On the Accuracy of Statistical Estimations of SAT Partitionings Effectiveness in Application to Discrete Functions Inversion Problems

Alexander Semenov and Oleg Zaikin

Matrosov Institute for System Dynamics and Control Theory SB RAS,
Irkutsk, Russia biclop.rambler@yandex.ru, zaikin.icc@gmail.com

Abstract. In this paper we study the problem of estimating the time required to process decompositions of hard SAT instances encoding inversion problems of some cryptographic functions. In particular, we consider one type of SAT decompositions, usually referred to as SAT partitioning. The effectiveness of a specific SAT parititioning is the total time required to solve all SAT instances from this partitioning. In the paper we construct statistical estimations of effectiveness of SAT partitionings with the help of computational scheme of the Monte Carlo method. We discuss the problem of accuracy of such estimations that arises because of drastic difference between the sizes of random samples, that can be processed in reasonable time, and the size of statistical population. We propose the method for improving constructed statistical estimations by using sets of random samples of increasing size followed by the extrapolation of obtained relation to the size of statistical population.

Keywords: discrete functions, cryptanalysis, SAT, Monte Carlo method

1 Introduction

State-of-the-art algorithms for solving Boolean satisfiability problem (SAT) are computationally quite effective methods, that are successfully applied to combinatorial problems from various areas for several decades [1]. In recent years there were published a lot of papers in which SAT was used to solve cryptanalysis problems [16, 20, 18, 5, 25, 23, 22]. The application of SAT to such problems is justified by a number of reasons. First, ciphering functions in symmetrical cryptography usually can be naturally represented as transformations of bit arrays. Therefore, the problems of inversion of the corresponding functions have relatively compact SAT encodings. Second, in the last few years a dramatic progress in the development of SAT solving technologies has been achieved. It motivates researchers to apply SAT approach to more and more computationally hard problems. Third, cryptographic functions used in the real life situations are based on some arguments that justify the complexity of their inversion problems. That is why corresponding SAT instances can be considered as convincingly hard tests.

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

The technologies that make it possible to solve such tests using SAT approach may prove useful for solving other combinatorial problems that can be effectively reduced to SAT.

In the present paper we describe an approach in which SAT solving algorithms are applied to the following problem. Assume we have a discrete function $f : \{0,1\}^n \to \{0,1\}^m$ which is total in $\{0,1\}^n$ and is specified by some algorithm (program) $A(f)$ (by $\{0,1\}^n$, $n \in N$ we denote the set formed by all binary words of length n). For an arbitrary $y \in Range\ f$, $Range\ f \subseteq \{0,1\}^m$ we need to find such $x \in \{0,1\}^n$ that $f(x) = y$. Below we will refer to this problem as the inversion problem for function f. It has an evident cryptographic context. For example, suppose that f is an algorithm implementing some keystream generator, y is a keystream fragment produced by the generator and x is an unknown secret key. Then the inversion problem for f is the problem of cryptanalysis of keystream generator based on the known keystream fragment.

One of the ways commonly employed to improve the effectiveness of solving inversion problems for cryptographic functions is the use of parallel and distributed computing environments. The choice of parallelization strategy that yields high effectiveness of corresponding solving process is a very nontrivial task. For a number of functions with secret key of small length the use of parallel computing makes it possible to solve inversion problems via exhaustive search (also known as brute-force search or brute-force attacks in the context of cryptography) [9]. However, for ciphers with keys of length > 80 this approach has no prospects. In such cases we can try to decrease the complexity of inversion problem solving relative to the complexity of brute-force attacks by employing intellectual parallel algorithms, based on the strategies aimed at reducing the search space. From our point of view state-of-the-art SAT solving algorithms, in particular, CDCL solvers [15] are well-suited for such purposes.

In the present paper we study the problem of decomposing an original hard SAT instance into a family of easier SAT instanses, that can be solved independently of each other. Such families are called SAT partitionings. Partitionings can be constructed in many different ways, thus the time required to process different partitionings can greatly vary.

The problem of estimating the time of processing of an arbitrary SAT partitioning is highly non-trivial in general case. Earlier [22] we generalized previously known results and proposed an approach to solving this problem, based on estimating values of special predictive function via the computational scheme of the Monte Carlo method. To find SAT partitionings with good time estimations we employed general scheme of local search in finite search space, improved by various metaheuristics (in particular, we used simulated annealing and tabu search). Each time we compute the value of predictive function it is neccessary to estimate the expected value of some random variable. Unfortunately, in general case we have no additional knowledge regarding the distribution of this variable. In these conditions the accuracy of constructed estimations can critically depend on the size of random samples used.

In the present paper we propose an approach aimed at improving the accuracy of estimations constructed according to the Monte Carlo method by using a set of random samples of increasing size and then by extrapolating the obtained relation to the

size of statistical population. We show that this approach works well in application to constructing SAT partitionings for SAT instances encoding inversion of several cryptographic functions. We applied this approach to improve the time estimations for solving inversion problems for some cryptographic functions.

2 Mathematical Background and Previous Results

Boolean Satisfiability Problem (SAT) consists in the following: for an arbitrary Boolean formula F to answer the question whether it is satisfiable and to construct the satisfying assignment in case if the answer is positive. This problem for an arbitrarty formula F can be effectively (in polynomial time on the length of F binary code in general case) reduced to the satisfiability problem for Boolean formula F' in conjunctive normal form (CNF). Hereinafter we consider SAT in this context. SAT is a classical NP-hard problem [4]. Despite this fact, it is possible to effectively solve SAT for wide classes of the so-called industrial tests of relatively high dimension (tens of thousands of variables and hundreds of thousands of clauses). The corresponding algorithms in recent years developed into powerful software systems and are actively used in symbolic verification, bioinformatics, cryptography and other areas. Further we mainly concentrate on the application of SAT to inversion of discrete functions used in cryptography.

