Solving Non-Linear Random Sparse Equations over Finite Fields

Thorsten Schilling

UiB

May 6, 2009

Algebraic Cryptanalysis

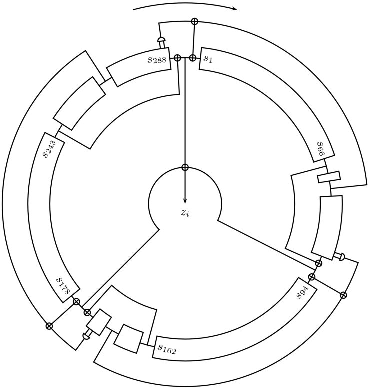
- Express cipher as system of equations
- Well known examples: AES, DES, Trivium etc.
- Obtaining solution to systems might break cipher

l-sparse Equation System

$$f_1(X_1) = 0, f_2(X_2) = 0, \ldots, f_m(X_m) = 0$$

- $X_i \subseteq X$
- $|X_i| \leq l$

Example Trivium



- 80 bits IV, 80 bits key
- 288 bits internal state
- Output bit is linear combination of state bits
- Overall very "simple" design

Example Trivium (2)

Expressed as equation system

$$
\begin{aligned}
a_i &= c_{i-66} + c_{i-111} + c_{i-110}c_{i-109} + a_{i-69} \\
b_i &= a_{i-66} + a_{i-93} + a_{i-92}a_{i-91} + b_{i-78} \\
c_i &= b_{i-69} + b_{i-84} + b_{i-83}b_{i-82} + c_{i-87}
\end{aligned}
$$

Output bit z_i

$$z_i = c_{i-66} + c_{i-111} + a_{i-66} + a_{i-93} + b_{i-69} + b_{i-84}$$

Example Trivium (2)

Expressed as equation system

$$\begin{aligned} a_i &= c_{i-66} + c_{i-111} + c_{i-110}c_{i-109} + a_{i-69} \\ b_i &= a_{i-66} + a_{i-93} + a_{i-92}a_{i-91} + b_{i-78} \\ c_i &= b_{i-69} + b_{i-84} + b_{i-83}b_{i-82} + c_{i-87} \end{aligned}$$

Output bit z_i

$$z_i = c_{i-66} + c_{i-111} + a_{i-66} + a_{i-93} + b_{i-69} + b_{i-84}$$

→ For state recovery solve system:

- 6-sparse
- 951 variables
- 663 quadric equations, 288 linear

Solving Strategies

- Linearization
- Gröbner Basis Algorithms

Solving Strategies

- Linearization
- Gröbner Basis Algorithms
- SAT-Solving
  - Is $\phi = (x_i \vee x_j \vee \ldots \vee x_k) \wedge \ldots \wedge (x_u \vee x_v \vee \ldots \vee x_w)$ SAT?
  - Theoretical worst case bounds [Iwama,04]:

sparsity	3	4	5	6
worst case	1.324^n	1.474^n	1.569^n	1.637^n

Solving Strategies

- Linearization
- Gröbner Basis Algorithms
- SAT-Solving
  - Is $\phi = (x_i \vee x_j \vee \ldots \vee x_k) \wedge \ldots \wedge (x_u \vee x_v \vee \ldots \vee x_w)$ SAT?
  - Theoretical worst case bounds [Iwama,04]:

sparsity	3	4	5	6
worst case	1.324^n	1.474^n	1.569^n	1.637^n

- Gluing & Agreeing [Raddum, Semaev]
  - Expected Running Times:

sparsity	3	4	5	6
Gluing[Semaev,WCC'07]	1.262^n	1.355^n	1.425^n	1.479^n
Agr.-Gluing[Semaev,ACCT'08]	1.113^n	1.205^n	1.276^n	1.334^n

SAT-solving - DPLL/Davis-Putnam-Logemann-Loveland

General structure of the algorithm (input ϕ in CNF):

- Extend a partial guess
- Propagate information
- Clause Resolution
- Backtrack if a conflict was detected

Past Major Enhancements

- Algorithmic [Silvia, Sakallah 1996]
  - Non-chronological backtracking
  - Conflict clauses
- Technical
  - Watched literals [Moskewicz et. al. 2001]
- Instance specific
  - Guessing heuristics

A New Approach

- Generalization of a backtracking solving method for sparse non-linear equation systems over finite fields
  - CNF-instances are specialization
- Exploit sparsity of equations

Equation as Symbol

- $f(X_i) = 0$ as pair of sets $S_i = (X_i, V_i)$
- X_i variables in which the equation is defined
- V_i its satisfying vectors/assignments

Random equation over $\mathbb{F}_2$ expected $|V_i| = 2^{|X_i|-1}$

Equation as Symbol

- $f(X_i) = 0$ as pair of sets $S_i = (X_i, V_i)$
- X_i variables in which the equation is defined
- V_i its satisfying vectors/assignments

