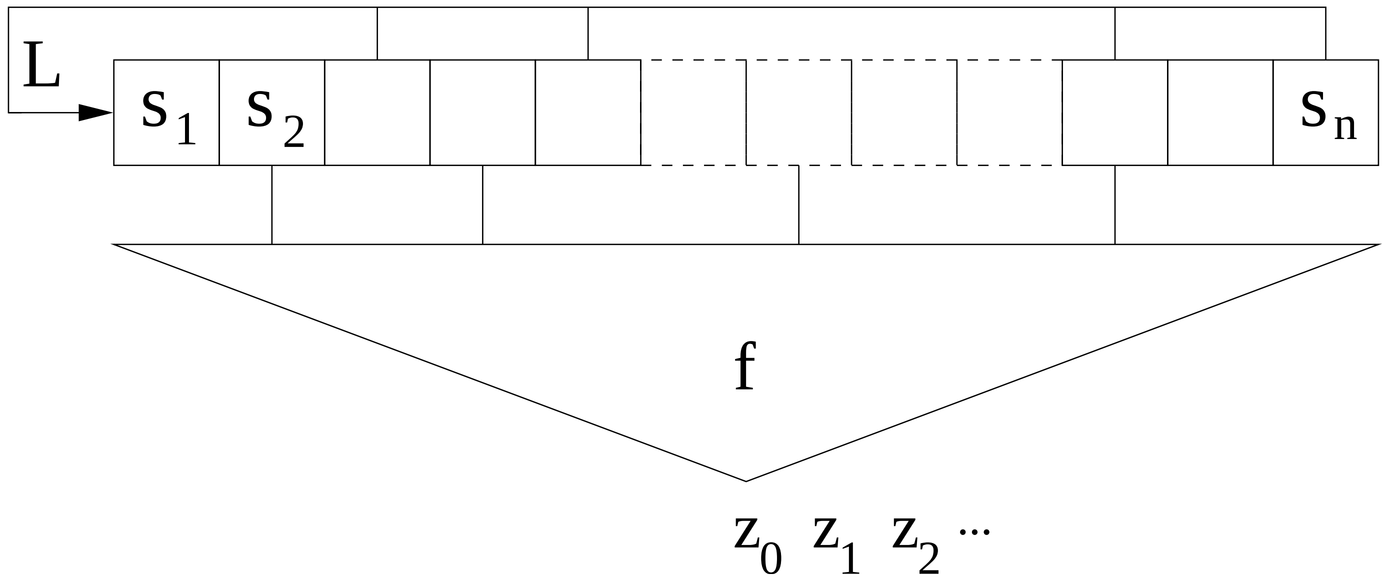
Using Wiedemann's algorithm to compute the algebraic immunity

Frédéric Didier
INRIA, projet CODES

Outline

Part 1
Introduction

A stream cipher : the filtered LFSR



Encoding/Decoding :

XOR the keystream bits(z_i) with the message bits

Secret Key : the initial state of the pseudo random generator

Algebraic attacks principle

Algebraic Normal Form (ANF) of a Boolean function f :

$$f(x_1, \dots, x_n) = \sum_{\mathbf{m} \in \mathbf{F}_2^n} f_{\mathbf{m}}\, x_1^{m_1} \cdots x_n^{m_n} \qquad f_{\mathbf{m}} \in \mathbf{F}_2$$

$$f(\mathbf{x}) = \sum_{\mathbf{m} \in \mathbf{F}_2^n} f_{\mathbf{m}}\, \mathbf{x}^{\mathbf{m}}$$

Basic idea : solve the algebraic system given by

$$f(L^i(s_1, \dots, s_n)) = z_i \quad \forall i \geq 0$$

Problem : The degree of f is usually too high

Algebraic attacks [Courtois Meier 03]

If we find a low degree Boolean function g such that

$$\forall \mathbf{x} \in \mathbf{F}_2^n \quad f(\mathbf{x})g(\mathbf{x}) = 0$$

It is called an annihilator of f and we have

$$\forall \mathbf{x} \in \mathbf{F}_2^n \quad f(\mathbf{x}) = 1 \quad \Longrightarrow \quad g(\mathbf{x}) = 0$$

So we get a new system in the degree of g

$$g(L^i(s_1, \ldots, s_n)) = 0 \quad \text{for } i \geq 0 \text{ and } z_i = 1$$

Remark : when $z_i = 0$ we use annihilators of $1 + f$

Our issue : computing annihilators of f

We can use Gröbner basis :

