Hill Climbing Algorithms and Trivium

Julia Borghoff[1], Lars R. Knudsen[1], and Krystian Matusiewicz[2]

[1] Department of Mathematics, Technical University of Denmark
{J.Borghoff,Lars.R.Knudsen}@mat.dtu.dk
[2] Institute of Mathematics and Computer Science, Wroclaw University of Technology
Krystian.Matusiewicz@pwr.wroc.pl

Abstract. This paper proposes a new method to solve certain classes of systems of multivariate equations over the binary field and its cryptanalytical applications. We show how heuristic optimization methods such as hill climbing algorithms can be relevant to solving systems of multivariate equations. A characteristic of equation systems that may be efficiently solvable by the means of such algorithms is provided. As an example, we investigate equation systems induced by the problem of recovering the internal state of the stream cipher Trivium. We propose an improved variant of the simulated annealing method that seems to be well-suited for this type of system and provide some experimental results.

Keywords: simulated annealing, cryptanalysis, Trivium.

1 Introduction

Cryptanalysis focuses on efficient ways of exploiting, perhaps unexpected, structure of cryptographic problems. It could be a difference which propagates with a high probability through the cipher as used in differential cryptanalysis [6,2] or a linear approximation of the non-linear parts of a cipher that holds for many of the possible inputs as is the case in linear cryptanalysis [20]. More recently, the so-called algebraic attacks have received much attention. They exploit the fact that many cryptographic primitives can be described by sparse multivariate non-linear equations over the binary field in such a way that solving these equations recovers the secret key or the initial state in the case of stream ciphers. In general, solving random systems of multivariate non-linear Boolean equations is an NP-hard problem [12]. However, when the system has a specific structure, we can hope that more efficient methods may exist.

One technique to tackle such equation systems is linearisation, where each non-linear term is replaced by an independent linear variable. It works only if there are enough linear independent equations in the resulting system. The XL algorithm [7] increases the number of equations by multiplying them with all monomials of a certain degree. It has been refined to the XSL algorithm [9], which, when applied to the AES, exploits the special structure of the equation

A. Biryukov, G. Gong, and D.R. Stinson (Eds.): SAC 2010, LNCS 6544, pp. 57–73, 2011.

system. Neither the XL nor the XSL algorithm have been able to break AES but algebraic attacks were successful in breaking a number of stream cipher designs [1,8].

In this paper we also investigate systems of sparse multivariate equations. The important additional requirement we make is that each variable appears only in a very limited number of equations. The equation system generated by the keystream generation algorithm of the stream cipher Trivium [10] satisfies those properties and will be examined in this paper as our main example. The fully determined Trivium system consists of 954 equations in 954 variables. Solving this system allows us to recover the 288-bit initial state.

Our approach considers the problem of finding a solution for the system as an optimization problem and then applies an improved variant of simulated annealing to it. As opposed to the XL and XSL algorithms, the simulated annealing algorithm does not increase the size of the problem, it does not generate more nor change the existing equations. The only additional requirement is an objective function, called the cost function, that should be minimized.

Simulated annealing has been studied in the context of cryptography before. An attack on an identification scheme based on the permuted perceptron problem (PPP) [19] was presented. An appropriate cost function was found which made it possible to solve the simpler perceptron problem as well as the PPP using a simulated annealing search. The attack showed that the recommended smallest parameters for the identification scheme are not secure. The same identification scheme was later a subject to an improved attack [5]. Simulated annealing was used to solve a related problem that had solutions highly correlated with the solution of the actual problem. Furthermore, timing analysis was applied where the search process is monitored and one can observe that some variables are stuck at correct values at an early state and never change again.

With the current experiments, we are not able to break Trivium in the cryptographic sense which means with a complexity equivalent to less than 2^{80} key setups and the true complexity of our method against Trivium is unknown. However, if one considers the Trivium system purely as a multivariate quadratic Boolean system in 954 variables this system can be solved significantly faster than exhaustive search, namely by around 2^{210} bit flips which is roughly equivalent to 2^{203} evaluations of the system. This shows that this variant of simulated annealing seems to be a promising tool for solving non-linear Boolean equation systems with certain properties.

2 Hill Climbing Algorithms

Hill climbing algorithms are a general class of heuristic optimization algorithms that deal with the following optimization problem. There is a finite set X of possible configurations. Each configuration is assigned a non-negative, real number called cost, or, in other words, a cost function is defined as $f : X \to \mathbb{R}$. For each configuration $x \in X$ a set of neighbours $\eta(x) \subset X$ is defined. The aim of the search is to find $x_{min} \in X$ minimizing the cost function $f(x)$,

$f(x_{min}) = \min\{f(x) : x \in X\}$, by moving from neighbour to neighbour depending on the cost difference between the neighbouring configurations.

