

Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks

Frederik Armknecht¹ Claude Carlet² Philippe Gaborit³
Simon Künzli⁴ Willi Meier⁴ Olivier Ruatta³

¹Universität Mannheim (Germany)

²Université Paris 8 and INRIA (France)

³Université de Limoges (France)

⁴FH Nordwestschweiz (Switzerland)

1 Introduction

- Filtered registers
- Algebraic attacks
- Fast algebraic attacks

2 Efficient Computation of Algebraic immunity

- Algebraic immunity
- Interpolation and algebraic immunity
- Experimental results

3 Efficient Computation for Fast Algebraic Attacks

- Arbitrary Functions
- Symmetric Functions
- Experimental Results

Stream ciphers: random generators

A natural candidate (fast): Linear Feedback Register

pb: as such easy to break

→ Idea: use more complex system based on LFSR:

→ Combined register, registers with memory, filtered registers

Boolean function : a function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ has a unique **algebraic normal form (ANF)** $f(x) = \bigoplus_{I \subset \{1, \dots, n\}} f_I \cdot \prod_{i \in I} x_i$.

The **algebraic degree** of a function, denoted $\deg f$, is the degree of its ANF.

Idea : For each clock, consider the values of a LFSR as the entries of a Boolean function in order to generate a keystream.

Remark : The linear complexity of the keystream is bounded above by

$$\sum_{i=0}^{\deg(f)} \binom{n}{i}.$$

Filtered LFSR

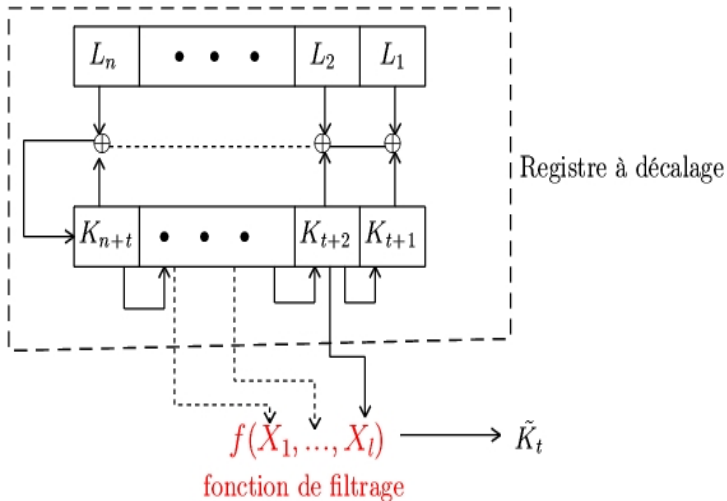


Figure: Filtered LFSR

Different kind of attacks on stream ciphers (correlation attacks, fast correlation attacks, algebraic attacks..).

- Principle of algebraic attacks : reduce the system to solving of an algebraic system.

Problem reduction

Let an LFSR with initial state $\mathbf{k} = (k_0, \dots, k_n)$ and defined by :

$$(K_{n+1+l}, \dots, K_{l+2}) = L * (K_{n+l}, \dots, K_{l+1})^t$$

and filtered by a function f , one considers the system :

$$\begin{cases} f(\mathbf{k}) = b_0 \\ f(L(\mathbf{k})) = b_1 \\ \vdots \\ f(L^l(\mathbf{k})) = b_l \end{cases}$$

where L^l is the $l^{\text{ième}}$ update of the LFSR.

Problem : Find \mathbf{k} , i.e. the $k_i, i \in \{1, \dots, n\}$, knowing L and f and the b_i .