Find low degree polynomial in the ideal

$$\langle 1 + f(X_1, \ldots, X_n), X_1^2 - X_1, \ldots, X_n^2 - X_n \rangle$$

But the linear algebra approach is more efficient in practice

Remark : We place ourselves in the case of a general f

Part 2

Linear algebra and algebraic immunity

Goal : existence/computation of relations

Let
- f be an n-variable Boolean function $(\mathbf{F}_2^n \rightarrow \mathbf{F}_2)$
- d and e be given degrees $(e \geq d)$

For normal algebraic attacks

- g $\quad deg(g) \leq d$ and $fg = 0$

For fast algebraic attacks

- g and h $\quad deg(g) \leq d$, $deg(h) \leq e$ and $fg = h$

Degree at most d Boolean function space

Using the Algebraic Normal Form, a g in this space can be writted in a unique way as

$$g(\mathbf{x}) = \sum_{|\mathbf{m}| \leq d} g_{\mathbf{m}} \mathbf{x}^{\mathbf{m}} \qquad |\mathbf{m}| \stackrel{\text{def}}{=} \sum_i m_i$$

Basis : $(\mathbf{x}^{\mathbf{m}})_{|\mathbf{m}| \leq d}$

Dimension : $D \stackrel{\text{def}}{=} \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{d}$

So we will represent g by D coefficients $(g_{\mathbf{m}})_{|\mathbf{m}| \leq d}$ in $\mathbf{F}_2$

Link with Reed-Muller codes

A Boolean functions g can be represented by its images list

$$g(\mathbf{0}), g(\mathbf{1}), \ldots, g(\mathbf{2^n - 1})$$

RM(d,n) is by definition the space of all Boolean functions of degree at most d with this representation

Previous slide : usual encoding for Reed-Muller codes

Degree d evaluation matrix

Let
- $\{\mathbf{m}_1, \ldots, \mathbf{m}_D\}$ an order on $\{\mathbf{m} \in \mathbf{F}_2^n, |\mathbf{m}| \leq d\}$
- $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ a set of N points in $\mathbf{F}_2^n$

$$\mathbf{V}^d_{\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}} \stackrel{\text{def}}{=} \left(\mathbf{x}_i^{\mathbf{m}_j}\right)_{i\in\{1,\ldots,N\},\ j\in\{1,\ldots,D\}}$$

For g such that $g(\mathbf{x}) = \sum_{i=1}^{D} g_{\mathbf{m}_i}\mathbf{x}^{\mathbf{m}_i}$ $(\deg(g) \leq d)$

$$\mathbf{V}^d_{\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}} \begin{pmatrix} g_{\mathbf{m}_1} \\ \vdots \\ g_{\mathbf{m}_D} \end{pmatrix} = \begin{pmatrix} g(\mathbf{x}_1) \\ \vdots \\ g(\mathbf{x}_N) \end{pmatrix}$$

Basic Algebraic attacks

Find g of degree $\leq d$ such that $fg = 0$

$$\forall \mathbf{x} \in \{\mathbf{x}, f(\mathbf{x}) = 1\} \quad g(\mathbf{x}) = 0$$

Finding g is the same as solving the $|f| \times D$ system

$$\mathbf{V}^{d}_{\{\mathbf{x}, f(\mathbf{x})=1\}}\, \overline{g} = 0$$

where the D unknown coeffs of g are in the vector $\overline{g}$

Equivalent problems

$\mathbf{V}^d_{\mathbf{F}_2^n}$ is actually the usual generator matrix of RM(d,n)

Solving the system $\mathbf{V}^d_{\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}}\,\overline{g} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$

- codewords/functions for which only the values at positions $\mathbf{x}_i$ are known (and equal to y_i)

  - Decoding over the erasure channel
  - Multivariate interpolation problem

Fast Algebraic attacks

g of degree $\leq d$ and h of degree $\leq e$ such that $fg = h$

$$\forall \mathbf{x} \in \mathbf{F}_2^n \quad f(\mathbf{x})g(\mathbf{x}) + h(\mathbf{x}) = 0$$

Finding g and h $\Leftrightarrow$ solving the $2^n \times (E + D)$ system

$$\left(\mathsf{Diag}(f(\mathbf{x})_{\mathbf{x}\in\mathbf{F}_2^n})\mathbf{V}^d_{\mathbf{F}_2^n} \;\middle|\; \mathbf{V}^e_{\mathbf{F}_2^n}\right)\begin{pmatrix} \overline{g} \\ \overline{h} \end{pmatrix} = 0$$