Johnson and Jacobsen [15] presented a unified view of many hill climbing algorithms by describing conditions on accepting a move from one configuration to another. The transition probability $p_k(x, y)$ of accepting a move from x to $y \in \eta(x)$ is defined as the product of a configuration generation probability $g_i(x, y)$ and a configuration acceptance probability $\Pr[R_k(x, y) \geq f(y) - f(x)]$, where $R_k(x, y)$ is a random variable and k is an iteration index that is increased by one after a fixed number of moves. Algorithm 1 presents a general form of a hill climbing algorithm.

Algorithm 1. General formulation of hill climbing algorithms

```
x_best ← x
while stopping criterion not met do
    k ← 0                                          ▷ set the outer loop counter
    while k < K do
        for m = 0, ..., M − 1 do
            generate a neighbour y ∈ η(x) with probability g_k(x, y)
            compute the cost function f(y) of the candidate
            if R_k(x, y) ≥ f(y) − f(x) then
                x ← y                                          ▷ accept the move
                if f(x) < f(x_best) then
                    x_best ← x                  ▷ store the best configuration
                end if
            end if
        end for
        k ← k + 1
    end while
end while
```

Note that when $R_k(x, y) = 0$, we obtain a local search algorithm as only moves that decrease the cost are accepted.

Simulated Annealing. The classical simulated annealing algorithm [18] is a special case of the general hill climbing algorithm presented above with a particular definition of the transition probability.

The simulated annealing algorithm uses a key parameter called the temperature t. The configuration generation probability is taken to be uniform, i.e., each neighbour is equally likely to be picked from each state. The acceptance probability depends on the difference $f(y) - f(x)$ in the cost function between the current state x and the selected neighbour y and the current temperature t_k. The move is always accepted when it decreases the cost and with probability $e^{-(f(y)-f(x))/t_k}$ when the cost increases. In terms of the general formulation presented above, we get this behaviour when we define $R_k(x, y) = -t_k \ln(U)$, where U is a uniform random variable on $[0, 1]$.

Note that when the temperature t_k is high, many cost-increasing moves are accepted. When the temperature is lower, worsening moves are less and less likely to be accepted.

The way the "temperature" t_k of the system decreases over time (k) is called the cooling schedule. The condition necessary for the global convergence of the method is that $t_k \geq 0$ and $\lim_{k\to\infty} t_k = 0$. In practice, two most commonly used cooling schedules are the exponential cooling schedule $t_k = \alpha \cdot \beta^k$ for some parameter $0 < \beta < 1$ and the logarithmic cooling schedule $t_k = \alpha / \log_2(k+1)$ proposed in [13], where α is a constant corresponding to the starting temperature.

3 Trivium System as an Optimization Problem

Trivium [10] is an extremely simple and elegant stream cipher that was submitted to the ECRYPT eStream project. It successfully withstood significant cryptanalytical attention [21,22,23,24,3] and became part of the portfolio of the eStream finalists.

To our knowledge, there is no attack on Trivium faster than the exhaustive key search so far. However, several attacks have been proposed which are faster than the naive guess-and-determine attack with complexity 2^{195} which was considered by the designers [10]. A more intelligent guess-and-determine attack with complexity 2^{135} using a reformulation of Trivium has been sketched in [17]. Furthermore, Maximov and Biryukov [21] described an attack with complexity $2^{85.5}$ and Raddum proposed a new algorithm for solving non-linear Boolean equations and applied it to Trivium in [23]. The attack complexity was 2^{164}. McDonald at al [22] considered Trivium as a Boolean satisfiability problem and used the SAT-solver MiniSAT in order to solve it. This approach was faster than exhaustive key search for small scale variants of Trivium called Bivium. However, for the full Trivium the estimated complexity is about $2^{159.9}$ seconds and is therefore worse than exhaustive search.

Trivium has an 80-bit key, an 80-bit IV and 288 bits of the internal state $(s_1, \ldots, s_{288})$. At each clock cycle it updates only three bits of the state and produces one bit of the keystream using the following procedure.

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

During the key setup phase, the key is loaded into the first 80 bits of the state, followed by 13 zero bits, then the IV is loaded into the next 80 bits of the state

and the remaining bits are filled with constant values. Then $4 \cdot 288$ clockings are computed without producing any keystream bits. Our results do not depend on this procedure.