To reduce to SAT the inversion problem of $f : \{0,1\}^m \to \{0,1\}^m$ in an arbitrary point $y \in Rangef$ one can use various approaches, including automated systems that construct SAT instance encoding the inversion of f in a considered point using the text of a program specifying f. Such software systems employ the concept of symbolic execution [14] to construct not machine code implementing f, but some Boolean formula. To construct all SAT instances used in our computational experiments we used the TRANSALG system [21].

Let $C_f(y)$ be a CNF, that encodes the inversion problem for some function $f : \{0,1\}^n \to \{0,1\}^m$ in point $y \in Rangef$. It means that $C_f(y)$ is satisfiable and from each of its satisfying assignments one can effectively extract such set of Boolean values $x \in \{0,1\}^n$ that $f(x) = y$. For the functions used in cryptography SAT for CNFs $C_f(y)$ is usually quite hard. Essentially for ciphers used in practice it is pointless to believe that the problems of such kind can be solved without employing high performance computing using parallel computing architectures. Below we describe one general approach to parallelization of SAT usually referred to as Partitioning approach and study the problem of estimating the effectiveness of an arbitrary SAT partitioning, and also the problem of search for partitionings with good effectiveness estimations.

Let us assume that we consider SAT for an arbitrary CNF C. The SAT partitioning of CNF C [11] is a family of formulas of the kind $C \wedge C_i, i \in \{1,\ldots,s\}$, such that any formula $C \wedge C_i \wedge C_j$ is unsatisfiable when $i \neq j$ and

$$C \equiv (C \wedge C_1) \vee \ldots \vee (C \wedge C_s)\,.$$

It is clear that C is satisfiable if and only if at least one CNF of the kind $C \wedge C_i$, $i \in \{1,\ldots,s\}$ is satisfiable. It is important to note that SAT for these CNFs can be solved in parallel.

There is a lot of different approaches to constructing SAT partitionings. A comprehensive study of many of them can be found in [11]. Unfortunately in the majority of cases the problem of estimating the time required to solve all SAT instances from a particular SAT partitioning is highly nontrivial. A number of approaches to constructing such estimations rely on trying to predict the solver behavior based on some service information produced by the solver during its work [27]. Another way implies that the statistical estimation of time required to process SAT partitioning is computed based on the time required to process its part. This approach was first proposed and later developed in the papers on SAT-based cryptanalysis [18, 5, 25, 23]. In [22] we have brought under the results proposed in cited papers the strict foundation in the form of Monte Carlo method in its classical sense [19]. Below let us briefly consider these results.

So, let C be an arbitrary CNF over the set of Boolean variables $X = \{x_1, \ldots, x_n\}$. Consider an arbitrary set $\tilde{X} = \{x_{i_1}, \ldots, x_{i_d}\}$, $\tilde{X} \subseteq X$. Let us call $\tilde{X}$ a decomposition set. Let g be an arbitrary minterm over $\tilde{X}$ and $G_{\tilde{X}} = \{g_i\}_{i=1,\ldots,2^d}$ be the set of all possible minterms over $\tilde{X}$. It is clear that an arbitrary formula $g \in G_{\tilde{X}}$ is satisfiable only on a single truth assignment of variables from $\tilde{X}$. Thus, the set

$$\Delta(C, \tilde{X}) = \{C \wedge g\}_{g \in G_{\tilde{X}}} \tag{1}$$

is a SAT partitioning of CNF C. Hereinafter we work with partitionings of the type (1).

Let C be an arbitrary CNF and $\Delta(C, \tilde{X})$ be its arbitrary SAT partitioning of the type (1). Let A be an arbitrary SAT solving algorithm. By $t_A(C, \tilde{X})$ we denote the total time required to process all SAT instances from $\Delta(C, \tilde{X})$ by algorithm A. In [22] we stated the following result.

Theorem 1. *Let A be an arbitrary complete SAT solving algorithm, i.e. its runtime is finite for an arbitrary input. Then for an arbitrary CNF C and any decomposition set $\tilde{X}$ there is a random variable $\xi_A(C, \tilde{X})$ with finite expected value $\mathrm{E}\left[\xi_A(C, \tilde{X})\right]$ such that the following equality holds*

$$t_A(C, \tilde{X}) = 2^{|\tilde{X}|} \cdot \mathrm{E}\left[\xi_A(C, \tilde{X})\right]. \tag{2}$$

Random variable $\xi = \xi_A(C, \tilde{X})$ from Theorem 1 is defined as follows. Define a uniform distribution on the set $\{0,1\}^d$, $d = |\tilde{X}|$. With each assignment $\alpha \in \{0,1\}^d$ we link minterm $g(\alpha)$ over $\tilde{X}$ that is satisfiable on α. With each α randomly chosen from $\{0,1\}^d$ we associate the value of random variable ξ equal to the time required by A to solve SAT for CNF $C \wedge g(\alpha)$.

