TU Graz

SCIENCE
PASSION
TECHNOLOGY

Algebraic Attacks

Markus Schofnegger

partially based on slides by Maria Eichlseder

Applied Cryptography 2 – ST 2020

The road so far...

- Exploit statistical information
  - Differential cryptanalysis (differences propagate with a certain probability)
  - Linear cryptanalysis (linear equations hold with a certain probability)
  - $\rightarrow$ rely on probabilities
- Block cipher encryption is deterministic
  - Can we find solutions such as keys (mostly) deterministically?

Algebraic Attacks

- Core procedure:
  1. Represent the cipher as a set of equations
  2. Solve resulting system for unknown variables (e.g., key variables)
- Many attack strategies (one has to be “creative”)
- Different solving techniques
- Complexities sometimes hard to estimate
- Can be powerful in low-data scenarios (more on that later)
- Efficiency depends on the structure of a cipher

Algebraic Attacks cont.

Algebraic attacks covered in this lecture:

- Interpolation Attacks
- Higher-Order Differential Attacks
- Attacks using Gröbner bases

...but first a short introduction

Writing Down Equations

Equations from an S-Box – Boolean Truth Table

Example (3-bit S-Box)

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
$S(x)$	0x0	0x1	0x5	0x6	0x7	0x2	0x3	0x4

x	$S(x)$
0x0	0x0
0x1	0x1
0x2	0x5
0x3	0x6
0x4	0x7
0x5	0x2
0x6	0x3
0x7	0x4

$\longrightarrow$

x_3	x_2	x_1	y_3	y_2	y_1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0

Equations from an S-Box – Algebraic Normal Form (ANF)

Example (3-bit S-Box)

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
$S(x)$	0x0	0x1	0x5	0x6	0x7	0x2	0x3	0x4

- Write the output bits y_3, y_2, y_1 as three Boolean functions of the input bits x_3, x_2, x_1 (*coordinate functions*, $\mathbb{F}_2^n \to \mathbb{F}_2$)
- In more detail: $y_i = \sum \prod x_i$ (over $\mathbb{F}_2$, the "XOR of ANDs")
- E.g., for the S-box above:

$$y_1 = x_1 + x_2 + x_3 + x_2x_3$$
$$y_2 = x_1x_2 + x_3$$
$$y_3 = x_2 + x_3 + x_1x_3$$

Equations from an S-Box – Algebraic Normal Form (ANF) cont.

Example (3-bit S-Box)

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
$S(x)$	0x0	0x1	0x5	0x6	0x7	0x2	0x3	0x4

- This representation is called the algebraic normal form (ANF) of S
- Note that all variables are bits (i.e., variables in $\mathbb{F}_2$, hence ${x_i}^2 = x_i, {y_i}^2 = y_i$)
- Maximum possible number of monomials in each coordinate function increases with the S-box size
- Maximum degree of all monomials in all coordinate functions is called the algebraic degree
- For the S-box above, the algebraic degree is 2 (due to x_2x_3, x_1x_2, and x_1x_3)

Equations from an S-Box – Working Over a Larger Field

Example (3-bit S-Box)

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
$S(x)$	0x0	0x1	0x5	0x6	0x7	0x2	0x3	0x4

Is it possible to work over a field $\mathbb{F}_n$ with $n > 2$? $\rightarrow$ 3-bit S-box, try $\mathbb{F}_{2^3}$

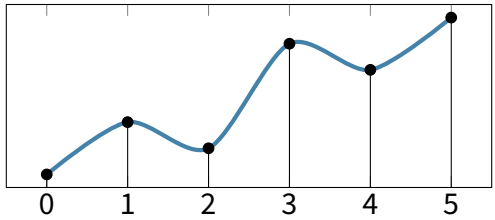
- Indeed:

$$S(x) = x^{2^n-2} = x^6 = \begin{cases} 0 & x = 0 \\ x^{-1} & \text{otherwise} \end{cases}$$

- Given 2^n $(x, S(x) = y)$ pairs, we can always find such a function by interpolation if its degree is $\leq 2^n - 1$
- Since S computes the inverse of x in $\mathbb{F}_{2^3}$ for all $x \neq 0$, we can also write $\forall x \neq 0 : x \cdot y = 1$ (now a degree-2 equation instead of a degree-6 one!)