The initial state which is the state of the registers at the time when the key generation starts can be expressed as system of sparse linear and quadratic Boolean equations [23]. 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 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

$$\begin{aligned} s_{66} \oplus s_{93} \oplus s_{91} \cdot s_{92} \oplus s_{171} &= s_{289} \\ s_{162} \oplus s_{177} \oplus s_{175} \cdot s_{176} \oplus s_{264} &= s_{290} \\ s_{243} \oplus s_{288} \oplus s_{286} \cdot s_{287} \oplus s_{69} &= s_{291} \\ s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288} &= z \end{aligned} \tag{1}$$

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 [23]. We only need to consider 954 equations and unknowns instead of $4 \cdot 288$ 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 system. We know that the initial state together with the corresponding updated state bits satisfies all the generated equations (1). On the other hand, for a random point each equation is satisfied with probability $\frac{1}{2}$. It is obvious that a random point satisfies the linear equation with probability $\frac{1}{2}$. A quadratic equation is satisfied if the quadratic term and the linear part have the same value. In the Trivium system each variable appears at most once per equation. Therefore the probabilities for the quadratic term and the linear part of the equation are independent. Hence the probability that a random point fulfills a quadratic equation of the Trivium system is $\Pr[\text{quadratic term} = 0] \cdot \Pr[\text{linear part} = 0] + \Pr[\text{quadratic term} = 1] \cdot \Pr[\text{linear part} = 1] = \frac{1}{2} \cdot \frac{3}{4} + \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{2}$.

If we consider the problem of solving the Trivium equation system as an optimization problem which is suitable for hill climbing algorithms (cf. Section 2) $X = \{0,1\}^{954}$ is the set of possible configurations. As a cost function $f : X \to \mathbb{R}$ we count the number of not satisfied equations in the system. We know that the minimum of the cost function is 0 and that the initial state of the Trivium system is a configuration for which the cost function is minimal. There might be other optimal solutions, however, it is easy to check if the solution we found is the desired one. If a configuration is an optimal solution for the

discrete optimization problem, it generates the same first 288 bits of keystream as the initial state we are looking for. But it is unlikely that the keystream will be the same for the following keystream bits. Therefore we can check if a solution is the desired one by observing a few more keystream bits and comparing them to the keystream generated by the solution. In our experiments it is unlikely that multiple solutions occur because we set some of the variables to their correct values and therefore consider a highly overdetermined equation system.

4 Properties of Trivium Landscapes

Hill climbing algorithms are sensitive to the way in which the cost function changes when moving between configurations. The best results are obtained when a move from a configuration $x \in X$ to one of the neighbours $\eta(x)$ does not change the value of the cost function too much.

In our case we move from one configuration to another by flipping the value of a single variable. Each variable appears in at most 8 equations and in 6 equations on average, so when moving to a neighbour of the current configuration the cost function will change by at most 8. Furthermore, each variable appears only once in an equation. Therefore changing the value of a single variable will change the value of the equation with probability 1 if the variable appears in a linear term and with probability $\frac{1}{2}$ if the variable appears in a quadratic term. In the latter case flipping the value of a variable will just change the outcome of the equation if the other variable in the quadratic term is assigned to '1'. If a variable appears in the maximum of eight equations it appears in two equations in the quadratic term only. The expected number of equations which change their outcome is 7. Additionally it is unlikely that flipping the value of a variable changes the outcome of all equations which contain this variable in the same direction or respectively it is unlikely that all equations which contain the variable have the same outcome for the configuration before the flip. (The case that a lot or even all equations have the same outcome will appear with higher probability the closer we are to the minimum.)

From these observations we infer that even if we move from a configuration x to one of its neighbours by flipping the value of a variable which appears in 8 equations we do not expect that the value of the cost function changes by 8 in almost all of the cases.

We confirmed this by the following experiment. We generated a Trivium system for a random key and calculated the cost function for a random starting point. Then we chose a neighbour configuration of our starting point and recorded the absolute value of the change in the cost function. To simulate being close to the minimum we set a number of bits to the correct solution but we allowed those bits to be flipped to move to a neighbouring configuration. The results are summarized in Table 5.

These properties of Trivium cost landscapes can be captured more formally using the notion of NK-landscapes and landscape auto-correlation as follows.

4.1 Trivium Systems and NK-Landscapes

NK-landscapes were introduced by Kaufmann [16] to model fitness landscapes with tunable "ruggedness". An NK-landscape is a set of configurations $X = \{0,1\}^n$ together with the cost function defined as

$$f(x) = \sum_{i=1}^{n} f_i(x_i; x_{\pi_{i,1}}, \ldots, x_{\pi_{i,k}}) \ ,$$

where each π_i is a tuple of k distinct elements from the set $\{1, \ldots, n\} \setminus \{i\}$. In other words, the cost function of an NK-landscape is a sum of n local cost functions f_i, each one of them depending on the main variable x_i and a set of k other variables. In a random neighbourhood model, the k indices are selected randomly and uniformly for each f_i. Depending on the value of k, we get either smooth landscapes with relatively few local minima when k is small and rugged landscapes for large values of k.

The Trivium optimization problem can be seen as a particular instance of such a combinatorial landscape. Consider the basic system of equations. We define each f_i as the contribution of i-th equation (either 0 or 1 depending on whether it is satisfied). Each equation depends on six distinct variables, we verified by a computer program that indeed we can always pick one of them as the main variable leaving exactly five other ones for each equation. Trivium optimization problem can thus be seen as an instance of NK-landscape with $n = 954$ and $k = 5$, a rather small value hinting at a certain smoothness of this landscape.

