Numerical solvers and cryptanalysis

Mario Lamberger, Tomislav Nad and Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria.
Tomislav.Nad@iaik.tugraz.at

Abstract. In this paper we present an approach to apply numerical methods in the cryptanalysis of modern cryptographic algorithms. We focus on the stream cipher Trivium. It is a stream cipher recommended by the eStream project in the hardware category. We use numerical methods to attack a reduced version of Trivium – called Bivium A. We first set up a system of equations describing the internal state of the cipher and convert it into a system over the reals. Four different techniques for the conversion are discussed. At this point we are able to apply numerical methods. We choose the DIRECT algorithm by D. R. Jones *et al.* and the *Interior Reflective Newton Method* by Coleman and Li. Results, occurring problems in this approach and possible future research directions are discussed.

Keywords: numerical solver, numerical analysis, Boolean equations, Trivium, Bivium, stream ciphers, cryptanalysis, optimization

1 Introduction

In the early 1990's first Biham and Shamir [3], and later Matsui [14] published two general techniques to cryptanalyze symmetric cryptographic algorithms: differential and linear cryptanalysis. These techniques had been successfully used to break many existing ciphers. Since the upcoming of these cryptanalytic methods, new design strategies have been proposed to resist this kind of analysis. Nowadays, newly developed cipher algorithms are proposed with an analysis of the resistance against linear and differential cryptanalysis. This does not rule out that new techniques may break them.

Recently, many attacks on cryptographic algorithms using their representation as a system of equations have been published. These attacks mainly use algebraic methods (*e. g.* Groebner Bases as in [4]) or SAT solvers (*cf.* [11]). In this paper we propose a new approach to solve such systems using numerical analysis, which is a large and well studied field of research. Many efficient algorithms exist to solve linear and nonlinear systems of equations numerically. We apply our approach on a reduced variant of the stream cipher Trivium [17], which is recommended by the eStream project [10]. Currently, Trivium resists all known attack methods, but its simple structure suggests that it may be a good target for other methods of cryptanalysis.

In Figure 1 our approach is outlined. From the system of equations over $GF(2)$ we use conversion methods to create an equivalent system over the reals. At this point one can apply numerical methods. The computed solution can be converted back to $GF(2)$ (with restrictions) which results in a solution for the original system.

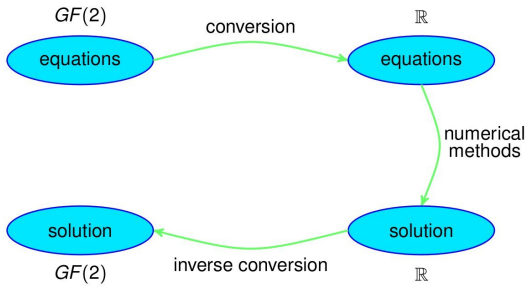

Fig. 1. Basic approach of *Numerical Cryptanalysis*

Notation: In this paper we define equations over $GF(2)$ and equations over the reals. To emphasize the difference, all variables and constants in equations over $GF(2)$ are capitalized and bold, *e. g.* $\mathbf{T}+\mathbf{S}$. Variables in equations over the reals are in lower case, *e. g.* $s+t-2\cdot s\cdot t$.

In Section 2 we give an overview of Trivium and its reduced variant Bivium A. Section 3 describes possibilities to convert a system to the real domain. A brief description of the chosen numerical solvers and important properties are given in Section 4. Finally, we present our experimental results for Bivium A in Section 5.

2 Trivium

A stream cipher is a symmetric key cipher, where the plaintext bits are combined with a pseudorandom bit stream (keystream). The keystream-bits are derived from a key and an initial value. Stream ciphers are in general fast and have limited or no error propagation, which makes them very useful for situations where transmission errors are highly probable or the plaintext is of unknowable length.

We are looking at Trivium, which was designed by Christophe De Cannière in 2005 [17,6]. Trivium is a synchronous stream cipher and generates up to 2^{64} bits of keystream from an 80-bit secret key (K) and an 80-bit initial value (IV). It consists of two phases: First the internal state of the cipher is initialized using the key and the IV, then the state is repeatedly updated and used to generate keystream-bits.

