The Cube Attack on Courtois Toy Cipher

Piotr Mroczkowski and Janusz Szmidt

Military Communication Institute
05-130 Zegrze, Poland
Military University of Technology,
Faculty of Cybernetics,
Institute of Mathematics and Cryptology
ul. Kaliskiego 2, 00-908 Warsaw, Poland

Abstract. The cube attack has been introduced by Itai Dinur and Adi Shamir [8] as a known plaintext attack on symmetric primitives. The attack has been applied to reduced variants of the stream ciphers Trivium [3, 8] and Grain-128 [2], reduced to three rounds variant of the block cipher Serpent [9] and reduced version of the hash function MD6 [3]. In the special case the attack has appeared in the M. Vielhaber ePrint articles [13, 14], where it has been named AIDA (*Algebraic Initial Value Differential Attack*) and applied to the modified versions of Trivium.
In this paper, we present the experimental results of application the cube attack to four rounds of the Courtois Toy Cipher (CTC) with the full recovery of 120-bit key. After that we extend the attack to five rounds by applying the meet-in-the-middle principle.

Key words: Cube attack, symmetric primitives, Boolean polynomials, CTC, the *meet-in-the-middle* method.

1 Introduction

In recent years there have been developed the methods of algebraic cryptanalysis of symmetric primitives, i.e. block and stream ciphers, hash and MAC functions. The idea is to represent the investigated algorithm as a system of multivariate polynomials involving as their variables the plaintext and ciphertext bits, the initial value bits or/and the key bits. Then to break the cryptosystem (to find the secret key) one must solve such a complicated system of algebraic equations. For real used ciphers such systems are to large to be solved by computational methods. There are considered the reduced and simplified versions of symmetric algorithms to investigate the applicability of algebraic cryptanalysis.

In this paper, we present recently introduced by Dinur and Shamir [8] the cube attack, as an example of algebraic technique in cryptanalysis. In fact, it is not pure algebraic method since it involves also some probabilistic tools. There were linear tests applied to approximate the complicated Boolean functions of several variables. When this approximation is possible with probability near one, then the cube attack is applicable.

The cube attack has been applied to the reduced variants of the stream ciphers Trivium [3, 8] and Grain-128 [2], to the reduced to three rounds variant of the block cipher Serpent [9] and to the reduced version of the hash function MD6 [3]. In the special case, the attack has appeared in the M. Vielhaber ePrint articles [13, 14], where it was named AIDA (*Algebraic Initial Value Differential Attack*) and applied to the modified versions of Trivium. In the second article [14], Vielhaber proposed also to use the Gaussian elimination and the Wavefront Model to extract the linear terms.

The CTC has been designed by Nicolais Courtois [5] to apply and test the methods of algebraic cryptanalysis. The security of this cipher and its modification [6] has been analysed by N. Courtois [5, 6], M. Albrecht [1], O. Dunkelman and N. Keller [10, 11]. The applications of the methods of algebraic, differential and linear cryptanalysis provided the attacks with complexity below that of the exhaustive key search; some of these attacks are theoretical ones. Although CTC and CTC2 are not practically used ciphers, cryptanalysts have payed the attention to them.

Our contribution is an application of the cube attack to the version of Courtois Toy Cipher with four rounds and 120-bit key and the extension of the original cube attack by combining it with the *meet-in-the-middle* method, where we add one round more to analyse. In this extended attack we assume that during the *preprocessing* phase an attacker can encrypt the chosen plaintexts and investigate the sums of the ciphertexts bits as a function of the key bits. The main task of this phase is to find linear (or affine) functions using the linear tests [4]. During the *preprocessing* phase the attacker collects many linear expressions in key bits and chooses linaerly independent ones; we have used here the MAGMA [15] package to do the needed calculations.

The *on-line* phase is a chosen plaintext attack, where one round is added to the cipher. The key is secret now and an attacker encrypts the plaintexts (obtained from the cubes found in the previous phase) and collects the ciphertexts after the added round. Now the attacker has no access to partial ciphertexts after the previous round. The *meet-in-the-middle* phase compares the right hand sides of the linear expressions obtained during the *preprocessing* phase (but without explicit calculation of them, as it was done in the original Dinur and Shamir cube attack) with the sums of bits obtained after inverting the last round of the cipher. The task is realized using the explicit formulae for output bits of the inversion of the last round.

The *meet-in-the-middle* attack has been practically realized for the five round CTC with 120-bit block and key size. In the experiments, randomly chosen keys have been retrieved during the *on-line* phase of the attack. It is worthy to mention that using the *BooleanPolynomial* class from the SAGE [17] package and the code written in Python [16] it is possible to do the fast calculations with quadratic Boolean functions depending on 240 binary variables. The similar calculations one can do for Boolean functions depending on more variables and having higher algebraic degree. It opens further possibilities to apply these tools in other context of algebraic cryptanalysis.