4.2 Landscape Auto-correlation

Another measure of landscape ruggedness is the notion of landscape correlation introduced by Weinberger [25]. We will follow the exposition by Hordijk [14]. The main idea is to perform a random walk on the landscape via neighbouring points. At each step, the cost function y_t is recorded. In this way a sequence $(y_t)_{t=1\ldots T}$ is obtained and we compute its auto-correlation coefficients.

The auto-correlation of a sequence (y_t) for the time lag i is defined as $\rho_i = Corr(y_t, y_{t+i}) = \frac{E[y_t \cdot y_{t+i}] - E[y_t]E[y_{t+i}]}{Var[y_t]}$ where E means the expected value and Var variance of a random variable. Estimates r_i of these auto-correlations ρ_i are $r_i = \frac{\sum_{t=1}^{T-i}(y_t - \bar{y})(y_{t+i} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$ where $\bar{y}$ means the mean value of the sample $y_1, \ldots, y_t$. Here a large auto-correlation coefficient corresponds to a smooth landscape. An important assumption that has to be made for such analysis to be meaningful is that the landscape is statistically isotropic. This means that the statistics of the time series generated by a random walk are the same, regardless of the starting point. Only then a random walk is "representative" of the entire landscape. By computing correlation coefficients for many random walks starting at different points we experimentally verified that the Trivium landscape can be seen as isotropic.

Selected correlation coefficients computed for a basic version of Trivium system and overdefined versions are presented in Table 1. We used sequences of

length 1000000 and averaged results over 100 runs with different keys, IVs and starting points of the walks. Clearly, generating the overdefined system makes the landscape smoother.

Table 1. Correlation coefficients for landscapes generated by Trivium systems of different sizes. n denotes the number of variables in the system.

keystream length	n	$r(1)$	$r(10)$	$r(20)$	$r(30)$	$r(40)$	$r(50)$
288	954	0.987	0.892	0.798	0.713	0.638	0.570
576	1818	0.993	0.942	0.889	0.839	0.791	0.747
1152	3546	0.997	0.970	0.942	0.914	0.886	0.862

5 Solving Trivium Systems with Modified Simulated Annealing

The properties of landscapes generated by the Trivium system of equations suggest that it might be possible to employ stochastic search methods such as simulated annealing to try to find a global optimum and thus recover the secret state of the cipher. In this section we report the results of our experiments in this direction.

Initial experiments with standard simulated annealing were not very encouraging. To be able to solve the Trivium system in reasonable time, we needed to simplify the initial system by setting around 600 out of 954 variables to their correct values throughout the search.

We experimented with the algorithm and its various modifications and found one that yielded a significant improvements over the standard algorithm. The algorithm works as follows. As with standard simulated annealing, we randomly generate a neighbour. If the cost decreases, we accept this move. If not, instead of accepting with probability related to the current temperature, we pick another neighbour and test that one. If after testing a certain number of neighbours we cannot find any cost decreasing move, we accept the increasing move with some probability, just as in the plain simulated annealing. The parameter of this procedure is the number of additional candidates to test before accepting cost increase.

If the parameter is zero, we get plain simulated annealing. On the other end of the spectrum, if we test all possible neighbours, it is easy to see that we get an algorithm that is equivalent to local search, we look for any possible decreasing move and we follow it. When we are in a local minimum, we enter a loop, we finally accept one of the cost increasing candidates but in the next move we always go back to the local optimum we found. Setting the parameter between those extremes yields an intermediate algorithm.

In practice, we used a probabilistic variant of this approach that randomly selects neighbours until it finds one with smaller cost or it exceeds the number of tests defined as a parameter nochangebound. This algorithm is presented in Alg. 2.

Algorithm 2. Modified version of simulated annealing

```
x_best ← x
T ← α                                          ▷ initial temperature parameter is α
k ← 0                                          ▷ set the outer loop counter
while T > 1 do
    for m = 0, ..., M − 1 do                   ▷ parameter M is the number of inner runs
        generate a neighbour y ∈ η(x) uniformly
        if f(y) < f(x) then                    ▷ if cost decreased
            x ← y                              ▷ accept the move
            if f(x) < f(x_best) then           ▷ found a new best value
                x_best ← x                     ▷ store the best configuration
                nc ← 0                         ▷ reset the neighbor counter
                if f(x_best) = 0 then          ▷ if we found a solution
                    return x_best              ▷ finish and return it
                end if
            end if
        else                                   ▷ the candidate cost is higher
            nc ← nc + 1
            if (nc > nochangebound) ∧ (exp((f(y) − f(x))/T > rnd[0, 1]) then
                x ← y                          ▷ accept the move
                nc ← 0                         ▷ reset the counter of tested neighbours
            end if
        end if
    end for
    k ← k + 1
    T = α/ log_2(k · M)                        ▷ Logarithmic cooling schedule
end while
```

The relationship between the number of neighbours tested and the time it took to find a solution (measured in the number of neighbours tested) is presented in Fig 1. Values of nochangebound below 25 result in running times exceeding 2^{40} flips. It suggests that the proper choice of nochangebound is critical for the efficiency of the simulated annealing, in particular, it cannot be too small.