Interpolation Attacks

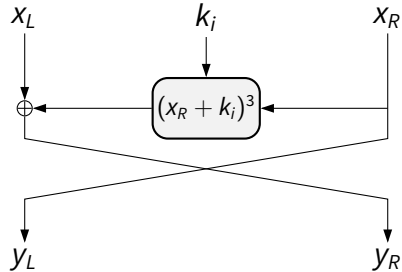
Interpolation Attacks



- Introduced in 1997 [JK97]
- Idea:
  1. Find polynomial f such that $f(p) = c$ for every plaintext p and ciphertext c without knowing the key
  2. Low-degree f allows interpolation using known (p, c) pairs
  3. Use the polynomial to recover the key

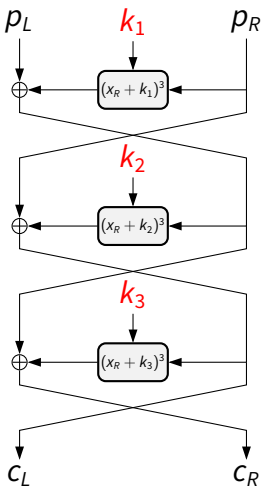
Interpolation Attack on the $\mathcal{PURE}$ Cipher

- Variant of the $\mathcal{KN}$ Feistel cipher proposed in 1995 [NK95] to be *provably resistant* against differential and linear attacks
- 64-bit blocks, 192-bit key $k = (k_i)_{i=1}^{6}$ with $k_i \in \mathbb{F}_{2^{32}}$
- Simplified round function (6 rounds in total):



- Computation of x^3 in $\mathbb{F}_{2^{32}}$

Interpolation Attack on the 3-Round $\mathcal{PURE}$ Cipher



- Tracing the encryption paths yields following equations:

$$c_R = {p_R}^{27} + \dots$$

$$\begin{aligned} c_L = {} & p_R{}^9 + p_R{}^8 k_1 + p_R k_1{}^8 + k_1{}^9 + p_L p_R{}^6 + p_L p_R{}^4 k_1{}^2 \\ & + p_L p_R{}^2 k_1{}^4 + p_L k_1{}^6 + p_R{}^6 k_2 + p_R{}^4 k_1{}^2 k_2 + p_R{}^2 k_1{}^4 k_2 \\ & + k_1{}^6 k_2 + p_L{}^2 p_R{}^3 + p_L{}^2 p_R{}^2 k_1 + p_L{}^2 p_R k_1{}^2 + p_L{}^2 k_1{}^3 \\ & + p_R{}^3 k_2{}^2 + p_R{}^2 k_1 k_2{}^2 + p_R k_1{}^2 k_2{}^2 + k_1{}^3 k_2{}^2 + p_L{}^3 \\ & + p_L{}^2 k_2 + p_L k_2{}^2 + k_2{}^3 + p_R \end{aligned}$$

- Note that c_L does not involve any k_3
- Still, given $p = (p_L \,||\, p_R)$ and $c = (c_L \,||\, c_R)$, it seems hard to solve for k_1, k_2, k_3

Interpolation Attack on the 3-Round $\mathcal{PURE}$ Cipher cont.

- Assume that p_L is constant (we know that the key is constant), then the equation for c_L simplifies to

$$c_L = {p_R}^9 + a_8{p_R}^8 + a_6{p_R}^6 + a_4{p_R}^4 + a_3{p_R}^3 + a_2{p_R}^2 + a_1 p_R + a_0,$$

  where each a_i is a function of the key and p_L
- With 7 (p, c) pairs we have 7 (linear) equations in 7 variables and can thus solve the system
- If p_L is not constant, more pairs are needed
- The full 6-round cipher needs 29 chosen plaintexts (the number can be kept low by choosing an advantageous initial structure and keeping the degrees and number of unknown coefficients low)

Interpolation Attack on the 3-Round $\mathcal{PURE}$ Cipher cont.