Idea introduced by **Shannon '48**

To solve multivariate linear systems, several approaches have been used:

- 1 Gröbner basis (Faugère et Ars) -this method may need less bits of the stream.
- 2 XL, XSL (Courtois et al.)
- 3 Linearization

If f has algebraic degree d_f for a register with N variables, then there are up to $\sum_{i=0}^{d_f} \binom{N}{i} \approx \binom{N}{d_f}$ monomials (which are considered independently), which leads to the solving of a multivariate linear system in $\mathcal{O}\left(\binom{N}{d_f}^\omega\right)$ where ω is greater than the exponent of linear algebra 2.37, in practice 2.8 or 3.

For $N \geq 128$ and $d \approx n$, one reaches very quickly the limits of this attack for actual systems, complexity grows with d_f .

Courtois and Meier '03 : If one can find a function g with small degree d_g such that fg or $(1 + f)g \equiv 0$ then the number of unknowns for an algebraic attack can be reduced from $\binom{N}{d_f}$ to $\binom{N}{d_g}$.

$f(L^i(\mathbf{k})) = b_i$ becomes $g(L^i(\mathbf{k})) * f(L^i(\mathbf{k})) = 0 = b_i g(L^i(\mathbf{k}))$ and hence the equations become $g(L^i(\mathbf{k})) = 0$ for i such that $b_i = f(L^i(\mathbf{k})) \neq 0$.

- the gain is on the number of unknowns since $d_g < d_f$,
- this attack is very efficient for TOYOCRYPT and LILI for which d_g is small or very small.
- **Algebraic attacks: 2 steps**
 - Relation step: find g with low degree such that fg or $(1 + f)g = 0$.
 - Solve the induced multivariate linear system

Courtois '03, Armknecht '04, and Hawkes-Rose '04

Idea of the attack: consider a register with N variables filtered by f , find a relation : $fg = h$ with $e = \deg g$ small and $d = \deg h$ bigger.

One considers that the sequence of $h(L^i(x_1, \dots, x_N))$ can be obtained as a LFSR with linear complexity $D_N = \sum_{i=0}^d \binom{N}{i}$.

One uses Berlekamp-Massey algorithm to eliminate all monomials of weight greater than e .

→ system with $E_N = \sum_{i=0}^e \binom{N}{i}$ unknowns.

4 steps complexity:

- **Relation step.** One searches g and h with small degrees such that $fg = h$. Worst complexity: solving a linear system with $D + E$ equations, let $\mathcal{O}((D + E)^3)$. In general one considers $e < d$.
- **Pre-computation step.** Computation of linear relations to eliminate the terms of degrees greater than e in the equations. Needs $2D_N$ bits of stream with complexity $\mathcal{O}(D_N \log^2(D_N))$.
- **Substitution step.** One eliminates the monomials of degree greater than e . Complexity in $\mathcal{O}(E_N^2 D_N)$ but DFT Hawkes and Rose: $\mathcal{O}(E_N D_N \log(D_N))$.
- **Solving step.** One solves the system with E_N linear equations in $\mathcal{O}(E_N^3)$.

Applications:

Courtois improves the attack on ToyoCrypt by decreasing by 1 the degree of the equations.

Recent attack by Courtois on the SFINKS stream cipher.

Relation between algebraic and fast algebraic attacks

if $fg = h$ then $ffg = h = fh$ and $(1 + f)h = 0$, hence h is an annihilator of f .

Courtois, Meier '03, Meier, Pasalic, Carlet '04

Definition

Let f be a Boolean function, the algebraic immunity of f , denoted $AI(f)$, is defined by: $AI(f) := \min\{\deg(g) \mid fg \equiv 0 \text{ ou } (1+f) * g \equiv 0\}$.

Proposition

If f is Boolean function with n variables then :

$$AI(f) \leq \left\lceil \frac{n}{2} \right\rceil.$$

→ the AI is an important criterium which permit to evaluate of the resistance to algebraic attack for **linearization** and a **necessary condition** for resistance to Gröbner basis attacks.

For a given f one searches to find a **non null** function g of smallest degree such that : $fg = 0$.