6 Experimental Results

In this section we report results of our computational experiments with the basic equation system generated by the problem of recovering internal state of Trivium. We took the fully determined system with 954 equations and variables obtained after observing 288 bits of the keystream.

We made some comparisons between exponential and logarithmic cooling schedules and from our limited experience the logarithmic cooling schedule performed better in more cases, so we decided to pick that one for our further tests.

The values of α were picked based on empirical observations. Too large α resulted in prolonged periods of almost-random walks where there was no clear sign that any optimization might occur. Too small values gave the behavior

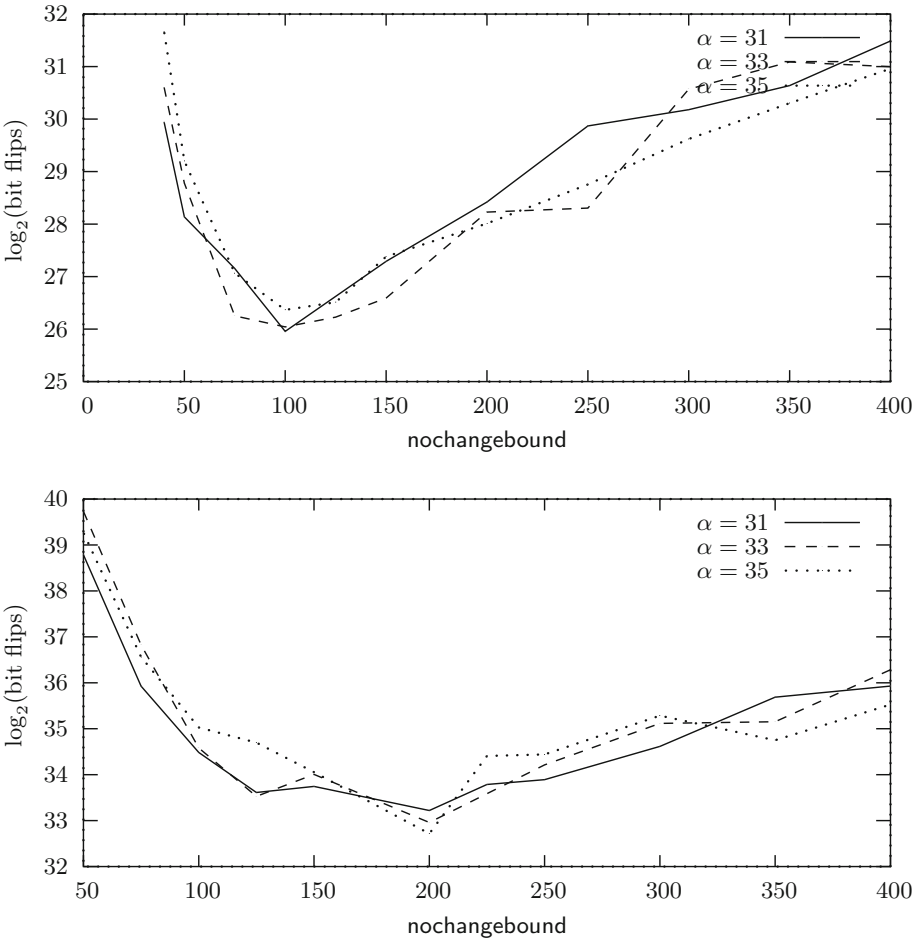

Fig. 1. Influence of nochangebound parameter on the efficiency of simulated annealing applied to basic Trivium system for three values of initial temperature α. Other parameters are $M = 1024$ (cf. Alg. 2), averages are over 10 tests. In the top figure we guessed 200 first bits of the state, in the bottom one 180 bits.

similar to a simple local search when the process was getting stuck in some shallow local optima. After a few trials we decided to use the initial temperature parameter $\alpha = 35$.

For each number of bits of the state fixed to their correct values (preassigned) we ran ten identical tests with different random seeds testing various values of nochangebound parameter (from the set 100, 150, 175, 200, 250, 300). After the test batch finished, we picked that value of nochangebound that yielded lowest search time. We managed to obtain optimal values for nochangebound for 200, 195, 190, 185, 180, 175 and 170 preassigned bits where we set the values of the first bits of the internal state. We use this optimal nochangebound to estimate the total complexity of the attack. The graph is presented in Fig. 2.

The total complexity is the product of the number of guesses we would need to make ($2^{preassigned}$) multiplied by the experimentally obtained running time of the search for the solution. We take the complexity for the correct guess. For a wrong guess the algorithm will not find a solution with costs 0 and terminate.