- Now we have a polynomial, which, given a plaintext, yields the corresponding ciphertext. But what about the key?
- Key-recovery attack:
  1. For every (p, c) pair and each of the possible last round keys, compute the 1-round decryption of c
  2. Construct the polynomial for the obtained $(r-1)$-round encryption
  3. Check with an additional (p, c) pair whether the key guess is correct
- For the $\mathcal{PURE}$ cipher: $27 + 1 + 1 = 29$ pairs and 2^{32} key guesses needed
- Protection (simplified): Increase number of coefficients $\neq 0$ (i.e., increase number of appearing monomials/make the polynomial dense)

Higher-Order Differentials and AIDA/Cube Attacks

Introduction

- Introduced in 1994 by X. Lai [Lai94] and L. Knudsen [Knu94]
- Use the algebraic normal form of Boolean functions
- Exploit low algebraic degree

Reminder

The algebraic degree is the maximum degree in all monomials of all coordinate functions describing the output bits.

- Build distinguishers and key-recovery attacks based on distinguishers

Deriving a Boolean Function – Mathematical Background

Derivative of $f : \mathbb{F}_2^n \to \mathbb{F}_2$ w.r.t. $a_i \in \mathbb{F}_2^n$

- 1^{st} order (same as in differential cryptanalysis):

$$\frac{\mathrm{d}}{\mathrm{d}a} f(x) = f(x) + f(x+a)$$

- k^{th} order (basis $a_1, a_2, \ldots, a_k$ of vector space $\mathcal{V}$):

$$\frac{\mathrm{d}}{\mathrm{d}a_1} \cdots \frac{\mathrm{d}}{\mathrm{d}a_k} f(x) = \frac{\mathrm{d}}{\mathrm{d}V} f(x) = \bigoplus_{v \in V} f(x+v) = \bigoplus_{w \in V+x} f(w)$$

- Familiar properties: sum rule, product rule, …, degree rule:

$$\deg(f) < d \implies \frac{\mathrm{d}}{\mathrm{d}a_1} \cdots \frac{\mathrm{d}}{\mathrm{d}a_d} f = 0$$

Zero-Sum Distinguishers

Definition (Zero Sum)

A zero sum of size s for $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a set $\{x_1, x_2, \ldots, x_s\} \subseteq \mathbb{F}_2^n$ such that

$$\sum_{i=1}^{s} x_i = \sum_{i=1}^{s} f(x_i) = 0.$$

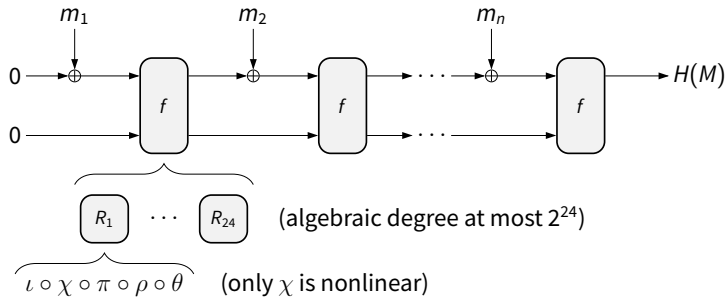
A zero-sum partition for f is a family of $2^n/s$ disjoint zero sums.

- For a random function f, zero-sum partitions should be hard to find
  - If not $\to$ nonrandom property, we can *distinguish* from random

Target: Keccak (SHA-3) [BDPA11]

Permutation-Based Hash Function

- State size: $5 \cdot 5 \cdot 64 = 1600$ bits
- Permutation f with 24 rounds, round function $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
- Sponge-based



KECCAK – Attack

- Components:
  - θ (sum columns, linear)
  - ρ (rotate words, linear)
  - π (permute words, linear)
  - χ (5-bit S-box, nonlinear)
  - ι (constant addition, linear)
- Focus on χ, the only nonlinear operation
- S-box has algebraic degree 2, hence the algebraic degree of the whole permutation is at most 2^{24}

KECCAK – Attack cont.

