Jan Długosz University in Częstochowa
Scientific Issues, Mathematics XVI, Częstochowa 2011

SAT-BASED CRYPTANALYSIS OF MODIFIED VERSIONS OF FEISTEL NETWORK

Paweł Dudek, Mirosław Kurkowski

Institute of Computer and Information Sciences
Częstochowa University of Technology
ul. Dąbrowskiego 73, 42-200 Częstochowa, Poland
e-mail: pdudek@icis.pcz.pl mkurkowski@icis.pcz.pl

Abstract. It is well known that Feistel Network (FN) is the foundation of many symmetric ciphers used in practice. In this paper we present some remarks and experimental results on SAT based cryptanalysis of several modified versions of FN. We investigate different cryptographic functions used in FN schema for better understanding their properties from a security point of view. In our work we study the notions widely used in many ciphers: the *xor* function, bits rotations, permutations and $S-boxes$.

1. Introduction

Boolean SATisfiability problem is the well known and celebrated NP-complete problem [2]. The Boolean encoding of some system models and checking satisfiability of obtained formulas sometimes gives the answer to the question about important system's properties [1, 8]. So far, there is no known algorithm that solves efficiently all the instances of SAT. It is generally believed that no such effective algorithm can already exist. On the other hand, in many instances a lot of Boolean formulas can be solved surprisingly efficiently, even very large formulas appearing naturally in description of various industrial systems as well as in decision and optimization problems [1, 2, 7]. There are many competing algorithms searching for a satisfying valuation for a given Boolean formula. A lot of them are highly optimised versions of the DPLL procedure of [4] and [5]. Usually SAT-solvers take input formulas in the conjunctive normal form (CNF). It is a conjunction of clauses, where a clause is a disjunction of literals, and a literal is a propositional variable or the complement of a propositional variable.

In this paper we use SAT for investigation of security properties of several modified versions of FN, that is some easy but very important cipher used as a basis for many strong symmetric ciphers applied in practice (see, for example, [9]). We show how using several different cryptographic functions as a main function F in FN can change security properties and computational complexity of FN's SAT based cryptanalysis. We show this on the well known functions used in many other symmetric ciphers: the *xor* function, bits rotation, permutation and $S-box$. Carrying out current research we want to check how SAT cryptanalysis works in the simple cases discussed in order to have the ability to select some other ciphers used in practice for future study. Their cryptanalysis may be promising.

The methodology is similar to that used in [3] and [8], and this paper presents an investigation additional to our previous paper [6].

The rest of this paper is organized as follows. In the second section, we introduce all the basic information on the FN cipher to the extent necessary for explaining our Boolean encoding method. The third section gives a process of a direct Boolean encoding of FN and the main functions which are considered. In the fourth section, we present some experimental results which have been obtained. Some conclusions and remarks concerning the future work are given in the last section.

2. Feistel Network

This section presents the basic information on FN which is needed for understanding the Boolean encoding of investigated ciphers. It is well known that FN is a symmetric-key block algorithm widely used as a design principle of many symmetric ciphers, including the famous Data Encryption Standard (DES). FN has the advantage that its encryption and decryption procedures are almost identical, requiring only a reversal of the key schedule. FN is an iterated algorithm which is executed many times on the same input. Due to a simple structure and easy hardware implementation, Feistel-like networks are widely used as a component of various cipher designs. Some famous, strong and used in practice FN are the following: MISTY1, Skipjack, Blowfish, RC5, Twofish (see, for example, [9]).

Consider a given bit block M that represents a plaintext. Let F denote the round main function of FN and $K_1, \ldots, K_n$ denote a sequence of keys obtained in some way from the main key K for the rounds $1, \ldots, n$, respectively. We use the symbol $\otimes$ for denoting the exclusive-OR (*xor*) operation.

The basic operations of FN are specified as follows:

1. break the plaintext block M into two equal length parts denoted by (L_0, R_0);
2. for each round $i = 0, 1, \ldots, n$ compute:
   a) $L_{i+1} = R_i$,
   b) $R_{i+1} = L_i \otimes F(R_i, K_i)$.

Then the ciphertext sequence is (R_{n+1}, L_{n+1}).

The structure of FN allows for an easy method of decryption. For explanation of the decryption procedure of FN, let us recall the basic properties of operation $\otimes$:

1. $x \otimes x = 0$,
2. $x \otimes 0 = x$,
3. $x \otimes (y \otimes z) = (x \otimes y) \otimes z$.

A given ciphertext (R_{n+1}, L_{n+1}) is decrypted by computing

$$R_i = L_{i+1}, \qquad L_i = R_{i+1} \otimes F(L_{i+1}, K_i)$$

for $i = n, n-1, \ldots, 0$.

It is easy to see that (L_0, R_0) is the plaintext again. Observe that we have the following equations:

$$R_{i+1} \otimes F(L_{i+1}, K_i) = (L_i \otimes F(R_i, K_i)) \otimes F(L_i, K_i)$$

$$= L_i \otimes (F(R_i, K_i) \otimes F(L_i, K_i)) = L_i \otimes 0 = L_i.$$

It should be noted that the power of the cipher depends on the choice of the function F. In practice, many different solutions are used in this case. In the next sections we investigate four of them: the *xor* function, bits rotation, permutation, and $S-box$.