2.1 Initialization

The internal state is represented by a 288-bit register. In the initialization step an 80-bit key and an 80-bit initial vector are loaded into the register (see Equation 1). Then, the state is rotated over four full cycles (without generating any keystream), where one cycle is the execution of the algorithm 288 times.

$$\begin{aligned}(\mathbf{S_1}, \mathbf{S_2}, \cdots, \mathbf{S_{93}}) &\leftarrow (\mathbf{K_1}, \mathbf{K_2}, \cdots, \mathbf{K_{80}}, \mathbf{0}, \cdots, \mathbf{0}) \\ (\mathbf{S_{94}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{177}}) &\leftarrow (\mathbf{IV_1}, \mathbf{IV_2}, \cdots, \mathbf{IV_{80}}, \mathbf{0}, \cdots, \mathbf{0}) \\ (\mathbf{S_{178}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{288}}) &\leftarrow (\mathbf{0}, \cdots, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1})\end{aligned} \tag{1}$$

2.2 Keystream generation

The following loop is a full description of the keystream generation:

$$\begin{aligned}&\textbf{For i=1 to N do} \\ &\qquad \mathbf{T_1} \leftarrow \mathbf{S_{66}} + \mathbf{S_{93}} \\ &\qquad \mathbf{T_2} \leftarrow \mathbf{S_{162}} + \mathbf{S_{177}} \\ &\qquad \mathbf{T_3} \leftarrow \mathbf{S_{243}} + \mathbf{S_{288}} \\ &\qquad \mathbf{Z_i} \leftarrow \mathbf{T_1} + \mathbf{T_2} + \mathbf{T_3} \\ &\qquad \mathbf{T_1} \leftarrow \mathbf{T_1} + \mathbf{S_{91}} \cdot \mathbf{S_{92}} + \mathbf{S_{171}} \\ &\qquad \mathbf{T_2} \leftarrow \mathbf{T_2} + \mathbf{S_{175}} \cdot \mathbf{S_{176}} + \mathbf{S_{264}} \\ &\qquad \mathbf{T_3} \leftarrow \mathbf{T_3} + \mathbf{S_{286}} \cdot \mathbf{S_{287}} + \mathbf{S_{69}} \\ &(\mathbf{S_1}, \mathbf{S_2}, \cdots, \mathbf{S_{93}}) \leftarrow (\mathbf{T_3}, \mathbf{S_1}, \mathbf{S_2}, \cdots, \mathbf{S_{92}}) \\ &(\mathbf{S_{94}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{177}}) \leftarrow (\mathbf{T_1}, \mathbf{S_{94}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{176}}) \\ &(\mathbf{S_{178}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{288}}) \leftarrow (\mathbf{T_2}, \mathbf{S_{178}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{287}}) \\ &\textbf{end for}\end{aligned} \tag{2}$$

The algorithm computes one keystream-bit $\mathbf{Z_i}$ in each step up-to N bits.

2.3 Reduced variant

In order to demonstrate our approach we chose a reduced variant called Bivium A, which was introduced by Raddum in 2006 [16]. Bivium A is constructed by removing the third register of Trivium and adapting the equations. The internal state has a length of 177 bits. In the initialization phase the third step of (1) is omitted and the keystream generation algorithm is changed to:

$$
\begin{aligned}
&\textbf{For i=1 to N do}\\
&\qquad \mathbf{T_1} \leftarrow \mathbf{S_{66}} + \mathbf{S_{93}}\\
&\qquad \mathbf{T_2} \leftarrow \mathbf{S_{162}} + \mathbf{S_{177}}\\
&\qquad \mathbf{Z_i} \leftarrow \mathbf{T_2}\\
&\qquad \mathbf{T_1} \leftarrow \mathbf{T_1} + \mathbf{S_{91}} \cdot \mathbf{S_{92}} + \mathbf{S_{171}}\\
&\qquad \mathbf{T_2} \leftarrow \mathbf{T_2} + \mathbf{S_{175}} \cdot \mathbf{S_{176}} + \mathbf{S_{69}}\\
&(\mathbf{S_1}, \mathbf{S_2}, \cdots, \mathbf{S_{93}}) \leftarrow (\mathbf{T_2}, \mathbf{S_1}, \mathbf{S_2}, \cdots, \mathbf{S_{92}})\\
&(\mathbf{S_{94}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{177}}) \leftarrow (\mathbf{T_1}, \mathbf{S_{94}}, \mathbf{S_{95}}, \cdots, \mathbf{S_{176}})\\
&\textbf{end for}
\end{aligned}
\tag{3}
$$

Another reduced variant is called Bivium B, which is the same as Bivium A except the keystream-bit computation, which is changed to $\mathbf{Z_i} \leftarrow \mathbf{T_1} + \mathbf{T_2}$.

2.4 Earlier work on Trivium

In a known-plaintext scenario, where the plaintext and corresponding ciphertext is known to the adversary, one can compute the keystream. Using this information the adversary tries to recover the initial key. If the initial key is successfully recovered, one can reproduce any sequence of keystream-bits. This should be difficult for a strong cipher.

