Information Security Group

On Cold Boots and Noisy Polynomials

Martin Albrecht & Carlos Cid

"Consider . . . the linear case in n variables. n equations are normally soluble, and $n + 1$ are not, so one of the $n + 1$ must be noisy. But which? It could be any of them."

– anonymous referee

PhD Seminar, Egham, 18.Feb. 2010

Outline

Outline

Coldboot Attacks I

Play

`http://citp.princeton.edu/memory/`

Coldboot Attacks II

Information Security Group

Definition (The Coldboot Problem)

We are given

1. $\mathcal{K} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^N$ where $N > n$,
2. two real numbers $0 \leq \delta_0, \delta_1 \leq 1$,
3. some **noisy** output $O = \mathcal{K}(k)$: each bit o_i is correct
   - with probability $1 - \delta_0$ if it is zero and
   - with probability $1 - \delta_1$ if it is one.
4. and some control function $\mathcal{E} : \mathbb{F}_2^n \rightarrow \{True, False\}$, which returns true for the pre-image of the noise free version of O.

The task is to recover k such that $\mathcal{E}(k)$ returns *True* or a noise-free O.

The Coldboot problem is equivalent to decoding a (non-)linear code with biased noise.

Coldboot Attacks III

Results in [3]:

Cipher	δ_0	δ_1	Success	Time
DES	0.10	0.001	100%	–
DES	0.50	0.001	98%	–
AES	0.15	0.001	100%	1s
AES	0.30	0.001	100%	30s

Can we do better and can we recover keys for more complicated key schedules like Serpent or Twofish?

Outline

PoSSo

We define polynomial system solving (**PoSSo**) as the problem of finding a solution to a system of polynomial equations over some field.

Definition (PoSSo)

Consider the set $F = \{f_0, \dots, f_{m-1}\}$ where each $f_i \in \mathbb{F}[x_0, \dots, x_{n-1}]$.

A solution to F is any point $x \in \mathbb{F}^n$ such that

$$\forall f_i \in F : f_i(x) = 0.$$

Note, that we restrict ourselves to solutions in the base field here.

Max-PoSSo I

We can define a family of **Max-PoSSo** problems, analogous to the well known Max-SAT family of problems.

`http://en.wikipedia.org/wiki/MAX-SAT`

Max-PoSSo II

Information Security Group

Definition (Max-PoSSo)

Find a point $x \in \mathbb{F}^n$ which satisfies the **maximum number** of polynomials in $F = \{f_0, \ldots, f_{m-1}\} \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$.

Max-PoSSo III

Information Security Group

Definition (Partial Max-PoSSo)

Find a point $x \in \mathbb{F}^n$ such that for **two sets of polynomials** $\mathcal{H}$ and $\mathcal{S}$ in $\mathbb{F}[x_0, \ldots, x_{n-1}]$

- $\forall f \in \mathcal{H} : f(x) = 0$ and
- the number of polynomials $f \in \mathcal{S}$ with $f(x) = 0$ is maximised.

- Max-PoSSo is Partial Max-Posso with $\mathcal{H} = \varnothing$.
- $\mathcal{H}$ for "hard" and $\mathcal{S}$ for "soft".
- Both terms are borrow from Partial Max-SAT.

Max-PoSSo IV

Information Security Group

Definition (Partial Weighted Max-PoSSo)

Find a point $x \in \mathbb{F}^n$ such that

- $\forall f \in \mathcal{H} : f(x) = 0$ and
- $\sum_{f \in \mathcal{S}} \mathcal{C}(f, x)$ is minimized

where $\mathcal{C} : f \in \mathcal{S}, x \in \mathbb{F}^n \to \mathbb{R}_{\geq 0}$ is a **cost function** which

- returns 0 if $f(x) = 0$ and
- some value > 0 if $f(x) \neq 0$.

Partial Max-PoSSo is Weighted Partial Max-PoSSo where $\mathcal{C}(f, x)$ returns 1 if $f(x) \neq 0$ for all $f \in \mathcal{S}$.

Coldboot as Partial Weighted Max-PoSSo

Information Security Group

- Let $F_{\mathcal{K}}$ be an equation system corresponding to $\mathcal{K}$.
- Assume that for each noisy output bit o_i there is some $f_i \in F_{\mathcal{K}}$ of the form $g_i + o_i$ where g_i is some polynomial.
- Assume that these are the only polynomials involving output bits.
- Denote the set of these polynomials $\mathcal{S}$.
- Denote the set of all remaining polynomials $\in F_{\mathcal{K}}$ as $\mathcal{H}$.
- Define the cost function $\mathcal{C}$ as a function which returns

$$\begin{array}{ll} \frac{1}{\delta_0} & \text{for } o_i = 0, f_i(x) \neq 0 \\ \frac{1}{\delta_1} & \text{for } o_i = 1, f_i(x) \neq 0 \\ 0 & \text{otherwise} \end{array}.$$

- Express $\mathcal{E}$ as a polynomial system which is satisfiable for k only and add these polynomials to $\mathcal{H}$.