2 The Cube Attack

We shall not distinguish at the moment between secret and public variables. Let p be a polynomial of n variables $x_1, \ldots, x_n$ over the field $GF(2)$. For a fixed subset of indices $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ let us take a monomial $t_I = x_{i_1} \ldots x_{i_k}$. Then we have a decomposition

$$p(x_1, \ldots, x_n) = t_I \cdot p_{S(I)} + q(x_1, \ldots, x_n),$$

where the polynomial $p_{S(I)}$ does not depend on the variables $x_{i_1}, \ldots, x_{i_k}$.

Definition 1. The maxterm of the polynomial p we call the monomial t_I, such that $deg(p_{S(I)}) = 1$, it means that the polynomial $p_{S(I)}$ corresponding to the subset of indices I is a linear one, which is not a constant.

The set of indices I defines the k-dimensional Boolean cube C_I, where on the place of each of the indices we put 0 or 1. A given vector $v \in C_I$ defines the derived polynomoal p_v depending on $n - k$ variables, where in the basic polynomoal p we put the values corresponding to the vector v. Summing over all vectors in the cube C_I we obtain the polynomial

$$p_I = \sum_{v \in C_I} p_v.$$

Theorem 1. Let p be a polynomial over the field $GF(2)$ and $I \subset \{1, \ldots, n\}$ the subset of indices. Then we have:

$$p_I = p_{S(I)},$$

where the polynomials are equal modulo 2.

Let us consider a cryptosystem described by the polynomial

$$p(v_1, \ldots, v_m, x_1, \ldots, x_n)$$

depending on m public variables $v_1, \ldots, v_m$ (the initial value or plaintext) and on n secret variables $x_1, \ldots, x_n$ (the key). The value of the polynomial represents the ciphertext bit. In general, the polynomial p is not explicitelly known; it can be a *black box.* We will consider the known plaintext attack, where at the *preprocessing* phase the attacker has also an access to secret variables.

The attack has two phases. In the *preprocessing* one the attacker can change the values of public and secret variables. The task is to obtain a system of linear equations on secret variables. In the second *on-line* phase of the attack the key is secret and the attacker can change the values of public variables. He adds the output bits, where the inputs run over some multi-dimennsional cube. The task is to obtain the right hand sides of linear equations. The system of linear equations can be solved giving some bits of the key.

The first task is to fix the dimension of the cube and the public variables over which we will sum; they are called *the tweakable* variables, and the other public variables are equal to zero. If we know the degree d of the basic polynomial, we fix the cube dimension to $d-1$. We do the summation over a fixed cube for several values of secret variables and collect the obtained values. We do the linear tests for the obtained function of secret variables and store it when it is linear. The linear test for a function $f(x)$ depending on a collection x of binary variables consists of checking the condition:

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

for some randomly chosen arguments x, x'. If the function f passes the linear test for a few hundreds of pairs x, x' and it is not a constant function (equal to zero or to one), then we can put the hypothesis that it is a linear (or affine) function. The theoretical explanation for these tests has been elaborated in the paper [4].

The next task is to calculate the explicit values of coefficients of the obtained linear function of secret variables. The free term of the linear function we obtain putting its all arguments equal zero. The coefficient of the variable x_i is equal 1 if and only if the change of this variable implies the change of value of the function. The coefficient of the variable x_i is equal 0 if and only if the change of this variable does not imply the change of value of the function. The task of the *preprocessing* phase of attack is to collect possible many independent linear terms - they constitute the system of linear equations on secret variables. This system of linear equations will be used in the *on-line* phase of attack. The *preprocessing* procedure is done only once in cryptanalysis of the algorithm.

In the *on-line* phase an attacker has access only to public variables (the plaintexts for block ciphers, the initial values for stream ciphers), which he can change and calculates the corresponding bits of the ciphertext under the unknown value of secret variables. The task of this phase of attack is to find some bits of secret key with complexity, which is lower than that of the exhuastive search in the brute force attack. The attacker uses the derived system of linear equations for secret variables (the unknown bits of the key), where the right hand sides of these equations are the values of sums of bits of ciphertexts obtained after summation over the same cubes as in the *preprocessing* phase, but now the key is not known.

The cube attack is applicable to symmetric ciphers for which the polynomials describing the system have relatively low degree. Then one can eventually find some bits of unknown key; the remaining bits of the key may be found by brute force search. After successful *preprocessing*, the *on-line* phase of the attack can be done many times for different unknown keys. In general, the cube attack is applicable to cryptosystems without knowing their inner structure. The attacker must have the possibility to realize the preprocessing phase and in the *on-line* one has an access to the implementation of the algorithm (to perform the summation over cubes under unknown key).

3 The Courtois Toy Cipher

3.1 The Specification

The CTC has been designed by Nicolais Courtois [5, 6] to apply and test methods of algebraic cryptanalysis. It is a SPN network with scalable number of rounds, the block and key size. Each round performs the same operations on the input data, except that a different round key is added each time. The number of rounds is denoted by N_r. The output of round $i-1$ is the input to round i. Each round consists of parallel application of B S-boxes (S), the application of the linear diffusion layer (D), and a final key addition of the round key (K_i). The round key K_0 is added to the plaintext block before the first round.

