Mixed-integer Linear Programming in the Analysis of Trivium and Ktantan

Julia Borghoff*

Department of Mathematics
Technical University of Denmark

Abstract. In this paper we present a rather new approach to apply mixed-integer optimization to the cryptanalysis of cryptographic primitives. We focus on the stream cipher Trivium, that has been recommended by the eSTREAM stream cipher project, and the lightweight block cipher Ktantan. Using these examples we explain how the problem of solving a non-linear multivariate Boolean equation system can be formulated as a mixed-integer linear programming problem. Our main focus is the formulation of the mixed-integer programming model (MIP model), which includes amongst others the choice of a conversion method to convert the Boolean equations into equations over the reals, different guessing strategies and the selection of binary variables. We apply the commercial solver Cplex to our problems. The results and further possible features of the approach are discussed.

1 Introduction

There is an increasing number of low-end computing devices such as RFID tags. These devices are deployed in many different applications and environments. With the increasing deployment of those devices comes a demand for new security solutions, because the standardized primitives are often not suitable for these extremely resource constrained environments. Thus, several new cryptographic primitives such as block and stream ciphers which are specially tailored to fit the constraints have been proposed in order to satisfy the demand.

Many of these primitives are designed on the edge. Aggressive design decision have been made to on the one side satisfy the restrictions given and on the other side maintain a certain security level. New proposals are designed to resist classic attacks like differential [3] and linear [10] cryptanalysis as well as newer attacks such as algebraic attacks [7, 6]. Thus, it is important to develop new techniques for cryptanalysis.

A new method using mixed-integer programming (MIP) has first been introduced to analysis Bivium [4], a small scale variant of the cipher Trivium [9]. The algebraic description of the cipher is the starting point for this rather new analysis approach. The main idea is to convert the Boolean equation system in

* The author is supported by the Danish research council for Technology and Production Sciences, Grant No.10-093667

an equation systems over the reals (where some variables might be restricted to integer values), use the resulting equation system to formulate a mixed integer optimization problem and solve this optimization problem. This method seems to be suitable for ciphers which can be describe as a system of equations with a low monomial degree.

Compared to the classical algebraic attacks such as Gröbner bases [5] or the XL algorithm [6], mixed integer programming offers a lot of flexibility such as considering over- or underdetermined systems. Moreover, it is rather simple to incorporate probabilistic equations as it e.g., has been done in [2]. Another feature of mixed-integer optimization is that we distinguish between integer or binary variables and continuous variables, where the integer or binary variables are the variables that are 'expensive'. Thus, we can make use of the strong dependency between variables, which the algebraic description of a cipher usually exhibits, and just declare the minimum number of variables as integers or binaries, while using algebraic attacks all variables are consider equivalent.

We apply mixed-integer programming to analyze the stream cipher Trivium [9], its small scale variant Bivium [11] and the block cipher family Ktantan [8]. Trivium is a hardware oriented stream cipher which is in the final portfolio of the eSTREAM stream cipher project [1]. Its 288-bit state consists of three interconnected NFSRs. During the keystream generation in each clocking three state bits are updated using quadratic update functions and one bit of keystream is generated in a linear manner. The state recovery problem of Trivium can be described as a quadratic multivariate Boolean equation system with around a thousand variables and equations [11]. Bivium is a small scale variant of Trivium which is obtain by removing the last NFSR. Thus, Bivium has an 17- bit state consisting of two NFSRs. There are two variants of Bivium called Bivium A and B, which are only distinguish by the keystream equation. In Bivium A the keystream only depends directly on two bits the second NFSR, while for Bivium B two bits of each register determine the keystream bit. Ktantan is a family of lightweight block ciphers, that uses the design idea of Bivium. It supports an 80 bit key and three different block size: 32, 48 or 64 bits. The plaintext is loaded into two interconnected NFSRs. These NFSRs are clocked over 254 rounds, where in each round, depending on the block size, two to six bits are updated using two bits of the masterkey. The content of the register after 254 rounds is output as ciphertext.

In [4] Bivium has been investigated using mixed-integer programming. Two different MIP models have been used, one based on the standard conversion methods and one based on the integer adapted standard conversion method which has been introduced in order to formulate such MIP problems. While [4] focuses on state recovery of Bivium A using a MIP model based on the standard conversion methods, we provide results on Bivium B and Trivium, as well as on the Ktantan family, where our focus is on MIP models based on the integer adapted standard conversion method. By comparing the two different conversion methods for Bivium B and Trivium we show that the integer adapted standard conversion is more suitable for equation systems with a higher monomial de-

gree. Using the MIP model based on the integer adapted standard conversion instead of the standard conversion method yields an improvement of factor 2 for Bivium B and an improvement of factor 6 for Trivium. The best confirmed attack complexity for Trivium corresponds to a search through 2^{180} keys. For Ktantan we only consider MIP models based on the integer adapted standard conversion. We compare different guessing strategies in order to preassign some of the variables in the problem and identify that it is best to guess the key bits that are mostly used.

The article is organized as follows. In Section 2 we introduce mixed-integer linear programming. Section 3 gives a definition of the two different methods to convert Boolean equations into equations over the reals. In Section 4 we give a description of the ciphers we are analyzing in this article. We explain how to formulate the problem of solving a Boolean equations system as a mixed-integer linear programming problem in Section 5 and present the results we obtain using this method in Section 6. In the last section we conclude.