Other Applications

Information Security Group

RFID security is often based on the LPN problem which is easily described as a Max-PoSSo problem.

Lattices security often rests on the LWE problem which is easily described as a Max-PoSSo problem.

Side-Channel data leakage is often noisy.

Algebraic Attacks can be improved by simplifying equation systems using probabilistic equations.

The family of Max-PoSSo problems has not be studied before as far as we can tell. There is some connection to solving polynomial systems over fixed precision real-numbers.

Outline

1. Coldboot Attacks
2. Polynomial System Solving with Noise
3. Mixed Integer Programming
4. Application

Mixed Integer Programming I

Integer optimization deals with the problem of minimising (or maximising) a function in several variables subject to linear equality and inequality constraints and integrality restrictions on some or all of the variables.

Definition (MIP)

A linear mixed-integer programming problem (MIP) is defined as a problem of the form

$$\min_{x}\{c^T x \mid Ax \leq b, x \in \mathbb{Z}^k \times \mathbb{R}^l\}$$

where

- A is an $m \times n$-matrix $(n = k + l)$,
- b is an m-vector and c is an n-vector.

Mixed Integer Programming II

Information Security Group

This means that we minimize the linear function $c^T x$ subject to linear equality and inequality constraints given by A and b.

We have that $k \geq 0$ variables are restricted to integer values while $l \geq 0$ variables are real-valued.

The set S of all $x \in \mathbb{Z}^k \times \mathbb{R}^l$ which satisfies the linear constraints $Ax \leq b$

$$S = \{x \in \mathbb{Z}_k \times \mathbb{R}_l, Ax \leq b\}$$

is called the feasible set.

If $S = \varnothing$ the problem is infeasible. Any $x \in S$ which minimises (or maximises) $c^T x$ is an optimal solution.

Mixed Integer Programming III

Information Security Group

Example

Maximise $x + 5y$, thus $c = (1, 5)$, subject to the constraints $x + 0.2y \leq 4$ and $1.5x + 3y \leq 4$ where $x \geq 0$ is real valued and $y \geq 0$ is integer valued.

The optimal value for $c^T x$ is $5\frac{2}{3}$ for $x = \frac{2}{3}$ and $y = 1$.

```
sage: p = MixedIntegerLinearProgram()
sage: x, y = p.new_variable(), p.new_variable()
sage: p.set_integer(y[0])
sage: p.add_constraint(x[0] + 0.2*y[0], max=4)
sage: p.add_constraint(1.5*x[0] + 3*y[0], max=4)
sage: p.set_min(x[0],0); p.set_min(y[0],0)
sage: p.set_objective(x[0] + 5*y[0])
sage: p.solve()
5.66666666666661
```

PoSSo as MIP I

Information Security Group

Consider some $f \in \mathbb{F}_2[x_0, \dots, x_{n-1}]$ and let $\mathcal{Z}$ a function that takes a polynomial over $\mathbb{F}_2$ lifts it to the integers. Analogous for elements in $\mathbb{F}_2$.

1. Restrict all x_i to binary values.
2. Evaluate $\mathcal{Z}(f)$ on all $\{\mathcal{Z}(x) \mid x \in \mathbb{F}_2^n, f(x) = 0\}$.
3. Let ℓ be the minimum value and u the maximum value.
4. Introduce some integer variable $\frac{\ell}{2} \leq m \leq \frac{u}{2}$.
5. Replace each monomial in $f - 2m$ by a new linearised variable, call the result g and add the linear constraint $g = 0$.
6. For each monomial $t = \prod_{i=1}^{N} x_i$
   - add a constraint $x_i \geq t$ and
   - add a constraint $0 \leq \sum_{i=1}^{N} x_i - t \leq N - 1$.

This is the **Integer Adapted Standard Conversion** [1].