Several papers have been proposed about cryptanalytic results on Trivium. Khazaei and Hassanzadeh [13] showed that Trivium is strong against the linear sequential circuit approximation attack in spite of the extra simplicity of its output function and next-state function. Turan and Kara [19] define the initialization step of Trivium as an 8-round function and try to attack the initialization with a smaller number of rounds. Maximov and Biryukov [15] developed two attacks on Trivium with decreased complexity compared to the work of Raddum [16]. Raddum developed a new technique to solve systems of equations and applied these to the equation system representing Trivium and the reduced variants. He successfully broke Bivium A and B, but with high complexity for the second variant. The full version of Trivium resists his attack. McDonald *et al.* attacked Bivium with MiniSat in [5], by transforming the equations into a satisfiability problem. They estimate the complexity for the attack on Bivium B to be about 2^{52} operations. This algebraic attack recovers the private key after observing only 1770 bits of keystream. Bivium A is completely broken requiring only 177 bits of the keystream. Eibach *et al.* [11] attacked also Bivium B with SAT solvers. Fischer *et al.* [1] can successfully recover the initial key by using statistical distinguishers with complexity about 2^{55} if the number of iterations in the initialization step is reduced down to 672. Dinur and Shamir [9] proposed a new method in cryptanalysis, called the cube attack. Using this technique, they can successfully recover the initial key with complexity about 2^{45} if the iterations in the initialization step are reduced down to 767.

2.5 System of equations representing Bivium A

For our approach we need a representation of the cipher as a system of equations and use a known-plaintext attack to compute the keystream-bits. In the same way as in [16] the system can be derived from the keystream generation algorithm (3). We denote the register bits $\mathbf{S_1}, \cdots, \mathbf{S_{177}}$ as the variables of the system. In each step we get three equations. One for the keystream-bit (4) and two for the register feedback (5).

$$\mathbf{S_{162}} + \mathbf{S_{177}} = \mathbf{Z_i} \tag{4}$$

To keep the system sparse and to avoid long equations, new variables are introduced instead of replacing the internal state with equations.

$$\begin{aligned} \mathbf{S_{162}} + \mathbf{S_{177}} + \mathbf{S_{175}} \cdot \mathbf{S_{176}} + \mathbf{S_{69}} &= \mathbf{S_{178}} \\ \mathbf{S_{66}} + \mathbf{S_{93}} + \mathbf{S_{91}} \cdot \mathbf{S_{92}} + \mathbf{S_{171}} &= \mathbf{S_{179}} \end{aligned} \tag{5}$$

To get a fully determined system, we need to clock Bivium A 177 times, which is equivalent to producing 177 keystream-bits. The last 132 equations can be dropped since the introduced variables are not used in any keystream-bit equation [16]. Additionally, 72 more equations and variables can be dropped since only $\mathbf{T_2}$ is used for the keystream-bit computation. In total we get 327 equations in 327 variables over $GF(2)$. The variables represent the internal state bits. To get a fully determined system we need to know the keystream-bits $\mathbf{Z_i}$. Finding a solution of the system is equivalent to reconstruct the internal state. Once the state is known one can clock the cipher backwards to reconstruct the secret key. Note, that it is possible to find more than one solution by solving the system, which does not lead to the correct key. However, in the attack we assume that for a strong cipher the amount of solutions is rather limited. To overcome this problem, one can generate more keystream-bits to get an overdetermined system of equations. Nevertheless, in this article we focus on solving such systems and therefore we do not pay more attention on the existence of more solutions in the Boolean domain.

3 Conversion methods

In our approach we try to solve a system over $GF(2)$ by using numerical methods. Since they operate on the real domain we convert the system into a system over the reals using different conversion techniques. Let S_G be the system over $GF(2)$ and S_R the system over the reals. A conversion method should at least guarantee that a solution for S_G results in a solution for S_R, after it is converted to the reals. Such a conversion can be done in several ways. We show four different possibilities, which should emphasize how much influence one has on the structure of the system over the reals, at this point. As a start one can represent a polynomial over $GF(2)$ as a polynomial over the reals. In [2] an overview of

possible representations is listed. Basically, the representation as a polynomial over the reals depends on the mapping of

$$t : GF(2) = \{\mathbf{0}, \mathbf{1}\} \rightarrow \{\mathbf{a}, \mathbf{b}\} \subset \mathbb{R}.$$

For the more "natural" choices of a and b, the following conversion rules for addition and multiplication can be derived:

– Standard representation

$$t : GF(2) = \{\mathbf{0}, \mathbf{1}\} \rightarrow \{0, 1\} \subset \mathbb{R}.$$

$$\begin{aligned} \mathbf{X_1} \cdot \mathbf{X_2} &\Longrightarrow x_1 \cdot x_2 \\ \mathbf{X_1} + \mathbf{X_2} &\Longrightarrow x_1 + x_2 - 2 \cdot x_1 \cdot x_2 \end{aligned}$$

– Fourier representation

$$t : GF(2) = \{\mathbf{0}, \mathbf{1}\} \rightarrow \{1, -1\} \subset \mathbb{R}.$$

$$\begin{aligned} \mathbf{X_1} \cdot \mathbf{X_2} &\Longrightarrow \frac{1}{2}(1 + x_1 + x_2 - x_1 \cdot x_2) \\ \mathbf{X_1} + \mathbf{X_2} &\Longrightarrow x_1 \cdot x_2 \end{aligned}$$

We want to note that since $GF(2)$ and the Boolean domain are equivalent, each Boolean operator and equation can be represented in a similar way.

In [2] also the dual and sign representations are mentioned. The sign representation is mostly the same as the Fourier representation. The dual representation results in a large increase of monomials in the polynomial over the reals, which is not very useful in our case.

Using the Fourier or standard representations the nonlinear equations of Bivium A result in degree 6 equations and the linear equations in degree 2 equations over the reals. The number of monomials is different and depends on the number of additions over $GF(2)$. One benefit of these conversions is the multilinearity of the resulting polynomials, *i. e.* the highest degree in one variable which can occur is 1. The Fourier representation has an additional benefit since variables can cancel out during the conversion, because $x^2 = 1$ holds. This effect can happen if one variable occurs in different monomials in the same equation.

In general we can say that if there are more additions than multiplications in the polynomial over $GF(2)$ then the Fourier conversion is the better choice. If the multiplication dominates, the standard conversion results in simpler equations. But the main problem of these conversion methods is the high increase of the degree and that linear equations over $GF(2)$ are nonlinear over the reals. Systems over the reals with high degrees make the problem harder for numerical methods. Additionally, we loose the benefit that part of the system is linear. Thus, different conversion methods are needed.

3.1 Adapted Standard Conversion

One can see that the system over $GF(2)$ for Bivium A consists of sparse equations with low degrees. The main idea behind the conversion method described in this section is to keep the structure of the equation when converting them to the real domain. This should result in short equations over the reals with low degrees. Our first approach to fulfill these requirements is called *Adapted Standard Conversion* (ASC) which maps $\{\mathbf{0}, \mathbf{1}\}$ to $\{0, 1\}$ but converts the equation as a whole, instead of each operation like in the representations mentioned above. As a start we divide the equations of Bivium A into two types. *Type I* covers the linear equations, like

$$\mathbf{S_{162}} + \mathbf{S_{177}} = \mathbf{Z_1}.$$

The nonlinear equations of the system, like

$$\mathbf{S_{162}} + \mathbf{S_{177}} + \mathbf{S_{175}} \cdot \mathbf{S_{176}} + \mathbf{S_{69}} + \mathbf{S_{178}} = \mathbf{0},$$

are covered by *type II*.

Conversion of type I As an example we convert the equations resulting from the first iteration of the keystream algorithm:

$$\mathbf{S_{66}} + \mathbf{S_{93}} = \mathbf{Z_1} = \begin{cases} s_{66} - s_{93} = 0 & \text{if } \mathbf{Z_i} = \mathbf{0} \\ s_{66} + s_{93} - 1 = 0 & \text{if } \mathbf{Z_i} = \mathbf{1}. \end{cases}$$

Note that $\mathbf{Z_i}$ is known to the adversary. The resulting equation over the reals is still linear and the truthtable using $\{0, 1\}^2$ is equivalent to the truthtable for the original equation in $\{\mathbf{0}, \mathbf{1}\}^2$.

Conversion of type II Let us consider the first type II equation of the first iteration:

$$\mathbf{S_{162}} + \mathbf{S_{177}} + \mathbf{S_{175}} \cdot \mathbf{S_{176}} + \mathbf{S_{69}} + \mathbf{S_{178}} = \mathbf{0} \tag{6}$$

Equations of this type have RHS (right hand side) equal to $\mathbf{0}$. We try now to keep the structure of the equation in the real domain. This is done by computing the truthtable of equation (6) and evaluate it over $GF(2)$. Then addition and multiplication over $GF(2)$ become the same operations in $\mathbb{R}$ and we evaluate the resulting equation over the reals.

