Using the algorithms for solving SAT to invert discrete functions with applications in cryptanalysis

Semenov A.A., ISDCT SB RAS

19.03.2021

$\{0,1\}^n, n \in \mathrm{N}$ is a set of arbitrary words of length n over $\{0,1\}$ (Boolean hypercube or Boolean hypercube of dimension n, the elements from $\{0,1\}^n$ are sometimes called Boolean vectors).

$\{0,1\}^+ = \bigcup_{n=1}^{\infty}\{0,1\}^n$ is a set of binary words of arbitrary finite length.

Hereinafter, by discrete functions we mean the functions of the kind $f: \{0,1\}^+ \to \{0,1\}^+$, and the functions of the kind $f: \{0,1\}^n \to \{0,1\}^m$, $n, m \in \mathrm{N}$.

For a function $f: \{0,1\}^n \to \{0,1\}^m$ the *preimage finding problem* or *inversion problem* is formulated as follows: given some $\gamma \in \mathrm{Range}\ f$ (we assume that $\mathrm{Range}\ f \neq \emptyset$), to find such $\alpha \in \{0,1\}^n$, that $f(\alpha) = \gamma$.

The majority of cryptanalysis problems can be viewed in the context of this problem.

Below we are mainly interested in discrete functions defined by programs (algorithms). From a theoretical point of view they are functions defined by programs for the binary Turing machine (such as the one described in [1]).

---

[1] Garey M., Johnson D. Computers and Intractability. 1979.

Formal basis.

Consider an arbitrary total function $f: \{0,1\}^+ \to \{0,1\}^+$, defined by program $M(f)$, the complexity of which is bounded from above by a polynomial in the length of an input. Let $f_n: \{0,1\}^n \to \{0,1\}^m$ be a function defined by $M(f)$ by constructing some $\gamma \in \{0,1\}^m$ for an arbitrary $\alpha \in \{0,1\}^n$.

The general inversion problem for f: given the text defining the program $M(f)$, an arbitrary $n \in \mathrm{N}$ and arbitrary $\gamma \in \mathrm{Range}\ f_n$ to find such $\alpha \in \{0,1\}^n$, that $f_n(\alpha) = \gamma$.

The basic theoretical result is the analogue of the Cook theorem (Cook-Levin theorem). In the context of the inversion problem for f it establishes its polynomial reducibility to the problem of finding a satisfying assignment of a satisfiable Boolean formula in Conjunctive Normal Form (CNF).

Let C be a CNF over the set of variables $X = \{x_1, \dots, x_N\}$ and f_C be a Boolean function defined by C

$$f_C: \{0,1\}^N \to \{0,1\}.$$

C is satisfiable, if there exists such $\alpha \in \{0,1\}^N$, that $f_C(\alpha) = 1$ (α is a satisfying assignment), otherwise C is unsatisfiable. The decision problem, that consists in determining whether an arbitrary C is satisfiable, is the classical NP-complete problem. Thus, the following problem is NP-hard: «for an arbitrary CNF to answer the question whether it is satisfiable or not, and if the answer is "yes" to construct an assignment of variables that satisfies this formula». Both problems are commonly denoted as SAT.

Despite its NP-hardness, in many specific cases SAT can be solved quite effectively. There are examples of C containing tens of thousands variables and hundreds thousands clauses, for which state of the art algorithms can determine whether it is satisfiable or not in minutes or even seconds.

It has been 20 years since the development of the CDCL algorithm [2], but in this short time, the SAT solvers based on it became the computational instrument widely employed to solve combinatorial problems from diverse areas, such as symbolic verification, software analysis, discrete optimization, bioinformatics, cryptanalysis, computational combinatorics, etc. In 2000 there was established the annual competition for SAT solvers (SAT competition) which are usually held in the context of "The International Conference on Theory and Applications of Satisfiability Testing".

In the present report the inversion problems described earlier are reduced to SAT. The examples show how the corresponding algorithms can be applied to cryptanalysis problems.

---

[2] Marques-Silva J., Sakallah K. GRASP: a search algorithm for propositional satisability. IEEE Transactions on Computers. 1999. Vol 48. Iss. 5. Pp. 506–521.

Reducibility theorem [3].

Theorem 1.

Let $f:\{0,1\}^+ \to \{0,1\}^+$ be a discrete function specified by a polynomial algorithm $M(f)$. Then there exists an algorithm M', which is polynomial in n, such that given the text of program $M(f)$ and the number $n, n \in \mathrm{N}$, it constructs CNF $C(f_n)$ over X, for which the following holds:

1. *For each $\gamma \in Range\ f_n$ and some $X^{in} \subset X, X^{out} \subset X$ КНФ $C(f_n, \gamma) = C(f_n)|_{\gamma/X^{out}}$ is satisfiable and from any of its satisfying assignments one can extract in polynomial time in n the assignment α of variables from X^{in}, such that $f_n(\alpha) = \gamma$;*
2. *For any $\gamma \notin Range\ f_n$ CNF $C(f_n, \gamma)$ is unsatisfiable.*

The CNF $C(f_n)$ is called template CNF.