- For r rounds of the permutation: degree at most 2^r
  - For example: 10 rounds have degree at most $2^{10} = 1024$

Definition (Cube)

A cube $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_2^n$, such that $n - k$ bits are constant and k bits loop through all 2^k values.
Example: $\mathcal{C} = (\star, 0, \star, 1) = \{(0,0,0,1), (0,0,1,1), (1,0,0,1), (1,0,1,1)\}$.

- Input a cube $\mathcal{C}$ of size $2^{1024+1} = 2^{1025}$
- Result is that

$$\sum_{x \in C} x = \sum_{x \in C} f(x) = 0$$

KECCAK – Attack cont.

- Now we have a distinguisher for 10 rounds of KECCAK with a complexity less than 2^{1600}
  - Not necessarily useful by itself
  - Might invalidate some security proofs
  - Or even better: lead to an attack (e.g., key-recovery attack)
- Can be extended to a cube attack [Vie07; DS09]
  - $\frac{d}{da_1} \cdots \frac{d}{da_d} f(x) = \overline{L}(x)$ introduced by a cube
  - $\overline{L}(x)$ is a linear function in x (i.e., it has degree 1)
  - Use linear equations from $\overline{L}(x)$ and solve for the key bits

Cube Attack on Toy Cipher

- Assume a toy cipher with key bits k_i, plaintext bits p_i, and ciphertext bits c_i
- Assume following equation holds for c_1 (algebraic degree 3):

$$c_1 = f_{c_1}(k_1, k_2, p_1, p_2) = p_1p_2k_1 + p_1p_2 + p_1k_1k_2 + p_2 + k_2$$

- Consider cube $\mathcal{C} = (k_1, k_2, \star, \star)$ with variables p_1 and p_2:

$$\begin{aligned}\frac{d}{d\mathcal{C}}f_{c_1} = \frac{d}{dp_1}\frac{d}{dp_2}f_{c_1} &= 0 \cdot 0 \cdot k_1 + 0 \cdot 0 + 0 \cdot k_1k_2 + 0 + k_2 \\ &+ 0 \cdot 1 \cdot k_1 + 0 \cdot 1 + 0 \cdot k_1k_2 + 1 + k_2 \\ &+ 1 \cdot 0 \cdot k_1 + 1 \cdot 0 + 1 \cdot k_1k_2 + 0 + k_2 \\ &+ 1 \cdot 1 \cdot k_1 + 1 \cdot 1 + 1 \cdot k_1k_2 + 1 + k_2 \\ &= k_1 + 1\end{aligned}$$

- $k_1 + 1$ is called the superpoly for p_1p_2
- Recover k_1 with $2^2 = 4$ chosen plaintexts and repeat for each output bit

Cube Attack – Strategy

Problem: For real-world ciphers, the ANF or the degree of f is not known!

Phases of the Cube Attack

- Offline phase
  1. Chosen keys and chosen inputs
  2. Precomputation to identify suitable superpolys for f
  3. Test linearity of $\overline{L}(x)$ for different $(a_1, a_2, \ldots, a_d)$
- Online phase
  1. Fixed (unknown) key and chosen inputs
  2. Evaluate equations for (plaintext, ciphertext) pairs
  3. Solve the linear system

Cube Attack – Offline Phase

Goal: Find cube $(p_1, p_2, \ldots, p_d)$ such that the superpoly is linear in $k_1, k_2, \ldots, k_n$

- General method:
  1. Generate new random cube candidate $(p_1, p_2, \ldots, p_d)$
  2. Verify linearity
  3. Compute coefficients of superpoly $\overline{L}(\cdot)$ of f:

$$\overline{L}(k) = \hat{c}_0 + \hat{c}_1 k_1 + \cdots + \hat{c}_n k_n$$

  4. Repeat until enough cubes with linear superpolys are found
- Steps 2 and 3 can be parallelized

Cube Attack – Offline Phase cont.

How to choose next cube?

- Depending on last superpoly. If it was...
  - Constant ($\hat{c}_i = 0 \forall i$): Remove a variable p_d
  - Nonlinear (linearity test failed): Add a variable p_{d+1}
- Alternatively, or if the algebraic degree of f is known: Randomly

How to verify linearity?