Existing methods :

- naive method: one fixes a degree for g and by linearization one solves a linear system by Gaussian elimination. This gives an algorithm in $\mathcal{O}\left(\binom{n}{d_g}^\omega\right)$ for the computation of the AI ($\omega = 2.8$ or 3) and usually one searches for d_g maximum in $\left\lceil \frac{n}{2} \right\rceil$.
- Gröbner basis (Ars, Faugère), this improves sometimes the complexity but on the average the complexity is (experimentally) of same order than with naive linearization.

Reduction of the problem of computation of the AI into a problem of multivariate Lagrange interpolation:

Proposition

Solving $f(z)g(z) = 0$ for any $z \in \mathbb{F}_2^n$ is equivalent to find g such that $g(z) = 0$ if $f(z) = 1$.

For $\mathcal{Z} = f^{-1}(1)$ this is equivalent to the following multivariate Lagrange interpolation problem:

Find a non nul function g of minimal (non null) degree such that $g(z) = 0$ for any $z \in \mathcal{Z}$.

Multivariate quadratic interpolation algorithm

Our new method generalizes the case of univariate Newton interpolation to multivariate Newton interpolation.

- **Univariate quadratic interpolation algorithm**

$$Z = \{z_1, \dots, z_d\} \subset \mathbb{C}, \mathbf{v} = (v_1, \dots, v_d) \in \mathbb{C}^d.$$

Definition

Let $Z = \{z_1, \dots, z_d\} \subset \mathbb{F}_q$, the Vandermonde associated matrix Z is :

$$V_Z = \begin{pmatrix} z_1^0 & \cdots & z_1^d \\ \vdots & \ddots & \vdots \\ z_d^0 & \cdots & z_d^d \end{pmatrix}$$

One wants to solve the system $V_Z \mathbf{g}^t = \mathbf{v}$ (we shall consider $\mathbf{v} \neq 0$ for the algorithm). If one proceeds naively one has to invert the matrix in $\mathcal{O}(d^3)$.

Rather than interpolate in the basis $1, x, x^2, \dots, x^d$, one interpolates in the Newton basis defined by $\mathcal{N}_{i+1} = (x - z_1) \cdots (x - z_i)$.

Computation of Newton basis : One computes the sequence $\mathcal{N}_{i+1} = (x - z_i) * \mathcal{N}_i$, with $\mathcal{N}_0 = 1$. For each i , one makes the subtraction of two vectors of size i , such that the computation of all basis elements costs $\mathcal{O}(d^2)$ arithmetic operations.

Incremental method: One adds the new points to interpolate one after the other.

Suppose one gets $f(z_1) = v_1$ and $f(z_2) = v_2$ with $f(x) = f_0 + f_1(x - z_1)$ and we add a new point to interpolate: $f(z_3) = v_3$ then consider:
$$f(x) = f_0 + f_1(x - z_1) + f_2(x - z_1)(x - z_2)$$

→ to find the new f one has NOT to modify f_0 and f_1 , but simply compute f_2 for a linear cost

Univariate Newton Interpolation

Computation of the coefficients of the Newton basis (divided differences), one evaluates the following tree:

$f[z_1, \dots, z_d]$					
$f[z_1, \dots, z_{d-1}]$	$f[z_2, \dots, z_d]$				
\vdots		\ddots			
\vdots			\ddots		
$f[z_1, z_2]$	$f[z_2, z_3]$	$f[z_3, z_4]$	\cdots	$f[z_{d-1}, z_d]$	
$v_1 = f(z_1)$	$v_2 = f(z_2)$	$v_3 = f(z_3)$	\cdots	$v_{d-1} = f(z_{d-1})$	$v_d = f(z_d)$

for $f[z_i, \dots, z_k] = \frac{f[z_i, \dots, z_{k-1}] - f[z_{i+1}, \dots, z_k]}{z_i - z_k}$.

The computation of the line i at the level i needs i subtractions and i divisions. The computation of all the elements of this tree costs $\mathcal{O}(d^2)$ arithmetic operations.