Now only rows where the result over $GF(2)$ is equal to $\mathbf{0}$ are considered. The possible real values for this case are $0, 2$ and 4. The converted equations over the reals have to cover all three possibilities. This is done by creating three real equations, which have the same structure as the original one, subtracted by the values for the three cases and multiplied by new variables. A fourth equation is added, ensuring that only one of the new variables should be equal to 1. The result is shown in equation (7).

$\mathbf{S_{162}}$	$\mathbf{S_{177}}$	$\mathbf{S_{175}}$	$\mathbf{S_{176}}$	$\mathbf{S_{69}}$	$\mathbf{S_{178}}$	over $GF(2)$	over $\mathbb{R}$
0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	1
0	0	0	1	1	0	1	1
0	0	0	1	1	1	0	2
0	0	1	0	0	0	1	1
0	0	1	0	0	1	0	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	0	0	1	0	4
1	1	1	0	1	0	1	3
1	1	1	0	1	1	0	4
1	1	1	1	0	0	1	3
1	1	1	1	0	1	0	4
1	1	1	1	1	0	0	4
1	1	1	1	1	1	1	5

Table 1. Evaluation of type II eqs. over the reals and $GF(2)$.

$$\begin{aligned} e_1 \cdot (s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{69} + s_{178}) &= 0 \\ e_2 \cdot (s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{69} + s_{178} - 2) &= 0 \\ e_3 \cdot (s_{162} + s_{177} + s_{175} \cdot s_{176} + s_{69} + s_{178} - 4) &= 0 \\ e_1 + e_2 + e_3 &= 1 \end{aligned} \tag{7}$$

This is done for every type II equation of Bivium A. Using ASC we get 777 equations and variables. The degree of converted type II equations is 3.

3.2 Splitting

Reducing the degree of the equations can also be done in a different way. Let us reconsider equation (6):

$$\mathbf{S_{66} + S_{93} + S_{171} + S_{91} \cdot S_{92} + S_{179} = 0}$$

We simplify it by splitting it up into three equations:

$$\begin{aligned} \mathbf{S_{91} \cdot S_{92}} &= \mathbf{X_1} \\ \mathbf{S_{66} + S_{93}} &= \mathbf{X_1 + X_2} \\ \mathbf{S_{171} + S_{179}} &= \mathbf{X_2} \end{aligned} \tag{8}$$

Now we apply the standard conversion on each side. This results in three equations (9), where the highest degree is two.

$$\begin{aligned} s_{91}s_{92} - x_1 &= 0 \\ s_{66} + s_{93} - 2s_{66}s_{93} - x_1 - x_2 + 2x_1x_2 &= 0 \\ s_{171} + s_{179} - 2s_{171}s_{179} - x_2 &= 0 \end{aligned} \tag{9}$$

For the whole system of Bivium A we get 627 equations and variables over the reals.

4 Numerical solvers

Numerical solvers are methods to approximate solutions for equations and systems of equations. We are interested in the special case of solving nonlinear polynomial systems of equations. For this problem iterative methods exist where an initial starting value is needed. We do not describe the used methods in detail, but we explain the reasons for the choice of the specific methods. Two general terms in numerical analysis are important to discuss different methods.

The term *locally convergent* refers to the situation that sufficiently good initial guesses (starting values) of the solution are assumed for convergence to the solution. Finding such a starting point is not trivial, especially if the dimension of the problem is high. In our case we can restrict the possible points to the hypercube $[0,1]^n$ (or $[-1,1]^n$ for the Fourier representation), where n is the dimension (number of variables). All possible restrictions to the initial guess for our problem are not sufficient for local convergence as mentioned in Section 4.4.

Globally convergent means that a rather general initial guess can be used for convergence. Efficient iterative methods should be able to cope with bad initial guesses. They represent a large and difficult topic in numerical analysis. Since we cannot provide an initial guess which is good enough for our problem, we focus on globally convergent methods.

In general there are two possibilities a numerical method can handle a problem. Either it is directly applied on a system of equations or the system is transformed to an optimization problem.

4.1 System of equations over the reals

A system of equations is defined as follows:

$$F = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} = 0, \text{ where } x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ and } f_i : \mathbb{R}^n \to \mathbb{R}. \tag{10}$$

The classical method to solve such a system is the well known Newton-Raphson method [8], which needs several properties of the system to be able to converge

to the solution. The method without extensions is locally convergent. Other Newton based methods include techniques for global convergence with specific prerequisites (*cf.* [8]).