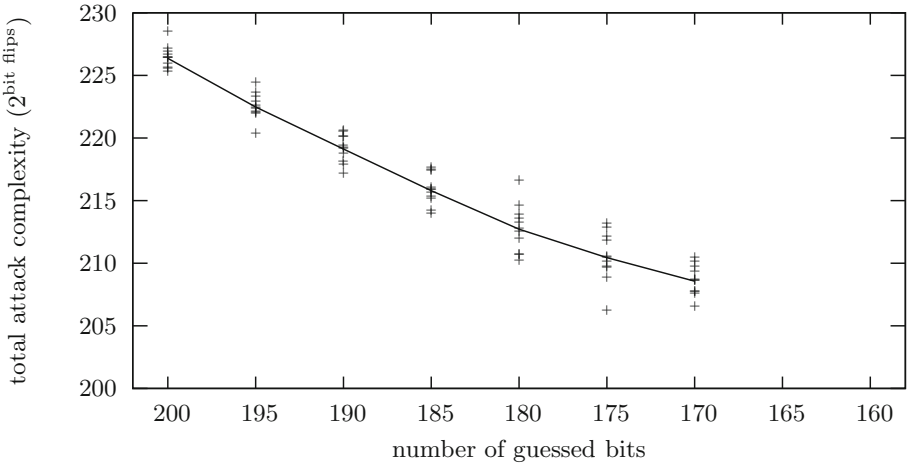

Fig. 2. Running times of the attack based on modified simulated annealing depending on the number of guessed bits. The numbers on the vertical axis are base two logarithms of the total number of moves necessary to find the solution. Crosses represent results of single experiments, the line connects averages.

The results show that the running time of the attack decreases with the smaller number of guessed bits since the increase in time of the search procedure is smaller than the decrease due to the smaller number of bits we have to guess. If the curve goes down below the complexity level corresponding to 2^{80} key setups of Trivium, it would constitute a state-recovery attack. However, the problem is that due to limited computational power we were not able to gather enough results for values of *preassigned* smaller than 170. Our program running on 1.1GHz AMD Opteron 2354 was able to compute 2^{35} bit flips per hour and tests with $preassigned = 170$ required around $2^{38} \sim 2^{39}$ bit flips.

It seems that trying to extrapolate the running times is rather risky, since we do not have any analytical explanation of the complexities we get as often is the case with heuristic search methods. Therefore we do not claim anything about the feasibility of such an attack on full Trivium. We can only conjecture that there might be a set of parameters for which such attack may become possible.

Due to the computational complexity, our experimental results are so far based on only rather small samples of runs for the fixed set of parameters. Therefore, they cannot be taken as a rigorous statisticial analysis but rather as a reconnaissance of the feasibility of this approach. However, we have noticed that for overwhelming fraction of all the experiments, the running times for different runs with the same set of parameters do not deviate from the average

exponent of the bit flips by more than ± 2, i.e. most of the experiments have the number of flips between 2^{avg-2} and 2^{avg+2}. Therefore, we believe that the results give some reasonable impression of the actual situation.

7 Some Variations

The previous section presented the set of our basic experiments. However, there is a multitude of possible variations of the basic setup which could possibly lead to better results. We report on some variations of the search problem we considered while looking for possible optimizations in the Appendix.

8 Conclusions and Future Directions

We presented a new way of approaching the problem of solving systems of sparse, multivariate equations over the binary field. We represent them as combinatorial optimization problems with the cost function being the number of not satisfied equations and then we apply some heuristic search methods, such as simulated annealing to solve them.

We showed that such systems may be relevant in cryptography by giving an example of the system generated by the problem of recovering the internal state of the stream cipher Trivium.

Our experimental results were focused on the Trivium system and they seem to be promising but for now they do not seem to pose any real threat to the security of this algorithm.

We hope that this paper will serve as a starting point for further research in this direction. There are many open problems in this area, the most obvious ones are the selection of better parameters of the search procedures and analytically estimating the possible complexity of such algorithms.

The other interesting direction seems to be the investigation of alternative cost functions. In all our experiments we use the simplest measure counting the number of not satisfied equations. However, many results in heuristic search literature suggest that the selection of a suitable cost function may dramatically change the efficiency of a search. The question of determining whether in our case there exist measures better than the ones we used is still open.

Acknowledgements. We would like to thank the anonymous reviewers of SAC 2010 and Tools for Cryptanalysis 2010 for their comments that helped to improve this paper.