Proposition

The unique solution of the Lagrange interpolation problem associated to Z and \mathbf{v} will be $f(x) = \sum_{i=1}^{d-1} f[z_1, \dots, z_i] \mathcal{N}_i(x)$.

Remark : Newton basis is the basis in which the Vandermonde matrix is **triangular**. A triangulation algorithm exploiting previous properties permits to effectuate a pivot on the Vandermonde matrix in $\mathcal{O}(d^2)$ arithmetic operations. Operations on divided differences can be interpreted in terms of linear algebra.

We generalize this algorithm to the multivariate case

Vandermonde determinant

For $X = x_1 x_2 \dots x_m$ one defines generalized exponents $\alpha = (\alpha_1, \dots, \alpha_m)$ by $X^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_m^{\alpha_m}$

One chooses an adequate graduated order for the monomials and one defines:

Definition

Let $X = \{z_1, \dots, z_D\} \subset \mathbb{F}_2^n$ and $E \subset \{\alpha_1, \dots, \alpha_D\} \subset \mathbb{F}_2^n$, one defines the Vandermonde matrix associated to X and E as:

$$V_{X,E} = \begin{pmatrix} z_1^{\alpha_1} & \dots & z_1^{\alpha_D} \\ \vdots & \ddots & \vdots \\ z_D^{\alpha_1} & \dots & z_D^{\alpha_D} \end{pmatrix}$$

and the Vandermonde determinant associated is $v_{X,E} = \det(V_{X,E})$.

this a generalization of the classical Vandermonde determinant.

Idea of incremental triangulation of $V_{Z,E}$

The idea is the generalization from the univariate case to the multivariate case for computing the solutions of the Lagrange interpolation problem:

- if one solves the Vandermonde system as a linear system one obtains $\mathcal{O}(D^3)$ arithmetic operations;
- if one solves the Lagrange system with an incremental pivot (one adds one point after another for a linear cost) one obtains a method in $\mathcal{O}(D^2)$ arithmetic operations.

Experimental results

The best known algorithms were cubic, our quadratic algorithm permits to compute the AI for functions with up to 20 – 25 variables rather than 16 – 18 in quadratic time.

n	d	weight	degree	non-linearity	AI
12	-1	2048	11	1984	5
13	-1	4096	12	4006	6
14	-1	8192	13	8064	6
15	-1	2^{14}	14	16204	6
16	-1	2^{15}	15	$2^{15} - 2^8$	7
17	-1	2^{16}	16	65174	7
18	-1	2^{17}	17	$2^{17} - 2^9$	7
19	-1	2^{18}	18	261420	7
20	-1	2^{19}	19	$2^{19} - 2^{10}$	7
20	$2^{18} - 2^9 + 1$	2^{19}	19	$2^{19} - 2^9$	9

Table: Algebraic immunity of inverse functions

Problem Statement

Given a Boolean function f :

find g and h of low degree, such that $f \cdot g = h$.

- f, g, h have n input variables, output is 0 or 1
- Given degrees: $\deg h = d, \deg g = e$
- Properties: d is at least $\mathcal{AI}(f)$, e is smaller

Example (with $n = 3, d = 2, e = 1$)

Let $f(x) = x_1 + x_1x_2x_3$.

With $g(x) = x_1 + x_3$, we find $h(x) = x_1 + x_1x_3$.

Problem Statement

Given a Boolean function f :

find g and h of low degree, such that $f \cdot g = h$.

- f, g, h have n input variables, output is 0 or 1
- Given degrees: $\deg h = d, \deg g = e$
- Properties: d is at least $\mathcal{AI}(f)$, e is smaller

Example (with $n = 3, d = 2, e = 1$)

Let $f(x) = x_1 + x_1x_2x_3$.

With $g(x) = x_1 + x_3$, we find $h(x) = x_1 + x_1x_3$.

