Cube Testers: Theory and Practice

Jean-Philippe Aumasson

University of Applied Sciences Northwestern Switzerland

(joint work with Itai Dinur, Willi Meier, and Adi Shamir)

Agenda

Cube attacks

Cube testers

Results

- on MD6
- on Trivium
- on Shabal

Conclusions

Cube attacks

2008 timeline

19 Aug: Shamir's CRYPTO 2008 talk

How to solve it: New Techniques in Algebraic Cryptanalysis

13 Sep: paper of Dinur and Shamir on ePrint

Cube Attacks on Tweakable Black Box Polynomials

27 Oct: attacks reported on reduced-round MD6

Targets of cube attacks

Primitives with **secret and public variables**

- keyed hash functions
- stream ciphers
- block ciphers
- MACs

which are based on **low-degree** components

- stream ciphers based on low-degree NFSR
- hash functions with only XORs and a few ANDs

Key observation 1

Any function

$$f : \{0,1\}^m \mapsto \{0,1\}^n$$

admits an algebraic normal form (ANF)

Example: $f : \{0,1\}^{10} \mapsto \{0,1\}^4$

$$\begin{aligned}
f_1(x) &= x_1x_2x_3 + x_1x_4x_5x_6x_7 + x_8x_9 \\
f_2(x) &= x_2x_4 + x_5x_6x_7x_8x_9x_{10} + x_6x_8x_9 \\
f_3(x) &= 1 \\
f_4(x) &= 1 + x_1 + x_3 + x_5
\end{aligned}$$

Key observation 2

Some monomial coefficients can be computed easily

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_3$$

Sum over all values of (x_1, x_2, x_3, x_4):

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

= coefficient of the monomial $x_1 x_2 x_3 x_4$!

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 + 0 \times x_1 x_2 x_3 x_4 + x_3$$

Key observation 3

Generalization: evaluation of factor polynomials

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_3$$

can be written

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 (x_3 + x_4) + x_3$$

Formal sum over all the values of (x_1, x_2):

$$\sum_{(x_1, x_2) \in \{0,1\}^2} f(x_1, x_2, x_3, x_4) = x_3 + x_4$$

Summary of cube attacks

Requirements

- a **low-degree ANF**
- only black-box access to the function
- negligible memory

Work in 2 phases

- **precomputation**: chosen keys and chosen IVs
- **online**: fixed unknown key and chosen IVs

Terminology

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1 x_2 (x_3 + x_4) + x_3$$

$(x_3 + x_4)$ is called the **superpoly** of the **cube** $x_1 x_2$

maxterm $=$ cube whose superpoly is **of degree 1 (linear)**

Evaluation of a superpoly

x_3 and x_4 fixed and unknown

$f(\cdot, \cdot, x_3, x_4)$ queried as a **black box**

ANF unknown, except: $x_1 x_2$'s superpoly is $(x_3 + x_4)$

$$f(x_1, x_2, x_3, x_4) = \cdots + x_1 x_2 (x_3 + x_4) + \cdots$$

With black-box queries, compute

$$\sum_{(x_1, x_2) \in \{0,1\}^2} f(x_1, x_2, x_3, x_4) = x_3 + x_4$$

Key-recovery attack

On a stream cipher with key k and IV v

$$f : (k, v) \mapsto \text{first keystream bit}$$

Precomputation: find maxterms and their superpolys

$$\begin{aligned}
f(k,v) &= \cdots + v_1 v_3 v_5 v_7 (k_2 + k_3 + k_5) + \cdots \\
f(k,v) &= \cdots + v_1 v_2 v_6 v_8 v_{12} (k_1 + k_2) + \cdots \\
\cdots &= \cdots \\
f(k,v) &= \cdots + v_3 v_4 v_5 v_6 (k_3 + k_4 + k_5) + \cdots
\end{aligned}$$

(reconstruct the superpolys, using linearity tests)

Online: evaluate the superpolys, solve the system

Applications

Stream cipher Trivium (reduced to 771 rounds):

- recover 80-bit key in $\approx 2^{36}$

Compression function of MD6...

Cube testers

In a nutshell

Like cube attacks:

- need only black-box access
- target primitives with secret and public variables and
- built on low-degree components

Unlike cube attacks:

- give distinguishers rather than key-recovery
- don't require low-degree functions
- need no precomputation

Basic idea: detect a structure...

Dichotomy structure/pseudorandomness

- many concepts of structure (e.g., linearity)
- given a structure, a pseudorandom object has “low correlation” with structured objects
- generalizing, a pseudorandom object has “low correlation” with **all** structured objects

$\Rightarrow$ if an object is not pseudorandom, then it has large structured component

see Tao, *FOCS’07*, arXiv:0707.4269

... in the superpolys

Primitive potentially vulnerable to cube testers when it has

- pseudorandom (very) low-degree terms
- structured (reasonably) high-degree terms

High degree terms are observed through **superpolys**

Need the structure to be **efficiently testable**

Algebraic property testing

Test if a function with finite domain and range satisfies a given property

Property $\equiv$ subset of functions

A tester on a function *f* for property $\mathcal{F}$:

- makes (adaptive) queries to *f*
- accepts if *f* satisfies the property (i.e. $f \in \mathcal{F}$)
- rejects with bounded probability otherwise

Efficiently testable properties

Examples:

- balance
- linearity
- low-degree
- constantness
- presence of linear variables
- presence of neutral variables

general characterization by Kaufman/Sudan, *STOC' 08*

Cube testers

Test properties of the superpolys

Example: testing a superpoly $\{0, 1\}^{10} \mapsto \{0, 1\}$

- if random, degree > 4 with prob. ≈ 1
- test if the superpoly has degree ≤ 4
- if yes, return **nonrandom**

Use Alon et al.'s test (*RAND -APPROX'03*):

$d \cdot 2^{2d}$ queries to test degree d

Cube testers

If a cube has n variables, each query (to its superpoly) costs 2^n queries (to the primitive attacked)

- need **efficient** property testers

Classical cube attacks only work with **linear** superpolys

Superpolys attackable by testing...

. . . **linearity**

$$\cdots + x_1 x_2 (x_3 + x_4) + \cdots$$

. . . **balance**

$$\cdots + x_1 x_2 x_3 (1 + x_6 x_7 x_8 x_9 x_{10}) + \cdots$$

Superpolys attackable by testing...

... **low-degree** (6)

$$\cdots + x_1 x_2 x_3 (x_2 x_3 + x_4 x_{21} + x_6 x_9 x_{20} x_{30} x_{40} x_{50}) + \cdots$$

... **neutral variables** (x_6)

$$\cdots + x_1 x_2 x_3 x_4 x_5 \cdot g(x_7, x_8, \ldots, x_{80}) + \cdots$$

... **linear variables** (x_6)

$$\cdots + x_1 x_2 x_3 x_4 x_5 \cdot (x_6 + g(x_7, x_8, \ldots, x_{80})) + \cdots$$

In practice

Compute e.g.

$$\sum_{(x_1,x_2)\in\{0,1\}^2} f(x_1, x_2, x_3, x_4) = x_3 + x_4$$

for chosen x_3 and x_4

x_i's are IV bits: **distinguisher**

x_i's are key or IV bits: **nonrandomness**

Results on MD6

MD6

Presented by Rivest at CRYPTO 2008

Submitted to the SHA-3 competition

- quadtree structure
- construction RO-indifferentiable
- low-degree compression function
- at least 80 rounds
- best authors' attack: 12 rounds

MD6's compression function

$$\{0,1\}^{64\times 89} \mapsto \{0,1\}^{64\times 16}$$

Input: 64-bit words $A_0.A_1,\ldots,A_{88}$

Compute the A_i's with the recursion

$$x \leftarrow S_i \oplus A_{i-17} \oplus A_{i-89} \oplus (A_{i-18} \wedge A_{i-21}) \oplus (A_{i-31} \wedge A_{i-67})$$
$$x \leftarrow x \oplus (x \gg r_i)$$
$$A_i \leftarrow x \oplus (x \ll \ell_i)$$

- round-dependent constant S_i
- quadratic step, at least 1280 steps

Properties exploitable by cube testers

- low-degree
- large state
- “late” inputs
- absorption of AND

Cube attacks on MD6

Key-recovery

- on the **14-round** compression function
- recover any 128-bit key
- in time $\approx 2^{22}$

Recall: at least 80 rounds recommended

Cube testers on MD6

Strategy

- identify “weak” input words
- force linearity of certain variables
- perturb & correct strategy
- test **balance** of Boolean components

Nonrandomness detected after **18 rounds**

Without the constants S_i: **66 rounds**

Results on Trivium

Trivium

Stream cipher by De Cannière and Preneel, 2005

eSTREAM HW portfolio

- 80-bit key and IV
- 3 quadratic NFSRs
- 1152 initialization rounds
- best attack on 771 rounds (cube attack)

Cube testers on Trivium

Test the presence of **neutral variables**

Distinguishers (only choose IVs)

- 2^{24}: 772 rounds
- 2^{30}: 790 rounds

Nonrandomness (also choose part of the key)

- 2^{24}: 842 rounds
- 2^{27}: 885 rounds

Full version: 1152 rounds

Results on Shabal

Shabal

Submitted to the SHA-3 competition

- compression function = keyed permutation $\mathcal{P}$
- $\mathcal{P}$ conjectured pseudorandom

Cube tester:

- test neutrality of key variables
- makes 2^{12} queries
- show that $\mathcal{P}$ is not pseudorandom

This doesn't affect Shabal's security...

but can make proofs on the structure inapplicable

Conclusions

Advantages and limitations

+

- more general than classical cube attacks
- no precomputation
- “polymorphic”

–

- only gives distinguishers
- only finds feasible attacks
- relevant for a minority of functions

Open issues

How to predict the existence of unexpected properties?

How to bound the degree of a quadratic recursion?

Optimal tradeoff 'cube size' / 'test complexity' ?

Which primitives are vulnerable to cube testers?

Cube Testers: Theory and Practice

Jean-Philippe Aumasson

University of Applied Sciences Northwestern Switzerland

(joint work with Itai Dinur, Willi Meier, and Adi Shamir)