Thus, if we know $E[\xi_A(C, \tilde{X})]$ we can obtain exact value of time required to process SAT partitioning $\Delta(C, \tilde{X})$. Unfortunately, in general case we are not able to effectively compute $E[\xi_A(C, \tilde{X})]$. However, we can estimate it using the Monte Carlo method in its classical sence [19], i.e. as method for estimating expected values of random variables. In more detail, according to this method, a probabilistic experiment that consists of N independent observations of values of an arbitrary random variable ξ is used to approximately calculate $E[\xi]$. Let $\xi^1, \ldots, \xi^N$ be results of the corresponding

observations. They can be considered as a single observation of N independent random variables with the same distribution, i.e. the following equalities hold

$$E[\xi] = E[\xi^1] = \ldots = E[\xi^N], Var(\xi) = Var[\xi^1] = \ldots = Var[\xi^N]$$

Let $E[\xi]$ and $Var(\xi)$ be both finite, also let γ be any fixed confidence level and $\gamma = \Phi(\delta_\gamma)$, where $\Phi(\cdot)$ is the normal cumulative distribution function. Then from the Central Limit Theorem [6] for some $N_0 = N_0(\gamma)$ and all $N \geq N_0$ the main formula of the Monte Carlo method is valid

$$\Pr\left\{\left|\frac{1}{N}\cdot\sum_{j=1}^{N}\xi^j - \mathrm{E}\left[\xi\right]\right| < \frac{\delta_\gamma\cdot\sigma}{\sqrt{N}}\right\} \geq \gamma. \tag{3}$$

Here $\sigma = \sqrt{Var\left(\xi\right)}$ stands for a standard deviation. It means that under the considered assumptions the value $\frac{1}{N}\cdot\sum\limits_{j=1}^{N}\xi^j$ is a good approximation of $\mathrm{E}\left[\xi\right]$, when the number of observations N is large enough.

Taking into account all of the above, the procedure of statistical estimation of the value of $t_A(C, \tilde{X})$ for a particular $\tilde{X}$ looks as follows. Assuming that there is specified a uniform distribution on $\{0,1\}^d$, $d = |\tilde{X}|$ we choose assignments $\alpha^1, \ldots, \alpha^N$. The obtained set of assignments forms a random sample. For each α^j, $j \in \{1, \ldots, N\}$ the algorithm A solves SAT for CNF $C \wedge g(\alpha^j)$. The runtime of A is the value ξ^j of the observed random variable ξ. After solving all SAT instances of the kind $C \wedge g(\alpha^j)$, $j \in \{1, \ldots, N\}$ we compute the following value:

$$F_{A,C}(\tilde{X}) = 2^d \cdot \left(\frac{1}{N}\cdot\sum_{j=1}^{N}\xi^j\right) \tag{4}$$

From (3) it follows that if N is large enough then the value of $F_{A,C}(\tilde{X})$ can be considered as a good approximation of (2). Therefore, instead of searching for a decomposition set with minimal value (2) one can search for a decomposition set with minimal value of $F_{A,C}(\tilde{X})$. Below we refer to function $F_{A,C}(\tilde{X})$ as to a predictive function.

In [22] we described the procedure for minimization of the function $F_{A,C}(\tilde{X})$ over the Boolean hypercube $\{0,1\}^n$. In this procedure we represent an arbitrary set $\tilde{X}$ with a Boolean vector of length n: in this vector 1-s correspond to variables from X that are included into $\tilde{X}$. Thus the values of $F_{A,C}(\tilde{X})$ are computed in points of the search space $\{0,1\}^n$. To minimize $F_{A,C}(\tilde{X})$ we used simulated annealing and tabu search metaheuristics. As we point out in [22] the tabu search scheme showed better performance on considered test instances.

The tabu search scheme for miminization of predictive function $F_{A,C}(\tilde{X})$, that we described in [22], was implemented in the form of MPI program PDSAT for computing clusters. We applied PDSAT to cryptanalysis instances of widely known A5/1 and Bivium keystream ciphers by reducing the corresponding problems to SAT. In the process of its work PDSAT used random samples of size 10^5. As a result we managed

to construct the time estimation for Bivium cryptanalysis that turned out to be almost three times better than record estimation at that time, constructed in [25] (if we scale the estimations according to the performance of computing platforms employed). In the process of computational experiments, however, we noticed that the size of random sample can critically influence the accuracy of estimation. It is quite surprising that in previous papers with similar results this problem had not been addressed at all. That is why in the following section we propose the original approach to improving the accuracy of values of predictive function $F_{A,C}(\tilde{X})$ computed via the Monte Carlo method.

3 The Method for Improving Statistical Estimations of SAT Partitionings Effectiveness

In this section we discuss the accuracy of estimations of the effectiveness of SAT partitionings produced by the Monte Carlo method. State-of-the-art SAT-solvers are quite complex programs, and even little modifications of their of parameters can greatly influence its performance. Authors of [7] note that often the SAT solver cannot solve some instance even after hours of work, but small modifications, such as, for example, changing variable assignment ordering, enable it to solve the instance in seconds or minutes. Back when SAT approach was not developed quite so well, in a number of papers similar behaviors were noted for Backtrack algorithms used to solve Constraint Satisfaction Problems. The study of these behaviors on randomly chosen instances showed that statistically observed data can be characterized by heavy-tailed distributions. The phenomenon of "heavy-tailed behavior" of state-of-the-art SAT solvers has been intensively studied in recent years [8, 3, 26, 7].

The results of the papers mentioned above make it possible to conclude that the complex and inhomogenious behavior of the SAT solver can negatively influence the consistency of the estimations of the effectiveness of SAT partitionings produced using the Monte Carlo method. The main danger here lies in the risk of obtaining overly optimistic estimations. Below we propose one method for solving this problem and justify its adequacy using computational experiments.

