Information Security Group

Cold Boot Key Recovery using Polynomial System Solving with Noise

Martin Albrecht & Carlos Cid

SCC 2010, Egham, UK

Outline

Outline

Coldboot Attacks I

Information Security Group

- 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

We are given

1. an efficiently computable function $\mathcal{KS} : \mathbb{F}_2^n \to \mathbb{F}_2^N$ with $N > n$,
2. two real numbers $0 \leq \delta_0, \delta_1 \leq 1$ and
3. some efficiently computable function $\mathcal{E}(k) \to \{True, False\}$.

Let $(K_0, \dots, K_{N-1}) = \mathcal{KS}(k)$. Compute $(K'_0, K'_1, \dots, K'_{N-1})$ with:

$$\begin{array}{rclcrcl} Pr[K'_i = 0 \mid K_i = 0] & = & 1 - \delta_1, & & Pr[K'_i = 1 \mid K_i = 0] & = & \delta_1, \\ Pr[K'_i = 1 \mid K_i = 1] & = & 1 - \delta_0, & & Pr[K'_i = 0 \mid K_i = 1] & = & \delta_0. \end{array}$$

$K'_i = 0$ is correct with probability $Pr[K_i = 0 \mid K'_i = 0] = \frac{(1-\delta_1)}{(1-\delta_1+\delta_0)} = \Delta_0$. Likewise for $K'_i = 1$.

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

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

Information Security Group

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

Information Security Group

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 $F = \{f_0, \dots, f_{m-1}\} \subset \mathbb{F}[x_0, \dots, x_{n-1}]$.

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 P. W. Max-PoSSo

Information Security Group

- 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

$$\begin{array}{cr} \frac{1}{1-\Delta_0} & \text{for } K_i' = 0, f(x) \neq 0, \\ \frac{1}{1-\Delta_1} & \text{for } K_i' = 1, f(x) \neq 0, \\ 0 & \text{otherwise.} \end{array}$$

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

Outline

1 Coldboot Attacks

2 Polynomial System Solving with Noise

3 Mixed Integer Programming

4 Application

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

Information Security Group

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

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 → Partial Weighted Max-PoSSo → MIP

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

Outline

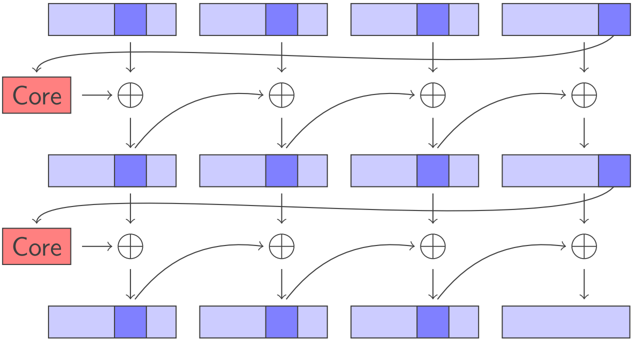
Simplifications

Information Security Group

- 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 $\mathcal{H}$.

Information Security Group

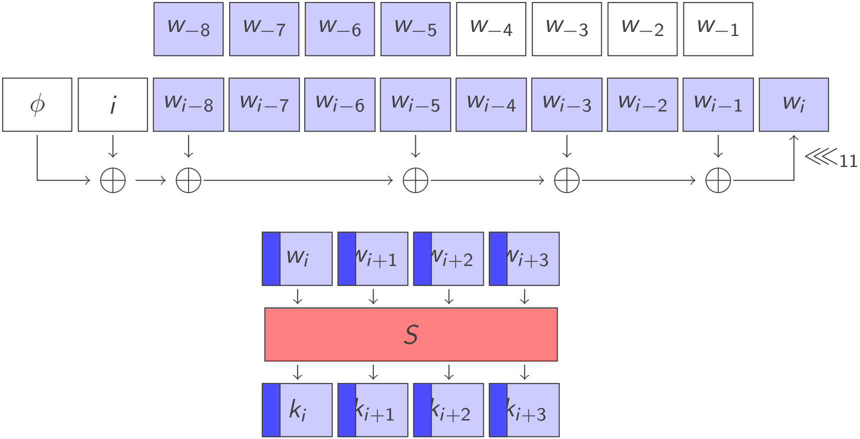
AES I



AES II

		Gurobi `http://www.gurobi.com`				
N	δ_0	a	#cores	cutoff t	r	max t
3	0.15	–	24	∞	100%	17956.4s
3	0.15	–	2	240.0s	25%	240.0s
3	0.30	+	24	3600.0s	25%	3600.0s
3	0.35	+	24	28800.0s	30%	28800.0s
		SCIP `http://scip.zib.de`				
3	0.15	+	1	3600.0s	65%	1209.0s
4	0.30	+	1	7200.0s	47%	7200.0s
4	0.35	+	1	10800.0s	45%	10800.0s
4	0.40	+	1	14400.0s	52%	14400.0s
5	0.40	+	1	14400.0s	45%	14400.0s

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 http://www.gurobi.com				
N	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
		SCIP http://scip.zib.de				
12	0.15	+	1	600.0s	32%	597.37s
16	0.15	+	1	3600.0s	48%	369.55s
20	0.15	+	1	3600.0s	29%	689.18s
32	0.15	+	1	3600.0s	21%	1105.58s
16	0.30	+	1	3600.0s	55%	3600.00s
20	0.30	+	1	7200.0s	57%	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 $\approx 2^{62}$.

Thank you!

Literature I

Information Security Group

- 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.