By $C(f_n)|_{\gamma/X^{out}}$ we denote the result of substitution of the set of values γ to variables from set X^{out} in $C(f_n)$.

---

[3] Semenov A., Otpuschennikov I., Gribanova I., Zaikin O., Kochemazov S. Translation of algorithmic descriptions of discrete functions to SAT with applications to cryptanalysis problems. Logical Methods in Computer Science. 2020. Vol. 16. Iss. 1, pp. 29:1-29-42.

The idea used as a foundation of the proof of Theorem 1 is that based on $M(f)$ and the number n it is possible to effectively construct a circuit $G(f_n)$, that uses functional elements from the basis $\{\wedge, \neg\}$, and specifies the function $f_n: \{0,1\}^n \to \{0,1\}^m$. Then using this circuit one can use an algorithm with linear complexity in the number of nodes in this circuit to transition to a template CNF $C(f_n)$ using the so-called Tseitin transformations [4]. The sets X^{in} and X^{out}, that were referred above are the sets of variables associated with the inputs and the outputs of circuit $G(f_n)$.

Note that it is the Tseitin transformations that are the main tool for reducing many verification problems to SAT. In particular it is true in the case of the problem of equivalence checking for two circuits.

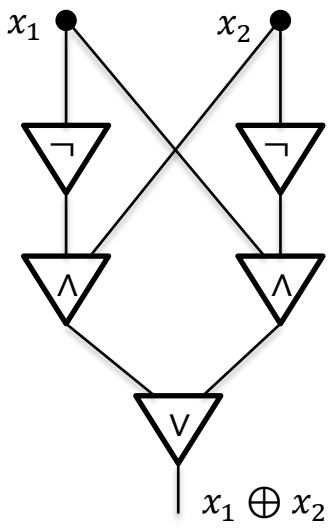
---

[4] Цейтин Г.С. О сложности вывода в исчислении высказываний// Записки научных семинаров ЛОМИ АН СССР. 1968. Т. 8. С. 234–259.

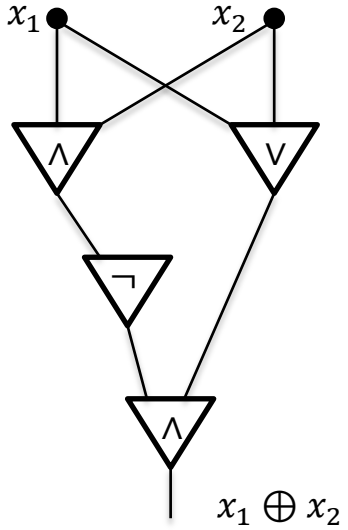
Equivalence checking problem.

Simple example: prove that the following two Boolean circuits are equivalent (i.e. they specify the same Boolean function) :

