this way it eliminates the stage of brute force search of some secret key bits which occured in previous cube attacks.

1 Introduction

The cube attack has been introduced by Itai Dinur and Adi Shamir [8] as a known plaintext attack on symmetric primitives. The method has been further developed in [2, 3, 9, 10]. The ciphertext bits produced by this algorithm are values of polynomials $p(v_1, \ldots, v_m, x_1, \ldots, x_n)$ depending on public variables $v_1, \ldots, v_m$ (bits of a plaintext for block ciphers or bits of an initial vector for stream ciphers) and depending on secret variables $x_1, \ldots, x_n$ (bits of a key). The attack consists of two stages. In the first *preprocessing* stage the attacker has access to public and secret variables. He sums up ciphertext bits obtained for chosen k-dimensional cubes in public variables

*This papre was presented at the 10th Central European Conference on Cryptology in Będlewo, Poland.

and fixed key variables. The attacker chooses different keys and obtains a function depending on key bits. The task of this stage is to find the cases where this function is affine or quadratic and reconstruct it. The tools to investigate the linearity or quadraticity of the resulting Boolean functions are the linearity and quadraticity tests, developed in [1, 5]. The *preprocessing* stage is the most time consuming part of the attack. Heuristic considerations and experiments are involved to find suitable cubes.

In the next *on line* stage of the attack a key is secret and the attacker only has access to public variables. He sums up over the same cubes as in the *preprocessing* stage to obtain the right hand sides of linear and quadratic equations. Having the system of all equations the attacker tries to solve it to get values of some key bits. To solve the resulting system of quadratic equations, linearization methods can be applied. The remaining, unknown bits of the key can be calculated by brute force searching. Dinur and Shamir [8] applied the cube attack to variants of stream cipher Trivium reduced to 672, 735 and 767 initialization rounds (the whole cipher runs over $4 \times 288 = 1152$ initial rounds before producing output key bits). They used linearity tests and obtained a system of linear equations for key bits. In the case of 767 initialization rounds they obtained 35 linear expressions for key bits. The remaining 80 - 35 = 45 bits of the secret key dominate the complexity of an attack being $O(2^{45})$ cipher executions which is below the complexity $O(2^{80})$ of the brute force search of all key bits. In [8] it was suggested to apply quadraticity tests within the cube attack.

We realize this idea and obtain quadratic and linear expressions involving key bits for a variant of Trivium reduced to 709 initialization rounds. Quadratic terms appear more often than linear ones and this leads to more equations for key bits. In our attack we used 22 and 23 dimensional cubes. We found 41 bits of a randomly chosen secret key using linear terms and 39 remaining bits of this key using quadratic equations. In fact, quadratic equations were solved *by hand*: substituting bits obtained from linear equations and doing some manipulations. This way the brute force searching of some bits of the secret key was completely eliminated. The complexity of the *on line* stage of our cube attack is about 2^{29} executions of reduced Trivium with $709 \div 713$ initialization rounds. The system of linear and quadratic expressions obtained during the *preprocessing* stage can be used to find any secret key.

It is important to have an effective implementation of the cipher in question to perform the cube attack. We used Paul Crowley's [7] implementation of Trivium, which was written using assembly *CorePy* tool. In this case 128 parallel strings of output key bits are prduced on the level of processor instructions. These strings are used to perform summation over chosen cubes and speed up the *preprocessing* stage of the attack.

2 Linearity and Quadraticity Tests

Let F_2^n be the n-dimensional vector space over the binary field F_2 and $f : F_2^n \longrightarrow F_2$ a Boolean function of n binary variables. The Boolean function f is affine if it satisfies the linearity test:

$$f(x \oplus x') = f(x) \oplus f(x') \oplus f(0)$$

for all $x, x' \in F_2^n$. Such a function has the following Algebraic Normal Form (ANF):

$$f(x_1, \ldots, x_n) = \bigoplus_{1 \le i \le n} a_i x_i \oplus a_0,$$

where $a_0, a_1, \ldots, a_n$ are binary coefficients.

