

# Bivium as a Mixed-0-1 Linear Programming Problem

## Extended Abstract

Julia Borghoff <sup>\*</sup>, Lars R. Knudsen, Mathias Stolpe

J.Borghoff@mat.dtu.dk  
Department of Mathematics  
Technical University of Denmark

### Introduction

Trivium is a stream cipher proposed for the eSTREAM project [5, 4]. Raddum introduced some reduced versions of Trivium, named Bivium A and Bivium B. The problem of recovering an internal state of Bivium can be described as a system of quadratic Boolean equations. The problem of solving a system of quadratic equations over  $GF(2)$  is known to be NP-hard. In this paper we propose a new approach to solve such a system using combinatorial optimization. We convert the Boolean equation system into an equation system over  $\mathbb{R}$  and formulate the problem of finding a 0-1-valued solution for the system as a mixed-0-1 programming problem. This gives us an attack on Bivium B in estimated time complexity of  $2^{64.5}$  seconds.

### Bivium

Bivium has a 177 bit initial state which is divided into two nonlinear feedback shift registers (NFSR). The 80-bit key is loaded into the first register and an 80-bit IV into the second. After  $4 \cdot 177$  clockings (steps) of the algorithm which do not produce any output a key stream bit is produced by an XOR of some state bits. The following pseudo code specifies how to generate one bit  $z$  of the key stream for Bivium B where the  $s_i$ s denote

---

<sup>\*</sup> The author is supported by a grant from the Danish research council for Technology and Production Sciences grant number 274-07-0246.

the state bits:

$$\begin{aligned}
t_1 &\leftarrow s_{66} + s_{93} \\
t_2 &\leftarrow s_{162} + s_{177} \\
z &\leftarrow t_1 + t_2 \\
t_1 &\leftarrow t_1 + s_{91}s_{92} + s_{171} \\
t_2 &\leftarrow t_2 + s_{175}s_{176} + s_{69} \\
(s_1, s_2, \dots, s_{93}) &\leftarrow (t_2, s_1, \dots, s_{93}) \\
(s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (t_1, s_{94}, \dots, s_{176})
\end{aligned}$$

Bivium A only differs in the key stream bit equation from Bivium B. In Bivium A the key stream bit  $z = t_2$  which means that the key stream bit only depends directly on the second register.

By introducing a new variable for each update of the NFSRs [6] we obtain a full description of an internal state after observing 177 key stream bits by equations of the form:

$$s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} = z \quad (1)$$

$$s_{66} \oplus s_{93} \oplus (s_{91} \wedge s_{92}) \oplus s_{171} \oplus s_{178} = 0 \quad (2)$$

$$s_{162} \oplus s_{177} \oplus (s_{175} \wedge s_{176}) \oplus s_{69} \oplus s_{179} = 0. \quad (3)$$

177 key stream bits yield a fully determined system of 399 equations in 399 unknowns.

### Mixed-0-1 Programming

Combinatorial 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-0-1 programming problem (MIP) is a problem of the form

$$\min_x \{c^T x \mid Ax \leq b, x \in \{0, 1\}^k \times \mathbb{R}^l\}$$

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 binary values while the other variables are real-valued.

The set  $S$  of all  $x \in \{0, 1\}^k \times \mathbb{R}^l$  which satisfy the linear constraints  $Ax \leq b$

$$S = \{x \in \{0, 1\}^k \times \mathbb{R}^l, Ax \leq b\}$$

is called a feasible set. An element  $x \in S$  is called a feasible point.

### Conversion Methods

The algorithms for solving mixed-0-1 programming problems work over the reals using integrality restrictions. The equation system generated by Bivium is a quadratic Boolean system. Hence we have to convert the Boolean equations to the real domain. Here the only requirement is that a solution for the Boolean system is also a solution for the system over the reals and each binary solution for the system over the reals is also a solution for the Boolean system. Additionally we want to map *TRUE* and *FALSE* to the set  $\{0, 1\}$ . One method to do this is the Standard Conversion Method [2]:

**Definition 1 (Standard Conversion)** *The Standard Conversion is defined by a mapping  $t : \{FALSE, TRUE\} \rightarrow \{0, 1\}$  with*