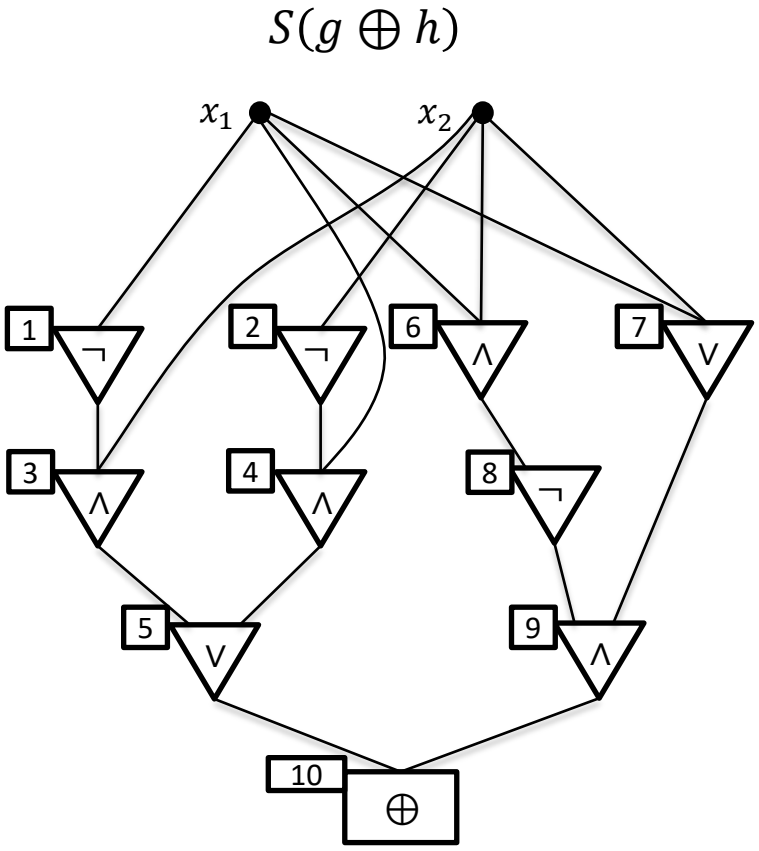
$S(g)$



$S(h)$



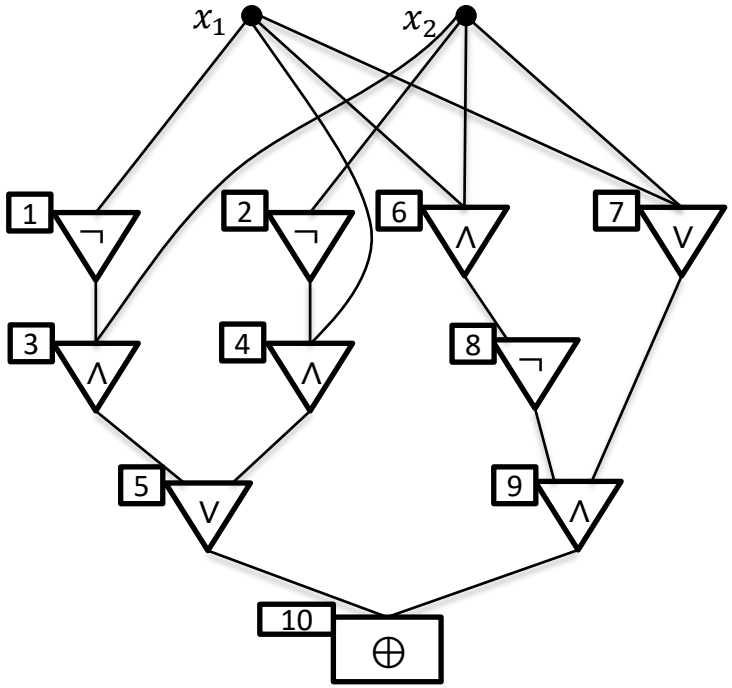
Reduce the equivalence checking problem for these two circuits to SAT. First , construct the following circuit.



«Glue» the inputs with the same names together and connect the outputs of original circuits via the XOR functional element.

It is clear that circuits $S(g)$ and $S(h)$ implement the same function if and only if the circuit $S(g \oplus h)$ is identically zero over $\{0,1\}^2$.

Next, construct CNF $C(g \oplus h)$ for circuit $S(g \oplus h)$. It is done via Tseitin transformations in linear time in the size of $S(g \oplus h)$. The idea of Tseitin transformations is very simple: with each node of a circuit except input nodes we associate a new Boolean variable and a Boolean formula that connects the inputs of this node with its output via logical equivalence.



1. $y_1 \equiv \neg x_1$
2. $y_2 \equiv \neg x_2$
3. $y_3 \equiv y_1 \wedge x_2$
4. $y_4 \equiv y_1 \vee y_2$
5. $y_5 \equiv x_1 \wedge x_2$
6. $y_6 \equiv x_1 \wedge x_2$
7. $y_7 \equiv x_1 \vee x_2$
8. $y_8 \equiv \neg y_6$
9. $y_9 \equiv y_7 \wedge y_8$
10. $y_{10} \equiv y_5 \oplus y_9$

Next, represent each formula from the constructed set as CNF over the corresponding set of Boolean variables (for example, the formula $y_1 \equiv \neg x_1$ transforms into $C_{y_1} = (y_1 \vee x_1) \wedge (\neg y_1 \vee \neg x_1)$, formula $y_3 \equiv y_1 \wedge x_2$ becomes CNF $C_{y_3} = (y_3 \vee \neg y_1 \vee \neg x_2) \wedge (\neg y_3 \vee y_1) \wedge (\neg y_3 \vee x_2)$, etc.).

Then, consider the CNF

$$C(g \oplus h) = C_{y_1} \wedge \cdots \wedge C_{y_{10}} \wedge y_{10}$$

From the properties of the Tseitin transformations it follows that the circuit $S(g \oplus h)$ represents the Boolean function that is identically zero over $\{0,1\}^n$ (in our case over $\{0,1\}^2$), if and only if CNF $C(g \oplus h)$ is unsatisfiable.

If the circuits are not equivalent then this CNF is satisfiable and there exists its satisfying assignment – the certificate that proves that two circuits are not equivalent (on the corresponding input values one scheme outputs 0, while the other outputs 1).

Cryptanalysis. Again, let $f: \{0,1\}^n \to \{0,1\}^m$ be a discrete function that transforms binary words of length n into binary words of length m, which is defined everywhere over $\{0,1\}^n$ and is specified by an algorithm $M(f)$. Given $\gamma \in Range\ f \subseteq \{0,1\}^m$, we need to find $\alpha \in \{0,1\}^n$, such that $f(\alpha) = \gamma$ (inversion problem).

This problem is typical for cryptography. Example: Let $M(f_{SHA-256})$ be the program that defines the SHA-256 hash function:

$$f_{SHA-256}: \{0,1\}^{512} \to \{0,1\}^{256}$$

Now let γ_k denote the image of function $f_{SHA-256}$, in which k first bits are zeros. Then the problem of finding *some* pair (α, γ_k), such that $f_{SHA-256}(\alpha) = \gamma_k$ is essentially a cryptocurrency mining problem.