The plaintext bits $p_0 \dots p_{Bs-1}$ are identified with $Z_{0,0} \dots Z_{0,Bs-1}$ and the ciphertext bits $c_0 \dots c_{Bs-1}$ are identified with $X_{N_r+1,0} \dots X_{N_r+1,Bs-1}$ to have an uniform notation ($s = 3$ is the size of the S-box). The S-box was chosen as the permutation

$$[7, 6, 0, 4, 2, 5, 1, 3].$$

It has $2^3 = 8$ inputs and 8 outputs. The output bits are quadratic Boolean functions of the input bits which can be expressed as

$$y_0 = x_0x_1 + x_0 + x_1 + x_2 + 1,$$
$$y_1 = x_0x_2 + x_1 + 1,$$
$$y_2 = x_0x_1 + x_0x_2 + x_1x_2 + x_1 + x_2 + 1,$$

and for the inverse S-box:

$$x_0 = y_0y_1 + y_2,$$
$$x_1 = y_0y_1 + y_0y_2 + y_1 + 1.$$
$$x_2 = y_0y_1 + y_1y_2 + y_0 + y_1.$$

The explicit form of these functions will be used when we apply *the meet-in-the-middle* method.

The diffusion layer (D) is defined as

$$Z_{i,257 mod Bs} = Y_{i,0},$$

for $i = 1, \dots, N_r$, and

$$Z_{i,(1987j+257) mod Bs} = Y_{i,j} + Y_{i,(j+137) mod Bs}$$

for $j \neq 0$ and all i, where $Y_{i,j}$ represent input bits and $Z_{i,j}$ represent output bits.

The key schedule is a simple permutation of bits:

$$K_{i,j} = K_{0,(i+j) mod Bs}$$

for all i and j, where K_0 is the main key. Key addition is performed bit-wise:

$$X_{i+1,j} = Z_{i,j} + K_{i,j}$$

for all $i = 1, \dots, N_r$ and $j = 0, 1, \dots, Bs-1$, where $Z_{i,j}$ represent output bits of the previous diffusion layer, $X_{i+1,j}$ the input bits of the next round, and $K_{i,j}$ the bits of the current round key.

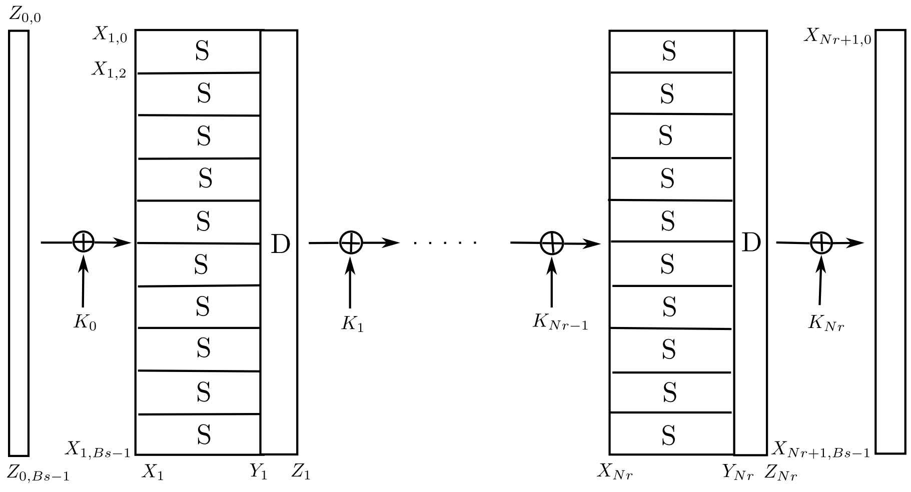

Figure 1: CTC overwiev for B = 10.

3.2 The Cube Attack on CTC

We have applied the cube attack to the version of Courtois Block Cipher with four rounds and 120-bit block and key size. There have been found the maxterms corresponding to four dimensional cubes and we have collected 120 related to them linearly independent linear polynomials which are given in Table 1 in Appendix. The table contains the indices of the cubes, the corresponding linear expressions and the ouput bits after four rounds of the CTC for which there were found these expressions after summation over the cubes. There were performed 1000 linear tests for each expression to state its linearity. In fact these 120 linearly independent functions were chosen among 610 linear ones generated during the *preprocessing* phase. It is difficult to estimate explicitelly the complexity of this phase. The first task was to find for which dimension of cubes there appear the maxterms with corresponding linear polynomials. This phenomenon has appeared for four dimensional cubes after four rounds of 120-bit CTC. Each round of CTC can be described by quadratic Boolean functions, hence the output bits of four round CTC are described by Boolean polynomials of degree $2^4 = 16$ regarded as a function of the plaintext bits and the key bits. According to general principles there should exist linear expressions corresponding to 15-dimensional cubes, but we have not found any up to now; probably the probability to detect any such is very low. The existence of linear terms for four dimensional cubes in this case may be related to some diffussion effects.