2 Mixed-Integer Programming

Combinatorial and integer optimization deals with the problem of minimizing (or maximizing) a function of several variables subject to equality and inequality constraints and integrality restrictions on some or all of the variables. A linear mixed-integer programming problem (MIP) is a problem of the form

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{Z}^k \times \mathbb{R}^l\}$$

where c is an n-vector, A is an $m \times n$-matrix ($n = k + l$) and b is an m-vector. This means that we minimize a linear function subject to linear equality and inequality constraints. Additionally, some of the variables are restricted to integer values while the other variables are real-valued.

The set S of all $x \in \mathbb{Z}^k \times \mathbb{R}^l$ which satisfies the linear constraints $Ax \leq b$

$$S = \{x \in \mathbb{Z}^k \times \mathbb{R}^l, Ax \leq b\}$$

is called a feasible set. An element $x \in S$ is called a feasible point. The MIP is called feasible if $S \neq \emptyset$ and infeasible if $S = \emptyset$. The function $z = c^T x$ is the objective function that we want to minimize.

An MIP has either an optimal solution, is unbounded, or is infeasible. If there exists for any $w \in \mathbb{R}$ an $x \in S$ such that $c^T x < w$, the MIP is unbounded. A solution for a MIP is optimal if a feasible point $x_S \in S$ exists with $c^T x_S \leq c^T x$ for all $x \in S$.

Special cases of MIP are the linear programming problem (LP)

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{R}^l\}$$

where all variables are continuous and the pure integer programming problem (IP)

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{Z}^k\}$$

where all variables are integer-valued.

To solve a MIP problem different techniques are applied such as branch-and-bound and cutting plane methods. Many of this methods have in common that they use relaxations, in particular the LP-relaxation, in the processes of finding an optimum. The relaxation of a MIP problem is obtained by removing constraints that make solving the problem difficult. This means in particular that the integer restrictions are removed in order to obtain a LP-relaxation of a MIP problem. The branch-and-bound algorithms uses the LP-relaxation for instance for estimating the best achievable objective value in a node of the search tree. Without going into detail it is intuitively clear that the more accurate this estimate is the better the algorithm will perform.

Furthermore, the solution time of a mixed-integer linear programming problem depends on the size of problem, meaning the number of variables, especially the number of variables with restrictions, the number of constraints but also on the complexity and density of the constraints. Thus, it is very important how the MIP problem is formulated. In this article we will compare two different formulating of a problem that based on different conversion methods.

3 Conversion Methods

The internal state of Trivium or the secret key of Ktantan can be described as a system of non-linear multivariate Boolean equations. A mixed-integer programming problem, however, consists of a real-valued objective function and real-valued constraints where some variables are possibly restricted to be integers. Thus, we have to represent the Boolean functions as polynomials over the reals where some of the variables may have integer restrictions. The conversion method should ensure that every solution of the Boolean function is also a solution of the polynomial over the reals, while additional real-valued solutions are not of interested. We consider two different conversion methods: the standard conversion method and the integer adapted standard conversion method.

3.1 The Standard Conversion Method

If we want to represent a Boolean function as a polynomial over the reals we have to map the values FALSE and TRUE to real numbers. The representation of the Boolean operators follows from this mapping. The most natural representation is the standard representation.

Definition 1 (Standard Representation).
Given a Boolean function $f(x_1, \ldots, x_n)$ *with* $x_i \in \{FALSE, TRUE\}$ *for* $1 \leq i \leq n$ *and a mapping* $t : \{FALSE, TRUE\} \to \{0, 1\}$ *such that*

$$t(x) = \begin{cases} 0 \text{ if } x{=}FALSE \\ 1 \text{ if } x{=}TRUE, \end{cases}$$

r *is the standard representation of* f *if*

$$r(t(x_1), \ldots, t(x_n)) = t(f(x_1, \ldots, x_n))$$

holds for all possible configurations of $(x_1, \ldots, x_n)$.

This representation of FALSE and TRUE as real numbers leads to the polynomial expressions of the Boolean operators which is given in Lemma 1. A Boolean function can be converted recursively into a polynomial over the reals using these polynomial expressions under consideration of the distributive law in the Boolean algebra.

Lemma 1. *(Standard Conversion Method)*
Let f be a Boolean function and r the corresponding standard representation. Then it holds

1. $f(x_1, x_2) = x_1 \wedge x_2 \implies r(y_1, y_2) = y_1 y_2$
2. $f(x_1, x_2) = x_1 \vee x_2 \implies r(y_1, y_2) = y_1 + y_2 - y_1 y_2$
3. $f(x_1, x_2) = x_1 \oplus x_2 \implies r(y_1, y_2) = y_1 + y_2 - 2y_1 y_2$
4. $f(x_1) = \neg x_1 \implies r(y_1) = 1 - y_1$

where $y_i = t(x_i)$ for $i = 1, 2$.

Lemma 1 can be proven by inspecting a truth table.

3.2 The Integer Adapted Standard Conversion Method (IASC)

This is a conversion method that only applies to Boolean polynomials, meaning Boolean functions in algebraic normal form, and was first introduced in [4]. The integer adapted standard conversion methods is deduce from presenting an equation modulo 2 as an equation over $\mathbb{Z}$ and works as follows: We consider a multivariate Boolean polynomial f over $\mathbb{F}_2$ and interpret this polynomial as a polynomial g over the integers by replacing XOR by addition and AND by multiplication. All solutions of the Boolean equation $f = 0$ will yield a multiple of 2 when plugged into g. Thus, for $x \in \{x | f(x) = 0\}$ it holds that $g(x) = 2 \cdot k$. Lets l be the smallest possible value for k and u be the largest. Then we obtain an integer equation by subtracting a multiple of 2 from g:

$$g - 2 \cdot y_{Int} = 0 \text{ where } l \leq y_{Int} \leq u.$$

As an example we consider the Boolean equation

$$x_1 \oplus x_2 \oplus (x_3 \wedge x_4) \oplus x_5 \oplus x_6 = \text{FALSE} \tag{1}$$

If we evaluate the corresponding real-valued polynomial $y_1 + y_2 + y_3 y_4 + y_5 + y_6$ for all solutions of (1) we get $0, 2, 4$ as results. That means that a solution of (1) is a solution to the following equation over the integers

$$y_1 + y_2 + y_3 y_4 + y_5 + y_6 - 2y_{\text{Int}} = 0$$

where $y_{\text{Int}} \in \{0, 1, 2\}$ and $y_i \in \{0, 1\}$ for $i = 1 \ldots 6$. The degree of the polynomial over the integer stays the same and the number of variables and monomials per equation is increased only by one compared to the Boolean polynomial.