The functions of the kind f, that are employed in cryptography, are typically very fast to compute. Therefore, the problems of their inversion can be effectively reduced to SAT. There are several software systems that make it possible to automatically perform such reductions. One of them is the Transalg system [5].

---

[5] Otpuschennikov I., Semenov A., Gribanova I., Zaikin O. Kochemazov S. Encoding Cryptographic Functions to SAT Using Transalg System // European Conference on Artificial Intelligence (ECAI) 2016, Frontiers in Artificial Intelligence and Applications. 2016. Vol. 285. Pp. 1594–1595.

A5/1 keystream generator

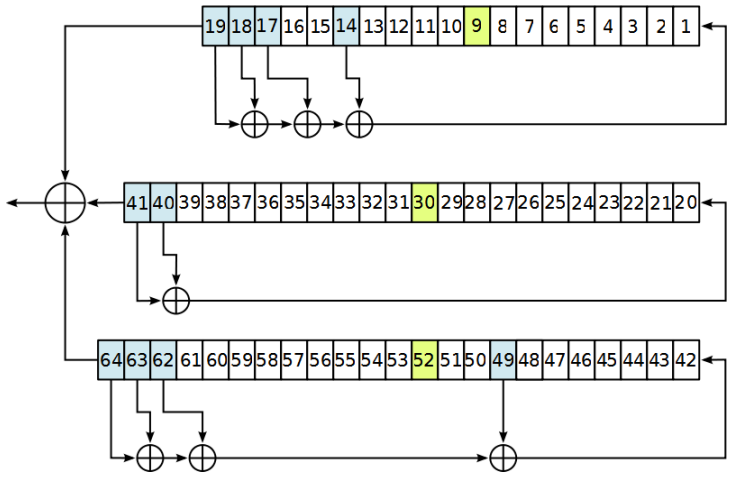
Keystream generator is a discrete function

$$f_n: \{0,1\}^n \to \{0,1\}^+$$

Cryptanalysis problem: given a known algorithm for computing f_n and a fragment of keystream to find the initial values of generator registers.

$\alpha = (\alpha_1, \dots, \alpha_n)$ — input (initial) value (secret key).
$\gamma = (\gamma_1, \dots, \gamma_m)$ — keystream fragment of length m.
The goal is given y to use the knowledge about the algorithm for computing f_n to find x.

A5/1 keystream generator



A5/1 is based on three LFSRs with the following feedback polynomials:
LFSR1:$X^{19} + X^{18} + X^{17} + X^{14} + 1$
LFSR2:$X^{22} + X^{21} + 1$
LFSR3:$X^{23} + X^{22} + X^8 + 1$
The size of the secret key is 64 bits. Let $b_1^\tau, b_2^\tau, b_3^\tau$ be the values of cells number 9, 30 and 52 at time moment τ. The register number $j \in \{1,2,3\}$ is shifted at moment τ if and only if

$$b_j^\tau = majority(b_1^\tau, b_2^\tau . b_3^\tau)$$
$$majority(x, y, z) = x \wedge y \vee y \wedge z \vee x \wedge z$$

Example. Encoding A5/1 to SAT

```
define len 128;
__in bit regA[19];
__in bit regB[19];
__in bit regC[19];
__in out stream[len];
int midA = 8;
int midB = 8;
int midC = 8;
//Shift LFSR A
bit shift_rslosA()
{
     bit x = regA[18];
     bit y = regA[18]^regA[17]^regA[16]^regA[13];
     for(int j = 18; j > 0; j=j-1)
     {
          regA[j] = regA[j-1];
     }
     regA[0] = y;
     return x;
}
//Shift LFSR B
bit shift_rslosB()
{
     bit x = regB[21];
     bit y = regB[21]^regB[20];
     for(int j = 21; j > 0; j=j-1)
     {
          regB[j] = regB[j-1];
     }
     regB[0] = y;
     return x;
}
```

```
//Shift LFSR C
bit shift_rslosC()
{
     bit x = regC[22];
     bit y = regC[22]^regC[21]^regC[20]^regC[7];
     for(int j = 22; j > 0; j=j-1)
     {
          regC[j] = regC[j-1];
     }
     regC[0] = y;
     return x;
}

bit majority(bit x, bit y, bit z){
     return x & y | x & z | y & z;
}

void main(){
     for (int i=0; i < len; i = i+1){
          bit maj = majority(regA[midA],
regB[midB],regC[midC]);
          if (!(maj^regA[midA])) shift_regA();
          if (!(maj^regB[midB])) shift_regB();
          if (!(maj^regC[midC])) shift_regC();
          stream[i] = regA[18]^regB[21]^regC[22];
     }
}
```

The corresponding CNF contains 39936 clauses over the set of 8768 variables.

The problem of cryptanalysis of A5/1 in the SAT form was solved in [6] (using a specific distributed computing environment). The employed parallelization technique was later developed into a general method applicable to construction of decompositions for a wide variety of SAT instances.

A new class of cryptographic attacks of the “guess and determine” kind, including the state of the art ones, was described in [8].

