Direct SAT-Based Cryptanalysis of Some Symmetric Ciphers

Miroslaw Kurkowski

Institute of Computer Science
Card. St. Wyszynski University, Warsaw

Poraj, 18-21.09.2017

Ministerstwo Nauki i Szkolnictwa Wyższego

Organizacja IX Konferencji Modelowanie Matematyczne w Fizyce i Technice (MMFT 2017)
- zadanie finansowane w ramach umowy 829/P-DUN/2017 ze środków Ministra Nauki i Szkolnictwa Wyższego przeznaczonych na działalność upowszechniającą naukę.

Outline

Outline

Outline

Outline

Symmetric ciphers

Algorithms

- Feistel Network 1972,
- DES - Data Encryption Standard, 1974, 56-bits key,
- Feistel based algorithms - MISTY1, Skipjack, Threefish, Blowfish, Camellia, CAST-128, FEAL, ICE, LOKI97, Lucifer, MARS, MAGENTA, RC5, TEA, Twofish, XTEA, GOST 28147-89.
- Rijndael - AES current Encryption Standard, 2001, 128, 192, 256-bits key.

Main operations

- xor,
- permutations,
- bits rotations,
- S-boxes.

Symmetric ciphers

Algorithms

- Feistel Network 1972,
- DES - Data Encryption Standard, 1974, 56-bits key,
- Feistel based algorithms - MISTY1, Skipjack, Threefish, Blowfish, Camellia, CAST-128, FEAL, ICE, LOKI97, Lucifer, MARS, MAGENTA, RC5, TEA, Twofish, XTEA, GOST 28147-89.
- Rijndael - AES current Encryption Standard, 2001, 128, 192, 256-bits key.

Main operations

- xor,
- permutations,
- bits rotations,
- S-boxes.

Methods of cryptanalysis

Knowledge point of view

- ciphertext-only attack,
- known-plaintext/ciphertext attack,
- chosen-plaintext attack,
- chosen-ciphertext attack.

Computational point of view

- statistical cryptanalysis,
- brute force,
- differential cryptanalysis,
- linear cryptanalysis,
- SAT-based approach.

SAT based methods

Verification methods

- software testing,
- simulations,
- formal model verification.

SAT based methods

- translating system and investigated property into a boolean propositional formula,
- testing satisfiability using SAT solvers.

SAT based methods

Verification methods

- software testing,
- simulations,
- formal model verification.

SAT based methods

- translating system and investigated property into a boolean propositional formula,
- testing satisfiability using SAT solvers.

SAT

Formulas

Formulas used in this-type verification are translated into Conjunction Normal Form (CNF - conjunction of clauses), for example

$$(p_1 \lor \neg p_2 \lor \neg p_5) \ \land \ (\neg p_2 \lor p_3 \lor \neg p_4) \ \land \ (p_3 \lor \neg p_4 \lor p_5)$$

Exponential translation algorithm that preserves equivalence, but linear algorithm that preserve satisfiability.

SAT solvers

- SAT Race conferences,
- MiniSAT, BerkMin, CryptoMiniSAT and many others.

SAT

Formulas

Formulas used in this-type verification are translated into Conjunction Normal Form (CNF - conjunction of clauses), for example

$$(p_1 \vee \neg p_2 \vee \neg p_5) \ \wedge \ (\neg p_2 \vee p_3 \vee \neg p_4) \ \wedge \ (p_3 \vee \neg p_4 \vee p_5)$$

Exponential translation algorithm that preserves equivalence, but linear algorithm that preserve satisfiability.

SAT solvers

- SAT Race conferences,
- MiniSAT, BerkMin, CryptoMiniSAT and many others.