So, what information can we use to estimate how well the sample mean approximates the expected value of the random variable considered? From the formula (3) it follows that the accuracy increases with the increase of the sample size (if expected value and variance are finite). In application to estimating the value of (2) the severity of the issue regarding the consistency of the Monte-Carlo estimations becomes apparent when we compare the random samples sizes with the total number of subproblems in the SAT partitioning. For example if the decomposition set consists of 50 variables, then the size of the corresponding partitioning is 2^{50}. Therefore the sample of size 10000 is approximately $8 \cdot 10^{-10}\%$ of the total size of the statistical population. Can we consider the estimation produced with the use of such a small sample to be consistent?

The suggested approach for improving the estimations of effectiveness of SAT partitionings, obtained via Monte Carlo, is based on the following simple idea. We consider the problem of estimating $\mathrm{E}[\xi]$, assuming that random variable ξ has finite expected

value and variance. Also we assume that the statistical population has finite size represented by number N^*. Consider the sequence of random samples sizes

$$N_1 < N_2 < \ldots < N_l \approx N^* \tag{5}$$

Below we will use the following notation:

$$F_{N_i} = \frac{1}{N_i} \cdot \sum_{j=1}^{N_i} \xi^j$$

Here $\left\{\xi^j\right\}_{j=1}^{N_i}$ is a random sample of size N_i from the statistical population of size N^*. Hereinafter we assume that for each N_i a different random sample of corresponding size is considered. Suppose that we can in affordable time calculate the values $F_{N_1}, F_{N_2}, \ldots, F_{N_k}, k < l$. Based on this data we need to predict the behavior of function F_N when $N \in \{N_{k+1}, \ldots, N_l\}$. Below we study this problem in detail.

Unfortunately we cannot make any reasonable assumptions about the type of a function to be used to extrapolate F_N. That is why we introduce the auxiliary function, to which we refer as *jump function*. This function is denoted as $\varepsilon(N)$ and is defined as follows:

$$\varepsilon(N_i) = max\left\{\frac{F_{N_{i-1}}}{F_{N_i}}, \frac{F_{N_i}}{F_{N_{i-1}}}\right\}, i \in \{2, \ldots, l\}$$

In essence, each value of $\varepsilon(N_i)$ shows how values of F_N for $N = N_{i-1}$ and $N = N_i$ differ from each other. Below we describe the behavior of function $\varepsilon(N)$ as N goes to infinity. It will allow us to choose the correct type of the function to extrapolate $\varepsilon(N)$ with. Let us show that the following theorem holds.

Theorem 2. *Consider a probabilistic experiment in which the outcome is a random variable that takes positive values. Assume that this variable has finite expected value and variance. Consider the set*

$$\{N_1, N_2, \ldots, N_l, \ldots\}$$

such that $N_{i+1} = \lfloor \lambda \cdot N_i \rfloor$ *for each* i *and some constant* $\lambda > 1$. *Then for any confidence level* γ *there exists a constant* $\alpha(\gamma)$ *such that for some* $k_0 = k_0(\gamma)$ *and all* $k \geq k_0$ *the following holds:*

$$\Pr\left\{1 \leq \varepsilon(N_k) \leq 1 + \frac{\alpha(\gamma)}{\sqrt{N_k}}\right\} \geq \gamma^2$$

Proof. Let ξ be a random variable satisfying all conditions of the theorem. The finiteness of the expected value and of the variance of ξ allows us to apply the Central Limit Theorem. Let us fix some confidence level γ. Assume $a = \mathrm{E}[\xi]$, $b = \delta_\gamma \cdot \sigma$. Since ξ takes only positive values, then $a > 0$. Consider an arbitrary set $\{N_1, \ldots, N_l, \ldots\}$ where $N_{i+1} = \lfloor \lambda \cdot N_i \rfloor$, $\lambda > 1$ is some constant. Note, that the events

$$A_k = \left\{|F_{N_k} - a| < \frac{b}{\sqrt{N_k}}\right\}, A_{k+1} = \left\{\left|F_{N_{k+1}} - a\right| < \frac{b}{\sqrt{N_{k+1}}}\right\} \tag{6}$$

are independent and therefore from (3) for some $k_0 = k_0(\gamma)$ and all $k \geq k_0$ the following holds:

$$\Pr\{A_k \wedge A_{k+1}\} \geq \gamma^2$$

From (6) and from $N_{k+1} = \lfloor \lambda \cdot N_k \rfloor$ for some $\lambda > 1$ it follows that with probability $\geq \gamma^2$ the following inequalities hold:

$$\begin{cases} a - \frac{b}{\sqrt{N_k}} < F_{N_k} < a + \frac{b}{\sqrt{N_k}} \\ a - \frac{c_1}{\sqrt{N_k}} < F_{N_{k+1}} < a + \frac{c_2}{\sqrt{N_k}} \end{cases} \tag{7}$$

where c_1, c_2 are some positive constants. Because ξ takes only positive values, then $F_{N_k} > 0$, $F_{N_{k+1}} > 0$. Since $\mathrm{E}[\xi] > 0$ then for some k_0 and all $k \geq k_0$ the values $a - \frac{b}{\sqrt{N_k}}$ and $a - \frac{c_1}{\sqrt{N_k}}$ are positive. It means that for an arbitrary $k \geq k_0$ from (7) with probability $\geq \gamma^2$ the following inequalities hold

$$\begin{gathered} \frac{a - \frac{b}{\sqrt{N_k}}}{a + \frac{c_2}{\sqrt{N_k}}} < \frac{F_{N_k}}{F_{N_{k+1}}} < \frac{a + \frac{b}{\sqrt{N_k}}}{a - \frac{c_1}{\sqrt{N_k}}} \\ 1 - \frac{C}{a\sqrt{N_k} + c_2} < \frac{F_{N_k}}{F_{N_{k+1}}} < 1 + \frac{D}{a\sqrt{N_k} - c_1} \end{gathered} \tag{8}$$

for some positive constants C, D. It is easy to construct similar bounds for $\frac{F_{N_{k+1}}}{F_{N_k}}$. Since it is evident that $\varepsilon(N_{k+1}) \geq 1$ then from (8) the assertion of the theorem follows. □

The theorem proved allows us to use the function of the kind $1 + \frac{\alpha}{\sqrt{N}}$ to predict the behavior of $\varepsilon(N)$. Here the constant $\alpha > 0$ is picked up individually for each random variable ξ based on the known first values of $\varepsilon(N)$.