- Again, if the algebraic degree is known: Easy
- BLR linearity test: Check for some random x, y if

$$\sum_{P_j \in \mathcal{C}} f(P_j, 0) + \sum_{P_j \in \mathcal{C}} f(P_j, x) + \sum_{P_j \in \mathcal{C}} f(P_j, y) = \sum_{P_j \in \mathcal{C}} f(P_j, x + y)$$

- Verify also that $\overline{L}(\cdot)$ is not constant (constant $\rightarrow$ no information about key)

Cube Attack – Offline Phase cont.

How to compute coefficients of $\overline{L}(k)$ of f?

- Let C be the cube with d cube variables $p_1, p_2, \ldots, p_d$
- Assume superpoly $\overline{L}(\cdot)$ is linear
  - Hence, $\overline{L}(k) = \hat{c}_0 + \hat{c}_1 k_1 + \cdots + \hat{c}_n k_n$
- 2^d chosen inputs P_j (all zero except $p_1 \mid\mid p_2 \mid\mid \cdots \mid\mid p_d = j$)
- $n + 1$ chosen keys: K_0 (all zero) and K_i (all zero except $k_i = 1$)
- Compute coefficients $\hat{c}_i$ by summing over inputs in cube C:

$$\hat{c}_0 = \sum_{P_j \in C} f(P_j, K_0)$$

$$\hat{c}_i = \sum_{P_j \in C} f(P_j, K_i) + \hat{c}_0,\ 1 \leq i \leq n$$

Cube Attack – Online Phase

Goal: Sum over cubes to get linear equations in $k_1, k_2, \ldots, k_n$

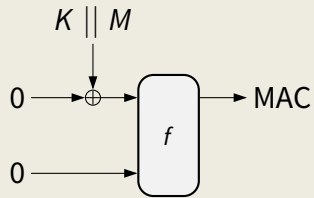
- General method:
  1. For each found cube $\mathcal{C}$ with superpoly coefficients $\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_n$:

$$\underbrace{\hat{c}_0 + \hat{c}_1 k_1 + \cdots + \hat{c}_n k_n}_{\text{write down LHS of linear equation}} \quad = \quad \underbrace{\sum_{P_j \in \mathcal{C}} f(P_j, K)}_{\substack{\text{evaluate RHS with queries} \\ \text{(i.e., the } c_i\text{'s)}}}$$

  2. Solve linear equation system
  3. Verify and/or find missing key bits by brute force
- Complexity: at most $n \cdot 2^d$ data and time (inputs and equations), plus complexity for brute-force step (if necessary)

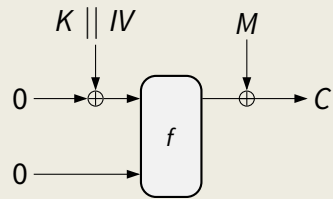
Target: Keyed KECCAK (capacity $c = 576$) [DMP+15]

KECCAK as MAC



Usable public parts for one block:
1. Input: M (896 bits)
2. Output: MAC (128 bits)

KECCAK as Stream Cipher



Usable public parts for one block:
1. Input: IV (128 bits)
2. Output: $M \oplus C$ (1024 bits)

- Attack on 5-round MAC: complexity $\approx 2^{35}$, 31 cube variables
- Attack on 6-round stream cipher: complexity $\approx 2^{36}$, 31 cube variables

Gröbner Basis Attacks

Background

- Let $F = (f_1, f_2, \ldots, f_n)$ be a system of n linear (and linearly independent) equations in n variables – how do we solve for the variables?
  - Easy: apply Gaussian elimination, solve the last (univariate) equation, substitute, ...
- But what if the equations are nonlinear?
  - Linearize them! For example: use the XL algorithm [CKPS00] (proposed for quadratic equations, but also works in the general case)
  - Idea: introduce new variables instead of increasing the degrees (e.g., ${x_1}^2 \rightarrow x_2$)
  - Problem: many variables...
- For AES and both when working in $\mathbb{F}_2$ and $\mathbb{F}_{2^8}$: thousands of quadratic equations and variables, and solving these systems is NP-hard