However, this conversion method can only be applied if we can restrict the newly introduced variable y_{Int} to be integer, the equation does in general not hold over the reals.

Note, that opposed to the standard conversion methods the integer adapted standard conversion method can be generalized for prime fields. Let $f(x) = 0$ be an equation over $\mathbb{F}_p$ then $f(y) - py_{Int}$ is the corresponding equation over $\mathbb{Z}$.

4 Trivium and Ktantan

In this section we give a short description of the stream cipher Trivium [9] including the small-scale variant Bivium [11] and the lightweight block cipher Ktantan [8] which uses design ideas from Bivium.

4.1 Trivium

Trivium [9] was recommended as a stream cipher for hardware application in the portfolio of the ECRYPT eSTREAM project [1]. Trivium takes an 80 bit key and 80 bit IV which are loaded into the internal state of size 288 bits consisting of three interconnected NFSRs. As key-setup the algorithm is clocked $4 \cdot 288$ times without producing any output. Afterwards the keystream is produce as described in the following pseudecode where s_j denotes the jth bit of the state and z_i is the keystream bit at time i:

for $i = 1, 2, \ldots$ **do**
  $z_i \leftarrow s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$ ▷ Generate output bit z_i
  $t_{i,1} \leftarrow s_{66} + s_{93} + s_{91} \cdot s_{92} + s_{171}$
  $t_{i,2} \leftarrow s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{264}$
  $t_{i,3} \leftarrow s_{243} + s_{288} + s_{286} \cdot s_{287} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_{i,3}, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_{i,1}, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_{i,2}, s_{178}, \ldots, s_{287})$
end for

Here the '+' and '$\cdot$' operations stand for addition and multiplication over $\mathbb{F}_2$ respectively.

Bivium B [11] is a small-scale variant of Trivium which is obtained by removing the last NFSR while the structure is kept. That means that the internal state of Bivium B is of size 177 bits, the keystream depends linearly on 4 state bits and two state bits are updated each clocking of the algorithm.

Trivium as an Equation System In [11] it is stated that the initial state, which is the state of the registers at the time when the key generation starts, (or any other internal state) can be expressed as a system of sparse linear and quadratic Boolean equations. We consider the initial state bits as variables and label them with $s_1 \ldots, s_{288}$. In each clocking of the Trivium algorithm three state bits are updated. The update function is a quadratic Boolean function

Algorithm 1 Bivium B:Pseudo code producing one bit of keystream.

for $i = 1, 2, \ldots$ **do**
  $z_i \leftarrow s_{66} + s_{93} + s_{162} + s_{177}$ ▷ Generate output bit z_i
  $t_{i,1} \leftarrow s_{66} + s_{93} + s_{91} \cdot s_{92} + s_{171}$
  $t_{i,2} \leftarrow s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_{i,2}, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_{i,1}, s_{94}, \ldots, s_{176})$
end for

of the state bits. In order to keep the degree low and the equations sparse we introduce new variables for each updated state bit $t_{i,1}$, $t_{i,2}$, $t_{i,3}$. We get the following equations from the first clocking

$$\begin{aligned} s_{66} \oplus s_{93} \oplus s_{91} \cdot s_{92} \oplus s_{171} &= s_{289}, \\ s_{162} \oplus s_{177} \oplus s_{175} \cdot s_{176} \oplus s_{264} &= s_{290}, \\ s_{243} \oplus s_{288} \oplus s_{286} \cdot s_{287} \oplus s_{69} &= s_{291}, \\ s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288} &= z. \end{aligned} \tag{2}$$

where the last equation is the keystream equation with z being the known keystream bit.

After observing 288 keystream bits we can set up a fully determined system of 954 Boolean equations in 954 unknowns [11]. We only need to consider 954 equations and unknowns instead of 1152 since we do not care about the last 66 state updates for each register. These variables will not be used in the keystream equation because the new bits are not used for the keystream generation before 66 further clockings of the cipher. By clocking the algorithm more than 288 times we can easily obtain an overdetermined equation system.

In a similar way Bivium can be represented as a non-linear Boolean equation system in 399 equations and variables after observing 177 bits of keystream.

