Modeling of algebraic analysis of PRESENT cipher by SAT solvers

Ekaterina Maro
Southern Federal University, Chehova str., 2, Taganrog city, 347922, Russia
E-mail: eamaro@sfedu.ru

Abstract. Paper presents approaches to algebraic analysis of the reliability of symmetric block ciphers by solving Boolean satisfiability (SAT) problem. A description of the application of decision search algorithms by the DPLL method is given. The principle of formation of the SAT problem for algebraic analysis of PRESENT cipher is proposed, as well as the parameters of the obtained SAT problems for a different number of cipher rounds and plaintexts.

1. Algebraic analysis

Referring to the work of C. Shannon [1], we can say that to analyze a reliability of information security algorithms, it is necessary to do “as much work as to solve a system of equations with a large number of unknowns”. For a long time main attention for block encryption algorithms reliability was focused on statistical methods. Algebraic methods that describe global approaches to the problem of analyzing reliability of algorithms were not sufficiently considered. The methods of algebraic analysis are based on use of nonlinear primitives of encryption algorithms in order to describe the encryption algorithm by systems of equations connecting desired key and known data (plain texts, cipher texts and etc.).

Regard to block encryption algorithms algebraic analysis can be presented in two stages. At the first stage, it is necessary to present an information protection algorithm and some additional information about algorithm in the form of systems of nonlinear equations over the field GF(2) or another finite field. The second stage consists in solving the system of equations and obtaining the secret key from the solution of the system.

Various approaches to solving nonlinear systems of Boolean equations have been developed. There are three main families of algorithms for solving systems of Boolean equations that are often used in the assessment of information security [2]:

- SAT-Solvers: MiniSat2, CryptoMiniSat, CaDiCaL, Plingeling, etc.
- Methods based on Gröbner basis: Buchberger algorithm, F4, F5, etc.
- Methods based on the linearization principle: relainization, extended linearization (XL), extended sparse linearization, ElimLin, etc.

In this paper we consider the possibility of using SAT solvers to analyze security properties for lightweight block encryption standard PRESENT [3].

2. Solving a system of Boolean algebraic equations using SAT-solvers

Satisfiability (SAT) problem consists of two main subtasks - checking the satisfiability of an arbitrary Boolean function represented in conjunctive normal form (CNF) and finding a set of values at which...
such a CNF is performed. The majority of SAT-solvers are based on the DPLL algorithm (Devis, Putnam, Logemann, Loveland), which was proposed in 1962 precisely to determine the feasibility of Boolean formulas written in conjunctive normal form, i.e. to solve the SAT-problem. For more than half a century, the DPLL algorithm has been the basis for most effective solvers for SAT-problems. The basic idea of the DPLL algorithm is to apply methods to bypass the search tree in depth and use the single clause rule [4].

Each time a variable is assigned and CNF is simplified according to the following rules:
- Variable propagation. If only one variable remains in the clause, then assign it such a value that the clause will become true (put the variable in the subset A, if there is no negation in the clause, or put it in the set B - if there is a negative).
- Elimination of “pure variables”. If a variable is found in a formula with only negation or only without negation, then it is called “pure” and it can be assigned such a value that it is always “true”, thereby reducing the number of free variables for analysis.

Consider an example of finding solutions of CNF using the DPLL algorithm. Take for example the formula:

$$(x_1 \lor x_3) \land (x_2 \lor x_3 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_4) \land (\overline{x}_2 \lor x_3) \land (x_1 \lor \overline{x}_3 \lor x_4)$$

Using the DPLL algorithm, 4 solution sets were found. The search tree of solution sets for the formula is shown in figure 1.

![DPLL algorithm for searching sets of CNF solutions](image)

**Figure. 1.** DPLL algorithm for searching sets of CNF solutions.

To find all possible solutions, the second rule (elimination of “pure” variables) should be excluded from the implementation of the DPLL algorithm. Then the search tree of the solution set for the formula (1) will have the structure shown in figure 2.
Algebraic analysis using SAT-solvers can be represented by an algorithm consisting of three main stages:

- Representation of the transformation of information security in the form of a system of Boolean equations in the in algebraic normal form (ANF).
- Change the equations system from ANF to CNF.
- Search for a set of solutions using SAT-solvers.