SAT based methods in security

SAT in security

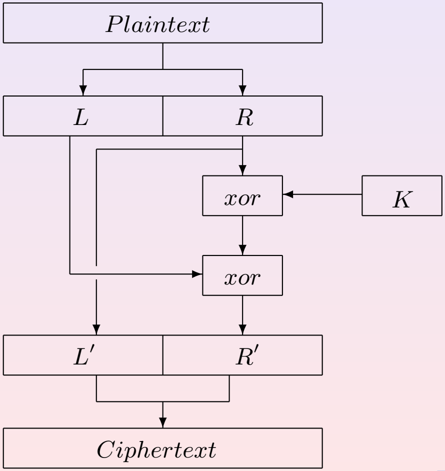
- **F. Massacci**, *Using walk-SAT and rel-sat for cryptographic key search*, In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI99), 1999.
- **I. Mironov, and L. Zhang**, *Applications of SAT Solvers to Cryptanalysis of Hash Functions*. In Theory and Applications of Satisfiability Testing - SAT, LNCS, vol. 4121, pp 102–115, Springer-Verlag, 2006.
- **M. Kurkowski, W. Penczek, A. Zbrzezny**, *SAT-based Verification of Security Protocols Using Networks of Automata*, in Proceedings of MoChArt'06, pp. 146-165, LNCS, vol. 4428, Springer Verlag 2007.
- **M. Kurkowski, W. Penczek**, *Verifying Timed Security Protocols via Translation to Timed Automata*, Fundamenta Informaticae, vol. 93 (1-3), pp. 245-259, IOS Press 2009.
- **P. Morawiecki and M. Srebrny**, *A SAT-based preimage analysis of reduced KECCAK hash functions*, in Proceedings of Second SHA-3 Candidate Conference, Santa Barbara. 2010.

Feistel network

Feistel network

- Introduced by Horst Feistel for IBM Lab in 1972.
- Block cipher.
- Main idea of Feistel network is used in DES, IDEA and many others ciphers.

Feistel network



Feistel equations

Ciphertext

1. $L' = R$,
2. $R' = L \; xor \; (R \; xor \; K)$.

xor properties

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

Decrypting

1. $L' = R$,
2. $R' \; xor \; (L' \; xor \; K) \; = L \; xor \; (R \; xor \; K) \; xor \; (R \; xor \; K) \; = L$.

Other DES components

Components

1. permutations,
2. bits rotations,
3. S-boxes

S-box

S-box is a nonlinear component, it is a matrix with 16 columns and 4 rows, where each of rows contains different permutation of $\{0, 1, 2, \ldots 15\}$.

Other DES components

Components

1. permutations,
2. bits rotations,
3. S-boxes

S-box

S-box is a nonlinear component, it is a matrix with 16 columns and 4 rows, where each of rows contains different permutation of $\{0, 1, 2, \dots 15\}$.

Feistel encoding

One round

Let $(p_1, \dots, p_{64})$ be a plaintext vector, where $p_i \in \{0,1\}$, for $i = 1, \dots, 64$ and $(k_1, \dots, k_{32})$ be a key vector, where $k_j \in \{0,1\}$, for $j = 1, \dots, 32$. We denote by $(c_1, \dots, c_{64})$ a cipher vector, where $c_s \in \{0,1\}$, for $s = 1, \dots, 64$.

Encoding formula for one round

$$\Phi_F^1 : \bigwedge_{i=1}^{32} (c_i \Leftrightarrow p_{i+32}) \ \wedge \ \bigwedge_{i=1}^{32} [c_{i+32} \Leftrightarrow (p_i \ xor \ p_{i+32} \ xor \ k_i)]$$

Feistel encoding

Many rounds

Let $(p_1^1, \ldots, p_{64}^1)$ be a plaintext vector, where $p_i^1 \in \{0,1\}$, for $i = 1, \ldots, 64$ and $(k_1, \ldots, k_{32})$ be a key vector, where $k_j \in \{0,1\}$, for $j = 1, \ldots, 32$. We denote by $(c_1^j, \ldots, c_{64}^j)$ a cipher vector after j rounds, where $c_s^j \in \{0,1\}$, for $s = 1, \ldots, 64$.

Encoding formula for j rounds

$$\Phi_F^j : \bigwedge_{i=1}^{32} \bigwedge_{s=1}^{j} (c_i^s \Leftrightarrow p_{i+32}^s) \wedge \bigwedge_{i=1}^{32} \bigwedge_{s=1}^{j} [c_{i+32}^s \Leftrightarrow (p_i^s \; xor \; p_{i+32}^s \; xor \; k_i)] \wedge$$
$$\wedge \bigwedge_{i=1}^{64} \bigwedge_{s=1}^{j-1} (p_i^{s+1} \Leftrightarrow c_i^s).$$

DES encoding

Permutations, rotations

- Consider P - the initial permutation function of DES. Let $(p_1, \ldots, p_{64})$ be a sequence of variables representing the plaintext bits.
- Denote by $(q_1, \ldots, q_{64})$ a sequence of variables representing the block bits after permutation P.

We can encode P as the following formula:

$$\bigwedge_{i=1}^{64} (p_i \Leftrightarrow q_{P(i)}).$$

In a such way, we can encode all the permutations, expansions, reductions, and rotations of DES.

