Advanced Cryptology

Algebraic Cryptanalysis

Mgr. Martin Jureček

Czech Technical University in Prague
Faculty of Information Technology
Department of Computer Systems

Příprava studijních programů Informatika pro novou fakultu ČVUT je spolufinancována Evropským sociálním fondem a rozpočtem Hlavního města Prahy v rámci Operačního programu Praha — adaptabilita (OPPA) projektem CZ.2.17/3.1.00/31952 – „Příprava a zavedení nových studijních programů Informatika na ČVUT v Praze“.
Praha & EU: Investujeme do vaší budoucnosti

„Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type.“

C.E. Shannon, 1949

Algebraic Cryptanalysis

Introduction

- Algebraic cryptanalysis, abbr. AC, is a new area of cryptanalysis which has gained much attention in recent years.
- The principle of AC consists in transferring the problem of breaking the cryptosystem to the problem of solving system of polynomial equations over a finite field.
- AC is mainly used in symmetric cryptanalysis:
  - examples of block ciphers: AES, DES
  - examples of stream ciphers: E0 - Bluetooth, Toyocrypt,

  however AC was used also in assymetric cryptography.

Process

- Process of AC is divided into the following two steps:

  1: Convert the cipher and possibly some supplemental information into a system of polynomial equations.
  2: Apply some algorithm to calculate the solution of the system of polynomial equations and derive a secret key.

- However there is a fundamental problem: solving a system of quadratic equations over any finite field is NP-Complete.
- Why should we convert the problem of breaking a cipher to the problem for which we do not know fast algorithm(i.e. with polynomial complexity)?

Algebraic Cryptanalysis

1.step

- The first step of AC consists of using the structure of cipher and supplemental information for creating a system of equations, that describe behavior of ciphers for a specific case.
- We consider the system of equations over a finite field, usually GF(2).
- Our aim is to obtain the smallest possible system of equations, that contains the polynomials with the lowest degree. Method which derives the system of equations depends on the cipher.
- If the system contains only linear equations, then we use e.g. Gaussian elimination, which has a cubic complexity and we receive the result relatively quickly.
- Well proposed ciphers provide a system of polynomial equations that we can not solve in a short time.

2.step

- The main part of AC is the second step, in which we solve the system of the polynomial equations over a finite field.
- Example: AC of AES-128 system includes approximately 8000 equations with 1600 variables and AC of AES-256 system has 22400 equations with 4480 variables.
- There are several methods for calculating these nonlinear systems, however none of them is fast.
- If there is an efficient algorithm for solving this problem (i.e. with polynomial complexity), then P = NP and that would be considered as surprise.

Algebraic Cryptanalysis

Methods for solving a system of polynomial equations

- Method of the type „guess and determine“ is simply, however it is not very effective.
- We „guess“(using brute force) values of appropriate variables and then we can easily calculate the rest of the system.
- Other methods are: linearization, that we will introduce later, XL algorithm and the method of Gröbner bases.
- Gröbner bases is very perspective method and has been successfully applied e.g. on AES.

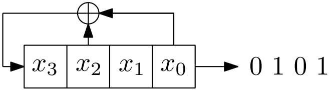
Algebraic Cryptanalysis

Application AC on some types of stream ciphers

- Examples of using AC - three classes of stream ciphers:
  - LFSR(Linear Feedback Shift Register) - cipher consists of only one LFSR
  - NLCG(Nonlinear Combination Generator) - cipher consists of more LFSRs and nonlinear boolean function which is used to compute output(function uses only output bits from LFSRs)
  - NLFG(Nonlinear Filter Generator) - cipher consists of only one LFSR and nonlinear boolean function which is used to compute output(function uses all bits from LFSR)
- AC is „known-plaintext attack“ since we assume that attacker knows bits of keystream.

Example of application AC on LFSR

- Let's consider the following simple LFSR of length 4:



- Our goal is to gain values $x_0, \ldots, x_3 \in \mathrm{GF}(2)$.
-

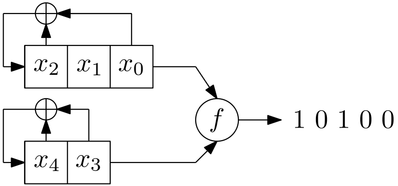
clock	LFSR state	output	equations
1.	(x_3, x_2, x_1, x_0)	x_0	$x_0 = 0$
2.	$(x_0 \oplus x_2, x_3, x_2, x_1)$	x_1	$x_1 = 1$
3.	$(x_1 \oplus x_3, x_0 \oplus x_2, x_3, x_2)$	x_2	$x_2 = 0$
4.	$(x_0, x_1 \oplus x_3, x_0 \oplus x_2, x_3)$	x_3	$x_3 = 1$

- In this case the system of equations is solved immediately.

Algebraic Cryptanalysis

Example of application AC on NLCG

- Let's consider the following example with two LFSRs:



  and nonlinear function $f(v_1, v_2) = v_1 + v_1 v_2$, where v_1 is the output bit of the first LFSR and v_2 of the second one.