After the formation of Boolean equations system the expression of the input and output vectors of the S-box is performed through known plain texts using knowledge of encryption algorithm structure. System of Boolean equations presented in ANF is obtained at this stage.

To be able to use SAT-solvers, the formed system of Boolean equations in the ANF should be transferred to CNF. The algorithm for the representation of equations formed for block ciphers in the ANF should first be simplified.

We use the following conversion algorithm for representation ANF to CNF [5]:

- Replacing a constant of 1 with a new unknown, since CNF should not contain constants.
- Convert the original nonlinear system to a linear form by replacing all products of unknowns with new variables.
- Fragmentation of long chains formed as addition modulo two of unknowns into substrings of shorter length (for example, 4 unknowns in total).
- Representation of the transformed system in CNF.

In general terms $2^p-1$ clause is required to represent sum of unknowns with a length of $p$.

Then the system of equations in the CNF is transferred to the SAT-solver algorithm. We propose to apply CaDiCaL and Plingeling solvers as the most efficient and suitable SAT-solvers for such tasks.

---

**Figure 2.** DPLL algorithm for searching set of CNF solutions without using the second rule.
3. Mathematical models of algebraic analysis of PRESENT cipher

PRESENT is a cipher based on substitution-permutation network (SP-network) [3]. PRESENT consists of 31 rounds, each of which applies the operations of addition modulo 2 with the round key, substitution with one of 16 S-boxes based on 4-bit vectors, and bit permutations. We consider algebraic analysis of PRESENT with independent round keys (without using the structure of round key generation algorithm). The general structure of PRESENT is shown in Fig. 3. Substitution and permutation transforms are defined by Table 1 and Table 2. Examples of the implementation of the algebraic analysis of the PRESENT cipher are described in [6-8].

| Table 1. PRESENT substitutions |
|---------------------------------|
| X    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| S(X) | 12| 5 | 6 | 11| 9 | 0 | 10| 13| 3 | 14| 15 | 8 | 4 | 7 | 1 | 2 |

| Table 2. PRESENT permutations |
|---------------------------------|
| b    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| H(b) | 0 | 16| 32| 48| 1 | 17| 33| 49| 2 | 18| 34| 50| 3 | 19| 35| 51 |
| b    | 16| 17| 18| 19| 20| 21| 22| 23| 24| 25| 26| 27| 28| 29| 30| 31 |
| H(b) | 4 | 20| 36| 52| 5 | 21| 37| 53| 6 | 22| 38| 54| 7 | 23| 39| 55 |
| b    | 32| 33| 34| 35| 36| 37| 38| 39| 40| 41| 42| 43| 44| 45| 46| 47 |
| H(b) | 8 | 24| 40| 56| 9 | 25| 41| 57| 10| 26| 42| 58| 11| 27| 43| 59 |
| b    | 48| 49| 50| 51| 52| 53| 54| 55| 56| 57| 58| 59| 60| 61| 62| 63 |
| H(b) | 12| 28| 44| 60| 13| 29| 45| 61| 14| 30| 46| 62| 15| 31| 47| 63 |

The round subkey generation algorithm from the original private key K = (k79, ..., k0) consists of the following transformations:
- Rotation by 61 position to the left;
- Substitution of 4 leftmost bits in S-boxes;
- Adding the current round index to the bits (k19, ..., k15) modulo 2.