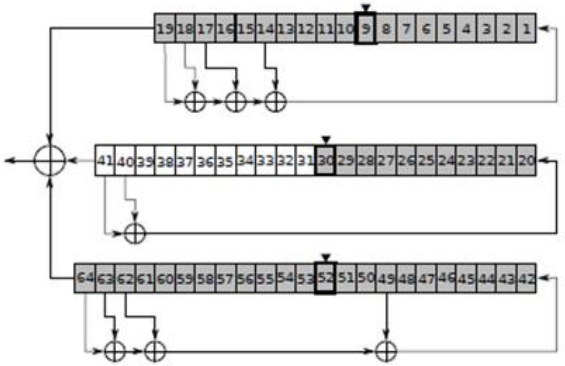
---

[6] Semenov A., Zaikin O., Bespalov D., Posypkin M. Parallel logical cryptanalysis of the generator A5/1 in BNB-grid system. Lecture Notes in Computer Science. 2011. Vol. 6873. Pp. 473-483.

[7] Semenov A., Zaikin O. Algorithm for finding partitionings of hard variants of Boolean satisfiability problem with application to inversion of some cryptographic functions. SpringerPlus, 5(1). Pp. 1-16, 2016.

[8] Semenov A., Zaikin O., Otpuschennikov I., Kochemazov S., Ignatiev A. On cryptographic attacks using backdoors for SAT. In Proc. of the Thirty-Second AAAI Conference on Articial Intelligence (AAAI), pages 6641-6648, 2018.

A guess and determine attack looks as follows. Given a system of equations (or a Boolean formula) that encodes the cryptanalysis problem, one substitutes into it the values of several bits. These substitutions weaken the original system making it linear. A good example of such attack is the Anderson's attack [9] on the A5/1 generator. The set of guessed bits described by Anderson is shown on the following figure.



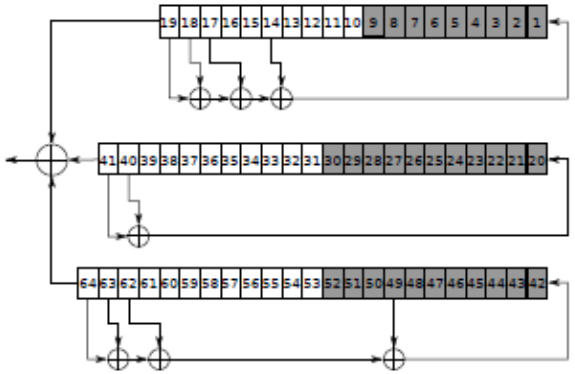
This set contains 53 bits (marked gray). Assigning any values to all variables corresponding to gray cells, when taking into account several specific features of the generator (register shifts) results in a system of linear equations over GF(2). Thus, in the Anderson's attack one needs to solve 2^{53} systems of linear equations. Using several modern GPUs this attack shows very reasonable time.

Another approach used in guess-and-determine attacks is to use some combinatorial algorithms (designed to tackle instances of NP-hard problems) to solve weakened cryptanalysis equations: in this case we do not require the corresponding problems to be polynomially solvable. Instead, we expect that the algorithm applied to the problem will manage to cope with the vast majority of the subproblems.

---

[9] Anderson, R.: A5 (was: Hacking digital phones). Newsgroup Communication (1994), http: //yarchive.net/phone/gsmcipher.html

In the paper [6] cited above we used the SAT solving algorithms for this purpose, and the set of guessed bits looked as follows:



This set contains only 31 variables, but to mount this attack one has to use the computing cluster or with general purpose processing units (CPUs) because CDCL algorithms do not work well on GPUs. Volunteer computing project, such as SAT@home [10], can be viewed as a good alternative of a computing cluster.

In paper [7] (actually, the first publication in this direction was published in Russian back in 2012) it was proposed to automatically construct guess-and-determine attacks by reducing the problem of finding an attack with a good runtime estimation to a black-box optimization problem. However, in case of the attacks from [7] their runtime estimations do not have any theoretical justifications that would prove their accuracy (however, for many functions the estimations constructed in this manner turn out to be quite good in practice). Such theoretical justifications can be constructed for a class of guess-and-determine attacks proposed in [8]. Let us consider them in more detail.

---

[10] Posypkin M., Semenov A., Zaikin O. Using BOINC Desktop Grid to Solve Large Scale SAT Problems. Computer Science. 13 (1). 2012. Pp. 25-34.

[7] Semenov A., Zaikin O. Algorithm for finding partitionings of hard variants of Boolean satisfiability problem with application to inversion of some cryptographic functions. SpringerPlus, 5(1). Pp. 1-16, 2016.

[8] Semenov A., Zaikin O., Otpuschennikov I., Kochemazov S., Ignatiev A. On cryptographic attacks using backdoors for SAT. In Proc. of the Thirty-Second AAAI Conference on Articial Intelligence (AAAI), pages 6641-6648, 2018.

Once again, consider the problem of inversion for function

$$f_n: \{0,1\}^n \to \{0,1\}^m,$$

specified by some algorithm. Let C_{f_n} be a template cnf for f_n.

Notation:

X – the set of all variables from C_{f_n}, X^{in} – the set of variables from X, which encode the inputs of f_n.