The Boolean function f is quadratic if it satisfies the quadraticity test:

$$f(x \oplus x' \oplus x'') = f(x \oplus x') \oplus f(x \oplus x'') \oplus f(x' \oplus x'')$$

$$\oplus f(x) \oplus f(x') \oplus f(x'') \oplus f(0)$$

for all $x, x', x'' \in F_2^n$. Such a function has the following ANF form:

$$f(x_1, \ldots, x_n) = \bigoplus_{1 \le i < j \le n} a_{ij} x_i x_j \oplus \bigoplus_{1 \le i \le n} a_i x_i \oplus a_0$$

for some binary coefficients a_{ij}, a_i, a_0.

Let us note that a function satisfying the linearity test also satisfies the quadraticity test, with all a_{ij} equal to zero. The main phenomenon appearing in the cube attack is that quadratic or affine Boolean functions can be detected with high probability by executing the quadraticity or linearity tests only for a very small number of triples (x, x', x'') or pairs (x, x'). Linearity tests and their information-theoretical applications were investigated by Blum, Luby and Ribenfeld [5]. Testing of low-degree polynomials over F_2 was developed by Alon, Kaufman, Krivelevich, Litsyn and Ron in [1].

The coefficients in the Algebraic Normal Form can be calculated by summing over suitable cubes. The task of the *preprocessing* stage is to collect, for the cipher under study, as many quadratic and linear expressions in key bits as possible.

3 Specification of Trivium

3.1 Key Stream Generation

The stream cipher Trivium contains a 288-bit inner state consisting of three registers of lengths 84, 93 and 111. The key stream generation is an iterative process which extracts the values of 15 specific state bits and uses them to update 3 bits of the state and to compute 1 bit of the key stream. The

state bits are then rotated and the process repeats. The generation of the ouput bitstring (z_i) of the maximal length of up to $N = 2^{64}$ bits, can be represented as follows:

for i=1 to N

$$t_1 \leftarrow s_{66} + s_{93}$$

$$t_2 \leftarrow s_{162} + s_{177}$$

$$t_3 \leftarrow s_{243} + s_{288}$$

$$z_i \leftarrow t_1 + t_2 + t_3$$

$$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$$

$$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$$

$$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$$

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$$

$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$$

$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$$

end for

3.2 Key and Initial Value Setup

The 288-bit inner state of Trivium is initialized in the following way:

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (k_1, k_2, \ldots, k_{80}, 0, \ldots, 0)$$

$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, IV_2, \ldots, IV_{80}, 0, \ldots, 0)$$

$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, 0, \ldots, 0, 1, 1, 1)$$

for i = 1 to 1152

$$t_1 \leftarrow s_{66} + s_{93}$$

$$t_2 \leftarrow s_{162} + s_{177}$$

$$t_3 \leftarrow s_{243} + s_{288}$$

$$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$$

$$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$$

$$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$$

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$$

$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$$

$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$$

end for

4 Results of the Cube Attack

4.1 Linear Expressions

In the *preprocessing* stage we found linear expressions involving unknown key bits. These expressions were obtained by summing up over cubes, which were chosen among initial value variables. 240 linearity tests were performed for each cube from Table 1. The dimensions of the cubes (22 and 23) were fixed experimentally. In fact, the cubes were chosen randomly but also we found a method to obtain a new cube from another one with a linear expression: one must increase the indices of a cube by one (if possible) and increase the number of initialization rounds by one.

Table 1. Linear expressions.

