Algebraic methods for cryptanalysis

Jean-Philippe Aumasson

1. Cube attacks and cube testers

[Dinur-Shamir-09]
[Aumasson-Dinur-Meier-Shamir-09]

Block cipher

$$E : \{0,1\}^k \times \{0,1\}^n \mapsto \{0,1\}^n$$

- k: secret key size
- n: block size
- e.g., $k = n = 128$
- family of permutations $\{E_K\}_{K \in \{0,1\}^k}$
- encryption: $M \mapsto C = E_K(M)$
- decryption: $C \mapsto M = E_K^{-1}(C)$
- ex: DES, AES, IDEA,

Stream cipher

$$E : \{0,1\}^k \times \{0,1\}^v \mapsto \{0,1\}^\ell$$

- k: secret key size
- v: initial value (IV) size
- ℓ: keystream size
- e.g., $k = 128$, $n = 96$, $\ell < 2^{64}$
- pseudo-random generator with seed (V, K)
- encryption/decryption: $X \mapsto X \oplus E_K(V)$
- ex: RC4, A5/1, Grain-128

Standard **adversarial model** for stream ciphers

- key K fixed and unknown
- adversary makes chosen-IV queries $E_K(V)$
- adversary tries to recover (information on) K
- adversary tries to **distinguish** E_K from a random generator

Stream ciphers often described as algorithms

Ex: RC4 [Rivest-94]

1. **for** $i = 0, \ldots, 255$
2. $\quad T[i] \leftarrow i$
3. $j \leftarrow 0$
4. **for** $i = 0, \ldots, 255$
5. $\quad j \leftarrow (j + T[i] + K[i]) \mod 256$
6. $\quad T[i] \leftrightarrow T[j]$

Any stream cipher $E : (K, V) \mapsto S \in \{0,1\}^\ell$ is associated with ℓ **polynomial equations** on GF(2), e.g.

$$\begin{aligned} S_0 &= K_0K_{10}K_{37}V_2V_7 + K_2K_3V_0V_9 + K_2 + K_5 + V_8 \\ S_1 &= K_3K_4V_0V_1V_2 + K_4V_3V_0V_9 + V_7 + V_8 \\ \cdots &= \cdots \\ S_{\ell-1} &= K_0K_1K_2K_3 + V_0V_1V_2V_3V_4 + 1 \end{aligned}$$

Any stream cipher $E : (K, V) \mapsto S \in \{0,1\}^\ell$ is associated with ℓ **polynomial equations** on GF(2), e.g.

$$\begin{aligned} S_0 &= K_0K_{10}K_{37}V_2V_7 + K_2K_3V_0V_9 + K_2 + K_5 + V_8 \\ S_1 &= K_3K_4V_0V_1V_2 + K_4V_3V_0V_9 + V_7 + V_8 \\ \cdots &= \cdots \\ S_{\ell-1} &= K_0K_1K_2K_3 + V_0V_1V_2V_3V_4 + 1 \end{aligned}$$

For security, equations should be

- dense
- of high degree

Ideally, each coefficient null with prob. $1/2$

Classical **algebraic attacks** on $E : (K, V) \mapsto S$

- find low-degree equations $f_i(K, V, S) = 0$
- solve system, to recover K when V and S known (NP-hard)

Classical **algebraic attacks** on $E : (K, V) \mapsto S$

- find low-degree equations $f_i(K, V, S) = 0$
- solve system, to recover K when V and S known (NP-hard)

State-of the art methods:

- find Gröbner bases of a polynomial ideal
- algorithms F_4, F_5, XL, XSL

Ex: 40 random quadratic equations in 20 variables over GF(2^8) solvable in 2^{45} cycles [Yang et al.-07]

How to exploit the algebraic structure without solving a nonlinear system?

Cube attacks

How to exploit the algebraic structure without solving a nonlinear system?

Cube attacks

General idea:

- compute high-order derivative to obtain linear equations
- solve a linear system in $O(n^3)$

Differentiation n times of a degree-n polynomial yields the coefficient of the highest-degree monomial

$$\begin{aligned} f(X_1, X_2, X_3, X_4) &= X_1 + X_1X_2X_3 + X_1X_2X_4 \\ &= X_1 + X_1X_2X_3 + X_1X_2X_4 + 0 \times X_1X_2X_3X_4 \end{aligned}$$

Sum over all values of (X_1, X_2, X_3, X_4):

$$f(0,0,0,0) + f(0,0,0,1) + f(0,0,1,0) + \cdots + f(1,1,1,1) = 0$$

Differentiation $m < n$ times of degree-n polynomial yields a polynomial of degree $\leq (n - m)$

$$
\begin{aligned}
f(X_1, X_2, X_3, X_4) &= X_1 + X_1X_2X_3 + X_1X_2X_4 \\
&= X_1 + X_1X_2(X_3 + X_4)
\end{aligned}
$$

Fix X_3 and X_4, sum over all values of (X_1, X_2):

$$
\begin{aligned}
\sum_{(X_1,X_2)\in\{0,1\}^2} f(X_1, X_2, X_3, X_4) &= 4 \times X_1 + (X_3 + X_4) \\
&= X_3 + X_4
\end{aligned}
$$

X_1 and X_2 public and variable (initial value)

X_3 and X_4 fixed and unknown (secret key)

Black-box queries to $f(\cdot, \cdot, X_3, X_4)$ with chosen (X_1, X_2)

X_1 and X_2 public and variable (initial value)

X_3 and X_4 fixed and unknown (secret key)

Black-box queries to $f(\cdot, \cdot, X_3, X_4)$ with chosen (X_1, X_2)