Fact ([11],[12], etc.): the substitution of an arbitrary $\alpha \in \{0,1\}^{|X^{in}|}$ to C_{f_n} and subsequent application of the Unit Propagation rule [13] results in derivation (in linear time) of values of all variables from X, including the variables that encode the value of a considered function on input α. Let us refer to the values of variables from X, that were constructed in such a way, as to values induced by α.

For an arbitrary $B \subseteq X$ denote by β_α the set of values of variables from B, that was induced by α. By γ_α denote the value of f_n on input α ($\gamma_\alpha = f_n(\alpha)$). Let $C_{f_n}(\beta_\alpha, \gamma_\alpha)$ be a CNF constructed by substituting the values β_α and γ_α into C_{f_n}. Assume that A is an arbitrary deterministic (possibly incomplete) algorithm for solving SAT.

Now let $\Omega = \{0,1\}^{|X^{in}|} = \{0,1\}^n$ be a space of elementary events, over which a uniform distribution is specified. With fixed B and $t > 0$ associate the random variable ζ_B, the value of which on an arbitrary input $\alpha \in \{0,1\}^n$ is equal to 1 if algorithm A finds the satisfying assignment of CNF $C_{f_n}(\beta_\alpha, \gamma_\alpha)$ in time at most t (denote this fact as $time\left(A\left(C_{f_n}(\beta_\alpha, \gamma_\alpha)\right) = \alpha\right) \leq t$). Otherwise, ζ_B is equal to 0 on an input α. Thus, ζ_B is a Bernoulli random variable.

---

[11] Bessiere C., Katsirelos G., Narodytska N., Walsh T. Circuit complexity and decompositions of global constraints. In IJCAI 2009, Proceedings of the 21st International Joint Conference on Articial Intelligence (IJCAI), Pp. 412-418, 2009.

[12] Семенов А.А. Декомпозиционные представления логических уравнений в задачах обращения дискретных функций. Известия РАН. Теория и системы управления. 2009. №5. С. 47-61.

[13] Marques-Silva J., Lynce I., Malik S. Conflict-Driven Clause Learning SAT Solvers. In handbook of Satisfiability. Pp. 131-153. 2009. IOS Press.

For each particular $B \subseteq X$ define the following value:

$$\rho_B(t) = \frac{\#\{\alpha | time\ (A(C_{f_n}(\beta_\alpha, \gamma_\alpha)) = \alpha) \leq t\}}{2^n}$$

The numerator of the formula contains the number of such α, that algorithm A in time $\leq t$ can find an assignment which satisfies $C_{f_n}(\beta_\alpha, \gamma_\alpha)$. In this context it is clear that $\rho_B(t)$ is the probability of success for variable ζ_B, thus, the spectrum of ζ_B is $\{1,0\}$ and the distribution is $\{\rho_B(t), 1 - \rho_B(t)\}$. Therefore, $\mathrm{M}[\zeta_B] = \rho_B(t)$.

For a fixed $t > 0$ let us refer to an arbitrary set B, $B \subseteq X$, $|B| = s$, such that $\rho_B(t) > 0$, as to Inverse Backdoor Set, IBS with parameters $(\rho_B(t), s, t)$.

Let B be an IBS with $\rho_B(t) > 0$ for function f_n. Let us describe a guess-and-determine attack on f_n, which is based on B.

Assume that we know $\gamma \in Range\ f_n$: $\gamma = \gamma_\alpha$ for some $\alpha \in \{0,1\}^n$. We need to find this α. For each $\beta \in \{0,1\}^{|B|}$ consider CNF $C_{f_n}(\beta, \gamma_\alpha)$ and launch algorithm A on this CNF, terminating it as soon as the runtime exceeds t. For some β: $\beta \neq \beta_\alpha$ A may even manage to prove unsatisfiability of $C_{f_n}(\beta, \gamma_\alpha)$ in time $\leq t$. The probability that A will find a satisfying assignment for $C_{f_n}(\beta_\alpha, \gamma_\alpha)$ in time $\leq t$ is $\rho_B(t)$.

Thus, when analyzing an arbitrary $\gamma \in Range\ f_n$ we need to traverse the whole set B. If it did not yield a result (when $\rho_B(t)$ is small), one can pick the next γ (it is assumed that each separate γ is induced by an independently chosen $\alpha \in \{0,1\}^n$).

This way it is possible to repeat the traversal of B again and again for different $\gamma \in Range\ f_n$. Then the probability that in at least one out of Q cases the traversal of $\{0,1\}^{|B|}$ will «hit» the correct β_α, is

$$P(Q) = 1 - \big(1 - \rho_B(t)\big)^Q.$$

Therefore, the runtime required to traverse $\{0,1\}^s$, $s = |B|$ at most Q times (i.e. the attacks runtime) is

$$t \cdot 2^s \cdot Q.$$

Question: how to choose Q in such a way that the resulting attack can be considered successful?