Representation of Boolean Functions

How to represent f , g , h ?

- Algebraic normal form $f(x) = \bigoplus_{\alpha} f_{\alpha} x^{\alpha}$
- Coefficient vector $C(f) = (f_0, \dots, f_{2^n-1})$
- Truth table $T(f) = (f(0), \dots, f(2^n - 1))$

We represent functions by $T(f)$, $C(g)$, $C(h)$.

- $C(g)$ is of size $E = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{e}$
- $C(h)$ is of size $D = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{d}$
- E is much smaller than D

Representation of Boolean Functions

How to represent f , g , h ?

- Algebraic normal form $f(x) = \bigoplus_{\alpha} f_{\alpha} x^{\alpha}$
- Coefficient vector $C(f) = (f_0, \dots, f_{2^n-1})$
- Truth table $T(f) = (f(0), \dots, f(2^n - 1))$

We represent functions by $T(f)$, $C(g)$, $C(h)$.

- $C(g)$ is of size $E = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{e}$
- $C(h)$ is of size $D = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{d}$
- E is much smaller than D

State of the Art

Q: How to solve the problem and find g , h ?

A: Given $T(f)$, set up a linear system in $C(g)$ and $C(h)$.

State of the art:

- Set up equations in g_β and h_γ for random values z :

$$f(z) \cdot \bigoplus_{\beta} g_{\beta} z^{\beta} = \bigoplus_{\gamma} h_{\gamma} z^{\gamma}$$

- $D + E$ variables, requires about $D + E$ equations
- Gaussian elimination in $\mathcal{O}((D + E)^3) = \mathcal{O}(D^3)$ time

Our approach:

separate equations for $C(g)$ and $C(h)$, solve smaller system ...

State of the Art

Q: How to solve the problem and find g , h ?

A: Given $T(f)$, set up a linear system in $C(g)$ and $C(h)$.

State of the art:

- Set up equations in g_β and h_γ for random values z :

$$f(z) \cdot \bigoplus_{\beta} g_{\beta} z^{\beta} = \bigoplus_{\gamma} h_{\gamma} z^{\gamma}$$

- $D + E$ variables, requires about $D + E$ equations
- Gaussian elimination in $\mathcal{O}((D + E)^3) = \mathcal{O}(D^3)$ time

Our approach:

separate equations for $C(g)$ and $C(h)$, solve smaller system ...

How to Set up New Equations

- Express a **single** coefficient h_γ in $C(g)$ and $T(f)$

Theorem

$$h_\gamma = \bigoplus_{\beta} \binom{\gamma}{\beta} A_{\gamma,\beta} \cdot g_\beta \quad \text{with} \quad A_{i,j} = \bigoplus_k \binom{i}{k} \binom{k}{j} f(k)$$

- Choose random γ with $|\gamma| > d$, then $h_\gamma = 0$
- Only E variables, requires about E equations
- With our algorithm, equations can be set up in $\mathcal{O}(DE^2)$ time
- Gaussian elimination in $\mathcal{O}(E^3)$ time

For any f , assess immunity in $\mathcal{O}(DE^2)$ time

How to Set up New Equations

- Express a **single** coefficient h_γ in $C(g)$ and $T(f)$

Theorem

$$h_\gamma = \bigoplus_{\beta} \binom{\gamma}{\beta} A_{\gamma,\beta} \cdot g_\beta \quad \text{with} \quad A_{i,j} = \bigoplus_k \binom{i}{k} \binom{k}{j} f(k)$$

- Choose random γ with $|\gamma| > d$, then $h_\gamma = 0$
- Only E variables, requires about E equations
- With our algorithm, equations can be set up in $\mathcal{O}(DE^2)$ time
- Gaussian elimination in $\mathcal{O}(E^3)$ time

