Information Security Group

On Cold Boots and Noisy Polynomials

Martin Albrecht & Carlos Cid

US Naval Academy Visit, Egham, 17. March 2010

Outline

Outline

Coldboot Attacks I

- In [3] a method is described for extracting cryptographic key material from DRAM.
- DRAM may retain large part of its content for several seconds after removing its power.
- Furthermore, time can potentially be increased by using cooling techniques.
- In the case of the AES and DES simple algorithms are also proposed in [3] to recover the key from the observed set of round subkeys in memory, which are however subject to errors (due to memory bits decay).

Coldboot Attacks II

Information Security Group

Definition (The Coldboot Problem)

We are given

1. $\mathcal{K} : \mathbb{F}_2^n \to \mathbb{F}_2^N$ where $N > n$,
2. two real numbers $0 \leq \delta_0, \delta_1 \leq 1$,
3. some **noisy** output $K = \mathcal{K}(k)$: each bit K_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 $\mathcal{E} : \mathbb{F}_2^n \to \{True, False\}$, which returns true for the pre-image of the noise free version of K.

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

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

Coldboot Attacks III

Information Security Group

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 such as Serpent?

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

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`

In fact, we can reduce Max-PoSSo to 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 Weighted Max-PoSSo)

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

- $\forall f \in \mathcal{H} : f(x) = 0$ 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 $f(x) \neq 0$.

Coldboot as Partial Weighted Max-PoSSo

- Let $F_{\mathcal{K}}$ be an equation system corresponding to $\mathcal{K}$.
- Assume that for each noisy output bit K_i there is some $f_i \in F_{\mathcal{K}}$ of the form $g_i + K_i$ 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

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

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

Outline

Mixed Integer Programming I

Information Security Group

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.

We minimise (or maximise) a linear function $c^T x$ subject to linear equality and inequality constraints given by some matrix A and a vector b as $Ax \leq b$.

We have that some variables are restricted to integer values while other variables are real-valued.

Mixed Integer Programming II

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.

PoSSo as MIP I

Information Security Group

Consider some $f \in \mathbb{F}_2[x_0, \ldots, x_{n-1}]$ 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].

Partial Weighted Max-PoSSo as MIP

- 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

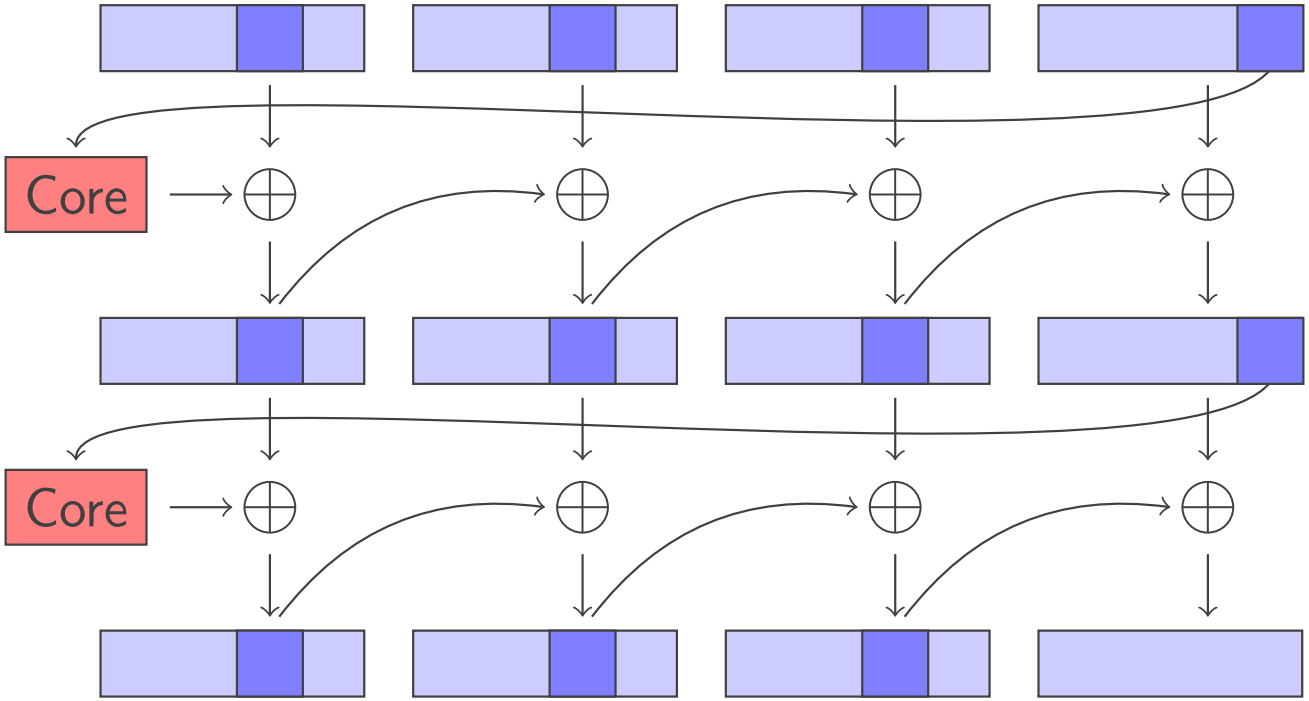
This approach is essentially the non-linear generalisation of decoding random linear codes with linear programming [2].