- Goal is to compute values $x_0, \ldots, x_4 \in \mathrm{GF}(2)$.

Generating the system of equations

- Let the first LFSR is R_1 and the second LFSR is R_2 and their output bits are v_1 and v_2 respectively

clock	state of R_1	state of R_2	v_1, v_2
1.	(x_2, x_1, x_0)	(x_4, x_3)	x_0, x_3
2.	$(x_0 \oplus x_2, x_2, x_1)$	$(x_3 \oplus x_4, x_4)$	x_1, x_4
3.	$(x_0 \oplus x_1 \oplus x_2, x_0 \oplus x_2, x_2)$	$(x_3, x_3 \oplus x_4)$	$x_2, x_3 \oplus x_4$
4.	$(x_0 \oplus x_1, x_0 \oplus x_1 \oplus x_2, x_0 \oplus x_2)$	(x_4, x_3)	$x_0 \oplus x_2, x_3$
5.	$(x_1 \oplus x_2, x_0 \oplus x_1, x_0 \oplus x_1 \oplus x_2)$	$(x_3 \oplus x_4, x_4)$	$x_0 \oplus x_1 \oplus x_2, x_4$

System of polynomial equations

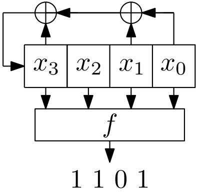
$$
\begin{aligned}
x_0 + x_0x_3 &= 1 \\
x_1 + x_1x_4 &= 0 \\
x_2 + x_2x_3 + x_2x_4 &= 1 \\
x_0 + x_2 + x_0x_3 + x_2x_3 &= 0 \\
x_0 + x_1 + x_2 + x_0x_4 + x_1x_4 + x_2x_4 &= 0
\end{aligned}
$$

- Can you solve it?
- (e.g. from the first equation is obvious that $x_0 = 1$)

Algebraic cCryptanalysis

Example of application AC on NLFG

- Let‘s consider the following example of LFSR:



  and nonlinear function $f(x_0, x_1, x_2, x_3) = x_0 + x_0x_1 + x_1x_3$.
- Our goal is to compute values $x_0, \ldots, x_3 \in$ GF(2).

Generating the system of equations

clock	state of LFSR
1.	(x_3, x_2, x_1, x_0)
2.	$(x_0 + x_1 + x_3, x_3, x_2, x_1)$
3.	$(x_0 + x_2 + x_3, x_0 + x_1 + x_3, x_3, x_2)$
4.	$(x_0, x_0 + x_2 + x_3, x_0 + x_1 + x_3, x_3)$

System of polynomial equations

$$\begin{aligned}
x_0 + x_0x_1 + x_1x_3 &= 1 \\
x_1 + x_0x_2 + x_2x_3 &= 1 \\
x_2 + x_0x_3 + x_3^2 &= 0 \\
x_3 + x_1x_3 + x_3^2 + x_0^2 + x_0x_1 &= 1
\end{aligned}$$

- Can you solve it?

Notes

- In NLFG, degree of the generated equations is upper bounded by the degree of nonlinear function f.
- The maximal degree of the equations is important factor and AC is more effective for lower degree.
- The important fact is that generating the equations does not depend on keystream.
- Therefore a generation of the system of polynomial equations can be done as a precomputation step and that's why the second step - solving of equations - is the most complicated.
- *Challenge: Try to create a system of equation for A5/1 stream cipher, which consists of 3 LFSRs and nonlinear clocking mechanism.*

Linearization

- At the end we present the basic technique for solving the system of multivariate polynomial equations called linearization.
- The algorithm is based on the following three steps:

  1: Substitute any product of variables by fresh variable.
  2: Solve the linear system (e.g. using Gaussian elimination).
  3: Plug the solution into the original system and check correctness of the solution.

- During linearization the linearized system can contain linear dependent equations and there have been proposed several improvements.

Linearization - example 1

- Consider the following system of equations over GF(2):

$$\begin{aligned} x + xy &= 1 \\ x + y &= 1 \\ x + y + xy &= 1 \end{aligned}$$

- In the first step we replace term xy with fresh variable z:

$$\begin{aligned} x + z &= 1 \\ x + y &= 1 \\ x + y + z &= 1 \end{aligned}$$

- In step 2 we easily compute the solution: $x = 1$, $y = 0$, $z = 0$
- In step 3 we check the correctness of the solution.

Linearization - example 2

- The following example illustrates the necessity of checking correctness of the solution:

$$\begin{aligned} x + xy &= 1 \\ x + y &= 0 \\ x + y + xy &= 0 \end{aligned}$$

- In step 1 we replace term xy with fresh variable z:

$$\begin{aligned} x + z &= 1 \\ x + y &= 0 \\ x + y + z &= 0 \end{aligned}$$

- By solving the system we obtain solution: $x = 1, y = 1, z = 0$.
- However after plugging it into the first equation we obtain: $1 + 1 * 1 \neq 1$, therefore checking the solution is necessary.