expression	cube indices	round
x14	{6,8,9,13,22,24,28,30,32,36,39,40,43,45,47,48,60,63,67,68,73,76,79}	709
x15	{2,9,15,17,27,28,32,40,44,46,52,54,59,64,68,70,71,72,73,74,76,78,79}	709
x16+1	{1,3,9,10,13,14,16,28,34,37,42,51,52,56,59,60,62,68,69,72,74,79}	709
x17	{4,5,6,10,11,12,17,19,21,26,32,40,44,49,54,58,60,61,67,72,74,77,78}	709
x18	{5,9,15,16,20,21,32,33,35,38,41,43,46,52,56,58,60,61,62,69,77,78,79}	709
x19+1	{6,8,13,17,23,27,28,33,44,45,46,53,54,56,60,61,67,68,72,74,75,77,79}	709
x20	{1,2,7,13,15,18,19,23,29,34,35,36,38,47,49,54,57,62,64,65,66,68,74}	709
x21	{4,7,10,11,19,20,22,23,24,30,32,33,38,41,49,52,54,59,66,67,69,74,77}	709
x22	{8,16,18,22,24,26,29,31,34,36,40,41,45,46,47,48,50,59,63,69,72,76,78}	709
x23+1	{6,10,13,16,19,25,28,35,39,42,44,48,57,61,62,63,64,65,67,68,73,77,78}	709
x24	{2,4,7,15,17,18,20,23,24,27,29,35,45,47,48,51,57,59,63,65,67,74,77}	709
x25	{3,5,8,16,18,19,21,24,25,28,30,36,46,48,49,52,58,60,64,66,68,75,78}	710
x33+1	{5,10,13,14,15,22,26,27,32,35,36,45,46,50,51,56,59,60,63,64,77,78,79}	709
x35+1	{2,6,8,9,19,23,24,29,32,33,34,42,47,49,51,52,53,57,61,64,73,77}	710
x39	{0,3,6,8,11,17,28,34,38,39,41,43,46,51,52,53,54,56,64,65,70,72,78}	709
x40	{5,6,11,19,27,31,32,39,40,44,47,49,51,52,56,58,59,63,65,66,69,71,79}	709
x41+1	{7,9,10,15,17,24,25,26,33,36,43,45,52,56,59,60,61,63,68,71,74,77}	709
x42+1	{8,10,11,16,18,25,26,27,34,37,44,46,53,57,60,61,62,64,69,72,75,78}	710
x43+1	{9,11,12,17,19,26,27,28,35,38,45,47,54,58,61,62,63,65,70,73,76,79}	711
x47	{4,5,7,13,15,18,27,30,33,34,36,39,42,44,45,46,51,53,57,63,75,77,78}	709
x48+1	{6,7,15,19,27,30,35,37,44,45,46,47,49,50,56,59,60,67,70,71,72,75,79}	709
x49	{0,8,14,18,25,28,31,35,38,42,44,45,51,52,58,60,66,67,70,73,76,77}	709
x50+1	{0,2,8,11,14,15,17,21,22,28,31,32,39,41,52,53,59,60,65,67,74,77,78}	709
x51	{1,8,10,15,18,26,28,29,33,35,37,38,42,51,53,55,57,60,61,65,66,67,75}	709
x21+x52	{7,10,11,12,15,21,29,32,37,39,41,44,47,53,56,57,59,62,63,66,70,76}	710
x53+1	{1,4,8,10,11,12,14,15,19,22,24,29,31,33,39,42,50,52,55,58,60,61,65}	710
x23+x54+1	{9,12,13,14,17,23,31,34,39,41,43,46,49,55,58,59,61,64,65,68,72,78}	712
x24+x55+1	{10,13,14,15,18,24,32,35,40,42,44,47,50,56,59,60,62,65,66,69,73,79}	713
x57+1	{1,3,6,10,11,14,15,16,23,25,28,35,40,41,42,44,46,52,58,66,68,69,75}	709
x21+x49+x58+1	{8,12,14,19,26,28,30,40,41,42,43,48,50,53,59,62,63,67,71,72,74,79}	709
x59+1	{6,14,16,31,37,40,43,48,50,53,54,55,57,58,60,61,62,68,72,73,74,76}	709
x60+1	{3,4,14,16,26,29,30,38,40,43,47,54,56,58,60,64,65,67,69,70,75,76,77}	709
x61	{3,8,11,14,16,17,18,20,22,24,27,33,35,38,44,48,52,53,59,66,73,77}	711
x62	{4,9,12,15,17,18,19,21,23,25,28,34,36,39,45,49,53,54,60,67,74,78}	712
x19+x63+1	{2,5,9,17,21,27,28,30,35,37,46,48,50,53,54,60,61,63,65,69,71,73,79}	709
x67	{1,7,12,15,18,27,30,41,44,46,47,48,49,52,53,54,56,59,62,63,66,69,79}	709
x72	{6,11,16,19,26,34,36,39,41,42,47,49,52,54,57,59,66,67,71,72,76,79}	709
x73	{1,3,4,6,12,14,15,19,25,26,28,29,35,40,49,52,57,64,66,67,68,72,75}	709
x74	{2,4,5,7,13,15,16,20,26,27,29,30,36,41,50,53,58,65,67,68,69,73,76}	710
x75	{3,5,6,8,14,16,17,21,27,28,30,31,37,42,51,54,59,66,68,69,70,74,77}	711
x76	{4,6,7,9,15,17,18,22,28,29,31,32,38,43,52,55,60,67,69,70,71,75,78}	712