Evaluate of $(X_3 + X_4)$ via order-2 derivative:

$$\sum_{(X_1, X_2) \in \{0,1\}^2} f(X_1, X_2, X_3, X_4) = X_3 + X_4$$

Just need to know that the factor of $X_1 X_2$ is $(X_3 + X_4)$

On a stream cipher $f : (K, V) \mapsto S$:

Phase 1: find monomials with linear derivative

$$\begin{aligned} f(K,V) &= \cdots + V_1V_3V_5V_7(K_2+K_3+K_5) + \cdots \\ f(K,V) &= \cdots + V_1V_2V_6V_8V_{12}(K_1+K_2) + \cdots \\ \cdots &= \cdots \\ f(K,V) &= \cdots + V_3V_4V_5V_6(K_3+K_4+K_5) + \cdots \end{aligned}$$

(reconstruct polynomials with linearity tests)

Phase 2: evaluate the polynomials in K, solve the system

Complexity: exponential in the order of derivatives, polynomial in the key size

Variant: cube testers

- make high-order differentiation
- compute statistics on values obtained

Use as distinguisher, not for key-recovery

Summary (cube attacks)

- recover keys of ciphers of low degree over GF(2)
- high-order derivative to obtain a linear system of equations

Open problems

- how to find good variables to differentiate?
- how to adapt to extensions of GF(2)?

2. Application to the cipher Grain-128

[Aumasson-Dinur-Henzen-Meier-Shamir-09]

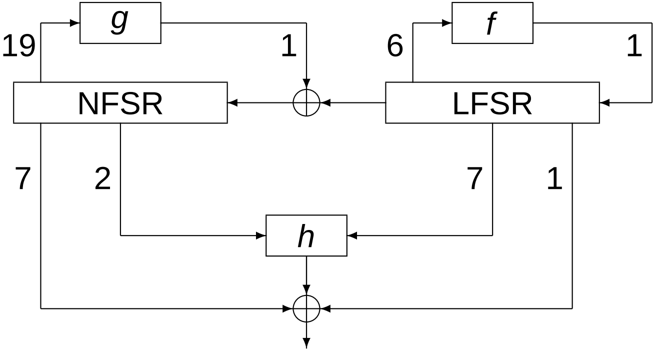
Grain-128

- state-of-the-art design (2006)
- by Hell, Johansson (Uni Lund), Meier (FHNW)
- developed within UE NoE project (eSTREAM)
- known attacks on reduced versions only
- implemented in the Bouncycastle library

Grain-128

128-bit key, 96-bit IV

degree-$(2+3)$ update function (deg NFSR$=2$, deg $h=3$)



Evolutionary algorithm for finding variables that give imbalanced polynomial after derivation

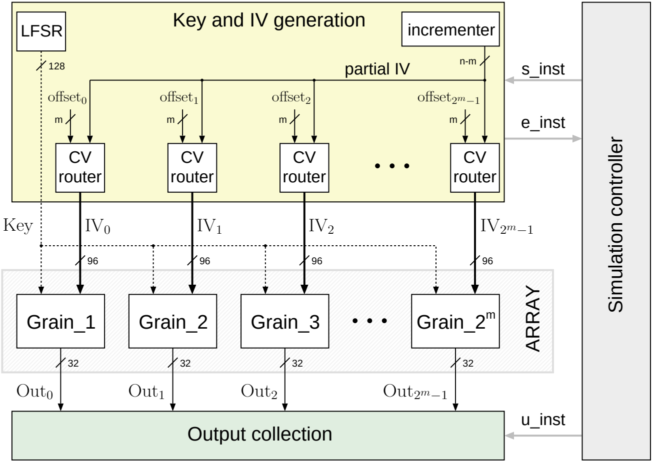
In a nutshell: population $=$ points in the search space

1. initialize population pseudorandomly
2. reproduction (crossover + mutation)
3. selection of best fitting individuals
4. go to 2.

#generations (steps 2-4) before halting $=$ parameter

Efficient implementation of derivation over several instances:

- on hardware field-programmable gate array (FPGA)
- parallelization 256×32



High-complexity attack

- 2^{40} for order-40 derivation
- 64 times
- 256 clockings per trial

2^{54} basic operations in total

High-complexity attack

- 2^{40} for order-40 derivation
- 64 times
- 256 clockings per trial

2^{54} basic operations in total

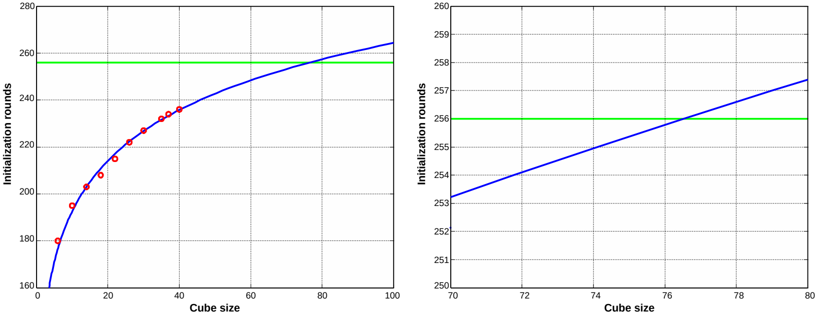
Results

Imbalance observed on reduced version with up to 237 initialization clockings (out of 256)

$\Rightarrow$ derivative is an imbalanced Boolean function

Extrapolation (Matlab)

By standard general linear regression



$\Rightarrow$ order-77 differentiation gives imbalanced function

Summary (attack on Grain-128)

- combines discrete optimization (EA) and cube testers
- first “cracking machine” for a stream cipher
- Grain-128 arguably broken (no 128-bit security)

Open problems

- which other ciphers are vulnerable?
- optimization: insights on the search space topology?

Algebraic methods for cryptanalysis

Jean-Philippe Aumasson