Gröbner Bases – Motivation

- Notion of Gröbner bases introduced in 1965 [Buc65]
- A different approach: Keep the number of variables low, but let the degree of the equations increase instead
- Combine three steps:
  1. Compute a Gröbner basis using our system of equations
  2. Change term order (more on that in a minute)
  3. Use factorization techniques to solve for a variable (also used as intermediate step in XL)

What is a Gröbner Basis? – Mathematical Background

Definition (Ideal)

Let $R = \mathbb{K}[x_1, x_2, \ldots, x_m]$ be a polynomial ring over some field $\mathbb{K}$ and $F = (f_1, f_2, \ldots, f_n)$ be a system of n equations in m variables. Then the ideal of F, denoted as $\mathrm{Id}(F)$, is

$$\mathrm{Id}(F) = \left\{ \sum_{i=1}^{n} r_i \cdot f_i \;\middle|\; n \in \mathbb{N}, r_i \in R, f_i \in F \right\}.$$

If the number of solutions for the ideal is finite, we call the ideal *zero-dimensional*.

- In words: the ideal of F is the set of all linear combinations of F over R (we also say that $\mathrm{Id}(F)$ is *generated* by F)
- Why do we need ideals?

What is a Gröbner Basis? – Mathematical Background cont.

- Given a set of equations $F = \{f_1, f_2, \ldots, f_n\}$, we convert it to a set of polynomials $P = \{p_1, p_2, \ldots, p_n\}$ (e.g., $x_1 + x_2 = x_3 \rightarrow x_1 + x_2 - x_3$)
- The set of solutions for F is precisely the set of solutions for P such that $p_1 = 0, p_2 = 0, \ldots, p_n = 0$
  - This set of solutions is called an algebraic variety
- Crucial point: the varieties of P and $\mathrm{Id}(P)$ are the same, which means they have the same solutions
  - ...but ideals are too large to use them efficiently

What is a Gröbner Basis? – Mathematical Background cont.

Definition (Gröbner Basis)

A Gröbner basis of an ideal is a polynomial equation system with the same variety and which is easier to solve.

- Computing a Gröbner basis for an ideal can be computationally expensive
- Algorithms involve polynomial divisions
  - Use the leading terms of the polynomials
  - The *term order* describes how the terms in a polynomial are ordered and what the leading term is
  - Huge impact on the efficiency of the computation

What is a Gröbner Basis? – Mathematical Background cont.

Lemma (Triangular Shape)

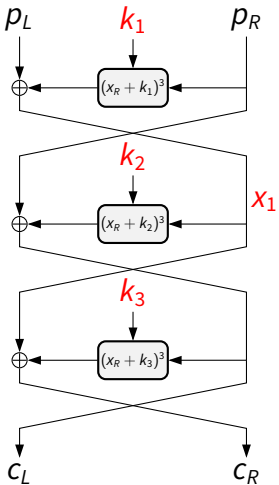
The reduced Gröbner basis $G = \{g_1, g_2, \ldots, g_n\}$ (in a specific term order) generating the zero-dimensional ideal I is of the form

$$\begin{aligned} g_1 &= {x_1}^d + h_1(x_1), \\ g_2 &= x_2 + h_2(x_1), \\ &\vdots \\ g_n &= x_n + h_n(x_1), \end{aligned}$$

where h_i is a polynomial in x_1 of degree at most $d - 1$.

- Note that g_1 is now a univariate equation and we can solve it by factorization!
- Use the result to solve for the other variables

Gröbner Basis Attack on the 3-Round $\mathcal{PURE}$ Cipher



- Our key variables are k_1, k_2, k_3
- We introduce an additional intermediate variable x_1 for our equations
- The system of equations describing the cipher is then

$$\begin{aligned} x_1 + (p_R + k_1)^3 + p_L &= 0, \\ c_L + (x_1 + k_2)^3 + p_R &= 0, \\ c_R + (c_L + k_3)^3 + x_1 &= 0 \end{aligned}$$

(p_L, p_R, c_L, c_R are known)

- But there is a problem...

Gröbner Basis Attack on the 3-Round $\mathcal{PURE}$ Cipher cont.