After collecting linear expressions and the corresponding cubes we performed the *on line* stage of cube attack. We chose a random 80-bit key (the secret in our experiment). Then we summed up the ciphertext bits obtained for the chosen cubes and the secret key, where the bits of initial vectors beyond the cube indices were equal to zero. This way we got the exact values of the terms from Table 1 and we obtained the system of linear equations

$$
\begin{array}{lll}
x14 = 1 & x15 = 0 & x16 = 0 \quad (1)\\
x17 = 0 & x18 = 0 & x19 = 1 \quad (2)\\
x20 = 1 & x21 = 0 & x22 = 1 \quad (3)\\
x23 = 0 & x24 = 0 & x25 = 1 \quad (4)\\
x33 = 1 & x35 = 0 & x39 = 1 \quad (5)\\
x40 = 1 & x41 = 0 & x42 = 0 \quad (6)\\
x43 = 1 & x47 = 1 & x48 = 1 \quad (7)\\
x49 = 1 & x50 = 1 & x51 = 0 \quad (8)\\
x21 + x52 = 0 & x53 = 0 & x23 + x54 = 1 \quad (9)\\
x24 + x55 = 0 & x57 = 1 & x21 + x49 + x58 = 1 \quad (10)\\
x59 = 1 & x60 = 1 & x61 = 0 \quad (11)\\
x62 = 0 & x19 + x63 = 1 & x67 = 0 \quad (12)\\
x72 = 1 & x73 = 0 & x74 = 0 \quad (13)\\
x75 = 0 & x76 = 1 & \quad (14)
\end{array}
$$

The equations (1) $\div$ (14) give the values of 41 bits of the key:

$$x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x33, x35,$$

$$x39, x40, x41, x42, x43, x47, x48, x49, x50, x51, x52, x53, x54, x55,$$

$$x57, x58, x59, x60, x61, x62, x63, x67, x72, x73, x74, x75, x76.$$

It remained to find 39 bits of the secret key.

4.2 Quadratic Expressions

In the *preprocessing* stage we also found the following cubes with corresponding quadratic terms (Tables 1 and 2). In fact, during this stage, 80 quadraticity tests were executed first, and if the function passed them, the linear tests were applied to check its affineness; additionaly it was checked whether the function was non-constant. The same method was applied to find some cubes with quadratic terms from similar ones by increasing indices and initialization rounds; then the corresponding quadratic polynomials have some regular structure. Having terms from Table 2, the *on line* stage of the

cube attack was done with the same secret key obtaining this way the values of these terms.

Table 2. Quadratic expressions.