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

and

$$s_1 \wedge s_2 \Rightarrow x_1 \cdot x_2$$

$$s_1 \oplus s_2 \Rightarrow x_1 + x_2 - 2x_1x_2$$

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

### Bivium as a mixed-0-1 programming problem

**Linearization** We want to convert the Boolean system of equations into a set of linear equalities and inequalities over the reals. We will use this set of linear equalities and inequalities as a set of constraints which describes the feasible set of a mixed-0-1 programming problem. If we additionally restrict the variables to be binary a feasible point will also be a solution for the Boolean equation system. It is important to note that we require the constraints to be linear. Furthermore the degree of the polynomial over the reals is equal to the number of variables in the Boolean polynomial (a Boolean polynomial is a Boolean function in ANF). Hence we introduce auxiliary variables so that none of the equations contains more than 4 variables:

*Example 1.* Consider

$$s_{66} \oplus s_{93} \oplus (s_{91} \wedge s_{92}) \oplus s_{171} \oplus s_{178} = 0.$$

We introduce two auxiliary variables and get three new equations

$$\begin{aligned} r_1 &= s_{66} \oplus s_{93}, \\ r_2 &= s_{91} \wedge s_{92}, \\ r_1 \oplus r_2 \oplus s_{171} \oplus s_{178} &= 0. \end{aligned}$$

Furthermore we split the equations into two halves, convert each side separately using the Standard Conversion Method and subtract the resulting polynomials afterwards.

*Example 2.* Consider the Boolean equation  $s_1 \oplus s_2 \oplus s_3 \oplus s_4 = 0$ . We can rewrite this as

$$s_1 \oplus s_2 = s_3 \oplus s_4.$$

Converting the left and the right hand side of the equation yields

$$x_1 + x_2 - 2x_1x_2 - x_3 - x_4 + 2x_3x_4 = 0.$$

In this way we can convert our system of Boolean equations into a system of quadratic equations over the reals. In order to linearize these equations we introduce a new binary variable  $y_{ij}$  for each quadratic term  $x_i x_j$  and three following inequalities which force the variable to take the correct value:

$$\begin{aligned} y_{ij} &\leq x_i, \\ y_{ij} &\leq x_j, \\ x_i + x_j - 1 &\leq y_{ij}. \end{aligned}$$

This linearization yields a system of 732 equality and 2529 inequality constraints in 1575 variables.

**The mixed-0-1 programming problem** As mentioned before we use the equality and inequality constraints to describe the feasible set of our mixed-0-1 programming problem. But there is more required to formulate a mixed-0-1 programming problem. As the objective function we can choose an arbitrary linear function. Experimentally we confirmed that the sum over all variables is a good choice for the objective function. Furthermore we showed that restricting only the initial state variables to  $\{0, 1\}$

improves the time complexity of the attack. Restricting these variables to be binary will automatically force the remaining variables to take binary values.

The remaining problem is the uniqueness of the solution. We cannot assume that the Boolean system or the corresponding mixed-0-1 programming problem has a unique solution. But in the case of Bivium it is easy to generate an overdetermined system by generating more key stream bits and for an overdetermined system it is likely that the solution is unique. We found out that generating 1/3 additional key stream bits and corresponding equations is a good choice.

## Results

We used the above described mixed-0-1 programming problem to attack Bivium A and Bivium B as well as variants of both with smaller state sizes. As a solver for the mixed-0-1 programming problem we used the commercial optimization tool CPLEX [1].

We are able to break Bivium A in less than 4.5 hour on average. This shows us that our approach is faster than Raddum [6] (about a day) but unfortunately slower than using MiniSAT [3] (21 sec).

In the case of Bivium B we have to reduce the complexity of the problem by guessing bits. We can determine the initial state for Bivium B in  $2^{64.5}$  seconds (assuming that we run the tests in parallel). Raddum says in [6] that he can search through  $2^{24}$  keys in  $2^{10.7}$  seconds. That means that the complexity for our approach corresponds to searching through  $2^{77.8}$  keys.

## References

1. ILOG CPLEX. <http://www.ilog.com/products/cplex/>.
2. BEIGEL, R. The polynomial method in circuit complexity. *Structure in Complexity Theory Conference* (1993), 82–95.
3. CAMERON McDONALD, CHRIS CHARNES, J. P. An algebraic analysis of trivium ciphers based on the boolean satisfiability problem. Cryptology ePrint Archive, Report 2007/129, 2007. [urlhttp://eprint.iacr.org/](http://eprint.iacr.org/).
4. CANNIÈRE, C. D., AND PRENEEL, B. Trivium specifications. *eSTREAM, ECRYPT Stream Cipher Project 2006*.
5. CANNIÈRE, C. D., AND PRENEEL, B. Trivium - a stream cipher construction inspired by block cipher design principles. *estream, ecrypt stream cipher*. Tech. rep., of Lecture Notes in Computer Science, 2005.
6. RADDUM, H. Cryptanalytic results on trivium, 03 2006.