PoSSo as MIP II

Information Security Group

Example

Consider $f = ac + a + b + c + 1$

- $\{x \mid x \in \mathbb{F}_2^3, f(x) = 0\} = \{(1,0,0),(0,1,0),(0,0,1),(1,0,1)\}$
- $\ell = 1,\ u = 2$
  1. $g = M + a + b + c + 1 - 2m = 0$
  2. $a \geq M$
  3. $c \geq M$
  4. $0 \leq a + c - M \leq 1$

PoSSo as MIP III

```
sage: attach anf2mip.py
sage: B.<a,b,c> = BooleanPolynomialRing()
sage: f = a*c + a + b
sage: bc = BooleanPolynomialMIPConverter()
sage: p = bc.integer_adapted_standard_conversion([f]); p
Mixed Integer Program  ( minimization, ...
sage: p.show()
Minimization:
  x_1 +x_2 +x_3 +x_4
Constraints:
  0 <= -2 x_0 +x_1 +x_2 +x_3 <= 0
  -1 <= x_2 -1 x_3 <= 0
  -1 <= x_2 -1 x_4 <= 0
  0 <= -1 x_2 +x_3 +x_4 <= 1
Variables:
  x_0 is an integer variable (min=0.0, max=1.0)
  x_1 is an boolean variable (min=0.0, max=1.0)
  x_2 is a real variable (min=0.0, max=1.0)
  x_3 is an boolean variable (min=0.0, max=1.0)
  x_4 is an boolean variable (min=0.0, max=1.0)
```

Partial Weighted Max-PoSSo as MIP

Information Security Group

- Convert each $f \in \mathcal{H}$ to linear constraints as before.
- For each $f_i \in \mathcal{S}$ add a new binary slack variable e_i to f_i and convert the resulting polynomial as before.
- The objective function we minimise is $\sum c_i e_i$ where c_i is the value of $\mathcal{C}(f, x)$ for some x such that $f(x) \neq 0$.

Any optimal solution $x \in S$ will be an optimal solution to the Weighted Partial Max-PoSSo problem.

Coldboot as MIP

Information Security Group

Coldboot $\rightarrow$ Partial Weighted Max-PoSSo $\rightarrow$ MIP

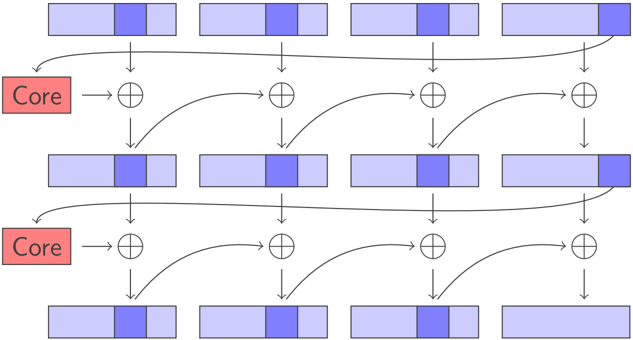
This approach is essentially the non-linear generalisation of decoding random linear codes with linear programming [2].

Outline

Simplifications

- We do not model $\mathcal{E}$ since its representation is often too costly; consequently we have no guarantee that the optimal k returned is indeed the k we are looking for.
- We do not include all equations available to us but restrict our attention to a subset (e.g. one or two rounds).
- We may use an "aggressive" modelling strategy where we assume $\delta_1 = 0$ which allows us to promote some polynomials from $\mathcal{S}$ to $\mathcal{H}$.

AES I

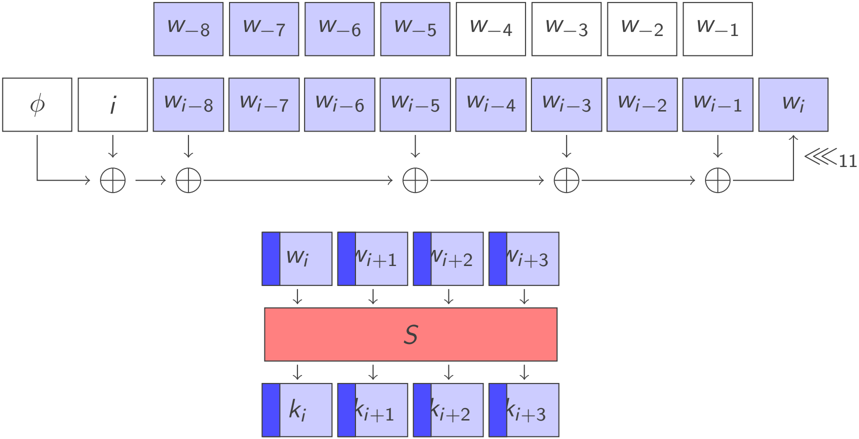


AES II

Information Security Group

		Gurobi			
instance	δ_0	#cores	cutoff t	r	max t
SR(2,4,4,4)	0.15	24	∞	100%	0.8s
SR(3,4,4,4)	0.30	24	∞	100%	41.41s
SR(3,4,4,4)	0.45	24	∞	60%	86.24s
SR(4,4,4,4)	0.45	24	∞	100%	976.0s
SR(2,4,4,8)	0.15	24	∞	100%	17956.4s
SR(2,4,4,8)	0.15	2	240.0s	25%	240.0s
aSR(2,4,4,8)	0.30	4	3600.0s	20%	3600.0s

