

Obtaining and solving systems of equations in key variables only for the small variants of AES

Stanislav Bulygin (joint work with Michael Brickenstein)

Applications of Computer Algebra 2008
Gröbner Bases and their Applications
RISC, Castle of Hagenberg, Austria

July 27, 2008

Symmetric cryptography

- Idea is to make some message m secret by applying an *encryption transformation* E dependent on a secret key e :
 $c = E_e(m)$ – **Encryption**.
- To e corresponds the decryption key d :
 $D_d(c) = D_d(E_e(m)) = m$ – **Decryption**.
- e and d are essentially the same \rightarrow name "symmetric".
- An adversary is assumed to have an access to some number of plaintext/ciphertext pairs $(m_i, c_i)_{1 \leq i \leq N} \rightarrow$ tries to figure out the key pair (e, d) used – *known-plaintext attack*

Introduction

Block ciphers

- A standard way to realize a symmetric cryptosystem is by means of a **block cipher**.
- Idea is to partition a message to blocks and then encrypt these blocks by doing some number of iterations of a certain transformation.
- One such transformation: *round*. S-P networks are frequently used: there one round is a successive application of a *substitution* (to provide *confusion*) and a *permutation* (to provide *diffusion*).
- Classical: DES (Data Encryption Standard) - encrypts 64-bit blocks with 56-bit keys.
- Current: AES (Advanced Encryption Standard) - encrypts 128-bit blocks with 128-bit keys.

Introduction

Algebraic cryptanalysis

- Represent a block cipher via a system of equations over a finite field. This system depends on the plaintext/ciphertext pair(s) and its solution should reveal the key used.
- Conventional methods of *statistical* cryptanalysis require many plaintext/ciphertext pairs, whereas algebraic may (in theory) work with just one or few pairs.

Historical development

- French: Courtois, Pieprzyk \longrightarrow description of AES and Serpent over \mathbb{F}_2
- British: Cid, Murphy, Robshaw \longrightarrow description of AES over \mathbb{F}_{2^8} introducing Big Encryption Standard (BES)
- German: Buchmann, Pyshkin, Weinmann \longrightarrow 0-dimensional representation of AES systems

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 attacks 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)

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.
- AES is an S-P network, so every block has similar structure and equations therein are of two types:
 - quadratic equations correspond to Substitution Box (nonlinear operation) – S-part;
 - linear equations correspond to the Diffusion Layer – P-part

Initial system

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

$$\begin{aligned}w_0 &= p + k_0, \\SBOX(x_i, w_{i-1}) &= 0, & i &= 1, \dots, n, \\w_i &= L(x_i) + k_i, & i &= 1, \dots, n-1, \\SBOX_K(s_i, k_{i-1}) &= 0, & i &= 1, \dots, n, \\k_i &= L_K(s_i) + L'_K(k_{i-1}), & i &= 1, \dots, n, \\c &= L(x_n) + k_n,\end{aligned}$$

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, 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

Method (Additional slide)

S-Box rewriting

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

- A quadratic S-Box from the system S :

$$\begin{aligned}x_2 w_3 + x_1 w_3 + x_3 w_2 + x_2 w_2 + x_3 w_1 + x_0 w_0 + 1 &= 0, \\x_3 w_3 + x_1 w_3 + x_2 w_2 + x_3 w_1 + x_0 w_1 + x_1 w_0 &= 0, \\x_1 w_3 + x_2 w_2 + x_0 w_2 + x_3 w_1 + x_1 w_1 + x_2 w_0 &= 0, \\x_1 w_3 + x_0 w_3 + x_2 w_2 + x_1 w_2 + x_3 w_1 + x_2 w_1 + x_3 w_0 &= 0.\end{aligned}$$

- A cubic S-Box from the system S_1 :

$$\begin{aligned}x_0 &= w_3 w_2 w_1 + w_2 w_1 w_0 + w_2 w_1 + w_2 w_0 + w_3 + w_2 + w_1 + w_0, \\x_1 &= w_3 w_1 w_0 + w_3 w_1 + w_2 w_1 + w_2 w_0 + w_1 w_0 + w_3, \\x_2 &= w_3 w_2 w_0 + w_3 w_0 + w_2 w_0 + w_1 w_0 + w_3 + w_2, \\x_3 &= w_3 w_2 w_1 + w_3 w_2 + w_3 w_1 + w_3 w_0 + w_3 + w_2 + w_1.\end{aligned}$$

Rewritten system

Rewriting S-Boxes yields the system S_1 :

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

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

Method

Software

Gröbner basis and normal form computations are done in PolyBoRi. Gröbner basis command: `groebner_basis`.

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 is quite cumbersome 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.

Meet-in-the-middle attack

- Meet-in-the-middle attack is quite standard tool in cryptanalysis.
- Idea: if we try to attack n rounds, reverse the last $n/2$ rounds and get some "matching" conditions with the first $n/2$ rounds.
- In our experiments we attack 2 rounds: inverse the last round and get equations in the key variables only using binding connections of the 1st and the 2nd rounds.
- For 2 rounds one gets degree- $(e - 1)$ equations instead of $r \cdot c \cdot e$.

Method

Meet-in-the-middle attack: experimental results

Here t_{red} is time to obtain key-variables-only equations from one pair (via normal form reductions), and t_{solve} is time to solve the final key-variables-only system

Cipher	# of pairs	t_{red} , sec.	t_{solve} , sec.	memory, MB
SR(2,2,4,4)	8	0.10	1.3	35
SR(2,4,2,4)	8	0.15	1.4	37
SR(2,4,4,4)	8	0.30	3.7	61
SR(2,2,2,8)	64	1.2	$\approx 3,300$	$\approx 1,200$
SR(2,1,4,8)	64	0.4	≈ 950	≈ 350

Note that for $e = 4$ -instances fast dense linear algebra implemented in libm4ri by Gregory Bard/Martin Albrecht is used, for $e = 8$, the system reduction was done using ZDDs (which consume less memory in this case).

Future work

- Try to exploit more structure in key-variables-only systems
- Understand better the "cutting technique"; do implementations
- Try the described method on other block ciphers (preferably with small S-Boxes)