- We have 3 equations in 4 variables (our system is *underdetermined*)
- Simple solution: Use a second (plaintext, ciphertext) pair
  - Introduce a new variable x_2 for the second pair (k_1, k_2, k_3 stay the same)
- Add equations:

$$
\begin{aligned}
x_2 + (p_R{}^{(2)} + k_1)^3 + p_L{}^{(2)} &= 0, \\
c_L{}^{(2)} + (x_2 + k_2)^3 + p_R{}^{(2)} &= 0, \\
c_R{}^{(2)} + (c_L{}^{(2)} + k_3)^3 + x_2 &= 0
\end{aligned}
$$

- Now we have 6 equations in 5 variables and we can solve it!
- Result: 96-bit key $k = (k_1, k_2, k_3)$ found in under 1 second on a normal laptop

Gröbner Bases – Complexity

- Reminder: Computing a Gröbner basis only one of the steps in the attack
  - In most cases, we expect it to be the most expensive one
  - Complexity difficult to estimate (number of variables and equations, degrees, …)
  - Last step (factorization) may also be a bottleneck
- Most theoretic results apply to “random” systems
  - However, cryptographic schemes tend to be well-structured
- Advantage: The attack does not need many (plaintext, ciphertext) pairs (sometimes, even one pair is enough!)
- Protection (simplified): Force attacker to use many variables, increase degrees of equations

Gröbner Bases – Recent Ciphers

- Not many successful attacks using Gröbner bases in literature
- However, recently they gain importance due to new ciphers which exhibit a "nice" algebraic structure
  - Design of such algorithms is motivated by new use cases (e.g., STARK proofs)
  - Gröbner bases can provide a very strong attack against such ciphers
- In general: difficult to apply Gröbner bases to bit-based schemes (i.e., working in $\mathbb{F}_2$)
  - Many variables
  - Necessary to add field equations ($x_i^2 = x$)

Algebraic Attacks – Summary

- Exploit the determinism of a cryptographic scheme
- Try to find precise solutions without relying (mostly) on probabilities
- Many different approaches, properties like degrees are important
- Work in many different fields ($\mathbb{F}_2$, $\mathbb{F}_p$, $\mathbb{F}_{2^n}$)
- Exploit constructions with a “nice” algebraic structure

Questions

1. What are some of the differences between statistical and algebraic attacks?
2. What are the main ideas of the interpolation attack on the $\mathcal{PURE}$ cipher?
3. You know that the algebraic degree of a function is 10. How many inputs (at most) to you need in order to generate a zero sum at the output?
4. What is a superpoly in the context of cube attacks? Why do we want to avoid constant superpolys?
5. Which parameters are mainly responsible for the complexity of a Gröbner basis computation?

References I

[BDPA11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. **The Keccak reference**. Round 3 submission to NIST SHA-3. 2011. URL: http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[Buc65] Bruno Buchberger. **Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal**. PhD thesis. University of Innsbruck, 1965.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. **Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations**. EUROCRYPT. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 392–407.

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. **Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function**. EUROCRYPT (1). Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 733–761.

References II

[DS09] Itai Dinur and Adi Shamir. **Cube Attacks on Tweakable Black Box Polynomials**. EUROCRYPT 2009. Vol. 5479. LNCS. 2009, pp. 278–299.

[JK97] Thomas Jakobsen and Lars R. Knudsen. **The Interpolation Attack on Block Ciphers**. FSE. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 28–40.

[Knu94] Lars R. Knudsen. **Truncated and Higher Order Differentials**. FSE. Vol. 1008. Lecture Notes in Computer Science. Springer, 1994, pp. 196–211.

[Lai94] Xuejia Lai. **Higher Order Derivatives and Differential Cryptanalysis**. Communications and Cryptography 1994. Kluwer Academic Publishers, 1994, pp. 227–233.

[NK95] Kaisa Nyberg and Lars R. Knudsen. **Provable Security Against a Differential Attack**. J. Cryptology 8.1 (1995), pp. 27–37.

[Vie07] Michael Vielhaber. **Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack**. IACR Cryptology ePrint Archive 2007 (2007), p. 413.