# Automatic Search of Attacks on round-reduced AES and Applications

**Charles Bouillaguet** Patrick Derbez Pierre-Alain Fouque

ENS, CNRS, INRIA Cascade

August 15, 2011

# Block-Cipher Cryptanalysis

## The Object: a Block Cipher

$$E : \underbrace{\{0,1\}^k}_{\text{key}} \times \underbrace{\{0,1\}^n}_{\textit{plaintext}} \rightarrow \underbrace{\{0,1\}^n}_{\textit{ciphertext}}$$

often $k = n$, but not always (e.g. AES-256: $n = 128$ and $k = 256$)

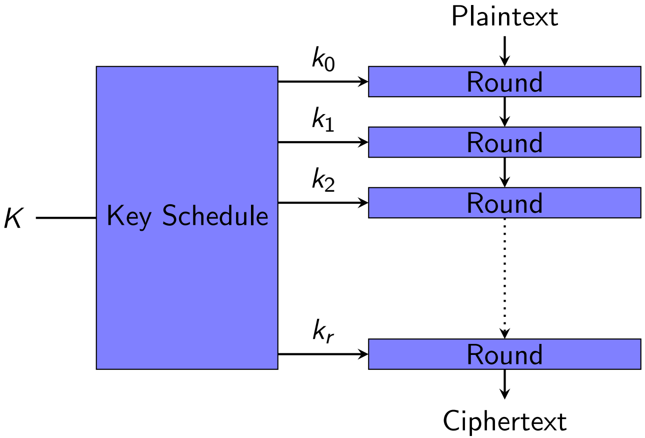
## The Subject: an Attacker

- Objective: **recover the secret key** (or maybe distinguish from random)
- Resources:
  - Time: less than $2^k$ encryptions
  - Data: less than $2^n$ plaintext/ciphertext pairs

Total Breaks of widely-used block ciphers are *relatively rare* (in comparison with hash functions/stream ciphers)

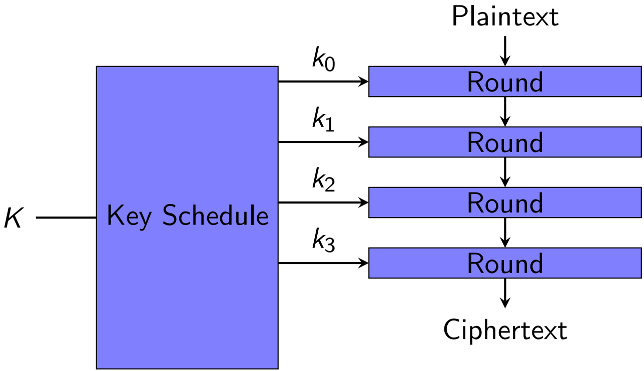
## What to do when block ciphers are too strong for us?

- **Solution # 1**:
  - First weaken it
  - Then break it



## What to do when block ciphers are too strong for us?

- **Solution # 1**:
  - First weaken it (reduce number of rounds)
  - Then break it



## What to do when block ciphers are too strong for us?

- **Solution # 2**:
  - First we get stronger
  - Then break it

## What to do when block ciphers are too strong for us?

- **Solution # 2**:
  - First we get stronger (chosen ciphertexts, )
  - Then break it


## What to do when block ciphers are too strong for us?

- **Solution # 2**:
  - First we get stronger (chosen ciphertexts, related keys, etc.)
  - Then break it


## Solution #3: Play Another Game

In this talk: **Low Data Complexity Attacks**

- Has to be faster than exhaustive search
- Only very few plaintext/ciphertext pairs available

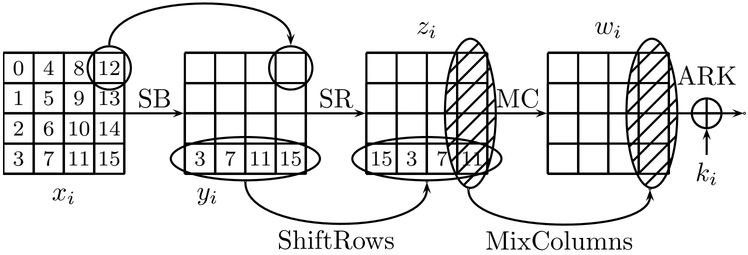
### Why ?

- Rather unexplored territory
- What is harder in practice?
  - **performing** $2^{50}$ elementary operations?
  - or **acquiring** 50 Plaintext/Ciphertext pairs?
- LDC attacks can sometimes be recycled, and used as sub-components in other attacks
  - *e.g.* attack on GOST uses a 2-plaintext attack on 8 rounds

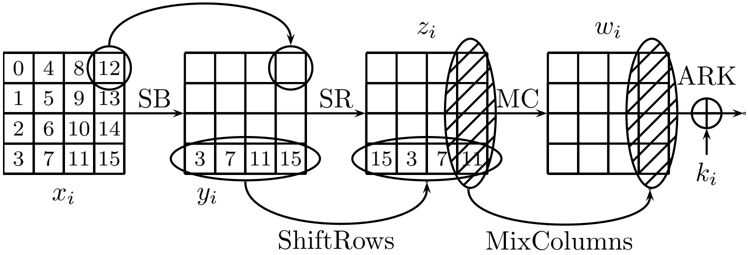
## Target Block Cipher: the Advanced Encryption Standard