References

1. Armknecht, F., Krause, M.: Algebraic attacks on combiners with memory. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 162–175. Springer, Heidelberg (2003)
2. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)

3. Borghoff, J., Knudsen, L.R., Stolpe, M.: Bivium as a mixed-integer linear programming problem. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 133–152. Springer, Heidelberg (2009)
4. Chardaire, P., Lutton, J.L., Sutter, A.: Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms. European Journal of Operational Research 86(3), 565–579 (1995)
5. Clark, J.A., Jacob, J.L.: Fault injection and a timing channel on an analysis technique. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 181–196. Springer, Heidelberg (2002)
6. Coppersmith, D.: The data encryption standard (DES) and its strength against attacks. IBM Journal of Research and Development 38(3), 243–250 (1994)
7. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
8. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 644–644. Springer, Heidelberg (2003)
9. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
10. De Cannière, C., Preneel, B.: Trivium – a stream cipher construction inspired by block cipher design principles. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 (2005), `http://www.ecrypt.eu.org/stream/papers.html` (April 29, 2005)
11. Eibach, T., Pilz, E., Steck, S.: Comparing and optismising two generic attacks on Bivium. In: Preproceedings of SASC 2008, pp. 57–68 (2008)
12. Fraenkel, A.S., Yesha, Y.: Complexity of solving algebraic equations. Information Processing Letters 10(4/5), 178–179 (1980)
13. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(6), 721–741 (1984)
14. Hordijk, W.: A measure of landscapes. Evolutionary Computation 4(4), 335–360 (1996)
15. Johnson, A.W., Jacobson, S.H.: A class of convergent generalized hill climbing algorithms. Applied Mathematics and Computation 125(2-3), 359–373 (2002)
16. Kauffmann, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
17. Khazaei, S.: Re: A reformuation of TRIVIUM. Posted on the eSTREAM Forum (2006), `http://www.ecrypt.eu.org/stream/phorum/read.php?1,448`
18. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
19. Knudsen, L.R., Meier, W.: Cryptanalysis of an identification scheme based on the permuted perceptron problem. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 363–374. Springer, Heidelberg (1999)
20. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
21. Maximov, A., Biryukov, A.: Two trivial attacks on TRIVIUM. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 36–55. Springer, Heidelberg (2007)

22. McDonald, C., Charnes, C., Pieprzyk, J.: An algebraic analysis of trivium ciphers based on the boolean satisfiability problem. Cryptology ePrint Archive, Report 2007/129 (2007), http://eprint.iacr.org/2007/129
23. Raddum, H.: Cryptanalytic results on Trivium. eStream (March 2006), http://www.ecrypt.eu.org/stream/papersdir/2006/039.ps
24. Turan, M.S., Kara, O.: Linear approximations for 2-round Trivium. In: Security of Information and Networks – SIN 2007, pp. 96–105. Trafford Publishing (2007)
25. Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics 63(5), 325–336 (1990)

A Variants of the Optimization Problem

In this appendix we present results of our experiments with different variations of the basic search problem.

A.1 Guessing Strategy

In order to lower the complexity of solving the equation system we set some of the variables to their correct values. However, the search complexity depends on which variables we choose.

We used different guessing strategies for pre-assigning variables and compared the influence on the running time of our algorithm. We used the following strategies to guess subsets of the state bits:

1. Select the first variables of the initial state.
2. Select the first variables of each register of the initial state.
3. Select the last variables of the initial state.
4. Select the last variables of the each register of the initial state.
5. Select the most frequent variables. These are the variables which are introduced by the update function at the beginning of the keystream generation. We guess values for variable s_{289} and the consecutive ones in this case.
6. An adaptive pre-assignment strategy which is similar to the ThreeFour strategy in [11] (see Subsection A.1).
7. Select the variables in such a way that the equation interdependence measure is minimal. (see Subsection A.1).

It turns out that the best guessing strategy of the ones we tested is to guess the first bits of the initial state. In addition to a pre-assignment of variables we can determine the value of further variables by considering the linear and quadratic equations (see below). We use this technique in the adaptive pre-assignment strategy.

Table 2. Running time for different pre-assignment strategies. nochangebound=110, 190 bits are preassigned, average taken over ten runs.

	first bits of the initial state	most frequent bits	first bit of every register	last bit of the initial state	last bit of each register
average	**29.5**	33.0	34.5	31.2	36.4

Adaptive Pre-Assignment Strategy. In this pre-assignment strategy we use the fact that assigning 5 of the variables in a linear equation will uniquely determine the 6th variable. Starting with an arbitrary linear equation we guess and pre-assign 5 of the 6 variables, determine the value of the remaining variable and assign this to its value. We know that a large fraction of the variables appear in two linear equations. So in the next round of pre-assignment we pick an equation in which at least one variable is already assigned. That means we only have to guess at most 4 variable to get one for free. We continue until we have made the maximum number of guesses or we cannot find an equation in which one variable is already assigned. In the latter case we just have to pick an equation without preassigned variables and run the algorithm again until we made the maximum number of guesses.

Additionally we also use the quadratic equations to determine the value of variables.

The advantage of this pre-assignment strategy is that we can assign many more variables than we actually have to guess. Table 3 gives us an impression of this advantage.

Table 3. The table shows how many bits additional to the guessed bits can be assigned using the adaptive pre-assignment strategy

# guessed bits	# assigned bits	additional assigned bits in %
5	6	20%
50	66	32%
100	135	35%
200	281	40.5%

The disadvantage is that we instead of making the equations sparser we fix some equations to be zero. That means that there are less equations left which contain free variables but the maximum number of equations in which a variables appears is still 8. Therefore a variable influences a higher percentage of equations.

