

# Attacking AES via Solving Systems in the Key Variables Only

Stanislav Bulygin, joint work with Michael Brickenstein

University of Kaiserslautern

First International Conference on Symbolic Computation and Cryptography  
April 28 – April 30  
Beijing 2008

April 29, 2008

## Motivation

- final goal: attack AES (Advanced Encryption Standard since 2001)
- originally encodes 128-bit blocks with 128-bit keys (4 by 4 arrays of bytes), 10 rounds
- test algorithms on easier, but similar ciphers
  - Small Scale Variants of AES
    - variable number of
      - rounds ( $n$ ) (1 – 10)
      - rows ( $r$ ), columns ( $c$ ) in the arrays (1,2, or 4)
      - size of a bit vector  $e$  (4 or 8)

## History

- French: Courtois, Pieprzyk  $\rightarrow \mathbb{F}_2$
- English: Cid, Murphy, Robshaw  $\rightarrow \mathbb{F}_{2^8}$
- German: Buchmann, Pyshkin, Weinmann  $\rightarrow$  0-dimensional representation

## Structure of equations

- System for AES can be seen as iterative blocks of equations, where "output" variables of one block are "input" variables for the next block (system  $S$ ). So blocks only intersect on a frontier.
- every block has similar structure and equations therein are of two types:
  - quadratic equations correspond to Substitution Box (nonlinear operation)
  - linear equations correspond to the Diffusion Layer

## Initial system

Schematically: at the beginning we have the system  $S$  of the form

$$w_0 = p + k_0,$$

$$SBOX(x_i, w_{i-1}) = 0, \quad i = 1, \dots, n,$$

$$w_i = L(x_i) + k_i, \quad i = 1, \dots, n,$$

$$SBOX_K(s_i, k_{i-1}) = 0, \quad i = 1, \dots, n,$$

$$k_i = L_K(s_i), \quad i = 1, \dots, n,$$

$$c = L(x_n) + k_n,$$

together with the field equations over  $\mathbb{F}_2$ . Here  $SBOX, SBOX_K$  are quadratic S-Box transformations for the encryption and the key schedule resp.;  $L, L_K$  are affine transformations.

## Rewriting

- System  $S$  has many variables, which we do not need!
- Rewrite equations in the S-Boxes so that every output variable could be expressed via input variables of the S-Box
  - + It is easier to see how every variable depends on preceding variables
  - Degree of equations rises from 2 to 3 (for  $e = 4$ ) or to 7 (for  $e = 8$ )
- By writing equations in such a way, we have that major part of this system  $S_1$  is already a Gröbner basis w.r.t to some degree ordering
- Doing the normal form reduction of the remaining equations modulo the major part, we obtain the system  $S_2$  in the initial key variables only

## Rewritten system

Rewriting S-Boxes yields the system  $S_1$ :

$$\begin{aligned}w_0 &= p + k_0, \\x_i &= \text{sbox}(w_{i-1}), \quad i = 1, \dots, n, \\w_i &= L(x_i) + k_i, \quad i = 1, \dots, n, \\s_i &= \text{sbox}_K(k_{i-1}), \quad i = 1, \dots, n, \\k_i &= L_K(s_i), \quad i = 1, \dots, n, \\c &= L(x_n) + k_n,\end{aligned}$$

together with the field equations over  $\mathbb{F}_2$ . Here  $\text{sbox}$ ,  $\text{sbox}_K$  are higher degree S-Box transformations for the encryption and the key schedule resp.

## Software

Gröbner basis and normal form computations are done in PolyBoRi. Gröbner basis algorithm: `symmgbGF2`.

## Experimental results

Cipher	# eqs. $S_1$	# eqs. $S_2$	time, sec.	memory, MB
SR(10,2,2,4)	576	16	1205	170
SR(10,1,1,4)	164	4	0.02	75
SR(10,1,2,4)	288	8	0.2	79
SR(10,1,1,8)	328	8	2	183

## Number of variables

For both  $S_1$  and  $S_2$  systems the number of equations is equal to the number of variables. We do not count field equations.