expression	cube indices	round
x25x26+x24+x51	{0,4,6,13,18,19,22,27,35,37,38,43,46,48,51,53,55,57,60,61,64,66,79}	709
x31x32+x30+x57	{3,10,11,24,25,30,34,38,40,41,43,44,51,54,57,59,61,62,65,66,70,76,79}	709
x32x33+x31+x58	{1,6,8,10,13,15,19,20,26,37,39,40,43,47,53,54,57,59,64,67,68,72,75}	709
x33x34+x32+x59	{1,3,10,15,16,18,24,26,28,29,33,37,39,40,43,46,49,51,52,54,59,61,78}	709
x35x36+x34+x61	{1,8,13,16,21,26,27,28,29,31,36,39,47,48,50,56,57,59,60,61,66,72,78}	709
x37x38+x36+x63	{0,1,8,9,10,12,14,18,21,22,23,26,32,33,40,49,52,54,57,67,75,78,79}	709
x38x39+x37+x64	{2,6,14,24,27,29,30,32,38,43,45,46,49,50,53,54,58,67,68,70,74,76,77}	709
x39x40+x38+x65	{0,3,4,11,13,18,20,32,36,43,48,49,52,56,59,63,64,66,67,71,73,76,78}	709
x18x19+x17+x44	{1,9,13,14,16,18,19,31,37,39,40,42,45,46,49,52,53,58,67,68,73,76,78}	709
x19x20+x18+x45 +x58+1	{0,8,13,14,17,19,20,21,22,24,30,32,39,44,47,52,55,59,60,66,68,77,78}	709
x47x48+x19+x40 +x46+x58+x73+1	{0,6,10,15,17,18,20,27,30,31,41,49,54,55,56,58,60,61,65,66,71,74,78}	709
x64x65+x21+x63	{2,4,5,7,10,13,14,16,23,25,28,33,36,42,43,56,59,61,63,68,74,76,79}	709
x66x67+x14+x23 +x65	{1,6,11,13,15,16,20,21,26,29,30,37,43,46,47,51,55,57,60,69,76,77,79}	709
x40x41+x39+x66	{3,8,12,14,22,24,26,30,32,36,43,44,45,46,49,51,52,53,57,59,72,75,78}	709
x68x69+x16+x25 +x67+1	{1,3,7,9,11,13,17,20,23,32,38,39,40,44,46,59,62,64,68,70,73,76,78}	709
x69x70+x17+x26 +x68	{0,1,6,7,8,13,16,17,18,25,26,33,37,41,49,56,58,61,62,68,71,78,79}	709
x51x52+x50+x77	{1,10,11,14,16,22,24,25,27,41,42,50,51,52,53,54,58,66,71,74,76,78}	709
x52x53+x51+x78+1	{0,1,7,13,15,18,21,29,30,32,39,42,47,48,55,57,63,65,66,67,72,76,77}	709
x53x54+x52+x79	{5,7,9,15,17,18,27,30,32,37,38,44,47,49,50,52,57,66,68,69,74,77,78}	709
x54x55+x11+x53	{2,11,13,16,25,31,36,39,42,46,47,55,58,63,65,67,68,69,70,73,75,76,77}	710
x55x56+x12+x54+1	{3,12,14,17,26,32,37,40,43,47,48,56,59,64,66,68,69,70,71,74,76,77,78}	711
x56x57+x13+x55+1	{4,13,15,18,27,33,38,41,44,48,49,57,60,65,67,69,70,71,72,75,77,78,79}	712

We got the following quadratic equations:

$$x25x26 + x24 + x51 = 0 \tag{15}$$
$$x31x32 + x30 + x57 = 1 \tag{16}$$
$$x32x33 + x31 + x58 = 0 \tag{17}$$
$$x33x34 + x32 + x59 = 0 \tag{18}$$
$$x35x36 + x34 + x61 = 0 \tag{19}$$
$$x37x38 + x36 + x63 = 1 \tag{20}$$
$$x38x39 + x37 + x64 = 0 \tag{21}$$
$$x39x40 + x38 + x65 = 0 \tag{22}$$
$$x18x19 + x17 + x44 = 0 \tag{23}$$
$$x19x20 + x18 + x45 + x58 = 1 \tag{24}$$
$$x47x48 + x19 + x40 + x46 + x58 + x73 = 0 \tag{25}$$
$$x64x65 + x21 + x63 = 0 \tag{26}$$
$$x66x67 + x14 + x23 + x65 = 0 \tag{27}$$

$$x40x41 + x39 + x66 = 1 \quad (28)$$
$$x68x69 + x16 + x25 + x67 = 0 \quad (29)$$
$$x69x70 + x17 + x26 + x68 = 0 \quad (30)$$
$$x51x52 + x50 + x77 = 0 \quad (31)$$
$$x52x53 + x51 + x78 = 0 \quad (32)$$
$$x53x54 + x52 + x79 = 1 \quad (33)$$
$$x54x55 + x11 + x53 = 1 \quad (34)$$
$$x55x56 + x12 + x54 = 1 \quad (35)$$
$$x56x57 + x13 + x55 = 0 \quad (36)$$