4.2 Ktantan

The Katan and Ktantan family of block ciphers is designed to provide 80 bit security with as few gates as possible. The main difference between Katan and Ktantan is the keyschedule. Both cipher come in three different block sizes 32, 48 and 64 bits. To begin with we describe the smallest member of the Ktantan family: Ktantan-32. The two other variants differ from Ktantan-32 in the size of the register, the tab positions, and the number of applications of the update functions in each round. The design is based on the design idea of Bivium. The cipher consists of two NFSR L_1 and L_2 of length 13 and 19 bits respectively and a known counter LFSR T of length 8. The plaintext is loaded in the two NFSRs and then the cipher is clock for 254 rounds. The details of the update of the registers are given as follows, where k_i denotes a subkey bit, IR is the irregular clocking determined by the counter T and $L_i(j)$ denotes the js bit of register L_i.

for $i = 1, \ldots, 254$ **do**
$f_a(L_1) \leftarrow L_1(12) + L_1(7) + L_1(8) \cdot L_1(5) + L_1(3) \cdot IR_i + k_{2i}$
$f_b(L_2) \leftarrow L_2(18) + L_2(7) + L_2(12) \cdot L_2(10) + L2(8) \cdot L2(3) + k_{2i+1}$
$(L_1(0), L_1(1), \ldots, L_1(12)) \leftarrow (f_b(L_2), L_1(0), \ldots, L_1(11))$
$(L_2(0), L_2(1), \ldots, L_2(18)) \leftarrow (f_a(L_1), L_2(0), \ldots, L_2(17))$
end for

For the Ktantan family the key is burnt and for each round of two masterkey bits are chosen as subkey bits. Which masterkey bit is used as subkey in each round of the encryption process is determined by the counter T. For Ktantan-32 each masterkey bits is used at least 5 and at most 7 times. The full key schedule can be found in Appendix A.

The register T is initialized with the all one state and runs until the all-one state is reached again. Thus. we obtain a multivariate non-linear quadratic Boolean equation system by introducing a new variable for each update state bit. Additionally the equation system contains 80 unknowns for the keybits. For one plaintext/ciphertext pair we introduce 32 variables for the initial register, 2 variables for the register update in every round and 80 variables for the key bits, furthermore we obtain $64 + 2 \cdot 254$ equations (note that we obtain 64 extra equations because the plaintext/ciphertext is known). For each additional pair we obtain $64 + 2 \cdot 254$ equations in only $32 + 2 \cdot 254$ unknowns. Thus, we can easily generate an overdetermined equation system by considering three plaintext/ciphertext pairs.

Ktantan-48 consists of two NFSRS of length 19 and 29 and the following two update functions, which ar applied twice in every round using the same subkey bits:

$f_a(L_1) \leftarrow L_1(18) + L_1(12) + L_1(15) \cdot L_1(7) + L_1(6) \cdot IR_i + k_{2i}$
$f_b(L_2) \leftarrow L_2(28) + L_2(19) + L_2(21) \cdot L_2(13) + L2(15) \cdot L2(6) + k_{2i+1}$

For each text pair we obtain $96 + 2 \cdot 2 \cdot 254$ equations in $48 + 2 \cdot 2 \cdot 254$ unknowns. Additionally, there are 80 unknowns describing the key variables. Thus, only two plaintext/ciphertext pair yield an overdetermined Boolean equation system. In the same manner two plaintext/ciphertext pair for Ktantan-64 yield an overdetermined equations system, because each text pair yields $128+2\cdot3\cdot254$ equations in $64+2\cdot3\cdot254$ unknowns. Ktantan-64 has two NFSRs of size 25 and 39 respectively, which are update by the following feedback functions where 3 state bits of each register are updated in every round using the same subkey bits

$f_a(L_1) \leftarrow L_1(24) + L_1(15) + L_1(20) \cdot L_1(11) + L_1(9) \cdot IR_i + k_{2i}$
$f_b(L_2) \leftarrow L_2(38) + L_2(25) + L_2(33) \cdot L_2(21) + L2(14) \cdot L2(9) + k_{2i+1}$.

It is interesting to note that for the two variants with bigger block size we only need two plaintext/ciphertext pairs to generate an overdetermined system. However, as the update functions are applied several times in each round, this does not result into smaller equation systems.

5 Boolean Equation Systems as Mixed Integer Programming Problems

In this section we explain how to convert the problem of solving a non-linear Boolean equation system into a mixed-integer linear programming problem.

To start with we recall what a mixed integer linear programming problem is. A mixed integer linear programming problem consists of a linear objective function, a set of linear equality and inequality constraints and a set of variables with integer restrictions. We distinguish two types of problems, an optimization problem, where we want to find a point that satisfies all constraints and restrictions and yields best possible value for the objective function, and a feasibility problem, where we are only interested in finding an element in the set of points that satisfy all constraints and restrictions. The size of the problem (meaning, number of constraints and variables) is an important factor when estimating if a problem is solvable, but it is not the most crucial. There are more factors we have to take into account when formulating a mixed integer program, because they influence the solvability. As mentioned in Section 2 most approaches for solving mixed-integer programming problem use relaxations of the problem during the solution process. This indicates that the tighter the bounds are which a relaxation yields the better. One way to achieve tight bounds is to keep the constraints sparse. Very complex constraints that contain many variables usually allow more solutions then sparse constraints that only contain very few variables. Furthermore, in a branch-and bound approach only variables with integer restrictions are considered as branching variables. Thus, the number of integer variable determines the size of the search tree in the worst case and influences thus the running time of the algorithm. Therefore, one should try to keep the number of variables with restrictions (binary or integer) low.

When transforming a Boolean equation system into a mixed integer programming problem we know that for each variable in the Boolean system we have to define at least one corresponding variable in the MIP and each equation in the Boolean system corresponds to at least one constraint of the MIP. Thus, the number of variables and constraints in the MIP is lower bounded by the number of variables and equations in the Boolean equation system and we derive the set of constraints of the MIP from the Boolean equations. As we consider a non-linear Boolean equation system on the one side and a system of linear constraints on the other side we will have to increase the number of variables and constraints in order to properly translate the one problem into the other. Later in the section we describe in detail how we can convert a Boolean equation system into a mixed integer linear programming problem.