The complexity of the *on-line* phase is equal to $120 \times 2^4 \approx 2^{11}$ encryptions of the four round CTC. In this phase the attacker has the derived system of linear equations and calculates (by summing over the cubes the ciphertext bits) the right hand sides of these equations. The solution of these system gives the key. We have performed the experiment for several randomly chosen keys and obtained their exact values. The all calculations involving linear algebra, e.g.

solving the systems of linear equations over binary field, has been done with Magma [15] computational system.

We present below the results obtained for the variant of CTC with six rounds and 15-bit block and key size. The maxterms with corresponding linear polynomials have appeared after summation over 14-dimensional cubes. Here is the system of 15 linearly independend equations with the right hand sides representing the sums of ciphertext bits after six rounds of this variant of CTC. The eight linear equations obtained for the cube {0,1,2,3,4,5,6,7,8,9,10,11,12,13}:

$$\begin{aligned}
x0 + x1 + x2 + x3 + x4 &= c0 \\
x2 + x3 + x9 &= c1 \\
x2 + x3 + x10 + x11 + x12 + x13 &= c2 \\
1 + x2 + x3 + x6 + x7 + x8 + x9 + x11 + x13 &= c3 \\
x0 + x1 + x2 + x4 + x7 + x8 + x10 + x11 + x12 &= c4 \\
x0 + x3 + x6 + x9 + x11 &= c6 \\
x5 + x11 &= c7 \\
1 + x0 + x3 + x6 + x7 + x8 + x10 + x12 + x13 &= c10
\end{aligned}$$

The seven linear equations obtained for the cube {0,1,2,3,4,5,6,7,8,9,10,11,12,14}:

$$\begin{aligned}
1 + x3 + x4 + x7 + x8 + x9 + x11 + x13 &= c0 \\
x1 + x3 + x5 + x10 + x11 + x12 + x13 + x14 &= c1 \\
1 + x0 + x2 + x3 + x5 + x6 + x8 + x9 + x14 &= c2 \\
x1 + x2 + x3 + x5 + x6 + x8 + x9 + x10 + x11 + x12 + x13 + x14 &= c3 \\
1 + x0 + x1 + x2 + x3 + x4 + x6 + x9 + x10 + x13 + x14 &= c4 \\
x0 + x2 + x3 + x7 + x10 + x13 + x14 &= c6 \\
1 + x0 + x1 + x2 + x4 + x5 + x6 + x7 + x9 + x14 &= c7
\end{aligned}$$

This example will be continued in the next point, where we will add one round more to extend the attack.

4 The Cube Attack and the *Meet-in-the-Middle* Method

We assume now that in the *preprocessing* phase the attacker has an access to keys and encryption data after four rounds of CTC (the variant with 120-bit block and key size) and then he collects the linear expressions in key bits which are given in Table 1 of Appendix. Now in the *on-line* phase we assume that the attacker can encrypt the plaintexts corresponding to the selected previously cubes and can collect the ciphertexts only after five rounds of the CTC. The task of this phase is to obtain the linear equations for unknown bits of the key.

We invert the last round of the cipher and obtain the exact formulae expressing the output bits as quadratic Boolean functions of the ciphertext bits (after five rounds) and the bits of the key. Summing these output bits over ciphertexts belonging to the given cube we obtain linear expression in unknown bits of the key: there is an even number of ciphertexts corresponding to the cube and the quadratic terms in key bits are canceled. Now we equal them to linear expressions given in Table 1 (obtained in *preprocessing* phase after four rounds) having this way the system of linear equations for the bits of the key. In fact, we compare the sums of the bits after four rounds with the sums of the bits obtained after decryption of the fifth round, but we do not collect the exact values of bits in the meeting point (these bits are equal and hence their sums are equal too). The exact formulae for the inverted last round are not presented here since they are to complicated. The main reason is that the inversion of the diffusion layer has not a simple form. We have generated them using the suitable program and they are included in the execution files (see below for the simplest case). The Figure 2 depicts the *meet-in-the-middle* attack.