Since first order information is used, the term *Jacobian matrix* of a system is important. It is defined as

$$F'(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

4.2 Optimization model

A system of equations can be modeled as a least square problem with upper and lower bounds. If we denote the equations for Bivium A as a vector function $F : \mathbb{R}^n \to \mathbb{R}^n$ then the optimization model is given by

$$\begin{aligned} \min \tfrac{1}{2}||F(s)||_2^2 \\ l_i \leq s_i \leq u_i \ (i = 1, \cdots, n) \end{aligned} \tag{11}$$

where $s, l, u \in \mathbb{R}^n$. l and u are the lower and upper bounds of s. In the case of the standard representation $l = (0, \cdots, 0)^t$ and $u = (1, \cdots, 1)^t$. Such a structure is called box-bounded optimization problem. Using this model for our approach has some advantages. Beside the fact that we can define upper and lower bounds for our problem we can add more information through additional constraints. Most modern and efficient algorithms for solving a system of equations transform the system in a least square problem.

When dealing with minimization problems, two terms are important: *local optimum* and *global optimum*. The global optimum in this case is 0, since in the solution all equations are equal to zero and so is their norm. A local optimum is any other static point. Global and local convergence in optimization problems have a slightly different meaning. Globally convergent optimization methods do not guarantee to find the global optimum, but it is ensured that for a general initial guess the algorithm converges at least to a local optimum.

4.3 General difficulties

Many different problems can occur when solving a system of equations numerically. Often the structure of the Jacobian matrix, which is used during the iterative computations, is important. Basic properties like regularity in the search area of these matrices are important. Singularities in the solution points or in an area near the solutions make the problem harder. Special algorithms use different strategies to overcome such problems.

Another essential point is determining a good *initial guess*, which depends on the amount and quality of available information. Only globally convergent methods are capable of using an arbitrary starting point (under restrictions). But

all iterative solvers have in common that the starting point has a high influence on the convergence. In our case we know in which region of the real domain the solution exists, which is $[0,1]^n$ (or $[-1,1]^n$ for Fourier representation), so we can restrict the search area to this subdomain. By guessing bits, which is a common approach in cryptanalysis, we can shrink the search area even more. Guessing bits means that an exhaustive search on a specific number of bits is done. First the correct values are guessed. Then the attack is applied. If the attack succeeds, the guess was correct, otherwise the attack with a new guess is applied again.

With the increase of dimension both restrictions to the search area lose in value. The difficulty of the start value determination with increasing dimension is often called "curse of dimensionality". If the available information is not good enough to provide good starting points the best one can do is to choose randomly values normally distributed in the search domain.

As a third point the *existence* and *uniqueness* of a solution is important. From the mentioned conversion methods it follows that if the system over $GF(2)$ has a solution, the converted system has at least one solution. Uniqueness of a solution for the real system cannot be guaranteed. In fact real-valued solutions may exist. Currently, we did not find any correlation between these solutions and the system over $GF(2)$. Finally, it is worth mentioning that each equation is continuously differentiable since the conversion result is always a polynomial.

4.4 Chosen methods

To be able to choose an appropriate solver we have to reconsider the known facts about our system. Our first experiments with locally convergent methods showed that we cannot provide a good starting point for these methods. This is why we focused on globally convergent methods. In the solution the Jacobian matrix associated to the system of Bivium A is regular, but in the hypercube a lot of singularities exists, which makes the system more unstable. Additionally, the Jacobian matrix is ill-conditioned. The high degree of the equations (in the case of Fourier and standard representation) and the dimension of the systems increase the difficulty of the problem.

Currently, a huge pool of different algorithms and strategies are available. There are general methods and specialized methods for specific problems. Since this is our first approach on applying numerical methods on Boolean systems, we decided to choose algorithms which are well known and used in commercial software packages. We tested different solvers and decided to focus on the *Interior Reflective Newton Method* and the *DIRECT algorithm*.

Interior Reflective Newton Method This method was invented by Coleman and Li in 1993 [7]. It is well studied and efficient implementations in mathematical software like Matlab™ are available. Another advantage of this method is the detailed convergence analysis [18], which is very valuable for a better understanding of the algorithm.

In the *Interior Reflective Newton Method* all iterates stay between the bounds. Since we know the exact bounds, this property is very useful. On the one hand it makes the analysis of the system easier, since we only have to consider the hypercube, and on the other hand it is ensured that bad properties outside of the cube do not influence the algorithm's convergence. The method has good global convergence properties but may get stuck at local minima.