We have already mentioned that the Boolean equations will be translated into a set of constraints. However, we cannot deduce the objective function directly from the original Boolean problem. That means that we can freely choose an objective function and thus, we consider a feasibility rather than an optimization problem. But even though we can freely choose the objective function it is important to put some thoughts in a proper choice of it because the objec-

tive function is extensively used in many solvers e.g. as bounding function and therefore influences the solution time significantly.

In the remainder of the section we show step by step how to obtain the set of constraints, how to decide which variables to restrict and how to choose an objective function. Some of the choice that are made are based on earlier experiments which are reported in detail in [4]. We consider two different formulations of the problem of solving a non-linear Boolean equation system as MIP, one based on the standard conversion method and one based on the integer adapted standard conversion method.

5.1 Converting the Boolean System into a Set of Linear Constraints

The main part when translating a Boolean equation system into a mixed integer programming problem is transforming the Boolean equations into linear constraints. This is the only part where the Boolean equation system is directly used in the modeling of the mixed integer programming problem. The translation from the set of Boolean equations into a set of linear constraints is basically performed in two steps. First, after a possible preprocessing of the Boolean equation system, the Boolean equations are converted into equations over the reals or the integers. This yields a system of non-linear equations. The next step is to linearize these equations by the replacing non-linear terms by new variables and additional inequality constraints which ensure that new variables behave according to the non-linear terms they are replacing. Depending on the choice of the conversion method we obtain a different set of constraints. Furthermore, the conversion methods implies integrality restrictions on some variables. Thus, the choice of the conversion method together with the set of Boolean equation mainly determines the feasible set.

Using the Integer Adapted Standard Conversion Method Applying the integer adapted standard conversion method is straight forward and yields, when we start with a quadratic Boolean equation system, a quadratic equation over the integers where all variables but the variables y_{Int}, that are introduced by the conversion method, are binary.

We linearize the constraints in the following way. We replace each quadratic term $x_i x_j$ by a new variable $q_{i,j}$ and add the following three inequalities to the set of constraints

$$q_{i,j} - x_i \leq 0 \tag{3}$$

$$q_{i,j} - x_j \leq 0 \tag{4}$$

$$x_i + x_j - q_{i,j} \leq 1 \tag{5}$$

The inequalities (3) and (4) ensure that $q_{i,j}$ is zero when x_i or x_j are zero, while inequality (5) forces $q_{i,j}$ to take the value one if both x_i and x_j are one. This method of linearizing equations can only be applied if the variables which are involved in the non-linear term are binary variables.

Using the Standard Conversion Method When we apply the standard conversion method to a Boolean function in ANF the total degree of the resulting polynomial over the reals equals the number of variables in the Boolean function. Furthermore, the number of monomials contained in the real-valued polynomial is exponential in the number of monomial in the Boolean polynomials.

Therefore we add a preprocessing phase where we prepare the Boolean equation system, before the application of the standard conversion method. We introduce auxiliary variables such that the Boolean polynomial contains at most four variables. Introducing new variables increases not only the number of variables but also introduces additional equations (one equation for each variable). Furthermore, we rewrite the equation such that left and the right hand side of the equation contain at most two variables. After this preprocessing we convert each side of the Boolean equation separately into a polynomial over the reals using the standard conversion method and consider the difference of these two polynomials as the desired real-valued polynomial. This yields us a system of quadratic real-valued equations when starting with a quadratic Boolean equation system.

In order to clarify the method we described above we use the following example. We consider the Boolean equation:

$$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \wedge x_6 \oplus x_7 = 0$$

We introduce two auxiliary variable s_1 and s_2 thus that we obtain three equations in total:

$$s_1 = x_1 \oplus x_2 \tag{6}$$
$$s_2 = x_5 \wedge x_6 \tag{7}$$
$$s_1 \oplus x_3 \oplus s_2 \oplus x_7 = 0 \tag{8}$$

The first two equations already satisfy the properties that each side of the equations contains at most two variables, while we rewrite the last equation as follows

$$s_1 \oplus x_3 = s_2 \oplus x_7 \tag{9}$$

The equations (8) and (9) have the same set of solution but converting Equation (9) yields a quadratic polynomial over the reals while Equation (8) yields a real-valued polynomial of degree 4. Applying the standard conversion method to each side of the equations separately and taking the difference yields the following three equations over the reals

$$s_1 - x_1 - x_2 + 2x_1x_2 = 0 \tag{10}$$
$$s_2 - x_5x_6 = 0 \tag{11}$$
$$s_1 + x_3 - 2s_1x_3 - s_2 - x_7 + 2s_2x_7 = 0 \tag{12}$$

Thus, we obtain quadratic polynomial equations over the reals which can be linearized using the approach described in Section 5.1.

5.2 Integer Restrictions

Some restriction on variables are implied by the conversion method that has been used. When using the integer adapted standard conversion the variables y_{Int} which are introduced by the conversion method have to be restricted to be integers; their bounds depend on the number of monomials in the Boolean polynomial. Usually, it holds $y_{Int} \in \mathbb{Z}$ and $0 \leq y_{Int} \leq k/2$ where k is the number of monomials in the corresponding Boolean polynomial.

The variables $q_{i,j}$ which are introduced during the linearization process directly depend on the variables in the quadratic term $x_i x_j$ which they replace. This is independent of the conversion method used and means, when x_i and x_j are binary, $q_{i,j}$ will automatically take a binary value. Thus, there is no additional binary restriction on $q_{i,j}$ necessary. The same holds for the auxiliary variables which are introduced to decrease the number of variables per equations as the expense of additional equations before applying the standard conversion method.