Now let us return to the problem of miminization of predictive function $F_{A,C}(\tilde{X})$ over Boolean hypercube $\{0,1\}^n$. As we already noted, in fact we minimize function $F(\chi)$, the value of which in an arbitrary $\chi \in \{0,1\}^n$ is defined as follows. Ones in a vector χ specify decomposition set $\tilde{X}$. For this $\tilde{X}$ we construct random sample $\alpha^1, \ldots, \alpha^N$, $\alpha_j \in \{0,1\}^d$, $d = |\tilde{X}|$. After this we compute the value of predictive function according to (4). The obtained value is the value of $F(\chi)$. Consider the following set of values of the parameter N (random sample size):

$$N_1, N_2 = \lambda \cdot N_1, \ldots, N_l = \lambda^{l-1} \cdot N_1$$

where λ, $\lambda > 1$, $\lambda \in \mathrm{N}$ is some constant. Also we suppose that $N_l \approx \left|\Delta(C, \tilde{X})\right|$, where $\tilde{X}$ is a decomposition set represented by the vector χ. Let us fix some k_*, $k_* < l$ and compute values

$$F_{N_1}(\chi), \ldots, F_{N_{k_*}}(\chi) \tag{9}$$

We will refer to this set as a *training set*. Our goal is to use the training set to compute values $\tilde{F}_{N_{k_*+1}}(\chi), \ldots, \tilde{F}_{N_l}(\chi)$ that predict the behavior of function $F_N(\chi)$ when $N \in \{N_{k_*+1}, \ldots, N_l\}$.

Using (9) we compute

$$\varepsilon(N_i), i \in \{2, \ldots, k_*\} \tag{10}$$

We assume that (10) are the values of function of the type $1+\frac{\alpha}{\sqrt{N_i}}$ for corresponding N_i. Using this information we pick the value of the constant α. Denote the function obtained as $\tilde{\varepsilon}(N_i)$. Let us compute the values of this function for N_i, $i \in \{k_*+1,\ldots,l\}$. The values $\tilde{F}_{N_i}$, $i \in \{k_*+1,\ldots,l\}$ we determine using the following formula

$$\tilde{\varepsilon}(N_i) = max\left\{\frac{\tilde{F}_{N_{i-1}}}{\tilde{F}_{N_i}}, \frac{\tilde{F}_{N_i}}{\tilde{F}_{N_{i-1}}}\right\}, i \in \{k_*+1,\ldots,l\}, \tilde{F}_{N_{k_*}}(\chi) = F_{N_{k_*}}(\chi)$$

When computing values of particular $\tilde{F}_{N_i}$ the main problem consists in avoiding overly optimistic estimations. That is why we choose to follow the "pessimistic scenario", so for each $i \in \{k_*+1,\ldots,l\}$ we assume that

$$\tilde{F}_{N_{i+1}}(\chi) \geq \tilde{F}_{N_i}(\chi)$$

It is easy to see that in this case for any $r > k_*$ the following holds:

$$\tilde{F}_{N_r}(\chi) = F_{N_{k_*}}(\chi) \cdot \prod_{s=k_*+1}^{r} \tilde{\varepsilon}(N_s) \tag{11}$$

We consider the value $F^*(\chi) = \tilde{F}_{N_l}(\chi)$ to be an improved estimation of the time required to process the partitioning $\Delta(C,\tilde{X})$, where $\tilde{X}$ is a decomposition set defined by vector $\chi \in \{0,1\}^n$.

The results of the application of the proposed technique aimed at improving the consistency of estimations are shown in the next section.

4 Computational Experiments

In this section we show the applicability of the suggested technique for improving the accuracy of estimations of predictive function values to SAT-based cryptanalysis of the Bivium cipher.

4.1 Time Estimations for Problems of SAT-based Cryptanalysis of Bivium

The Bivium keystream generator [2] uses two shift registers of a special kind. The first register contains 93 cells and the second contains 84 cells. To initialize the cipher, a secret key of length 80 bit is put to the first register, and a fixed (known) initialization vector (IV) of length 80 bit is put to the second register. All remaining cells are filled with zeros. An initialization phase consists of 708 rounds during which keystream output is not released.

In accordance with [17, 24] we considered cryptanalysis problems for Bivium and Grain in the following formulation. Based on the known fragment of keystream we search for the values of all registers cells at the end of an initialization phase. It means that we need to find 177 bits in case of Bivium and 160 bits in case of Grain. Usually it is believed that to uniquely identify the secret key it is sufficient to consider a keystream

fragment of length comparable to the total length of shift registers. Here we followed [5, 24] and set the keystream fragment length for Bivium cryptanalysis to 200 bits.