DIRECT algorithm Apart from global convergence we also need global optimization techniques to find a solution for the system. Therefore, we used the DIRECT (DIviding RECTangles) algorithm by Jones *et al.* [12]. It is a deterministic sampling method designed for finding the global minima for bound constrained optimization problems and does not need a starting point.

5 Experiments for Bivium A

For Bivium A we constructed four different systems over the reals (one for each conversion method) and investigated two numerical methods. We ran experiments with different starting point strategies for the *Interior Reflective Newton Method* and different guessing bits approaches for both.

5.1 Starting points

As mentioned in Section 4, the starting point for iterative methods has a high influence on the behavior. We used different strategies for a good initial guess.

- Random in $\{0,1\}^n$, $\{-1,1\}^n$
- Random in $[0,1]^n$, $[-1,1]^n$
- Part of the solution used in the starting point

Because of the large dimension and the results of locally convergent methods, we started with random points. We have two types of randomness. First points in $\{0,1\}^n$ (or $\{-1,1\}^n$ for Fourier representation) are chosen. Secondly, we set the starting point to a random point in $[0,1]^n$ (or $[-1,1]^n$). Additionally, we used part of the precomputed solution in the initial guess, to see how "close" the guess has to be for convergence to the solution.

5.2 Guessing bits

Guess and determine attacks are commonly used in cryptanalysis. We can do the same in our approach with a slightly different purpose. By guessing a few bits, which for our experiments means that we replace some variables with their correct values, we simplify the system before the conversion. By this reduction of the dimension, we also simplify the problem in the real domain. To get a good reduction with as few as possible guessed values, we have to choose the bits wisely. To achieve this goal we compute for each variable its distribution in

the system, *i. e.* in how many equations a specific variable occurs. We see for example that variables from $\mathbf{S_{94}}$ up to $\mathbf{S_{162}}$ occur in 7 equations. If we guess such a variable we can simplify 7 equations in one step.

The second usage of guessing bits is to determine a good starting point as mentioned above. In that case the system is not simplified but better starting information for a numerical method is provided. Since during the system creation new variables are introduced, more variables can be guessed than the number of the internal state bits.

5.3 Results

We ran about 1000 experiments for each starting point strategy. Table 2 lists the results for different conversion methods and starting point strategies for the *Interior Reflective Newton Method*.

conversion	starting point	convergence	objective function value
Fourier	random	local optimum	≈ 95.9189
	$c(75\%)$	solution found	0
Standard	random	local optimum	≈ 21.7087
	$c(75\%)$	solution found	0
ASC	random	local optimum	≈ 0.3366
	$c(75\%)$	solution found	0
Splitting	random	local optimum	≈ 9.2417
	$c(75\%)$	solution found	0

Table 2. Results for Bivium A using the *Interior Reflective Newton Method* and four different conversion methods. Starting points are random in $[0,1]^n$ (or $[-1,1]^n$). $c(x\%)$ are starting points where $x\%$ of the variables are guessed. The last column includes the average objective function value in the converged points.

For random starting points the *Interior Reflective Newton Method* achieved global convergence to local optima, which are not solutions of the system. All local optima are in the near of a corner of the hypercube where a large part of the values are either 0 or 1 (or -1, 1 for the Fourier representation). If we use part of the precomputed solution in the starting strategy we need at least 75% of all variables in the system to achieve convergence to the solution. Even if we assign the correct values for all variables representing the internal state bits, the algorithm finds only local optima. That means the objective function (11) has a lot of stationary points beside the one we are looking for.

The DIRECT algorithm does not need an initial guess since it is an deterministic algorithm. In our experiments the algorithm did not converge to any point, so we stopped the algorithm after three days. Other methods with global optimization techniques may be more successful.

In Table 2 we also present the values of the objective function. We see that we get closer to 0 for the systems with low degrees, even if they have more

equations and variables. For ASC the numerical method can reach a much lower objective function value, but does not find a solution. Other numerical methods may deliver better results.

For the simplified systems, where we replaced the guessed variables with their correct values before the conversion, we get similar results.

6 Conclusions and further work

Basically, every cryptographic algorithm can be described as a system of equations. In this paper we present an approach to use numerical methods to solve such systems. We demonstrate it on a reduced variant of the stream cipher Trivium, called Bivium A. The first step in our approach is to create an equivalent system over the reals, which can be done in several ways. Either by using one of the common known representations of Boolean equations over the reals or by exploiting the structure of the system. Since the degree of each equation in the system over the reals is a property which has high influence on how difficult it is to solve the system, we describe two conversion methods which keep the degree low. Beside the conversion methods in this paper, other techniques are possible. It is important to notice that it is unusual and of great benefit from a numerical point of view that one has such a high influence on the structure of the system over the reals.