3. Boolean encoding for cryptanalysis

This section presents a direct Boolean encoding of FN versions with different functions F. As it was mentioned above, FN constitutes the basic structure for many well respected symmetric ciphers. Hence, its Boolean encoding will be helpful in the SAT-based cryptanalysis which we want to pursue in the future.

For our explanation we consider FN with a 64-bit block of a plaintext and a 32-bit key. Let $p_1, \ldots, p_{64}$, $k_1, \ldots, k_{32}$ and $c_1, \ldots, c_{64}$ be the propositional

variables representing a plaintext, a key, and a ciphertext, respectively. Observe that following the Feistel algorithm for the first, left half of a plaintext we have:

$$\bigwedge_{i=1}^{32}(c_i \Leftrightarrow p_{i+32}).$$

It is easy to see that for the second, right half of a plaintext we have:

$$\bigwedge_{i=1}^{32}(c_i \Leftrightarrow (p_i \otimes F(k_i, p_{i+32})).$$

Hence the encoding formula for one round of FN is as follows:

$$\Phi^1_{Feistel} : \bigwedge_{i=1}^{32}(c_i \Leftrightarrow p_{i+32}) \ \wedge \ \bigwedge_{i=1}^{32}(c_i \Leftrightarrow (p_i \otimes F(k_i, p_{i+32})).$$

In the case of t rounds of FN we have the following: Let $(p^1_1, \ldots, p^1_{64})$, $(k_1, \ldots, k_{32})$ be a plaintext and a key vectors of variables, respectively. By $(p^j_1, \ldots, p^j_{64})$ and $(c^i_1, \ldots, c^i_{64})$ we describe the vectors of variables representing input of the jth round for $j = 2, \ldots, t$ and output of the ith round for $i = 1, \ldots, t-1$. We denote by $(c^t_1, \ldots, c^t_{64})$ the variables of a cipher vector after the tth round.

The formula which encodes the whole tth round of a Feistel Network is as follows:

$$\Phi^t_{Feistel} : \bigwedge_{i=1}^{32}\bigwedge_{s=1}^{t}(c^s_i \Leftrightarrow p^s_{i+32}) \ \wedge \ \bigwedge_{i=1}^{32}\bigwedge_{s=1}^{t}[c^s_{i+32} \Leftrightarrow (p^s_i \ \otimes \ F(k_i, p^s_{i+32}))] \wedge$$

$$\wedge \bigwedge_{i=1}^{64}\bigwedge_{s=1}^{t-1}(p^{s+1}_i \Leftrightarrow \ c^s_i).$$

Observe that the last part of $\Phi^t_{Feistel}$ states that the outputs from the sth round are the inputs of the $(s+1)$th round.

As we can see, the obtained formula is a conjunction of ordinary, or rather simple, equivalences. This is important from the viewpoint of translating into CNF. The second advantage of this description is that we can automatically generate the formula for many investigated rounds.

It is well known that the security of FN cipher depends on the function F. In our investigation, as a simple instantiation of the function F, we firstly use the function xor, denoted as before without any other changes of the used

bits. In this case we obtain the following formula that encodes the function F in the considered cipher:

$$F(k_i, p^s_{i+32}) \;\Leftrightarrow\; (k_i \otimes p^s_{i+32}).$$

The second considered approach is adding the rotation of key's bits into F. If we use one right rotation for one round of FN, we arrive at the following formula:

$$F(k_i, p^s_{i+32}) \;\Leftrightarrow\; (k_{i\oplus s} \otimes p^s_{i+32}),$$

where $\oplus$ denotes $+$ modulo 32.

The third example presents some permutation in the function F. In this work we consider the PC permutation – one of permutations used in the DES cipher [9]. In this case we get the following formula:

$$F(k_i, p^s_{i+32}) \;\Leftrightarrow\; (k_i \otimes p^s_{PC(i)\oplus 32}).$$

The last, most powerfull, considered modification of the function F is adding the $S-box$ into F. Like before, we use the first $S-box$ used in DES cipher [9]. Firstly, we increase the length of the block half from 32 to 48 by repeating the proper bits like in the DES algorithm. Observe that each of $S-boxes$ of this type is the matrix with four rows and sixteen columns, where in each row we have one different permutation of numbers belonging to Z_{16}. These numbers are denoted in binary form as four-tuples of bits. Following this, we can consider each $S-box$ as a function of the type $S_{box} : \{0,1\}^6 \rightarrow \{0,1\}^4$.

For simplicity, let us denote by $\overline{x}$ the vector $(x_1, \ldots, x_6)$ and by $S^k_{box}(\overline{x})$ the kth coordinate of the value $S_{box}(\overline{x})$ for $k = 1, 2, 3, 4$.