In general when describing a cipher we introduce new variables in order to keep the degree of the polynomials low; these are for example the new variables which are introduce for the updated register bits in Bivium, Trivium and Ktantan. These variables are depending on the initial state variables and in the case of Ktantan the key bits and are connected to them by equations with binary/integer coefficients. Thus, they will be binary if the initial variables are binary. Therefore, it is sufficient to restrict the initial state and the key variables to be binary and no restrictions on the variables coming from the register updates are necessary. It is important to note that even though we do not restrict the variables to be binary we still bound them with zero as lower and one as upper bound.

The initial state variables and the key variables have to be binary variables.

5.3 The Objective Function

We convert the Boolean equation system into a equation system over the reals or the integers and use it to describe the feasible set of the mixed-integer programming problem. Thus, we are rather interested in finding a feasible point than an optimal solution. If we assume that there is only one element in the feasible set we will find the same solution no matter which objective function we use (assuming that the solver finds a solution in a reasonable time). However, a good choice for the objective function significantly improves the solution time. In [4] experiments with different objective functions have been carried out and the solver performed best using the sum over all variables as objective function when using the standard conversion and the sum over all but the integer variables y_{Int} introduced by the conversion method when using the integer adapted standard conversion method. For a random point this objective function will yield as objective value about $\frac{1}{2} \times$ # variables, but for a feasible point the objective function value will be significantly lower than this value. The reason is that many of the newly introduced variables replace a quadratic term, thus for

a feasible point they are zero with probability $\frac{3}{4}$. This means by minimizing this function the algorithm gets information that enables it to consider nodes that more likely yield a feasible point first. Therefore, we choose in all our experiments the sum over all variables but the variables introduced by the conversion method as objective function.

6 Results

In this chapter we present experimental results on Bivium, Trivium and Ktantan when using mixed-integer linear programming in order to recover the initial state or the secret key, respectively. All experiments are performed using the commercial solver IBM Ilog Cplex 12.1 available under academic license on a standard PC. The only available measurement of the complexity is the running times of the algorithm in seconds. In order to be able to compare our results with exhaustive search, we provide an implementation[1] of the naive brute force search and measure the time needed to perform the search.

For Bivium and Trivium we compare the approaches of using the standard and the integer adapted standard conversion method to model the Boolean equation system as mixed-integer linear programming problem while for Ktantan we focus on the MIP using the integer adapted standard conversion only. It turns out that for all ciphers it takes too long time to solve the mixed-integer linear programming problem as it is, therefore we simplify the problem by guessing some bits.

6.1 Bivium and Trivium

We compare the standard and the integer adapted standard conversion when modeling Bivium and Trivium as a mixed-integer programming problem. In order to simplify the problem we guess the last bits of the internal state. In [4] different guessing strategies have been compared and this seems to be the best one. Furthermore, we consider an overdetermined equation system in order to ensure that there is with high probability only one solution and thus, only one element in the feasible set. For Bivium and Trivium it is easy to obtain an overdetermined system by observing additional keystream bits; after we have generated a fully determined system every keystream bit yields two new variables and three equation or three new variables and four equations, respectively.

Bivium/Trivium as MIP

Bivium/Trivium as MIP using the standard conversion method In [4] it is step by step explained how to convert Bivium A into a mixed-integer linear programming problem using the standard conversion. We use the same parameters for

[1] It is a naive implementation of the ciphers in C which is not optimized for key search but performs a simple testing. Thus, an optimized implementation might be faster.

Bivium B. We choose the sum over all variables as objective function, restrict only 177 variables corresponding to the initial state to be binary, and consider an overdetermined equation system that has been generated by considering 1/3 additional keystream bit, thus 177+59 keystream bits. This number has been experimentally identified. This yields an MIP in 2821 variables and 5865 constraints, 1388 of these are equality constraints.

In an equivalent manner we can model Trivium as a mixed-integer programming problem using the standard conversion method. After observing 384 bits of keystream this yields a mixed-integer programming problem in 7746 variables and 16638 constraints. We restrict only the initial 288 variables to be binary.

Bivium/Trivium as MIP using the integer adapted standard conversion Using the integer adapted standard conversion to model the MIP as been shortly consider for Bivium A in [4] but showed a worse performance than using the standard conversion method. However, already in [4] it was suggested that the integer adapted standard conversion method might be more suitable for Boolean equation system containing more complex equations as it is the case for Bivium B. We use the sum over all but the integer variables y_{Int} which are introduce by the conversion as objective function, the 177 variables corresponding to the initial state are restricted to be binary while the variables y_{Int} are integers and bounded depending on the number of monomials in the equation. We assume that already 5 additional keystream bits are enough to generate an overdetermined Boolean equation system which has a unique solution with high probability. We carried out some experiments in order to compare the solution time when adding a different number of additional keystream and the corresponding quadratic equations and found that it is sufficient to observe 5 additional keystream bits. The results are summarized in Table 1.

Table 1. Bivium B as MIP using the IASC with 50 preassigned bits. Comparison of the solution time for different numbers of additionally consider keystream bits.

add. keystream bits	5	30	59
time in sec.	983,84	1310,21	1332,17

This yields a mixed-integer programming problem in 1055 variables of which 177 are binary and 414 are integers after observing 182 bits of keystream. Of the 1110 constraints 414 are equality constraints.

For Trivium we obtain a mixed-integer linear programming problem in 2624 variables and 3017 constraints after observing 293 keystream bits. We restrict the 288 initial variables to be binary and we additionally have 974 variables with integer restrictions.

Results on Trivium and Bivium We compare the time needed for solving the mixed-integer problem that was obtained by using the standard and the

integer adapted standard conversion method. To simplify the problem we guess some of the last bits of initial state.