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

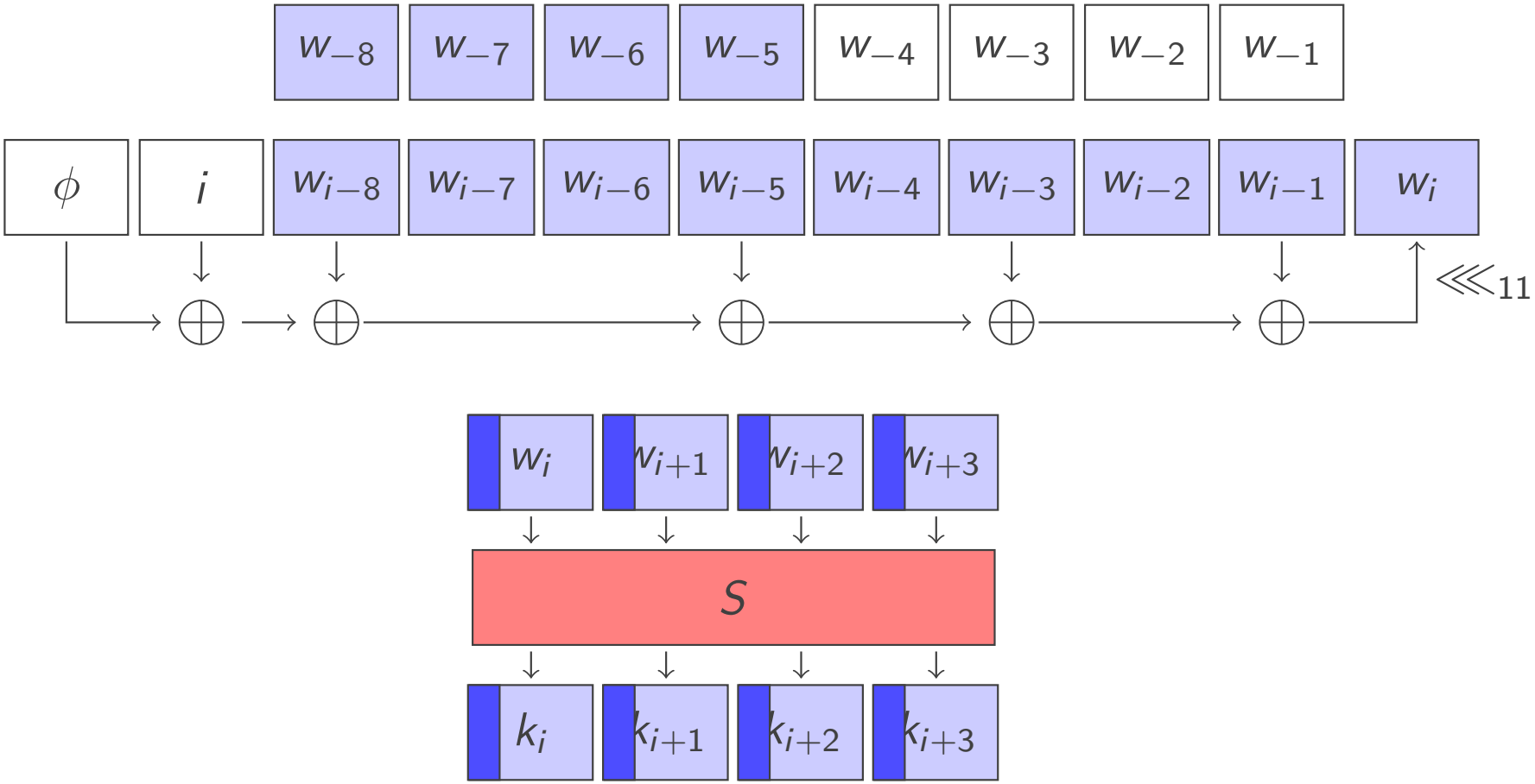


AES II

Problem	δ_0	a	Cutoff t	r	Max t
SR(2,4,4,4)	0.15	–	∞	100%	31.8s
SR(3,4,4,4)	0.30	–	3600.0s	95%	1684.42s
SR(3,4,4,4)	0.45	–	3600.0s	55%	3600.00s
SR(4,4,4,4)	0.45	–	7200.0s	10%	7200.00s
SR(2,4,4,8)	0.15	+	3600.0s	65%	3600.0s
SR(2,4,4,8)	0.15	+	3600.0s	64%	3600.0s
SR(2,4,4,8)	0.30	+	7200.0s	45%	7200.0s
SR(2,4,4,8)	0.35	+	10800.0s	10%	10800.0s
SR(2,4,4,8)	0.40	+	14400.0s	0%	14400.0s
SR(3,4,4,8)	0.40	+	14400.0s	10%	14400.0s

Solver: SCIP (`http://scip.zib.de`)

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

#words	δ_0	a	#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

Solver: Gurobi (`http://www.gurobi.com`)

Serpent III

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 $\approx 2^{62}$.

Information Security Group

Thank you!

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.