For the numerical part of our approach we chose two more general algorithms, after we excluded classical methods like Newton-Raphson. Two terms were important here: global convergence and global optimum. Both in combination represent a difficult problem in numerical and optimization research, respectively. Currently, we are not able to solve the system of Bivium A with the chosen methods, without providing much information, but we are still at the beginning of our research and there is a lot of available knowledge in the field of numerical analysis and optimization research. A lot of different strategies on the algorithm level and on the problem modeling level exist. It is possible to create a mixed integer nonlinear and even linear problem out of the system over $GF(2)$ or using different objective functions and add equations as constraints. At the moment we also did not modify a solver algorithm to exploit the specific structure of our system, which is often done in applied mathematics to solve specific problems numerically. This paper should mark a first step and offers several future research directions.

Acknowledgments

The work in this paper has been supported by the Austrian Science Fund (FWF), project P19863.

References

1. 0002, S.F., Khazaei, S., Meier, W.: Chosen iv statistical analysis for key recovery attacks on stream ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT. Lecture Notes in Computer Science, vol. 5023, pp. 236–245. Springer (2008)
2. Beigel, R.: The Polynomial Method in Circuit Complexity. In: Structure in Complexity Theory Conference. pp. 82–95 (1993), citeseer.ist.psu.edu/beigel95polynomial.html
3. Biham, E., Shamir, A.: Differential cryptanalysis of the full 16-round des. In: Brickell, E.F. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 740, pp. 487–496. Springer (1992), http://link.springer.de/link/service/series/0558/bibs/0740/07400487.htm
4. Buchmann, J., Pyshkin, A., Weinmann, R.P.: Block Ciphers Sensitive to Gröbner Basis Attacks. In: Pointcheval, D. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 3860, pp. 313–331. Springer (2006), http://dx.doi.org/10.1007/11605805_20
5. Cameron McDonald, Chris Charnes and Josef Pieprzyk: An algebraic analysis of Trivium ciphers based on the boolean satisfiability problem. In: International Conference on Boolean Functions: Cryptography and Applications, BFCA (2008)
6. Cannière, C.D.: Trivium: A stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. Lecture Notes in Computer Science, vol. 4176, pp. 171–186. Springer (2006)
7. Coleman, T.F., Li, Y.: An interior trust region approach for nonlinear minimization subject to bounds. Tech. rep., Department of Computer Science, Cornell University (1993)
8. Deuflhard, P.: Newton methods for nonlinear problems, Springer Series in Computational Mathematics, vol. 35. Springer-Verlag, Berlin (2004), affine invariance and adaptive algorithms
9. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 5479, pp. 278–299. Springer (2009)
10. ECRYPT: The estream project. http://www.ecrypt.eu.org/stream/
11. Eibach, T., Pilz, E., Völkel, G.: Attacking bivium using sat solvers. In: Büning, H.K., Zhao, X. (eds.) SAT. Lecture Notes in Computer Science, vol. 4996, pp. 63–76. Springer (2008), http://dx.doi.org/10.1007/978-3-540-79719-7_7
12. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. 79(1), 157–181 (1993)
13. Khazaei, S., Hassanzadeh, M.: Linear Sequential Circuit Approximation of the TRIVIUM Stream Cipher. eSTREAM submitted papers (2005)
14. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of feal cipher. In: EUROCRYPT. pp. 81–91 (1992), http://link.springer.de/link/service/series/0558/bibs/0658/06580081.htm
15. Maximov, A., Biryukov, A.: Two Trivial Attacks on Trivium. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography. LNCS, vol. 4876, pp. 36–55. Springer (2007), http://dx.doi.org/10.1007/978-3-540-77360-3_3
16. Raddum, H.: Cryptanalytic results on trivium. eSTREAM submitted papers (2007), http://www.ecrypt.eu.org/stream/papersdir/2006/039.ps
17. Robshaw, M.: New Stream Cipher Designs: The eSTREAM Finalists. Springer-Verlag, Berlin, Heidelberg (2008)

18. Thomas F. Coleman and Yuying Li: On the convergence of interior-reflective newton methods for nonlinear minimization subject to bounds. Math. Program. 67, 189–224 (1994)
19. Turan, M.S., Kara, O.: Linear approximations for 2-round trivium. In: Proc. First International Conference on Security of Information and Networks (SIN 2007). pp. 96–105. Trafford Publishing (2007)