Therefore in our experiments we needed to find initial values of 177 Bivium registers bits that lead to generation of known keystream fragment of size 200 bits. We encoded the corresponding problem to SAT using the TRANSALG software system [21]. Below we will use the notation we introduced in [22] when describing the tabu search algorithm for minimization of predictive function. By $\tilde{X}_{start}$ (χ_{start}, respectively) we denote the starting decomposition set, i.e. the point in $\{0,1\}^n$ from which the minimization process for function $F(\cdot)$ starts. This set is usually chosen based on features of the considered problem. In particular, it is easy to show that the value $F(\cdot)$ can be computed effectively (for a random sample of reasonable size) if $\tilde{X}_{start}$ is a Strong Unit Propagation Backdoor Set (SUPBS, [12]) for CNF C (here we assume that we use complete CDCL-algorithm [15] in the role of A). By χ_{best} we denote the point in $\{0,1\}^n$ in which the value of predictive function is minimal when the search procedure finishes working. In our experiments in the role of $\tilde{X}_{start}$ we chose the set of variables encoding the initial values of Bivium registers, thus $|\tilde{X}_{start}| = 177$. The PDSAT program, that is an MPI-implementation of tabu search algorithm for predictive function minimization, was launched on 160 cores of Opteron 6276 processor within Academician V.M. Matrosov computing cluster of Irkutsk Supercomputing Centre.

Further on the example of one instance of Bivium cryptanalysis we will show a typical situation, in which the sample size can critically affect the accuracy of the estimations. In the process of computing the random samples of size 10^5 were used. As a result PDSAT found the point χ_{best} for which $F(\chi_{best}) = 2.085 \times 10^{11}$ seconds. The corresponding $\tilde{X}_{best}$ contains 50 variables. After this we launched the computational experiment in which we used random samples of size 10^4. After one day of work PDSAT found the point χ'_{best} with predictive function value $F(\chi'_{best}) = 5.98 \times 10^6$ seconds. We decided to compute in this point the values of predictive function using random samples of sizes $N_2 = 2 \times 10^4$, $N_3 = 4 \times 10^4$ and $N_4 = 8 \times 10^4$. For these random samples we also computed the values of jump function $\varepsilon(N)$. The corresponding results are presented in Table 1.

Sample size	$N_1 = 10^4$	$N_2 = 2 \times 10^4$	$N_3 = 4 \times 10^4$	$N_4 = 8 \times 10^4$
F_{N_i}	5.98×10^6	1.10×10^7	8.60×10^6	2.59×10^{12}
$\varepsilon(N_i)$	–	1.83	1.27	3.01×10^5

Table 1. Evaluation of the estimation consistency

Note that $\varepsilon(N_4) = max\left\{\frac{F_{N_3}}{F_{N_4}}, \frac{F_{N_4}}{F_{N_3}}\right\} \approx 3 \times 10^5$. The explanation of this phenomena is that samples of size $N_1 = 10^4$, $N_2 = 2 \times 10^4$ and $N_3 = 4 \times 10^4$ contained only simple subproblems, resulting in overly optimistic estimations. However the sample of size $N_4 = 8 \times 10^4$ contained about a hundred of subproblems that were much harder than others. Therefore we can conclude that the estimation F_{N_1} for sample size $N_1 = 10^4$ is overly optimistic.

After this we returned to the point χ_{best} found using random samples of size $N_1 = 10^5$. Note that it defines the set $\tilde{X}_{best}$, $|\tilde{X}_{best}| = 50$. We computed the predictive function values in this point for random sample sizes

$$N_j = 2^{j-1} \times 10^5, \tag{12}$$

where $j \in \{2, \ldots, 14\}$. Obtained values $F_{N_j}(\chi_{best})$ we used as the training set. Then based on $\varepsilon(N_j)$, $j \in \{2, \ldots, 14\}$ we selected the value of constant α in the function $\hat{\varepsilon}(N_j) = 1 + \frac{\alpha}{\sqrt{N_j}}$, that extrapolates the behavior of $\varepsilon(N_j)$ for $j \in \{15, \ldots, 35\}$. Here $\left|\tilde{X}_{best}\right| = 50$, so $N_{35} \approx 2^{50}$. With the help of the MATLAB system we got $\alpha = 445.9$. The graphs of function $\varepsilon(N_j)$ for $j \in \{1, \ldots, 14\}$ and function $\hat{\varepsilon}(N_j)$ for $j \in \{1, \ldots, 35\}$ are shown on Figure 1. Note that function $\hat{\varepsilon}(\cdot)$ doesn't have big jumps. On Figure 2 the extrapolation of the behavior of the predictive function in the considered point χ_{best} constructed according to formula (11) is shown. The improved estimation for the predictive function value is $\hat{F}^*(\chi_{best}) = 7.37 \times 10^{11}$

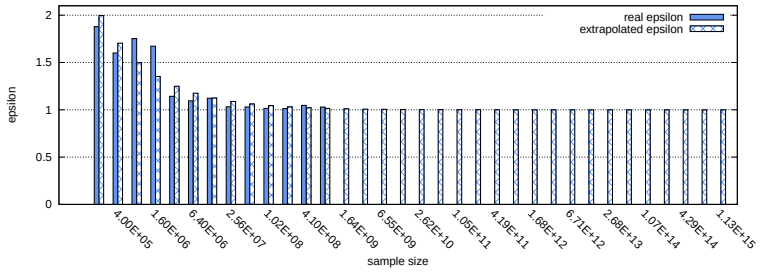

Fig. 1. Values of ε and extrapolated $\tilde{\varepsilon}$ for Bivium

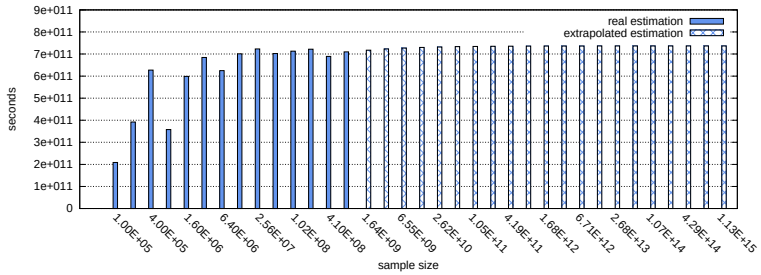

Fig. 2. Values of estimation F_{N_j} and extrapolated estimation $\tilde{F}_{N_j}$ for Bivium