It is clear, that when $Q \geq \left\lceil \frac{3}{\rho_B(t)} \right\rceil$ $P(Q) > 0{,}95$, and when $Q \geq \left\lceil \frac{4}{\rho_B(t)} \right\rceil$ $P(Q) > 0{,}98$. Hereinafter, we consider the runtime of a successful attack to be

$$t \cdot 2^s \cdot \left\lceil \frac{3}{\rho_B(t)} \right\rceil$$

Question: how to evaluate $\rho_B(t)$?

Important fact: Since $\rho_B(t) = \mathrm{M}[\zeta_B]$, it is possible to estimate $\rho_B(t)$ by estimating $\mathrm{M}[\zeta_B]$ via the Monte Carlo method [14],[15]. I.e.: 1) Observe the values of ζ_B on a sample of size N: generate N independent inputs α, then knowing β_α and γ_α launch A on $C_{f_n}(\beta_\alpha, \gamma_\alpha)$; if A finds satisfying assignment in time $\leq t$, then $\zeta_B = 1$, otherwise $\zeta_B = 0$; 2) compute the sum of values of ζ_B over the sample and divide it into N.

---

[14] Metropolis N., Ulam S. The Monte Carlo Method. J. Amer. statistical assoc., 44(247):335-341, 1949.
[15] Ермаков С.М. Метод Монте-Карло и смежные вопросы. М.: «Наука». 1971.

From the Chebyshev's inequality we have:

$$\Pr\left[\left|\rho_B(t) - \frac{1}{N}\cdot\sum_{j=1}^{N}\zeta_B^j\right| \le \epsilon\right] \ge 1 - \frac{1}{4\cdot\epsilon^2\cdot N}$$

Therefore, we can guarantee an arbitrarily good accuracy of the estimation of $\rho_B(t)$ (when $\rho_B(t) \neq 0$) by increasing the number of observations of ζ_B.

Thus, for a given IBS B we can consider the runtime of a corresponding attack to be equal to:

$$t \cdot 2^{|B|} \cdot \frac{3N}{\sum_{j=1}^{N}\zeta_B^j} \qquad (*)$$

The formula (*) is important because it allows one to view the problem of finding an attack (of the considered kind) with best effectiveness as a pseudo-Boolean optimization problem.

Indeed, represent an arbitrary set $B \subseteq X$ by vector λ of size $|X|$ (ones in this vector correspond to variables from X, which are present in B). Consider the function

$$\Psi_{C_{f_n},A}: \{0,1\}^{|X|} \to \mathrm{R} \qquad (**)$$

The value of (**) is computed for an arbitrary $\lambda \in \{0,1\}^{|X|}$ as follows: construct a set $B = B_\lambda$, specified by vector λ; generate a sample of N random inputs for f_n and observe the random variable ζ_B for N times; compute (*) (runtime estimation for an attack).

Consider a problem of minimization of pseudo-Boolean black-box function of the kind (**). One can use many different metaheuristic algorithm for solving it. In [8] we used the tabu search for this purpose, in [16] we applied (1+1)-Evolutionary Algorithm and a special variant of a genetic algorithm. The paper [17] contains a review on the application of metaheuristic algorithms for pseudo-Boolean black-box optimization to the problems of finding decompositions of hard SAT instances (including the problems of constructing guess-and-determine attacks on cryptographic functions).

---

[8] Semenov A., Zaikin O., Otpuschennikov I., Kochemazov S., Ignatiev A. On cryptographic attacks using backdoors for SAT. In Proc. of the Thirty-Second AAAI Conference on Articial Intelligence (AAAI), pages 6641-6648, 2018.

[16] Pavlenko, A., Semenov, A., Ulyantsev, V.: Evolutionary computation techniques for constructing SAT-based attacks in algebraic cryptanalysis. LNCS. 2019. Vol. 11454 (EvoApplications). Pp. 237–253.

[17] Semenov A., Zaikin O., Kochemazov S. Finding Effective SAT Partitioning via Black-Box Optimization. In book "Black Box Optimization, Machine Learning, and No-Free Lunch Theorems". Springer Optimization and Its Applications 170. (2021) (In Print)

Some computational results

	AES-128, 2,5 rounds, One portion=1KP	PRESENT-80 6 rounds One portion=2KP (finding 80-bit secret key)
SAT immunity estimation found by minimization of function (**)	$1{,}45 \times 10^{15}$ sec. |B|=42 SAT-solver ROKK (1 th.) 3KP The memory requirements are negligible.	$8{,}49 \times 10^{15}$ sec. |B|=39 SAT-solver ROKK (1 th.) 750 KP $2{,}85 \times 10^{15}$ sec. |B|=39 SAT-solver Painless (36 th.) 10 KP
Runtime estimation of the attack from [18]	2^{80} AES-operations; Approximately corresponds to $3{,}08 \times 10^{16}$ sec. |B|=80 (supposedly) 2KP Memory requirements: 2^{80} bits	---
Runtime estimation of the attack from [19]	---	$6{,}60 \times 10^{18}$ sec. |B|=52 Enhanced Binary Characteristic Set Algorithm (EBCSA) 1 KP (!)