We substituted the 41 known key bits to equations (15) $\div$ (36) to obtain new 22 values of key bits:

$$x11 = 1,\ x12 = 0,\ x26 = 0,\ x30 = 1,\ x31 = 1,\ x32 = 1,\ x34 = 0,\ x36 = 1$$

$$x37 = 0,\ x38 = 0,\ x44 = 0,\ x45 = 0,\ x46 = 1,\ x64 = 0,\ x65 = 1,$$

$$x66 = 0,\ x68 = 1,\ x69 = 1,\ x70 = 1,\ x77 = 1,\ x78 = 0,\ x79 = 1.$$

The equation (29) takes the form $x68x69 = 1$ and the equation (36) gives the relation $x56 = x13$. At this moment we were left with the remaining 17 unknown key bits:

$$x0, \ldots, x10,\ x13,\ x27,\ x28,\ x29,\ x56,\ x71.$$

These key bits were found by considering the following cubes and corresponding quadratic expressions.

Table 3. Quadratic expressions.

expression	cube indices	round
x8x9+x7+x34	{2,4,5,8,9,11,13,15,17,23,26,41,43,56,63,64,65,67,68,70,71,73,75}	709
x7x8+x6+x33	{1,4,8,9,16,22,25,28,29,34,42,48,49,53,55,57,59,62,70,73,75,76,79}	709
x9x10+x8+x35	{3,5,6,9,10,12,14,16,18,24,27,42,44,57,64,65,66,68,69,71,72,74,76}	710
x10x11+x9+x36	{4,6,7,10,11,13,15,17,19,25,28,43,45,58,65,66,67,69,70,72,73,75,77}	711
x11x12+x10+x37	{5,7,8,11,12,14,16,18,20,26,29,44,46,59,66,67,68,70,71,73,74,76,78}	712
x70x71+x27+x69	{0,2,5,8,13,14,16,17,27,35,38,43,50,52,53,56,57,60,65,69,75,78,79}	709
x6x7+x5+x32	{0,3,6,9,13,15,16,21,27,33,40,41,43,47,53,57,58,59,61,70,71,76,78}	709
x4x5+x3+x30	{1,4,9,13,14,16,19,24,33,37,38,43,44,45,46,47,54,56,59,68,70,73,76}	709
x3x4+x2+x29	{0,1,2,5,8,9,14,16,25,30,31,40,47,53,54,58,60,61,63,69,75,77,78}	709
x30x31+x29+x56	{8,11,15,20,23,24,27,31,32,36,42,45,50,52,53,54,56,59,60,61,62,68,75}	709
x5x6+x4+x31	{0,5,7,8,9,12,20,25,31,35,42,45,46,47,51,56,58,60,67,68,69,70,71}	709
x1x2+x0+x27+1	{7,8,11,14,15,31,36,37,40,41,44,48,49,55,58,59,62,64,66,67,69,75,78}	709
x2x3+x1+x28	{1,2,6,7,8,10,13,15,23,27,30,32,35,48,49,50,52,53,56,58,68,69,70}	709
x7x8+x16x53+x19x20 +x6+x27+x33+1	{0,2,5,19,20,22,28,29,31,36,37,39,40,50,54,55,65,68,71,72,75,77,79}	710
x71x72+x19+x28+x70	{0,2,6,9,11,17,21,36,37,42,45,46,47,48,51,53,57,58,60,71,75,77,78}	710
x29x30+x28+x55	{10,14,16,19,24,25,27,32,34,35,37,38,40,42,45,49,52,60,63,70,71,76,78}	710

The results of the *on line* stage of attack led us to the second system of quadratic equations:

$$x8x9 + x7 = 1 \qquad x7x8 + x6 = 0 \tag{37}$$
$$x9x10 + x8 = 1 \qquad x9 + x10 = 0 \tag{38}$$
$$x10 = 1 \qquad x27 + x71 = 0 \tag{39}$$
$$x6x7 + x5 = 0 \qquad x4x5 + x3 = 1 \tag{40}$$
$$x3x4 + x2 + x29 = 0 \qquad x29 + x56 = 1 \tag{41}$$
$$x5x6 + x4 = 0 \qquad x1x2 + x0 + x27 = 1 \tag{42}$$
$$x2x3 + x1 + x28 = 1 \qquad x7x8 + x6 + x27 = 0 \tag{43}$$
$$x28 + x71 = 0 \qquad x28 + x29 = 1 \tag{44}$$