Fast Algebraic attacks - improved system

$\mathbf{V}^e_{\{\mathbf{x},|\mathbf{x}|\leq e\}}$ is involutive

So, given the equation $h(\mathbf{x}) = f(\mathbf{x})g(\mathbf{x})$ we have

$$\mathbf{V}^e_{\{\mathbf{x},|\mathbf{x}|\leq e\}}\overline{h} \;=\; \mathrm{Diag}(f(\mathbf{x})_{|\mathbf{x}|\leq e})\mathbf{V}^d_{\{\mathbf{x},|\mathbf{x}|\leq e\}}\overline{g}$$

$$\overline{h} = \mathbf{M}\overline{g} \stackrel{\text{def}}{=} \mathbf{V}^e_{\{\mathbf{x},|\mathbf{x}|\leq e\}}\mathrm{Diag}(f(\mathbf{x})_{|\mathbf{x}|\leq e})\mathbf{V}^d_{\{\mathbf{x},|\mathbf{x}|\leq e\}}\overline{g}$$

and we get a new $2^n \times D$ system (actually $(2^n - E) \times D$)

$$\left(\mathrm{Diag}(f(\mathbf{x})_{\mathbf{x}\in\mathbf{F}_2^n})\mathbf{V}^d_{\mathbf{F}_2^n} + \mathbf{V}^e_{\mathbf{F}_2^n}\mathbf{M}\right)\overline{g} = 0$$

Fast matrix vector product for V

over $\mathbf{F}_2^4$, $\quad \mathbf{V}^4_{\mathbf{F}_2^4} = \begin{pmatrix} \bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet \\ &\bullet&&\bullet&&\bullet&&\bullet&&\bullet&&\bullet&&\bullet&&\bullet \\ &&\bullet&\bullet&&&\bullet&\bullet&&&\bullet&\bullet&&&\bullet&\bullet \\ &&&\bullet&&&&\bullet&&&&\bullet&&&&\bullet \\ &&&&\bullet&\bullet&\bullet&\bullet&&&&&\bullet&\bullet&\bullet&\bullet \\ &&&&&\bullet&&\bullet&&&&&&\bullet&&\bullet \\ &&&&&&\bullet&\bullet&&&&&&&\bullet&\bullet \\ &&&&&&&\bullet&&&&&&&&\bullet \\ &&&&&&&&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet \\ &&&&&&&&&\bullet&&\bullet&&\bullet&&\bullet \\ &&&&&&&&&&\bullet&\bullet&&&\bullet&\bullet \\ &&&&&&&&&&&\bullet&&&&\bullet \\ &&&&&&&&&&&&\bullet&\bullet&\bullet&\bullet \\ &&&&&&&&&&&&&\bullet&&\bullet \\ &&&&&&&&&&&&&&\bullet&\bullet \\ &&&&&&&&&&&&&&&\bullet \end{pmatrix}$

Moebius Transform : Computable in $O(2^n \log 2^n)$ over $\mathbf{F}_2^n$

The issue in term of linear algebra

For both attacks, we get a matrix $\mathbf{A}$

- Is the matrix $\mathbf{A}$ singular ?
  - to assert the immunity
- Find elements in the kernel of $\mathbf{A}$
  - to find relations and build an attack

Existing algorithms

Basic algorithm ▶ Gaussian elimination on $\mathbf{A}$

[Armknecht Carlet Gaborit Künzli Meier Ruatta] EuroC 06
[Didier Tillich] FSE 06
[Braken Lano Preneel] ACISP 06

▶ Use structure to improve elimination

[Didier] Indocrypt 06

▶ Idea based on fast matrix vector product with $\mathbf{A}$

Part 3
Wiedemann's Algorithm

General Facts for an $N \times N$ matrix

- One of the algorithms designed to solve large sparse linear system
- Huge litterature because of important applications (used in factorisation/discrete logarithm algorithms)
- Complexity in $O(N)$ matrix vector products
- Faster than Gaussian elimination as long as matrix vector product is faster than $O(N^2)$

Wiedemann's Algorithm for a square matrix

Let A an $N \times N$ matrix and b a vector in $\mathbf{F}_2^N$

The following Krylov sequence is lineary generated

$$b, Ab, A^2b, \ldots, A^Nb, \ldots$$

Let $P_b \in \mathbf{F}_2[X]$ be its minimal polynomial, that is

the minimal degree polynomial such that $P_b(A)b = 0$

P_b divide the minimal polynomial of A, so $\deg(P_b) \leq N$

From P_b to our problem solution

Assume P_b is known and that

$$P_b(X) = c_0 + XQ(X) \quad Q(X) \in \mathbf{F}_2[X]$$

If $c_0 \neq 0$ (and therefore $c_0 = 1$) then $AQ(A)b = b$

- $Q(A)b$ is a solution x to the system $Ax = b$

If $c_0 = 0$ then $AQ(A)b = 0$ (in particular A is singular)