Substitutions in a single block can be described by 21 linearly independent equations. In the research work following Boolean nonlinear equations were formed (where x, and y, are respectively the input and output bits of the S-box):

x1=x2 ⊕ x0 ⊕ x1 ⊕ x3 ⊕ y3=0,
x0=x1 ⊕ x0=x2 ⊕ x0+x1 ⊕ y0 ⊕ y2 ⊕ y3 ⊕ 1=0,
x0=x3 ⊕ x1=x3 ⊕ x1=y0 ⊕ x0=y1 ⊕ x0=y2 ⊕ x1 ⊕ x2 ⊕ y2=0,
x0=x3 ⊕ x0=y0 ⊕ x1=y1 ⊕ x0 ⊕ x2 ⊕ y2=0,
x0=x2 ⊕ x0=x0 ⊕ x0=x2 ⊕ x1=x2 ⊕ x3 ⊕ x0 ⊕ x2 ⊕ y2 ⊕ y3=0,
x0=x1 ⊕ x1=x3 ⊕ x1=x3+ x0+ x3 + y3=0,
x0=x2 ⊕ x0=y0 ⊕ x0=y2 ⊕ x0=y3=0,
x0=x1 ⊕ x0=x2 ⊕ x0=x3 ⊕ x0=x2 ⊕ x0=x2 ⊕ x0=x2 ⊕ x3 ⊕ y1 ⊕ y2 ⊕ y3 ⊕ 1=0,
x0=x1 ⊕ x0=x2 ⊕ x0=x3 ⊕ x0=x2 ⊕ x0=x2 ⊕ x3 ⊕ y1 ⊕ y3 ⊕ 1=0,
x0=x3 ⊕ x0=x3 ⊕ x0=x0 ⊕ x0=x1 ⊕ x2=x2 ⊕ x1 ⊕ y1 ⊕ y2 ⊕ 1=0,
x0=x2 ⊕ x2=x3 ⊕ x2=x3 ⊕ x2=x3 ⊕ x2=x3 ⊕ x2=x3 ⊕ x2=x3 ⊕ y1 ⊕ y2 ⊕ 1=0,
x0=x2 ⊕ x2=x1 ⊕ x2=x0 ⊕ x0 ⊕ x1 ⊕ y1 ⊕ 1=0,
x0=x1 ⊕ x0=x2 ⊕ x0=x3 ⊕ x1=x3 ⊕ x2=x3 ⊕ x0=x1 ⊕ x3=y1 ⊕ y3=0,
x0=x1 ⊕ x3=x2 ⊕ x1 ⊕ x2 ⊕ y1 ⊕ y2 ⊕ 1=0,
x0=x1 ⊕ x1=x3 ⊕ x2=x3 ⊕ x3=y3 ⊕ x1 ⊕ x2 ⊕ x3 ⊕ y1 ⊕ y2 ⊕ 1=0,
x1=x3 ⊕ x0=y1 ⊕ x0 ⊕ x1 ⊕ x2 ⊕ y1 ⊕ y3=0,
y0=y2 ⊕ x3 ⊕ y1 ⊕ y3 ⊕ 1=0,
x0=x1 ⊕ x1=x3 ⊕ x0=y1 ⊕ y0=y3 ⊕ x0 ⊕ x1 ⊕ x2 ⊕ y1 ⊕ y2 ⊕ y3 ⊕ 1=0,
\[ \begin{align*}
x_2 \cdot x_3 & \oplus x_0 \cdot y_1 \oplus y_1 \cdot y_2 \oplus x_1 \oplus x_3 \oplus y_2 \oplus y_3 = 0, \\
x_0 \cdot x_1 & \oplus x_1 \cdot x_3 \oplus x_2 \cdot x_3 \oplus x_0 \cdot y_0 \oplus x_0 \cdot y_2 \oplus y_1 \cdot y_3 \oplus x_0 \oplus x_3 = 0, \\
x_0 \cdot x_1 & \oplus x_2 \cdot x_3 \oplus x_0 \cdot y_1 \oplus y_2 \cdot y_3 \oplus x_2 \oplus y_2 = 0.
\end{align*} \]

Figure 3. General structure of Present cipher.

In general PRESENT encryption can be described by a set of \( m = i \cdot 21 \) equations with \( n = i \cdot 8 \) unknowns, where \( i \) is the number of S-boxes, which take part in encryption and round key generation. So the standard 31 rounds of PRESENT can be described by a set of 11067 nonlinear Boolean equations with 4216 unknowns; the total number of S-boxes is \( i = 16 \cdot 31 + 31 = 527 \).