4.2 Solving Cryptanalysis Problems for Weakened Ciphers and Justification of Proposed Method

To evaluate the consistency of the proposed technique for improving estimations we solved a number of weakened cryptanalysis problems for the Bivium cipher using the computing cluster. For this purpose, the PDSAT program was used. In this program in addition to the prediction mode there is a solving mode. In this mode for $\tilde{X}_{best}$ found during predictive function minimization it generates all $2^{|\tilde{X}_{best}|}$ assignments of variables from $\tilde{X}_{best}$ and solves all corresponding SAT instances. By *BiviumK* we denote the families of weakened cryptanalysis problems of the Bivium cipher. We call each individual problem from such family an instance. An arbitrary instance from *BiviumK* is a cryptanalysis problem for the corresponding cipher in the formulation described above with an addition of known values of K initial variables that encode values of the last K cells of the second generator register. In the computational experiments we considered series of 3 different instances for *Bivium14*. In all cases on the prediction stage we used random samples of size $N = 10^5$. We processed these series in the following manner: for the first instance in the series of three we used PDSAT to find the decomposition set $\tilde{X}_{best}$, defined by vector χ_{best}. Then we computed the improved estimation $\hat{F}(\chi_{best})$ for the corresponding set using the extrapolation via the jump function. After this the decomposition set $\tilde{X}_{best}$ found for the first instance was used to solve all 3 instances from the series in parallel. All instances were successfully solved. To solve one instance it took about 17 hours on 480 cores of Opteron 6276.

Thus for each problem of the kind *Bivium14* we found exact time required to process the SAT partitioning defined by $\tilde{X}_{best}$. After this we compared this value with the value obtained using the proposed technique for improving the estimated values of predictive function. Below we describe this experiment in more detail.

For each cryptanalysis instance *Bivium14* the PDSAT system found the set $\tilde{X}_{best}$ containing 35 variables. Here PDSAT used random samples of size $N_1 = 10^5$. Similar to the case of non-weakened Bivium we used $\lambda = 2$ and considered first 14 values N_j:

$$N_1 = 10^5, N_j = 2^{j-1} \times 10^5, j \in \{2, \ldots, 14\}$$

The obtained values $F_{N_j}(\chi_{best})$ we used in the role of training set. Since $|\tilde{X}_{best}| = 35$ and $2^{35} \approx N_{20}$ we needed to extrapolate the behaviour of functions $\varepsilon(N_j)$ and $F_{N_j}(\chi_{best})$ to $j \in \{15, \ldots, 20\}$. We extrapolate $\varepsilon(N_j)$ as a function of the type $1 + \frac{\alpha}{\sqrt{N_j}}$. With the help of MATLAB we got $\alpha = 72.273$. The results of extrapolation are showed on Figure 3.

Then we extrapolated the behaviour of $F_{N_j}(\chi_{best})$ for $j \in \{15, \ldots, 20\}$ using formula (11). The results are shown on Figure 4.

The last column of the diagram on Figure 4 shows the difference between $\hat{F}^*(\chi_{best})$ and the real time spent to process the corresponding SAT partitioning (solid line). Note that it does not exceed 7 % and $\hat{F}^*(\chi_{best})$ is not overly optimistic estimation.

5 Related Works

A lot of papers studied the questions regarding estimating the runtime of algorithms for solving SAT and Constraint Satisfaction problem on hard instances. Thus in [13] there

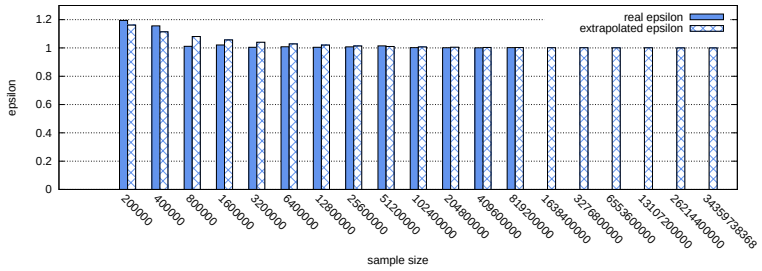

Fig. 3. Values of ε and extrapolated $\tilde{\varepsilon}$ for *Bivium14* (for the first instance from the series)

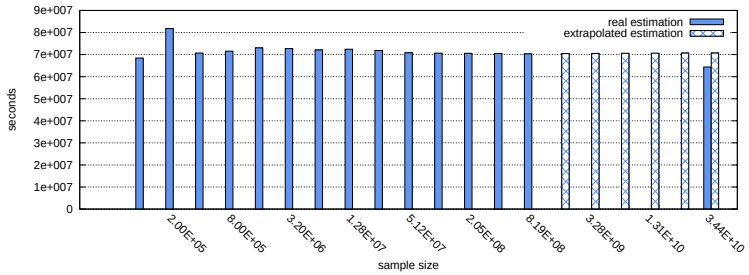

Fig. 4. Values of estimation F_{N_j} and extrapolated estimation $\tilde{F}_{N_j}$ for *Bivium14* (for the first instance from the series)

were proposed several methods to estimate the size of a backtracking search tree. These methods can be applied to estimate the time required to solve SAT using DPLL-based solvers. In [27, 10] some methods for estimating time to solve SAT using CDCL solvers were suggested. In these methods an estimation is computed based on the evaluation of a particular SAT instance and current results of CDCL solver.