Table 2. Bivium: Comparison of the solution time in seconds when using the standard conversion method or the integer adapted standard conversion method for modeling the MIP. The last 50 bits of the initial state are set to the correct value.

	SCM	IASC
average	207,4	113,9
log average	7,7	6,8
total time (log)	57,7	56,8

For Bivium B we guess the 50 last bits of the register. The results are summarized in the Table 2. According to our reference implementation we can search through $2^{10.1}$ keys per second. The average solution time for Bivium B as a mixed-integer programming problem when using the standard conversion method and guessing 50 bits is $207,45$ seconds. This is the average for the cases were we guessed correct, for wrong guesses the solver decides on average even faster that the problem is infeasible. Thus, when we incorporate the time for bit guessing, the initial state can be found in $2^{57.7}$ which corresponds to a search through $2^{67.8}$ keys[2]. Using the integer adapted standard conversion to model Bivium as MIP improves the running time by a factor of two compared to the standard conversion method. When guessing 50 bits the problem can be solved in $113,88$ seconds on average which leads to a total solution time of $2^{56.8}$ seconds or equivalent it means that we have to search through $2^{66.9}$ keys.

For Trivium we have to guess at least the 160 bits in order to obtain results in a reasonable time. Thus, this approach is clearly worse than exhaustive search. However, assuming that we are facing a Boolean equation system with around a 1000 variables, or assume that the 288 initial state variables form the secret the results are quite impressive and show the strength of using mixed integer programming for solving sparse Boolean equation systems. The results are summarized in Table 3. The MIPs modeled using the integer adapted standard conversion can in average be solved 6 times faster than when using the standard conversion method. Furthermore, we can observe that when we reduce the number of guess, the running time for a single instance increases but the overall running time decreases, because the increase in the solution time is less than the gain when guessing less bits. Using the integer adapted standard conversion method reducing the number of guessed variable from 165 to 160 improves

[2] In [4] it is stated that it takes $2^{64.5}$ seconds to solve the MIP corresponding to Bivium B using the standard conversion. The improvement by a factor of almost 2^7 can be explained by using an newer version of CPLEX on a faster machine. For the experiments carried out here we use CPLEX 12.2 on PC with an Intel Core 2 Duo E6850 3GHZ processor and 3 GB RAM.

the total running time by a factor of two and yields a average running time of $2^{171.3}$ seconds including the time for guessing the variables. This corresponds to a search through $2^{180,3}$ keys[3].

Table 3. Trivium: Comparison of the solution time in seconds when using the standard conversion method or the integer adapted standard conversion method for modeling the MIP. The last 165 bits or 160 bits of the initial state are set to the correct value.

seed	SCM 165 preassig.	IASC 165 preassig.	IASC 160 preassig.
average	1729,7	281,7	2461,1
log average	10,8	8,1	11,3
total time (log)	175,8	173,1	171,3

6.2 Ktantan

The experiments on Trivium showed that the integer adapted standard conversion method is more suitable when converting complex equations into constraints. Therefore we focus on the integer adapted standard conversion method to model the key recovery problem of Ktantan as mixed-integer linear programming problem. For Ktantan32 three plaintext/ciphertext pairs will yield an overdetermined system and thus with high probability determine the key uniquely. However, in order to simplify the problem we will guess several key bits. Guessing key bits fixes some of the variables and thus decreases the number of variables. Thus, it is sufficient to consider the equation system obtained from two plaintext/ciphertext pairs. This yields a Boolean equation system of 1168 variables in 1152 equations. The corresponding mixed-integer programming problem consists of 3728 variables and 5760 constraints. We only restrict the 80 key variables to be binary and the variables y_{Int} introduceed by the integer adapted standard conversion method to be integer. These are 1016 variables. As the registers are initialized with the known plaintext and the key variables are binary the updated register variables will automatically take binary values when bounded by zero and one. As objective function we take the sum over all variables that are bounded by zero and one. That are all variables except the variables introduced by the conversion method.

We have to guess 70 variables in order to obtain a solution in a reasonable time. We explore three different guessing strategies: The first keybits of the master key, the last keybits of the master key and the key bits which are used most often as subkeys. It turns out that guessing the most used bits is the best strategy and yields a running time of $2^{77,66}$. The results are summarized in Table 4. We can see that for very few problem no running time is given. In these cases

[3] According to our reference implementation we can search through 2^9 keys per seconds.

the running time exceeded one hour and an the computation has been aborted. In such cases we can guess an additional bit in order to speed up the calculation.

When testing a single plaintext/ciphertext pair we can search through $2^{16.3}$ keys per second. Thus, using mixed-integer programming in order to find the secret key corresponds to a search through 2^{94} keys and is therefore worse than exhaustive search. We formulate the key recovery problem of Ktantan48/64 as a mixed-integer programming problem. We guessed the 72 most used key bits. The results are summarized in Table 5. We can conclude, that Ktantan does not seem to be a suitable problem for mixed-integer programming. The only variables which are actually unkown in the problem are the 80 key variables, as we have to guess almost all variables of these variables and it takes additional 3-4 minutes to solve the resulting problem, an exhaustive search will be more efficient.

7 Conclusion

We investigated a rather new technique for cryptanalysis based on mixed-integer programming. We presented results on Bivium, Trivium and the Ktantan family, where we considered different MIP models as well as different guessing strategies. In the current version mixed-integer programming does not pose a threat to Trivium or Ktantan. However, especially the results on Trivium show that mixed-integer programming has potential to provide solution to structured Boolean equation systems where a part of the variables depend on some initial variables. Generally, this approach seems to be similar to algebraic attacks because our targets are cryptographic algorithms that can be describe as a system of sparse equations. However, mixed-integer programming is far more flexible than algebraic attacks as we can solve over- or underdetermined systems. Furthermore the fact that we consider optimization problem enables to easily incorporate probabilistic equation, which makes this attack suitable for side channel attacks, noisy keystream and general algebraic attack where we can find additional equations that do not hold in all cases. Therefore we think that mixed-integer programming is a valuable technique in cryptanalysis, but it is yet open to examine the real potential of it in different scenarios.