For three rounds of the PRESENT cipher the following replacements were made:

\[
X_1 = P \oplus K_1, \\
X_2 = H(Y_1) \oplus K_2, \\
X_3 = H(Y_2) \oplus K_3, \\
Y_3 = H^{-1}(C),
\]

where \( X_1, \ldots, X_3 \) are the 64-bit input values of S-box, \( Y_1, \ldots, Y_3 \) are the 64-bit output values of S-box, \( K_1, \ldots, K_3 \) are the 64-bit round keys, \( H() \) is the permutation transformation, \( H^{-1}() \) is the inverse transformation of \( H() \).

For four rounds of PRESENT the following relationship between variables was used:

\[
X_1 = P \oplus K_1, \\
X_2 = H(Y_1) \oplus K_2, \\
X_3 = H(Y_2) \oplus K_3, \\
X_4 = H(Y_3) \oplus K_4, \\
Y_4 = H^{-1}(C).
\]
where $X_1, ..., X_4$ are the 64-bit input values of S-box, $Y_1, ..., Y_4$ are the 64-bit output values of S-box, $K_1, ..., K_4$ are the 64-bit round keys, $H()$ is the permutation transformation, $H^{-1}()$ is the inverse transformation of $H()$.

The proposed algorithms were applied to perform algebraic analysis of Present without using the structure of round subkey generation. The obtained experimental estimations of algebraic analysis CNF parameters are shown in Table 3.

Table 3. Algebraic analysis CNF parameters for PRESENT cipher

| Number of text pairs | Number of equations | Number of unknowns | Number of literals | Number of clauses | Number of solutions |
|----------------------|---------------------|-------------------|-------------------|------------------|-------------------|
| 3 rounds of PRESENT cipher |                     |                   |                   |                  |                   |
| 2                    | 2016                | 448               | 8786              | 137530           | $>10^3$           |
| 3                    | 3024                | 576               | 16083             | 274534           | $>10^4$           |
| 6                    | 6048                | 960               | 23863             | 410800           | 1                 |
| 4 rounds of PRESENT cipher |                     |                   |                   |                  |                   |
| 3                    | 4032                | 832               | 18805             | 313070           | $>2\cdot 10^4$   |
| 5                    | 6720                | 1216              | 30585             | 520686           | 256               |
| 8                    | 10752               | 1792              | 48784             | 832128           | 16                |

4. Conclusion

We have investigated the method of PRESENT encryption transform representation as a set of algebraic boolean equations. We also described reduction of PRESENT transform to SAT problem. The number of literals and clauses, which are encountered with different numbers of known text pairs, has been found. Finding of a 162-bit and 256-bit subkeys of PRESENT encryption with different number of text pairs has been modeled. The experimental results on search timings for SAT problem of 3-4 round PRESENT analysis are presented.

Acknowledgements

This work was supported by a scholarship program of the President of Russian Federation for young scientists and postgraduate students No. SP-1907.2019.5.

References

[1] Ссылка (стиль ‘Reference’)

[2] Shannon C E 1949 Bell System Technical Journal 28(4) 656–715

[3] Pasalic E 2008 Arithmetic of Finite Fields, Proceedings of Second International Workshop (WAIFI, Siena, Italy) pp 189-204

[4] Bogdanov A et al. 2018 Cryptographic Hardware and Embedded Systems 4727 450-66

[5] Semenov A A. 2014 XII All-Russian meeting on control problems pp 7361-74

[6] Bard G, et al 2007 Cryptology ePrint Archive 24

[7] Bard G 2009 Algebraic Cryptanalysis p 356

[8] Nakahara J et al 2009 8th International Conference on Cryptology and Network Security, CANS ’09, New York pp 58-75

[9] Babenko L K, Maro E A and Anikeev M V 2017 IEEE inc., USA pp 272-8