Random equation over $\mathbb{F}_2$ expected $|V_i| = 2^{|X_i|-1}$

Example:

$$f_e(x_1, x_2, x_3) = x_1x_2 \oplus x_3 = 0$$

becomes

S_e	x_1	x_2	x_3
a_0	0	0	0
a_1	0	1	0
a_2	1	0	0
a_3	1	1	1

Gluing

Create new symbol $S_1 \circ S_2 = (X_1 \cup X_2, U)$ from two symbols S_1, S_2 where

$$U = \{(a_1, b, a_2) | (a_1, b) \in V_1 \text{ and } (b, a_2) \in V_2\}$$

Gluing

Create new symbol $S_1 \circ S_2 = (X_1 \cup X_2, U)$ from two symbols S_1, S_2 where

$$U = \{(a_1, b, a_2) | (a_1, b) \in V_1 \text{ and } (b, a_2) \in V_2\}$$

Example:

S_0	1	2	3
a_0	0	0	0
a_1	0	1	0
a_2	1	0	0
a_3	1	1	1

$\circ$

S_1	3	4	5
b_0	1	0	0
b_1	1	0	1
b_2	1	1	1

$=$

$S_0 \circ S_1$	1	2	3	4	5
c_1	1	1	1	0	0
c_2	1	1	1	0	1
c_3	1	1	1	1	1

Agreeing

Delete from symbols S_1, S_2 vectors whose projections do not match in their common variables $X_1 \cap X_2$. If symbol gets "empty" $\rightarrow$ contradiction.

Agreeing

Delete from symbols S_1, S_2 vectors whose projections do not match in their common variables $X_1 \cap X_2$. If symbol gets "empty" $\rightarrow$ contradiction.

Example:

S_0	1	2	3
a_0	0	0	0
a_1	0	1	0
a_2	1	0	0
a_3	1	1	1

,

S_1	3	4	5
b_0	1	0	0
b_1	1	0	1
b_2	1	1	1

become

S_0	1	2	3
a_3	1	1	1

,

S_1	3	4	5
b_0	1	0	0
b_1	1	0	1
b_2	1	1	1

Gluing-Agreeing Algorithm

Glue intermediate symbol with another symbol, then agree the intermediate equation system.

Agreeing2

Aim: Reduce number of steps for Agreeing

Addresses of common projections stored as tuples

S_0	1	2	3
a_0	0	0	0
a_1	0	1	0
a_2	1	0	0
a_3	1	1	1

,

S_1	3	4	5
b_0	0	0	0
b_1	1	0	0
b_2	1	0	1
b_3	1	1	1

$\longrightarrow \{a_0, a_1, a_2; b_0\}, \{a_3; b_1, b_2, b_3\}$

Mark groups of common projections

Example: a_0, a_1, a_3 marked $\Rightarrow$ b_1, b_2, b_3 marked

$\{a_0, a_1, a_2; b_0\}, \{a_3; b_1, b_2, b_3\}$

S_0	1	2	3
a_0	0	0	0
a_1	0	1	0
a_2	1	0	0
a_3	1	1	1

,

S_1	3	4	5
b_0	0	0	0
b_1	1	0	0
b_2	1	0	1
b_3	1	1	1

$\{\cancel{a_0}, \cancel{a_1}, a_2; b_0\}, \{\cancel{a_3}; \cancel{b_1}, \cancel{b_2}, \cancel{b_3}\}$

S_0	1	2	3
a_2	1	0	0

,

S_1	3	4	5
b_0	0	0	0

Agreeing2

Advantages

- Information propagation through tuples
- Asymptotically faster
- Overhead for reading and writing decreases

Continous Implicit Agreeing

- Tree search through possible Gluings
- Works only on tuple markings
- Keep obtained knowledge
  - Persistent marking
- Flexible in the Gluing order

General Algorithm

1. Pick symbol
2. Mark all yet unmarked assignments, except one
    - Guess assignment
3. Run Agreeing2
4. Backtrack on contradiction

Guessing Heuristics

- Which symbol keeps search tree narrow
- Gluing complexity roughly $2^{max_i |X(i)| - i} \longrightarrow$ sort and keep $|X(i)| - i = |X_0 \cup X_1 \cup \ldots X_i| - i$ low
- Better: choose symbol with smallest $|V_i|$ (only unmarked assignments)
- Other heuristics possible, e.g. maximum $|X_i|/|V_i|$ etc.

Example Result

$n = m = 150, l = 5$

MiniSAT		dynglue	
Decisions	6844	Guesses	1549
Conflicts	5046	Contradictions	280
Propagations	120239	Tuple Propagations	541128
Time	0.15s	Time	0.08s

Open Questions

- Conflict handling
- Learning
  - Dynamic
  - Static
- Heuristics
  - Random systems
  - Equation systems from ciphers