The first work that used SAT solvers for cryptanalysis was [16]. The authors of [18, 5, 25, 24] presented some estimations of the time required for logical cryptanalysis of the Bivium cipher. The Monte Carlo term was first used in application to such estimations in [24].

Note that in the cited papers neither random variables nor their expected values estimated via Monte Carlo method were not strictly defined. It was first done in [22].

The main novelty of the results of the present article lies in the technique for improving the effectiveness estimations for SAT partitionings by means of extrapolating the values of predictive function to random samples of gradually increasing size. We showed the applicability of the corresponding technique on SAT instances encoding the inversion of some cryptographic functions.

Acknowledgments The research was funded by Russian Science Foundation (project No. 16-11-10046).

References

1. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
2. 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. LNCS, vol. 4176, pp. 171–186. Springer (2006)
3. Chen, H., Gomes, C.P., Selman, B.: Formal models of heavy-tailed behavior in combinatorial search. In: Walsh, T. (ed.) CP. Lecture Notes in Computer Science, vol. 2239, pp. 408–421. Springer (2001)
4. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA. pp. 151–158 (1971)
5. 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)
6. Feller, W.: An introduction to probability theory and its applications. Vol. II. Second edition, John Wiley & Sons Inc., New York (1971)
7. Gomes, C.P., Sabharwal, A.: Exploiting runtime variation in complete solvers. In: Biere et al. [1], pp. 271–288
8. Gomes, C.P., Selman, B., Crato, N., Kautz, H.A.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. J. Autom. Reasoning 24(1/2), 67–100 (2000)
9. Güneysu, T., Kasper, T., Novotný, M., Paar, C., Rupp, A.: Cryptanalysis with copacobana. IEEE Trans. Comput. 57(11), 1498–1513 (Nov 2008)
10. Haim, S., Walsh, T.: Online Estimation of SAT Solving Runtime. In: Büning, H.K., Zhao, X. (eds.) SAT. Lecture Notes in Computer Science, vol. 4996, pp. 133–138. Springer (2008)
11. Hyvärinen, A.E.J.: Grid Based Propositional Satisfiability Solving. Ph.D. thesis, Aalto University (2011)
12. Järvisalo, M., Junttila, T.A.: Limitations of restricted branching in clause learning. Constraints 14(3), 325–356 (2009)
13. Kilby, P., Slaney, J.K., Thiébaux, S., Walsh, T.: Estimating search tree size. In: AAAI. pp. 1014–1019. AAAI Press (2006)
14. King, J.C.: Symbolic execution and program testing. Commun. ACM 19(7), 385–394 (Jul 1976)
15. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning sat solvers. In: Biere et al. [1], pp. 131–153
16. Massacci, F., Marraro, L.: Logical Cryptanalysis as a SAT Problem. J. Autom. Reasoning 24(1/2), 165–203 (2000)
17. Maximov, A., Biryukov, A.: Two trivial attacks on trivium. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 4876, pp. 36–55. Springer (2007)
18. Mcdonald, C., Charnes, C., Pieprzyk, J.: Attacking Bivium with MiniSat. Tech. Rep. 2007/040, ECRYPT Stream Cipher Project (2007)
19. Metropolis, N., Ulam, S.: The Monte Carlo Method. J. Amer. statistical assoc. 44(247), 335–341 (1949)
20. Mironov, I., Zhang, L.: Applications of sat solvers to cryptanalysis of hash functions. In: Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing. pp. 102–115. SAT'06, Springer-Verlag, Berlin, Heidelberg (2006)
21. Otpuschemmikov, I., Semenov, A., Kochemazov, S.: Transalg: a tool for translating procedural descriptions of discrete functions to SAT. In: WCSE 2015-IPCE: Proceedings

of The 5th International Workshop on Computer Science and Engineering: Information Processing and Control Engineering. pp. 289–294 (2015)
22. Semenov, A., Zaikin, O.: Using monte carlo method for searching partitionings of hard variants of boolean satisfiability problem. In: Malyshkin, V. (ed.) Parallel Computing Technologies - 13th International Conference, Proceedings. Lecture Notes in Computer Science, vol. 9251, pp. 222–230. Springer (2015)
23. Semenov, A., Zaikin, O., Bespalov, D., Posypkin, M.: Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System. In: Malyshkin, V. (ed.) PaCT. Lecture Notes in Computer Science, vol. 6873, pp. 473–483. Springer (2011)
24. Soos, M.: Grain of Salt - an Automated Way to Test Stream Ciphers through SAT Solvers. In: Tools'10: Proceedings of the Workshop on Tools for Cryptanalysis. pp. 131–144 (2010)
25. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT Solvers to Cryptographic Problems. In: Kullmann, O. (ed.) SAT. LNCS, vol. 5584, pp. 244–257. Springer (2009)
26. Williams, R., Gomes, C.P., Selman, B.: Backdoors to typical case complexity. In: Gottlob, G., Walsh, T. (eds.) IJCAI. pp. 1173–1178. Morgan Kaufmann (2003)
27. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based Algorithm Selection for SAT. J. Artif. Intell. Res. (JAIR) 32, 565–606 (2008)