## Analysis

- The method of  $S_2$  system gives an opportunity to use many plaintext/ciphertext pairs, it was not possible if working with the systems  $S$  or  $S_1$
- Drawback: high degree ( $r \cdot c \cdot e$ ) dense equations (every term appears practically with probability  $1/2$ ) in the resulting system  $S_2$  composed of only key variables.

## Cutting technique

- Let  $f_i(x_1, \dots, x_n) = 0, i = 1, \dots, m$  be a polynomial system over  $\mathbb{F}_2$ . If  $a \in \mathbb{F}_2^n$  is a solution:  $f_i(a) = 0 \forall i$ , such that  $\text{weight}(a) := \#\{i | a_i \neq 0\} = s$ , then  $a$  is also a solution of  $\tilde{f}_i(x_1, \dots, x_n) = 0, i = 1, \dots, m$ . Here  $\tilde{f}_i$  is obtained from  $f_i$  by dropping out the monomials of degree  $> s$
- So, if we are looking for solutions of low weight in a system composed of high degree polynomials, it is sufficient to consider low-degree parts of every polynomial in the system.

## Cutting technique

- Let  $\text{supp}(a) = \{i | a_i \neq 0\}$  and  $\text{weight}(a) = s$ . Perform a coordinate change  $x_i \mapsto x_i + 1$  for  $i \in I \subset \text{supp}(a)$  on the initial system  $f_i(x_1, \dots, x_n) = 0, i = 1, \dots, m$
- Let  $s' = s - |I|$ , then there is a solution  $a'$  for the system  $\tilde{f}_i(x_1, \dots, x_n) = 0, i = 1, \dots, m$ , where each  $\tilde{f}_i$  is obtained from  $f_i$  by dropping out the monomials of degree  $> s'$
- By doing  $x_i \mapsto x_i + 1$  for  $i \in I$  again we are able to find the solution  $a$  of weight  $s > s'$ , working with a system composed of polynomials of degree  $s'$
- Thus, by doing coordinate change  $x_i \mapsto x_i + 1$  for several  $i$ 's on the initial system  $S_1$  and working with linear (or quadratic) parts of equations we reduce the problem of finding a key to solving many simple linear (or quadratic) systems instead of one large  $S_2$

## Cutting technique

- Using this cutting technique even with a naive PYTHON script half of the key space of
  - AES-10-2-2-4 can be scanned in  $< 3$  min
  - AES-10-1-2-8 can be scanned in  $< 30$  min
  - negligible memory consumption
- Exhaustive search turns out to be a particular case of the above, if we consider only constant terms in the system every time; coordinate change corresponds to a trial key selection. It is faster, than linear or quadratic analogue
- Further analysis may reveal benefits of the linear (or quadratic) "cutting technique"

## Future work

- Apply the method to many plaintext/ciphertext pairs approach
- Understand better the cutting technique; do implementations
- Try the described method on other block ciphers

## S-Box rewriting

An example on how an S-Box changes for  $e = 4$  follows.

- A quadratic S-Box from the system  $S$ :

$$x_2 w_3 + x_1 w_3 + x_3 w_2 + x_2 w_2 + x_3 w_1 + x_0 w_0 + 1 = 0,$$

$$x_3 w_3 + x_1 w_3 + x_2 w_2 + x_3 w_1 + x_0 w_1 + x_1 w_0 = 0,$$

$$x_1 w_3 + x_2 w_2 + x_0 w_2 + x_3 w_1 + x_1 w_1 + x_2 w_0 = 0,$$

$$x_1 w_3 + x_0 w_3 + x_2 w_2 + x_1 w_2 + x_3 w_1 + x_2 w_1 + x_3 w_0 = 0.$$

- A cubic S-Box from the system  $S_1$ :

$$x_0 = w_3 w_2 w_1 + w_2 w_1 w_0 + w_2 w_1 + w_2 w_0 + w_3 + w_2 + w_1 + w_0,$$

$$x_1 = w_3 w_1 w_0 + w_3 w_1 + w_2 w_1 + w_2 w_0 + w_1 w_0 + w_3,$$

$$x_2 = w_3 w_2 w_0 + w_3 w_0 + w_2 w_0 + w_1 w_0 + w_3 + w_2,$$

$$x_3 = w_3 w_2 w_1 + w_3 w_2 + w_3 w_1 + w_3 w_0 + w_3 + w_2 + w_1.$$