For any f , assess immunity in $\mathcal{O}(DE^2)$ time

Symmetric Functions

- Is it easier to set up equations for **symmetric** functions f ?
- $f(x)$ depends only on weight $|x|$ of input, notation $f^s(|x|)$
- Derive a much simpler relation for h_γ

Theorem

$$h_\gamma = \bigoplus_\beta \binom{\gamma}{\beta} A_{|\gamma|,|\beta|}^s \cdot g_\beta \quad \text{with} \quad A_{i,j}^s = \bigoplus_k \binom{i-j}{i-k} f^s(k)$$

- $A_{i,j}^s$ computed in negligible time
- With our algorithm, equations can be set up in $\mathcal{O}(E^2)$ time
- Gaussian elimination in $\mathcal{O}(E^3)$ time

For symmetric f , assess immunity in $\mathcal{O}(E^3)$ time

Symmetric Functions

- Is it easier to set up equations for **symmetric** functions f ?
- $f(x)$ depends only on weight $|x|$ of input, notation $f^s(|x|)$
- Derive a much simpler relation for h_γ

Theorem

$$h_\gamma = \bigoplus_\beta \binom{\gamma}{\beta} A_{|\gamma|,|\beta|}^s \cdot g_\beta \quad \text{with} \quad A_{i,j}^s = \bigoplus_k \binom{i-j}{i-k} f^s(k)$$

- $A_{i,j}^s$ computed in negligible time
- With our algorithm, equations can be set up in $\mathcal{O}(E^2)$ time
- Gaussian elimination in $\mathcal{O}(E^3)$ time

For symmetric f , assess immunity in $\mathcal{O}(E^3)$ time

Attack on the Majority Function

- Majority function with n input variables:

$$f(x) = \begin{cases} 0 & \text{if } |x| \leq \lfloor n/2 \rfloor \\ 1 & \text{otherwise} \end{cases}$$

- f is symmetric and has **maximum AI**
- What about immunity against fast algebraic attacks?

Theorem

There exist g and h with $d = \lfloor n/2 \rfloor + 1$ and $e = d - 2^j$ (with maximum $j = 0, 1, \dots$ such that $e > 0$).

- Example: for $n = 9$, we find $d = 5$ and $e = 5 - 2^2 = 1$

Majority function has weak immunity against FAA's

Attack on the Majority Function

- Majority function with n input variables:

$$f(x) = \begin{cases} 0 & \text{if } |x| \leq \lfloor n/2 \rfloor \\ 1 & \text{otherwise} \end{cases}$$

- f is symmetric and has **maximum AI**
- What about immunity against fast algebraic attacks?

Theorem

There exist g and h with $d = \lfloor n/2 \rfloor + 1$ and $e = d - 2^j$ (with maximum $j = 0, 1, \dots$ such that $e > 0$).

- Example: for $n = 9$, we find $d = 5$ and $e = 5 - 2^2 = 1$

Majority function has weak immunity against FAA's

Experimental Results

Non-symmetric functions f with maximum \mathcal{AI} :

- Examples in [Dalai, Gupta, Maitra] at FSE'05
- We find g and h with:

$$\deg h = \mathcal{AI}(f), \deg g = 1$$

- **Perfect situation** for a fast algebraic attack!
- Reaction from [DGM]: related functions with better immunity

Open problems:

- Find Boolean functions with a good \mathcal{AI} AND a good generalized \mathcal{AI} .
- Possible candidate functions, no general proven results.

Experimental Results

Non-symmetric functions f with maximum \mathcal{AI} :

- Examples in [Dalai, Gupta, Maitra] at FSE'05
- We find g and h with:

$$\deg h = \mathcal{AI}(f), \deg g = 1$$

- **Perfect situation** for a fast algebraic attack!
- Reaction from [DGM]: related functions with better immunity

Open problems:

- Find Boolean functions with a good \mathcal{AI} AND a good generalized \mathcal{AI} .
- Possible candidate functions, no general proven results.

Thank you!

