TU Graz

SCIENCE
PASSION
TECHNOLOGY

Cube Attacks

Lukas Lamster

January 2021

GF(2)

- Smallest possible field
- Characteristic 2
- Equivalent to $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}_2$ or $\mathbb{F}_2$
- Additive identity 0
- Multiplicative identity 1

GF(2) Operations

- XOR ($+$) and AND ($\cdot$)

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Field Properties?

- $\{0, 1\}$ is an abelian group w.r.t. $+$ with identity 0 $\checkmark$
- $\{1\}$ is an abelian Group w.r.t $\cdot$ with identity 1 $\checkmark$
- Distributive law holds $\checkmark$

Additional Operations

- OR ($\vee$) and NOT ($\neg$)

$\vee$	0	1
0	0	1
1	1	1

$\neg$	0	1
-	1	0

Extension to GF(2^n)

- GF(2^n) is an extension field
- Consists of polynomials
- Coefficients drawn from GF(2)
- Example: GF(2^2)

$$\mathrm{GF}(2^2) = \{0, 1, x, 1 + x\}$$
$$\text{modulus} = x^2 + x + 1$$

Boolean Functions

- Mapping $\mathbb{F}_2^n \rightarrow \{0, 1\}$
- n input bits mapped to one output bit
- Example:

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

Algebraic Normal Form

- Similar to DNF or CNF
- Sum of products (monomials/cubes)

$$y = \sum_{I \subseteq \{1,\dots,n\}} k_I \prod_{j \in I} x_j$$

- k_I is 1 or 0
- Example

$$y = x_1 x_2 + x_3 \implies k_I = 1 \text{ for } I = \{1, 2\} \text{ and } I = \{3\}$$

Algebraic Degree

- Similar to degree of 'normal' polynomial
- Example:

$$x_1 x_2 + x_1 x_3 + x_2 \qquad \Longrightarrow \text{degree} = 2 \text{ in } x$$

- Equal to the multivariate degree
- $\delta(y) = d = \max\{|I| \, | \, k_I \neq 0\}$

Cubes

- Index subset I defines cube
- A cube with k variables is a k-dimensional subspace of $\mathbb{F}_2^n$
- Only the k variables change
- Example

$$I = \{1, 2, 4\}$$
$$t_I = x_1 x_2 x_4$$

Overview

- Proposed by Dinur and Shamir (2008/09)
- Algebraic attack
- Strongly related to AIDA by Vielhaber (2007)
- System seen as polynomial
- Ciphertext bits functions of plaintext and key bits

Observations

- Let $I \subseteq \{1, \ldots, n\}$ index the term t_I
- We can write every polynomial as

$$p(x_1, \ldots, x_n) = t_I \cdot p_{S(I)} + q(x_1, \ldots, x_n)$$

- $p_{S(I)}$ is called the *superpoly* of I in p
- If $\delta(p_{S(I)}) = 1$, t_I is a *maxterm* of p

Observations

Given $p(x_1, \ldots, x_n) = t_I \cdot p_{S(I)} + q(x_1, \ldots, x_n)$

- $p_{S(I)}$ has no common variable with t_I
- Each term in $q(x_1, \ldots, x_n)$ misses at least one variable from I
- What happens if we sum over the cube t_I?
- Cube with size $k \rightarrow 2^k$ possible combinations

Summing over a cube

How can we sum over a given cube t_I?

- Only modify cube variables, keep other variables fixed (set to 0 or 1)
- Sum over all possible combinations of cube bits
- Example

Let t_I be defined by $I = \{1, 2\}$ and $p(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 + x_2$

$$\sum_{t_I} p(x_1, x_2, x_3) = (0 + 0 + 0) + (0 + 0 + 0) + (0 + 0 + 1) + (1 + 0 + 1)$$

Observations

$$\text{Given } p(x_1,\ldots,x_n) = t_I \cdot p_{S(I)} + q(x_1,\ldots,x_n)$$
$$\sum_{t_I} \left(t_I \cdot p_{S(I)} + q(x_1,\ldots,x_n)\right) \equiv p_{S(I)} \bmod 2$$

Proof?

- We know that we sum over 2^k combinations
- No term t_J in $q(x_1,\ldots,x_n)$ is influenced by all variables in t_I
- Every t_J is summed an even number of times
- $t_I \cdot p_{S(I)}$ is only non-zero iff $t_I = 1$

Conclusion

- Summing over a cube is equivalent to differentiating w.r.t. the cube
- The result is equal to the superpoly $p_{S(I)}$
- If t_I was a maxterm, the result will be a linear function

The Attack

- Two phases, offline and online
- Assume attacker has access to blackbox polynomial
- Polynomial and degree unknown
- Attacker can evaluate arbitrary input in offline phase

Offline Phase

- Create random cubes
- Sum over cubes
- Check if superpoly is linear
- Repeat until enough polynomials are found
- Calculate coefficients of key bits to get equations

How do we check linearity?
How can we calculate the coefficients?

BLR Linearity Test

Given a function f, we want to know if f is linear
Idea

- Sample $x, y \in \mathbb{F}_2^n$ from uniform random distributions
- Evaluate $f(x), f(y)$ and $f(x + y)$
- Check if $f(x) + f(y) = f(x + y)$
- If the equality does not hold $\rightarrow$ f is certainly non-linear
- Else f is *probably* linear

BLR Linearity Test for our use case

- We cannot just compute $f(x), f(y)$ and $f(x+y)$
- We need to sum over the whole cube for each input
- Use caching / save old calculations to speed up the computation

Calculating Coefficients

Given a linear superpoly $p_{S(I)}$, how can we reconstruct an equation here?

Assume the polynomial has the form $p_{S(I)} = c_0 + c_1 x_1 + \ldots + c_n x_n$

- For a linear equation, changing one variable flips the output
- Test if variable x_j influences the output
  - Sum over the cube with all variables set to zero $\rightarrow$ get c_0
  - Sum over the cube with x_j set to 1
  - Compare results

Calculating Coefficients

- If results differ $\rightarrow c_j = 1$
- Doing this for all x_j will reveal $p_{S(I)}$
- This can partially be done during linearity checking

Online Phase

- Now only the plaintext can be altered
- Use previously gathered cubes and equations
- Apply cubes on fixed-key system
- Solve linear equations for key bits

Applications

- Algebraic degree is limiting factor
- Number of possible cubes grows exponentially
- Apply to ciphers with easy algebraic structures

Usage in System Security

- Not only practical for crypto
- Practical for reverse engineering

Setting

- Modern computers utilize shared caches
- Locations in caches are not distributed randomly
- Undocumented hash function hard-wired in CPU
- Linear functions for 2^n-core CPUs
- Non-linear functions for other core counts

Setting

- Unknown hash mapping address to cache slice
- Start by guessing algebraic degree
- Orientate on existing functions

Data Collection

- Calculate all cubes up to degree
- Measure and sum slice mappings for cubes
- Determine if cube is used by checking sum
- After finding all cubes of current degree correct truth table

Advantages

- No need to measure all addresses
- Results are can be cached
- Nonlinear function is reconstructed

Disadvantages

- Again, exponential in degree
- Resulting function contains large amount of cubes
- Post-processing needed