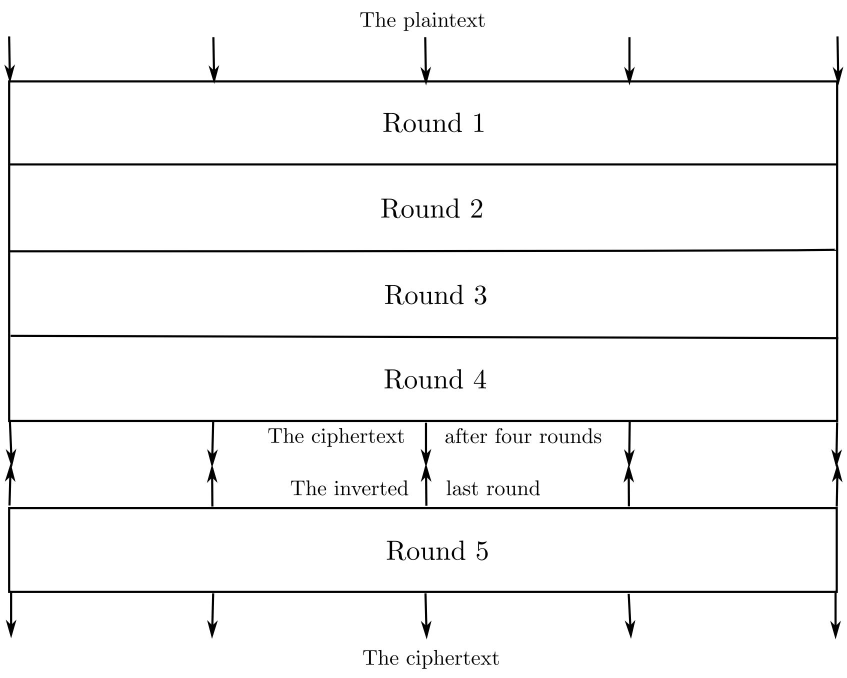

Figure 2: The *meet-in-the middle* after four rounds.

As an example, the formulae for the inversion of diffusion layer in the case of CTC with five S-boxes (i.e., 15-bit plaintext and key size) are given below. Here $z0, \ldots, z14$ are the inputs bits and $y0, \ldots, y14$ the output bits of the diffusion layer.

$$\begin{aligned}
y0 &= z2\\
y1 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9\\
y2 &= z0 + z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12\\
&\quad + z13 + z14\\
y3 &= z2 + z3 + z4 + z5 + z6 + z7 + z8\\
y4 &= z0 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12 + z13\\
&\quad + z14\\
y5 &= z2 + z3 + z4 + z5 + z6 + z7\\
y6 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12 + z13 + z14\\
y7 &= z2 + z3 + z4 + z5 + z6\\
y8 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12 + z13\\
y9 &= z2 + z3 + z4 + z5\\
y10 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11 + z12\\
y11 &= z2 + z3 + z4\\
y12 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10 + z11\\
y13 &= z2 + z3\\
y14 &= z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10
\end{aligned}$$

We have performed the described above *meet-in-the-middle* phase of the attack with the inverted seventh round of the 15-bit CTC and here there are the obtained linear equations.

$$\begin{aligned}
x0 + x1 + x2 + x3 + x4 + x9 &= 0\\
x2 + x3 + x9 &= 1\\
x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 &= 0\\
x0 + x2 + x3 + x6 + x7 + x8 + x10 + x12 + x14 &= 0\\
x1 + x2 + x4 + x7 + x8 + x9 + x13 + x14 &= 1\\
x1 + x2 + x4 + x5 + x9 + x11 + x14 &= 0\\
x5 + x6 + x9 + x10 + x12 + x13 &= 1\\
x1 + x2 + x4 + x6 + x7 + x8 + x10 + x14 &= 0\\
x0 + x1 + x3 + x4 + x7 + x8 + x9 + x10 + x12 + x14 &= 0\\
x1 + x2 + x4 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 &= 0\\
x0 + x4 + x7 + x14 &= 1\\
x4 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + 1 &= 0\\
x0 + x5 + x7 + x11 + x12 &= 0\\
x0 + x2 + x3 + x7 + x10 + x13 + x14 &= 0\\
x0 + x1 + x2 + x4 + x5 + x6 + x7 + x9 + x14 &= 0
\end{aligned}$$

The solution of these equations is the key:

$$(x0, x1, \ldots, x14) = (1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1),$$

randomly chosen at the beginning of the experiment.

We have performed the same experiment for the five round CTC with 120-bit block and key size. We have exploited the linear expressions (Appendix, Table 1) obtained after four rounds during the *preprocessing* phase. In fact, each of them corresponds here to different 4-dimensional cube. In the *on-line* phase which is now the *meet-in-the-middle* we have collected the ciphertexts after five rounds of 120-bit CTC obtained after encryptions with the key (which is assumed to be unknown in the experiment) corresponding to the same cubes. The system of 120 linear equations is to large to write it down here (see the extended version [12] of the article). All manipulations with the Boolean polynomials depending on $120 + 120 = 240$ binary variables have been done in SAGE package [17] and the related program has been written using the Python language. It appears that the rank of this system of linear equations is equal to 119, hence one bit of the key must be guessed. The performed experiments have confirmed the correctness of the method for several randomly chosen 120-bit keys. The complexity of the *on-line* phase here is the 2^{11} encryptions of five round CTC and storage of the 2^{11} 120-bit ciphertexts. The complexity of the linear part of caculations is negligible.

In general, this *meet-in-the-middle* extenion of the cube attack would work in the situation when we are able to realize successfully the *preprocessing* phase of the cube attack for n rounds of block cipher and the invertion of $n + 1$-st round leads to a system of equations which could be solved.