[18] Bouillaguet C., Derbez P., Fouque P.-A. Automatic search of attacks on round-reduced AES and applications // Lecture Notes in Computer Science. 2011. Vol. 6841 (CRYPTO). Pp. 169–187.
[19] Yeo S.L., Li Z., Khoo K., Low Y.B. An Enhanced Binary Characteristic Set Algorithm and Its Applications to Algebraic Cryptanalysis // Lecture Notes in Computer Science. 2017. Vol. 10355 (ACNS). Pp. 518–536.

Some computational results. Estimations from paper [16]

Table 1. Experimental results for six cryptographic algorithms. The leftmost column contains the name of a keystream generator for which the cryptanalysis problem is considered. The remaining table is divided into three sections corresponding to strategies (1+1)-EA, GA and Tabu Search. The first column of each section contains the power of guessed bits set, which corresponds to the best guess-and-determine attack found. The second column contains a time estimation (in seconds) for the attack

	(1+1)-EA		GA		Tabu Search	
	power of guessed bits set	G&D attack (seconds)	power of guessed bits set	G&D attack (seconds)	power of guessed bits set	G&D attack (seconds)
Trivium-Toy 64/75	21	**3.19e+07**	22	5.36e+07	17	4.30e+07
Trivium-Toy 96/100	33	1.28e+13	40	**2.09e+12**	34	3.14e+12
Bivium 177/200	32	2.60e+12	39	**1.49e+12**	40	4.29e+12
ASG 72/76	9	5604.8	8	6155.19	8	**5601.33**
ASG 96/112	13	6.76e+06	16	**3.72e+06**	14	3.95e+06
ASG 192/200	47	2.27e+18	44	2.84e+17	47	**1.14e+16**

[16] Pavlenko, A., Semenov, A., Ulyantsev, V.: Evolutionary computation techniques for constructing SAT-based attacks in algebraic cryptanalysis. LNCS. 2019. Vol. 11454 (EvoApplications). Pp. 237–253.

Some computational results

	Grain 1.0 **Full-round** **One portion=1KP** **(finding 80-bit secret key)**	**PRESENT-80** **6 rounds** **One portion=2KP** **(finding 80-bit secret key)**
SAT immunity estimation found by minimization of function (**)	$1{,}27 \times 10^{27}$ sec. |B|=79 SAT-solver ROKK (1 th.) 2KP	$1{,}09 \times 10^{22}$ sec. |B|=61 SAT-solver ROKK (1 th.) 6KP

The results of these experiments are especially curious. PRESENT [20] is a popular lightweight block cipher. Grain 1.0 [21] is a stream cipher with very high encryption speed. Traditionally, block ciphers are viewed as more resistant than keystream ones. However, here we see that the runtime estimation for an attack on PRESENT is almost 10^5 times smaller than that of the attack on Grain!

[20] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. CHES 2007. LNCS, vol. 4727, pp. 450–466 (2007)

[21] Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. Int. J. Wire. Mob. Comput. 2(1), pp. 86–93 (2007)

On the definition of SAT immunity

The first definition of SAT immunity was introduced by N.T. Courtois with co-authors in [22]

Definition (SAT Immunity or satisfaction immunity). We define the SAT Immunity of a given cipher and for M plaintext/ciphertext pairs of the cipher as being the smallest possible number of key bits which can be fixed so that given M KP we can compute the secret key by the best available SAT solver in a relatively short time, say less than 1000 seconds." (N.T. Courtois et al. [22])

Further, N. Courtois notes, that "*This notion is as precise as it can be*", and it is impossible to agree with this comment. And the problem here is not even in the performance of SAT solving algorithms and the underlying hardware. The definition is incorrect in principle. Let us give a simple example.

---

[22] Courtois N.T., Gawinecki J.A., Song, G. Contradiction immunity and guess-then-determine attacks on GOST // Tatra Mountains Mathematical Publications. 2012. Vol. 53. Pp. 65–79.

Imagine the following two situations:

1. For a considered cryptanalysis problem we found a set of guessed bits of size, say, 40. Assume that fixing the values of the corresponding variables results in SAT instances that can be solved by a SAT solver under 0.001 seconds.;
2. We found a set of guessed bits of size 32, but solving an arbitrary SAT instance constructed by fixing the values of corresponding variables takes from 0,91 to 0,99 seconds.

Question: what is the UNSAT immunity of the considered cipher? Is it 40? Or32? Seemingly, it is 32 in accordance with the definition by Courtois. However, the total runtime of the corresponding guess-and-determine attacks is:

<109 951 16 28 seconds in the first case ($|B| = 40$)

>386 547 05 66 seconds in the second case ($|B| = 32$)

Thus, it is necessary to define the resistance of ciphers to algebraic attacks (including the attacks that employ SAT solvers) by other means.

In light of the above, it is natural to consider the following value to be the correct estimation of SAT immunity (for a fixed algorithm A and time limit t) :

$$\min_{B \in 2^X} \left(2^{|B|} \cdot t \cdot \left\lceil \frac{3}{\rho_B(t)} \right\rceil \right)$$

Thank you for your attention!