Serpent I


w_{-8} w_{-7} w_{-6} w_{-5} w_{-4} w_{-3} w_{-2} w_{-1}
ϕ i w_{i-8} w_{i-7} w_{i-6} w_{i-5} w_{i-4} w_{i-3} w_{i-2} w_{i-1} w_i
$\lll 11$
w_i w_{i+1} w_{i+2} w_{i+3}
S
k_i k_{i+1} k_{i+2} k_{i+3}

Serpent II

		Gurobi			
#words	δ_0	#cores	cutoff t	r	max t
8	0.05	2	60.0s	50%	16.22s
12	0.05	2	60.0s	85%	60.00s
8	0.15	24	600.0s	20%	103.17s
12	0.15	24	600.0s	55%	600.00s
*12	0.30	24	7200.0s	20%	7200.00s

Serpent III

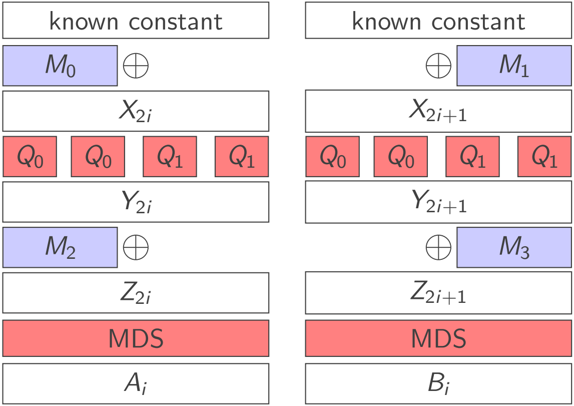
Ad-hoc approach:

- We wish to recover a 128-bit key, so we need to consider at least 128-bit of output.
- On average the noise free output should have 64 bits set to zero.
- In order to consider an error rate up to δ_0, we have to consider

$$\sum_{i=0}^{\lceil \delta_0 \cdot 64 \rceil} \binom{64 + \lceil \delta_0 \cdot 64 \rceil}{i}$$

  candidates and test them.
- If $\delta_0 = 0.15$ we have $\approx 2^{36.87}$.
- If $\delta_0 = 0.30$ we have $\approx 2^{62}$.

Twofish I



The output of the key schedule is then

$$A_i \boxplus B_i$$

and

$$A_i \boxplus 2 \cdot B_i.$$

Twofish II

Information Security Group

Ad-hoc approach:

- We wish to recover a 128-bit key, so we need to consider at least 128-bit of output.
- On average the noise free output should have 64 bits set to zero.
- In order to consider an error rate up to δ_0, we have to consider

$$\sum_{i=0}^{\lceil \delta_0 \cdot 64 \rceil} \binom{64 + \lceil \delta_0 \cdot 64 \rceil}{i}$$

  candidates and test them.
- If $\delta_0 = 0.15$ we have $\approx 2^{36.87}$.
- If $\delta_0 = 0.30$ we have $\approx 2^{62}$.
- Due to the lack of inner diffusion solving the system for each instance is easy.

Twofish III

Other approaches:

- We have more information from the key dependent S-boxes which give us 64 bits worth of linear equations. However, including them makes the final solving step much harder.
- We can attempt to recover the noise-free version of O using MIP and then solve only once.

Thank you!

Drinks at 6 in the Happy Man?

Literature I

- Julia Borghoff, Lars R. Knudsen, and Mathias Stolpe.
  Bivium as a Mixed-Integer Linear programming problem.
  In Matthew G. Parker, editor, *Cryptography and Coding – 12th IMA International Conference*, volume 5921 of *Lecture Notes in Computer Science*, pages 133–152, Berlin, Heidelberg, New York, 2009. Springer Verlag.

- Jon Feldman.
  Decoding Error-Correcting Codes via Linear Programming.
  PhD thesis, Massachusetts Institute of Technology, 2003.

- J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten.
  Lest we remember: Cold boot attacks on encryption keys.
  In *Proceedings of 17th USENIX Security Symposium*, pages 45–60, 2008.