- $Q(A)b$ is a non-trivial kernel element of A

Computing P_b with Berlekamp-Massey

Choose a random vector u^t in $\mathbf{F}_2^N$ then compute

$$u.b, u.Ab, u.A^2b, \ldots, u.A^{2N}b$$

and its minimal polynomial $P_{u,b}$ with Berlekamp-Massey

P_b ? $\quad P_{u,b}/P_b$ and equality with probability $\geq 1/(6 \log N)$

Complexity

- $2N$ matrix vector product of A
- $O(N^2)$ for the Berlekamp-Massey part

Version we used to assert the immunity

Over $\mathbf{F}_2$ and for a random choice of b and u,
if A is singular then $X/P_{u,b}$ with probability $\geq 1/4$

A singular ?

- Try i different values of u and b
- If $X/P_{u,b}$ then yes
- If $\deg(P_{u,b}) = N$ and $X \nmid P_{u,b}$ then no
- Otherwise, no with pb $\geq 1 - (3/4)^i$

Non square case over $\mathbf{F}_2$

If A is a $N \times k$ matrix, an algorithm exist to construct a “random” sparse $k \times N$ matrix Q such that

- If A is of full rank, then QA is a $k \times k$ non-singular matrix with a probablity bounded away from 0
- The number of 1 in Q is in $O(N \log N)$

Now :

- We can just run Wiedemann’s algorithm on QA
- However, more pass are needed

Part 4

Results

Wiedemann's complexity summary

Degree d immunity of a n-variable Boolean function

Square case :

- $O(D)$ matrix vector products in $O(n2^n)$
- $O(2^n)$ memory
- Moebius transform is vectorizable (`SSE2`)

Non square case :

- $O(D)$ evaluations in $O(n2^n)$
- $O(n2^n)$ memory for storing the matrix Q
- product by Q is not vectorizable

Complexity summary for normal AA

Algorithm	complexity	memory
Gaussian elimination	$O(\|f\|D^2)$	$O(D^2)$
Eurocrypt $2006^{(1)}$	$O(D^2)$	$O(D^2)$
FSE $2006^{(2)}$	$O(D)$, fixed d, $n \to \infty$	$O(D)$
Wiedemann's	$O(n2^nD)$	$O(n2^n)$

(1) : Average complexity, worst should be around $O(2^nD)$

(2) : Average complexity and memory to assert the immunity only, in the general case no better results than the Gaussian elimination, but a lot faster in practice

Complexity summary for Fast AA

Algorithm	complexity	memory
Gaussian elimination	$O(2^n(E+D)^2)$	$O((E+D)^2)$
Eurocrypt $2006^{(1)}$	$O(ED^2)$	$O(D^2)$
ACISP $2006^{(1)}$	$O(ED^2+E^2)$	$O(ED)$
FSE $2006^{(2)}$	$-$	$-$
Wiedemann's$^{(3)}$	$O(n2^nD)$	$O(n2^n)$

(1) : Average complexity, worst should be around $O(2^nDE)$

(2) : Adaptable to this case, should give the same kind of result as for normal AA, but no theoretical proof

(3) : Best algorithm for most values of the degree constraint

Square case record (P4/3.2Ghz/2Gb)

Degree d immunity for a n-variable balanced Boolean function $(n = 2d + 1)$

Algorithm	d, n	time
Gaussian elimination	$d = 8 \quad n = 17$	a few hours
Eurocrypt 2006	$d = 9 \quad n = 19$	-
FSE 2006	$d = 9 \quad n = 19$	6h
Wiedemann's algorithm (one pass)	$d = 9 \quad n = 19$	102s
	$d = 11 \quad n = 23$	11h
	$d = 12 \quad n = 25$	20d

Wiedemann's algo for non-square case

We loose more than a factor 32 (no vectorization)

- for AA, same limit as with previous algo $(n = 19)$
- Same limit for FAA ! almost no dependance on e

Improvement : by using block Wiedemann's algorithm we can expect the same kind of performance

Advantages of this approach

- Uses well-known algorithms and has good complexity
- Memory efficient $\longrightarrow$ can deal with many variables
- Almost the same algorithm for AA and FAA
- Leads to an efficient decoding over the erasure channel for all codes that can be generated efficiently