We can encode the $S-box$ as the following Boolean formula:

$$\Phi_{S_{box}} : \bigwedge_{\overline{x}\in\{0,1\}^6} (\bigwedge_{i=1}^{6} (\neg)^{1-x_i} r_i \;\Rightarrow\; \bigwedge_{j=1}^{4} (\neg)^{1-S^j_{box}(\overline{x})} q_j),$$

where $(r_1, \ldots, r_6)$ is the input vector of the $S-box$ and $(q_1, \ldots, q_4)$ is the output one. Additionally, by $(\neg)^0 r$ and $(\neg)^1 r$ we mean r and $\neg r$, respectively. Using this, we can encode the $S-box$ used as 256 simple implication. This number is equal to the size of the $S-box$ matrix. Due to the strongly irregular and random character of $S-boxes$, we are sure that this is the simplest method of their Boolean encoding. Having them, we can encode any given number of rounds of modified FN as a Boolean propositional formula. The next step of our investigation is applying our cryptanalysis procedure to the obtained formulas.

Rounds	Variables	Clauses	Encoding Time (s.)	Encoding Memory (MB)	Solving Time (s.)	Solving Memory (MB)
4	672	9952	0.008	8	0.012	8
8	1248	19808	0.016	8	0.181	9
12	1824	29664	0.021	9	90.74	25
16	2400	39520	0.028	9	11370.5	231

Table 1: Experimental results with $S-box$.

4. Cryptanalysis procedure and experimental results

The cryptanalysis procedure used in our investigation proceeds as follows:

1) encode a single round of the cipher considered as a Boolean propositional formula;
2) generate automatically the formula encoding an iterated desired number of rounds;
3) convert the obtained formula into the CNF form;
4) (randomly) choose a plaintext and the key vector as the $0, 1$ valuation of the variables representing them in the formula;
5) insert the chosen valuation into the formula;
6) calculate the corresponding ciphertext using an appropriate key and insert it into the formula;
7) run some SAT-solver to find a satisfying valuation, including a valuation of the key variables.

To test how the SAT based cryptanalysis works for the functions mentioned above we have used the previously outlined procedure for four different number of rounds of a modified version of the FN: 4, 8, 12, and 16. The obtained results show that the SAT based cryptanalysis is similar to other methods of cryptanalysis. The SAT based cryptanalysis proceeds along very well with simple functions such as xor, rotations, and permutations. The results obtained for these features show a strictly linear relationship to receive the key from the known plaintext ciphertext pair with increasing the number of iterations of the algorithm.

In the case of adding the $S-box$ into the function F we obtain results that show an exponential relation to receiving the key with increasing the number of iterations of the algorithm. We can see these results in Table 1.

In this Table we show the number of variables and clauses of the encoding formula obtained for proper iterations of FN, time of generating the formula and time of searching key satisfiable valuation (break the key).

The computer used to perform the experiments was equipped with the processor Intel Pentium D (3000 MHz), 2 GB main memory, the operating system Linux, and the SAT-solver MiniSat.

5. Conclusion and the future work

In this paper we have shown how the SAT based cryptanalysis works for breaking some modified versions of Feistel Network. We have investigated four different main functions of FN using the *xor* function, bits rotation, permutation, and *S–box*. The obtained results show that this type of cryptanalysis proceeds well with the ciphers with the *xor* function, bits rotation, and permutation. Using the $S-box$ in cipher algorithm increases computational complexity of this type cryptanalysis into an exponential one. The next step of our work will consist in choosing some of the ciphers used in practice that have functions simply from the SAT based cryptanalysis and trying to break them. We hope that this investigation will be helpfull in choosing such ciphers. We are sure that the Kazumi cipher will be a good example for our work. Clearly, the success of our method depends on finding a cipher which can be broken.

References

[1] A. Armando, L. Compagna. Sat-based model-checking for security protocols analysis. *Int. J. Information Security*, **7**(1), 3–32, 2008.

[2] A. Biere, M. Heule, H. van Maaren, T. Walsh. *Handbook of Satisfiability*, vol. 185 of Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam 2009.

[3] N. Courtois, G. V. Bard. Algebraic cryptanalysis of the data encryption standard. In: *Proc. of the 11th IMA Int. Conf. on Cryptography and Coding*, vol. 4887 LNCS, pp. 152–169, Springer, Berlin 2007.

[4] M. Davis, G. Logemann, D. W. Loveland. A machine program for theoremproving. *Commun. ACM*, **5**(7), 394–397, 1962.

[5] M. Davis, H. Putnam. A computing procedure for quantification theory. *J. ACM*, **7**(3), 201–215, 1960.

[6] P. Dudek, M. Kurkowski, M. Srebrny. Towards parallel direct SAT-based cryptanalysis. *Proc. of PPAM'11*, LNCS, Springer (accepted).

[7] M. Kurkowski, W. Penczek, A. Zbrzezny. SAT-based verification of security protocols using networks of automata. In: *Proc. of MoChArt'06*, vol. 4428 LNCS, pp. 146–165, Springer, Berlin 2007.

[8] F. Massacci. Using Walk-SAT and Rel-SAT for cryptographic key search. In: T. Dean (ed.), *Proc of 16th Int. Joint Conf. on Artificial Intelligence*, pp. 290–295, Morgan Kaufmann Publishers, San Francisco 1999.

[9] A. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton 1996.