The above equations after solving them (*by hand*) gave the following key bits:

$$\begin{array}{ccc} x0 = 1 & x1 = 0 & x2 = 1 \\ x3 = 1 & x4 = 0 & x5 = 0 \\ x6 = 0 & x7 = 1 & x8 = 0 \\ x9 = 1 & x10 = 1 & x27 = 0 \\ x28 = 0 & x29 = 1 & x56 = 0 \\ x71 = 0 & & \end{array}$$

Finally, we got $x13 = 0$, since $x13 = x56$. These are all key bits:

$$[x0, \ldots, x79] =$$

$$[1,0,1,1,0,0,0,1,0,1,1,1,0,0,1,0,0,0,0,1,1,0,1,0,0,1,0,0,0,1,1,$$
$$1,1,1,0,0,1,0,0,1,1,0,0,1,0,0,1,1,1,1,1,0,0,0,1,0,0,1,0,1,1,0,$$
$$0,0,0,1,0,0,1,1,1,0,1,0,0,0,1,1,0,1]$$

We would like to indicate that, in the above cube attack with quadraticity tests on a variant of Trivium reduced to 709 initialization rounds, the resulting system of quadratic equations was fairly simple. The method of solving it in two steps enabled us to find solutions just using only some elementary tricks. We have chosen this number of initialization rounds because a reduced variant of Trivium (called Bivium) having only two registers and $4 \times 177 = 708$ initialization rounds was previously investigated. In an attack on Trivium with more initialization rounds and using quadraticity tests to recover more key bits than only those from linear tests requires more time or computational resources.

References

[1] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. *Testing Low-Degree Polynomials over GF(2).* RANDOM 2003 and APPROX 2003, S. Arora, K. Jansen, J.D.P. Rolim, and A. Sahai, editors. LNCS, vol 2764, pp. 188-199. Springer 2003.

[2] J-P. Aumasson, W. Meier, I. Dinur, and A. Shamir. *Cube Testers and Key Recovery Attacks on Reduced Round MD6 and Trivium.* Fast Software Encryption 2009. Orr Dunkelman, editor. LNCS, vol 5665, pp. 1-22. Springer 2009.

[3] J-P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir. *Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128.* IACR Cryptology ePrint Archive, 2009/218.

[4] S. S. Bedi and R. Pillai. *Cube Attacks on Trivium.* IACR Cryptology ePrint Archive. 2009/15.

[5] M. Blum, M. Luby, and R. Rubinfeld. *Self-Testing/Correcting with Applications to Numerical Problems.* Journal of Computer and System Sciences, vol 47(1993), pp. 549-595.

[6] Ch. DeCannière and B. Preneel. *Trivium.* M.J.B. Robshaw and O. Bilet, editors. *New Stream Cipher Designs.* LNCS, vol 4817, pp. 244-246. Springer 2008.

[7] P. Crowley, *Trivium, SSE2, CorePy, and the "cube attack".* Published on http://www.lshift.net/blog/

[8] I. Dinur and A. Shamir. *Cubic Attacks on Tweakable Black Box Polynomials.* EUROCRYPT 2009. A. Joux, editor. LNCS, vol 5479, pp. 278-299. Springer 2009.

[9] I. Dinur and A. Shamir. *Side Channel Cube Attacks on Block Ciphers.* IACR Cryptology ePrint Archive, 2009/127.

[10] P. Mroczkowski and J. Szmidt. *The Cube Attack on Courtois Toy Cipher.* IACR Cryptology ePrint Archive, 2009/497. To appear in Proceedings of WEWoRC 2009. LNCS. Springer.

[11] P. Mroczkowski and J. Szmidt. *The Cube Attack on Stream Cipher Trivium and Quadraticity Tests.* Rump Session. CRYPTO 2010.

[12] M. Vielhaber, *Breaking ONE.TRIVIUM by AIDA an Algebraic IV Differential Attack.* IACR Cryptology ePrint Archive, 2007/413.

[13] M. Vielhaber. *AIDA Breaks (BIVIUM A and B) in 1 Minute Dual Core CPU Time.* IACR Cryptology ePrint Archive, 2009/402.