DES encoding

S-boxes

- We can consider each S-box as a function of type $S_{box} : \{0,1\}^6 \to \{0,1\}^4$. We denote a vector $(x_1, \ldots, x_6)$ by $\overline{x}$ and by $S^k_{box}(\overline{x})$ the k-th coordinate of value $S_{box}(\overline{x})$, for $k = 1, 2, 3, 4$.

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

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

where $(p_1, \ldots, p_6)$ is the input vector of S-box and $(q_1, \ldots, q_4)$ the output one. Additionally, by $(\neg)^0 p$ and $(\neg)^1 p$ we mean p and $\neg p$, respectively.

Algorithm

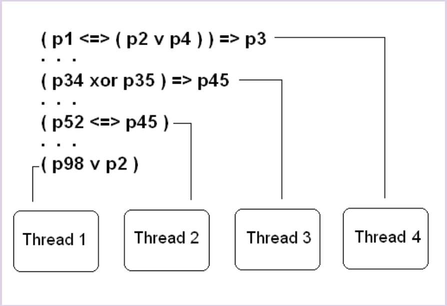
Our cryptanalysis procedure

1. encoding a single round of the cipher considered as a Boolean propositional formula;
2. automatically generating of the formula encoding a iterated desired number of rounds (the whole cipher);
3. converting the formula obtained its CNF;
4. (randomly) choosing a plaintext and the key vector as a $0, 1$ valuation of the variables representing them in the formula;
5. inserting the chosen valuation into the formula;
6. calculating the corresponding ciphertext using an appropriate key and insert it into the formula;
7. using SAT-solver to find a satisfying valuation, including a valuation of the key variables.

Optimisation and parallelisation

Parallelisation

- Parallelisation of the process of encoding formula generation. An acceleration about 15%



Optimisation and parallelisation

Optimisation

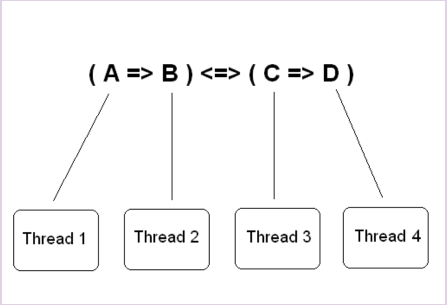
- Optimising the formula obtained by removing all not necessary equivalences. We can decrease the number of variables used sometimes by 50% and the number of clauses by 10%.

Rounds	Variables	Variables optimised	Clauses	Clauses optimised
4	1024	568	10496	9472
8	1976	1016	20866	18944
16	3768	1912	41601	37888

Optimisation and parallelisation

Parallelisation

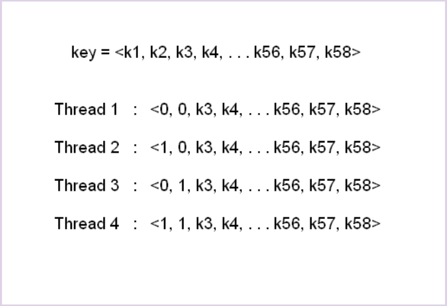
- Parallel realization of translating formula into a CNF form. 10% speedup in the translation time.



Optimisation and parallelisation

Parallelisation

- Parallel application of SAT-solver using 2^n cores.



key = <k1, k2, k3, k4, . . . k56, k57, k58>

```
Thread 1  :  <0, 0, k3, k4, . . . k56, k57, k58>
Thread 2  :  <1, 0, k3, k4, . . . k56, k57, k58>
Thread 3  :  <0, 1, k3, k4, . . . k56, k57, k58>
Thread 4  :  <1, 1, k3, k4, . . . k56, k57, k58>
```

Optimisation and parallelisation

Rounds	Time (s.)	Time with parallelisation (s.)
3	0,052	0,036
4	34,686	6,542
5	12762	2438
6 + 20 key bits	47,539	2,216

Parallelisation

In the last row of Table, for six rounds of DES with added valuations of 20 key bits, the whole key was solved in 2.2 secs with all our optimisation and parallelisation. In the same case but without the improvements presented above, the whole key was solved in 145 secs. The best result known so far in SAT-based cryptanalysis for this case was 68 secs.

Future work

Future

- Other methods of formulas optimalization.
- Parametrized cryptanalysis with many pairs plaintext/ciphertext.
- Testing several SAT-solvers.
- Testing other ciphers like KAZUMI, RC family or recently introduced hash functions like Grostl, Grain128.
- Create dedicated SAT-solver for cryptanalysis.

That is all :)

Thank You