References

1. M. Albrecht. *Algebraic Attacks on the Courtois Toy Cipher.* Master Thesis. Department of Computer Science. University of Bremen. 2006.
2. J-P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir. *Efficient FPGA Implementations of High-Dimensional Cube Teters on the Stream Cipher Grain-128.* IACR Cryptology ePrint Archive, 2009/218.
3. J-P. Aumasson, I. Dinur, W. Meier, and A. Shamir. *Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium.* In: Fast Software Encryption 2009. O. Dunkelman, editor. LNCS. Springer, to appear.
4. 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.
5. N. Courtois. *How Fast can be Algebraic Attacks on Block Ciphers ?.* IACR Cryptology ePrint Archive, 2006/168.
6. N. Courtois. *CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited.* IACR Cryptology ePrint Archive, 2007/152.
7. C. De Canniere and B. Preneel. Trivium. Ecrypt Stream Cipher Project. Website: http://www.ecrypt.eu.org/stream/
8. I. Dinur and A. Shamir. *Cube Attacks on Tweakable Black Box Polynomials.* In: EUROCRYPT 2009. A. Joux, editor. LNCS, vol 5479, pp. 278-299. Springer.

9. I. Dinur and A. Shamir. *Side Channel Cube Attacks on Block Ciphers.* IACR Cryptology ePrint Archive, 2009/127.
10. O. Dunkelman and N. Keller. *Linear Cryptanalysis of CTC.* IACR Cryptology ePrint Archive, 2006/250.
11. O. Dunkelman and N. Keller. *Cryptanalysis of CTC2.* In: CT-RSA 2009. M. Fischlin, editor. LNCS, vol 5473, pp. 226-239. Springer.
12. P. Mroczkowski and J. Szmidt. *The Cube Attack on Courtois Toy Cipher.* IACR Cryptology ePrint Archive, 2009/.
13. M. Vielhaber, *Breaking One.Fivium by AIDA an Algebraic IV Differential Attack*, IACR Cryptology ePrint Archive, 2007/413.
14. M. Vielhaber. *AIDA Braeks BIVIUM (A and B) in 1 Minute Dual Core CPU Time.* IACR Cryptology ePrint Archive, 2009]402.
15. Magma V2.14-17. Computational Algebra Group. School of Mathematics and Statistics. University of Sydney. Website: http://magma.maths.usyd.edu.au
16. Python Programming Language. Website: http://www.python.org
17. SAGE Mathematical Software. Version 2.6. Website: http://www.sagemath.org

Appendix

Table 1: The linear expressions for CTC with 4 rounds and 120-bit key.

cube indices	expression	out. bit	cube indices	expression	out. bit
{78,84,86,113}	x80	c69	{26,28,63,118}	x64+x65	c69
{71,85,107,116}	1+x69+x70	c21	{5,46,86,103}	1+x84+x85	c37
{32,63,64,77}	1+x30+x31	c17	{22,84,110,113}	x86	c76
{10,11,12,50}	x13+x14	c17	{49,62,68,113}	1+x48+x50	c87
{25,74,100,101}	1+x24	c102	{32,65,73,89}	1+x87+x88	c93
{49,76,85,86}	1+x75	c102	{4,20,32,84}	x85+x86	c39
{36,37,110,115}	x108	c99	{18,20,62,73}	1+x60+x61	c63
{20,23,112,114}	x116	c106	{1,8,64,77}	1+x63+x65	c53
{0,13,61,92}	x2	c117	{37,38,91,115}	1+x90+x92	c99
{41,56,78,110}	x79+x80	c51	{37,67,97,109}	1+x66	c93
{14,20,46,51}	x53	c96	{18,20,47,114}	x115+x116	c3
{38,53,79,80}	x36	c81	{40,45,98,119}	1+x46	c31
{7,11,47,52}	1+x6+x8	c43	{13,59,92,101}	x99	c113
{25,46,83,104}	1+x45+x47	c16	{3,12,14,97}	1+x4	c30
{0,2,94,98}	1+x93+x95	c93	{10,58,70,101}	1+x99+x100	c54
{11,23,79,92}	1+x78	c70	{5,26,59,97}	x57	c48
{11,35,43,118}	1+x33+x34	c1	{34,75,87,89}	x76+x77	c17
{0,52,98,112}	x1+x2	c32	{5,56,58,104}	1+x57+x59	c35
{12,14,56,89}	1+x54+x55	c117	{7,35,53,70}	1+x51+x52	c1
{61,62,89,102}	x104	c48	{41,82,83,94}	x39	c84
{24,28,53,107}	x26	c19	{21,41,49,77}	x22+x23	c114
{17,49,81,101}	x82+x83	c54	{28,74,88,98}	1+x96+x97	c8
{3,97,98,101}	x4+x5	c35	{38,69,100,101}	x71	c114
{62,97,113,117}	1+x118	c94	{22,27,82,107}	x29	c19
{8,75,83,115}	1+x76	c39	{18,26,58,71}	1+x19	c102