Minimizing Equation Interdependency. If all the equations used different sets of variables, it would be trivial to solve the system by a simple local search. However, variables appear in many equations and changing the value of one of them influences other equations at the same time. This suggests the idea of

Table 4. Running times and landscape auto-correlation coefficients ξ for bit pre-assignment strategies minimizing equation interdependence. Experiments used $\alpha = 33$, $M = 1024$, nochangebound $= 110$.

strategy:	reference	Case 1	Case 2
avg:	29.34	38.9	28.72
ξ	90.1	97.4	96.1

guessing (pre-assigning) bits to minimize the number of variables shared by many equations and thus reduce the degree of mutual relationships between equations.

Capturing this intuition more formally, let E_i be an equation and let $\mathcal{V}(E_i)$ denote the set of *not preassigned* variables that appear in the equation. We can define the measure of interdependence of two equations E_i, E_j as $IntrDep(E_i, E_j) = |\mathcal{V}(E_i) \cap \mathcal{V}(E_j)|$. If the measure is zero, equations use different variables and we can call them separated. Note that pre-assigning any bit that is used by both equations decreases the value of interdependence.

To capture the notion of equations interdependence in the whole system of Trivium equations E, the following measure could be used

$$\sum_{e,g \in E, e \neq g} |\mathcal{V}(e) \cap \mathcal{V}(g)|^2 \ . \tag{2}$$

We used the sum of squares to prefer systems with more equations with only few active (non-preassigned) variables over less equations that have more active variables, but it is possible to use an alternative measure without the squares,

$$\sum_{e,g \in E, e \neq g} |\mathcal{V}(e) \cap \mathcal{V}(g)| \ . \tag{3}$$

The algorithm for pre-assigning bits to minimize the above measure is rather simple. We start with computing the initial interdependence of the system. Then, we temporarily pick a free variable and assign it to compute the new interdependence of the system. If this value is smaller than the current record, we remember it as a new record. After we test all possible candidates, we pick the record one and assign it for good. We repeat this procedure until we get the required number of preassigned bits.

We performed an experiment that compared the results of the reference preassignment strategy fixing the first 190 bits of the state with two variants minimizing (2) and (3). Results presented in Table 4 are interesting. It seems that in spite of significant smoothening of the landscape indicated by higher values of the coefficient ξ the first strategy minimizing (2) significantly worsens the running time. A possible explanation may be that the landscape became more like "golf-course" with large areas without any direction and only very small attraction basins leading to global solution(s). Another possibility is that for such systems, different parameter of nochangebound is preferred. The second variant minimizing (3) seems to be only slightly better than setting the first bits, but more tests would be needed for more parameters to decide any definite advantage.

A.2 Using Overdefined Systems

Results on landscape auto-correlation suggest that using overdefined systems may yield landscape with better structural properties. However, this happens at the expense of a larger set of variables and equations we have to deal with. Our experimental results on overdefined systems suggest that the gain we get from a better landscape is offset by the larger system size so search times are actually not better.

A.3 Variable Persistence

According to [4,5] while using simulated annealing to some optimization problems, one can observe a bias in the frequency of assigning values to variables during the simulated annealing procedure. This bias is related to the solution of the system and observing it can give some information on the solution we are looking for.

We made some experiments that investigated if configurations of local minima (states we run into after a long cooling run) have variables correlated with the global minimum state. In our limited experiments with the basic Trivium system we did not observe any such correlations.

Furthermore, we could see from our preliminary experiments that for a local minimum with a cost value around 40 the current solution still had a large hamming distance to the known optimal solution.

B Cost Change Statistics

Table 5. Change of the cost function when moving to a neighbour configuration: The first row denotes the number of preassigned bits we use to simulate different distances from the minimum. We count how often out of 10000 trials the cost function changes by 0 to 8 units. The last row gives us the average change of the cost function.

i	0	100	200	300	400	500	600	700	800	900	954
0	1714	1702	1685	1560	1309	1052	944	767	601	264	0
1	3253	3246	3297	3158	2641	2143	1856	1550	1120	389	34
2	2248	2235	2240	2241	1930	1720	1385	1172	937	1001	1062
3	1557	1571	1550	1659	1821	1757	1488	1278	1258	1515	1537
4	675	665	668	754	1024	1020	911	810	741	596	648
5	386	400	380	409	691	940	1088	1078	1024	1068	1160
6	127	128	130	164	409	916	1372	1630	1866	2002	2049
7	32	44	41	46	165	439	837	1352	1854	2297	2534
8	8	9	9	9	10	13	119	363	599	868	976
average change	1.81	1.824	1.814	1.9	2.32	2.83	3.32	3.85	4.38	4.97	5.3