A Ktantan Key Schedule

The master key of the Ktantan family is burnt into the device. For each round two bits of the master bits are chosen as subkey bits. The counter register T is used to determine the subkey bits.

The masterkey K is split into $K = w_4||w_3||w_2||w_1||w_0$ where the least significant bit of w_0 is the least significant bit of K. Let $a_i = MUX16to1(w_i, T_7T_6T_5T_4)$ where $MUX16to1(x, y)$ gives the yth bit of x. Then

$$k_a = \bar{T}_3\bar{T}_2(a_0) \oplus (T_3 \vee T_2)MUX4to1(a_4a_3a_2a_1, T_1T_0)$$

Table 4. Results on Ktantan-32. Running time in seconds when 70 key bits are guessed. The second column gives the running time when guessing the key bits which are used most often during the encryption, column 3 and 4 contain the running times when we guess the first or the last bits of the key respectively. "'-"' denotes that the algorithm has been stopped after exciding a running time of 1000 sec.

seed	most used	first	last
12341	372,9	72,7	65,07
12342	113,5	295,2	179,26
12343	231,5	228,4	195,49
12344	165,4	151,9	315,2
12345	336,4	464,1	390,16
12346	220,7	598,7	154,07
12347	108,4	437,5	134,5
12348	170,2	10,6	106,81
12349	-	227,3	185,56
12350	165,9	611,1	246,09
12351	336,1	467,6	30,64
12352	445,6	397,18	114,22
12353	195,0	1215,96	140,26
12354	51,8	64,12	247,75
12355	121,5	287,17	83,76
12356	203,4	443,84	282,05
12357	33,5	234,03	249,93
12358	227,5	-	388,07
12359	70,4	364,04	115,27
12360	261,5	148,61	172,58
12361	156	376,82	139,65
12362	150,9	951,86	290,19
12363	303,4	457,83	369,96
12364	200,2	336,68	138,59
12365	330,3	548,69	161,27
12366	232,9	116,72	224,14
12367	25,1	624,28	241,3
12368	199,2	308,34	219,68
12369	291,4	295,44	54,6
12370	234,7	497,39	267,75
12371	8,7	290,38	222,27
12372	148,3	493,29	184,85
12373	214,2	392,72	199,32
12374	117,8	374,17	96,36
12375	226,1	658,81	599,03
12376	67,5	202,88	240,51
12377	376,8	-	137,76
12378	300,5	363,7	171,73
12379	242,4	2234	151,62
12380	308,3	637,06	432,2
12381	224,4	429,47	-
12382	30,6	223,99	517,45
12383	19,4	336,03	70,11
12384	207,7	417,01	230,49
12385	438,4	568,3	274,97
12386	218,5	305,25	29,94
12387	155	111,98	64,18
12388	198,8	258,68	81,21
12389	150,8	-	347,82
12390	279,8	391,89	271,43
average	201,8	423,9	208,7
log	7,7	8,7	7,7

Table 5. Running times for Ktantan48 and Ktantan64

Cipher	time in sec	log	total running time in sec
Ktantan48	99,32	6,63	78,63
Ktantan64	317,1	8,31	80,31

$$k_b = \bar{T_3}T_2(a_4) \oplus (T_3 \vee \bar{T_2})MUX4to1(a_3a_2a_1a_0, T_1\bar{T_0}),$$

and $k_2i = k_a$ and $k_{2i+1} = k_b$.

References

1. The eSTREAM project. http://www.ecrypt.eu.org/stream/.
2. Albrecht, M., and Cid, C. Cold boot key recovery by solving polyonomial systems with noise. IACR eprint http://eprint.iacr.org/2011/038.pdf, 2011.
3. Biham, E., and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993. ISBN: 0-387-97930-1, 3-540-97930-1.
4. Borghoff, J., Knudsen, L. R., and Stolpe, M. Bivium as a mixed-integer linear programming problem. In *Cryptography and Coding, 12th IMA International Conference* (2009), M. G. Parker, Ed., vol. 5921 of *Lecture Notes of Computer Science*, Springer, pp. 133–152.
5. Buchberger, B. Ein algorithmisches Kriterium fü r die Lö sbarkeit eines algebraischen Gleichungssystems. *Aequationes Mathematicae 4* (1970), 374–383.
6. Courtois, N., Klimov, E., Patarin, J., and Shamir, A. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology – EUROCRYPT 2000* (2000), B. Preenel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 392–407.
7. Courtois, N., and Meier, W. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology – EUROCRYPT 2003* (2003), E. Biham, Ed., vol. 2656 of *Lecture Notes of Computer Science*, Springer, pp. 644–644.
8. De Cannière, C., Dunkelman, O., and Knežević, M. KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009* (2009), vol. 5747/2009 of *Lecture Notes in Computer Science*, pp. 272–288.
9. De Cannière, C., and Preneel, B. Trivium specifications. *eSTREAM, ECRYPT Stream Cipher Project* (2006).
10. Matsui, M. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT '93* (1994), T. Helleseth, Ed., vol. 765 of *Lecture Notes in Computer Science*, Springer.
11. Raddum, H. Cryptanalytic results on Trivium. eSTREAM report 2006/039, 2006. http://www.ecrypt.eu.org/stream/triviump3.html.