cube indices	expression	out. bit	cube indices	expression	out. bit
{26,80,95,102}	x103+x104	c117	{67,88,95,106}	1+x87+x89	c5
{30,72,73,85}	x32	c75	{29,34,35,112}	1+x27+x28	c36
{4,23,50,92}	1+x3+x5+x21	c9	{17,67,68,103}	1+x102+x104	c108
{39,43,68,71}	x41	c34	{17,59,100,113}	1+x15+x16	c30
{76,105,118,119}	x106+x107	c0	{48,77,106,107}	x50	c18
{76,77,104,108}	1+x109	c35	{17,46,86,105}	x107	c42
{33,49,71,80}	x34+x35	c46	{12,20,89,101}	x14	c20
{11,40,41,42}	x44	c72	{13,20,41,70}	1+x69	c4
{13,29,73,107}	x27	c48	{46,79,82,104}	1+x45	c79
{16,50,76,90}	x92	c111	{8,56,91,107}	1+x6+x7	c27
{11,43,80,107}	1+x105+x106	c102	{1,26,85,93}	1+x0+x2+x94	c119
{34,59,91,111}	1+x112	c25	{16,73,104,109}	1+x108+x110	c9
{51,65,97,104}	1+x52	c104	{77,92,116,118}	1+x114+x115	c33
{21,38,79,92}	1+x22	c33	{70,73,90,101}	1+x91	c67
{14,36,40,119}	x37+x38	c9	{7,54,55,74}	x72	c117
{77,113,117,119}	x111	c102	{35,41,66,73}	x68	c117
{17,19,61,86}	1+x18	c52	{7,23,35,44}	x21	c19
{11,38,114,116}	1+x36+x37	c39	{10,11,83,88}	x81	c72
{15,16,43,89}	1+x42+x44	c30	{11,14,36,38}	1+x9+x10	c9
{11,62,77,117}	x119	c72	{7,28,94,115}	1+x93+x114	c18
{6,13,72,74}	x8	c117	{20,51,53,82}	1+x81+x83	c96
{8,26,48,50}	1+x24+x25	c33	{57,65,91,97}	1+x58	c21
{5,9,74,80}	1+x10	c36	{1,32,64,98}	1+x0+x2	c39
{17,83,95,115}	x15	c36	{11,31,73,80}	1+x30	c64
{4,8,65,77}	x63	c18	{24,26,39,85}	1+x40	c69
{35,37,116,119}	1+x117+x118	c57	{49,58,86,111}	x112+x113	c49
{47,50,73,116}	1+x72+x74	c82	{24,25,96,103}	x98	c87
{40,41,66,110}	1+x67	c27	{50,70,71,87}	x89	c17
{10,22,28,101}	1+x9+x11	c18	{26,52,53,96}	1+x97	c54
{34,56,76,88}	1+x33	c12	{42,47,88,113}	1+x43	c79
{23,56,58,80}	x54	c55	{37,38,71,72}	1+x73	c99
{17,19,49,97}	1+x18+x20	c93	{32,50,55,56}	1+x48+x49	c57
{50,60,82,103}	1+x61	c3	{20,61,68,99}	x101	c52
{15,41,76,82}	x17	c12	{8,13,59,97}	1+x12+x14	c33
{25,35,61,86}	1+x60+x62	c59	{31,32,54,59}	x55+x56	c18

As an example, there are the first ten linear equations (of the 120 ones) for the key bits obtained after applying the *meet-in-the-middle* method.

$$
\begin{aligned}
x22 + x24 + x26 + x28 + x30 + x32 + x80 + x83 + x85 + x87 + x89 + x91 &= 0 \\
x64 + x65 &= 0 \\
x22 + x24 + x26 + x28 + x30 + x32 + x34 + x36 + x38 + x40 + x42 + x44 + x46 +& \\
x48 + x50 + x52 + x54 + x56 + x69 + x70 + x83 + x85 + x87 + x89 + x91 + x93 +& \\
x95 + x97 + x99 + x101 + x103 + x105 + x107 + x109 + x111 + x113 + x115 &= 0 \\
x84 + x85 &= 1 \\
x0 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x9 + x11 + x13 + x15 + x16 + x17 + x18 +& \\
x19 + x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x32 + x33 +& \\
x34 + x35 + x36 + x37 + x38 + x39 + x40 + x41 + x42 + x43 + x44 + x45 + x46 +& \\
x47 + x48 + x49 + x50 + x51 + x52 + x53 + x54 + x55 + x56 + x57 + x58 + x59 +& \\
x60 + x61 + x62 + x63 + x64 + x65 + x66 + x67 + x68 + x70 + x72 + x74 + x75 +& \\
x76 + x77 + x78 + x79 + x80 + x81 + x82 + x83 + x84 + x85 + x86 + x87 + x88 +& \\
x89 + x90 + x91 + x92 + x93 + x94 + x95 + x96 + x97 + x98 + x99 + x100 + x101 +& \\
x102 + x103 + x104 + x105 + x106 + x107 + x108 + x109 + x110 + x111 + x112 +& \\
x113 + x114 + x115 + x116 + x117 + x118 + x119 &= 1 \\
x86 &= 0 \\
x13 + x14 &= 0 \\
x22 + x24 + x26 + x28 + x30 + x48 + x50 + x83 + x85 + x87 + x89 + x91 &= 1 \\
x0 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 +& \\
x14 + x15 + x16 + x17 + x19 + x21 + x23 + x25 + x26 + x27 + x28 + x29 +& \\
x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37 + x38 + x39 + x40 + x41 +& \\
x42 + x43 + x44 + x45 + x46 + x47 + x48 + x49 + x50 + x51 + x52 + x53 +& \\
x54 + x55 + x56 + x57 + x58 + x59 + x60 + x61 + x62 + x63 + x64 + x65 +& \\
x66 + x67 + x68 + x69 + x70 + x71 + x72 + x73 + x74 + x75 + x76 + x78 +& \\
x80 + x82 + x84 + x85 + x86 + x87 + x88 + x89 + x90 + x91 + x92 + x93 +& \\
x94 + x95 + x96 + x97 + x98 + x99 + x100 + x101 + x102 + x103 + x104 +& \\
x105 + x106 + x107 + x108 + x109 + x110 + x111 + x112 + x113 + x114 +& \\
x115 + x116 + x117 + x118 + x119 &= 0 \\
x87 + x88 &= 1
\end{aligned}
$$

The rank of the whole system is equal 119. After applying the Gaussian elimination we obtain the system of 119 equations:

$$
\begin{array}{rrrr}
x0 = 1, & x1 + x118 = 1, & x2 = 1, & x3 = 0, \\
x4 = 1, & x5 = 0, & x6 = 0, & x7 = 1, \\
x8 + x118 = 1, & x9 = 0, & x10 = 1, & x11 = 0,
\end{array}
$$

$$\begin{array}{rrrr}
x12 = 1, & x13 = 0, & x14 = 0, & x15 = 1, \\
x16 = 1, & x17 = 0, & x18 = 1, & x19 = 1, \\
x20 + x118 = 0, & x21 = 1, & x22 = 1, & x23 + x118 = 1, \\
x24 + x118 = 0, & x25 + x118 = 0, & x26 = 0, & x27 = 1, \\
x28 + x118 = 0, & x29 = 1, & x30 = 1, & x31 + x118 = 1, \\
x32 + x118 = 1, & x33 + x118 = 0, & x34 + x118 = 0, & x35 + x118 = 0, \\
x36 = 0, & x37 = 1, & x38 + x118 = 0, & x39 + x118 = 1, \\
x40 + x118 = 0, & x41 = 1, & x42 = 1, & x43 = 1, \\
x44 = 0, & x45 = 1, & x46 = 0, & x47 + x118 = 0, \\
x48 = 1, & x49 + x118 = 1, & x50 = 1, & x51 + x118 = 1, \\
x52 + x118 = 0, & x53 = 0, & x54 = 1, & x55 = 1, \\
x56 = 0, & x57 = 1, & x58 + x118 = 0, & x59 + x118 = 1, \\
x60 = 1, & x61 = 0, & x62 + x118 = 0, & x63 = 0, \\
x64 = 1, & x65 = 1, & x66 + x118 = 1, & x67 = 0, \\
x68 + x118 = 0, & x69 + x118 = 1, & x70 = 1, & x71 = 1, \\
x72 + x118 = 1, & x73 = 1, & x74 + x118 = 1, & x75 + x118 = 0, \\
x76 + x118 = 1, & x77 = 1, & x78 = 0, & x79 = 1, \\
x80 + x118 = 0, & x81 = 0, & x82 + x118 = 0, & x83 + x118 = 1, \\
x84 + x118 = 0, & x85 + x118 = 1, & x86 = 0, & x87 = 1, \\
x88 = 0, & x89 = 0, & x90 = 1, & x91 = 0, \\
x92 = 1, & x93 + x118 = 1, & x94 = 0, & x95 + x118 = 0, \\
x96 + x118 = 0, & x97 = 0, & x98 = 1, & x99 = 0, \\
x100 = 1, & x101 = 0, & x102 + x118 = 1, & x103 = 1, \\
x104 + x118 = 1, & x105 + x118 = 1, & x106 + x118 = 0, & x107 + x118 = 0, \\
x108 = 0, & x109 = 1, & x110 = 1, & x111 + x118 = 0, \\
x112 = 1, & x113 = 1, & x114 + x118 = 1, & x115 + x118 = 1, \\
x116 + x118 = 0, & x117 = 0, & x119 = 0. &
\end{array}$$

Taking $x118 = 1$ we obtain the key: $[x0, x1, \ldots, x118, x119] =$
[1,0,1,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1,0,0,1,1,1,0,1,1,0,
1,1,1,1,0,1,0,1,1,0,1,0,1,0,1,1,0,1,1,0,1,0,1,0,1,1,0,0,1,0,1,1,0,1,0,1,0,1,0,1,
1,0,1,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,1,0,1,0,0,1,0,0,1,1,0,1,1,1,1,1,0,0,1,0,1,0].