- Designed by Rijmen and Daemen for AES competition
- Selected as the AES in 2001
- One of the most widely used encryption primitive
- AES basic structures :
  - Substitution-Permutation network
  - Block size: 16-bytes (128 bits)
  - key lengths: **128**, 192 or 256 bits
  - 10 rounds for the 128-bit version

# Description of the AES



## Description of the AES




- Single-key attacks up to :
  - 8 rounds on AES-128
  - 9 rounds on AES-192/256
- Related-subkey attacks on the full AES-256/AES-192
- Complexities just slightly less than the naturals bounds

# Techniques for Low Data Complexity Attacks

**The problem with "Usual" attack techniques**

- Statistical attacks (*e.g.*, differential, impossible,linear)
- "Golden-plaintext" attacks (*e.g.*, reflexion, slide)

They require **(VERY) LARGE QUANTITY** of data

What's left?

- Algebraic Attacks/SAT-solvers ?
- Guess-and-Determine attacks
- Meet-in-the-Middle attacks

## Techniques for Low Data Complexity Attacks

**The problem with "Usual" attack techniques**

- Statistical attacks (*e.g.*, differential, impossible,linear)
- "Golden-plaintext" attacks (*e.g.*, reflexion, slide)

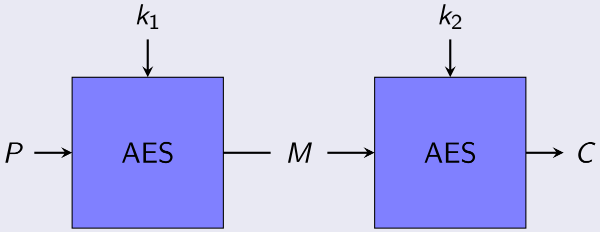
They require **(VERY) LARGE QUANTITY** of data

What's left?

- ~~Algebraic Attacks/SAT-solvers~~
- Guess-and-Determine attacks
- Meet-in-the-Middle attacks

# Meet-in-the-Middle Attacks

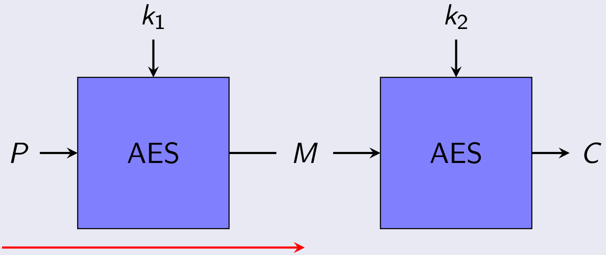
## A very bad way to build an AES with 256-bit keys




$$E_{k_1,k_2} = AES_{k_1} \circ AES_{k_2}$$

## Meet-in-the-Middle Attacks

### A **very bad way** to build an AES with 256-bit keys

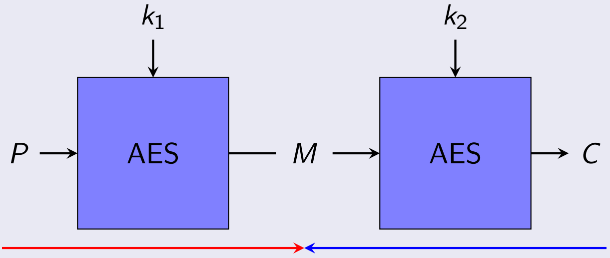



$$E_{k_1,k_2} = AES_{k_1} \circ AES_{k_2}$$

- For all $k_1$, store $AES_{k_1}(P) \to k_1$ in a hash table

## Meet-in-the-Middle Attacks

**A very bad way to build an AES with 256-bit keys**

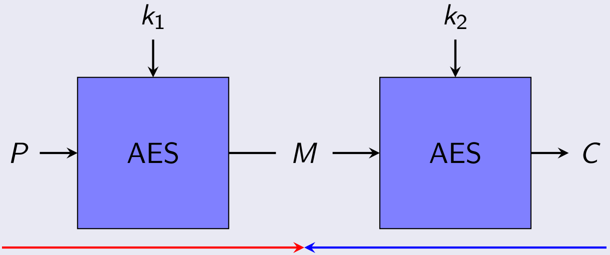



$$E_{k_1,k_2} = AES_{k_1} \circ AES_{k_2}$$

- For all $k_1$, store $AES_{k_1}(P) \to k_1$ in a hash table
- For all $k_2$, look-up $AES^{-1}_{k_2}(C)$ in the hash table

## Meet-in-the-Middle Attacks

### A very bad way to build an AES with 256-bit keys




$$E_{k_1,k_2} = AES_{k_1} \circ AES_{k_2}$$

- For all $k_1$, store $AES_{k_1}(P) \to k_1$ in a hash table
- For all $k_2$, look-up $AES^{-1}_{k_2}(C)$ in the hash table
- We expect $\approx 1$ value of $k_1$ per value of $k_2$

Time complexity $\approx 2^{128}$ encryptions, with 256-bit keys!

## Cryptanalytic Tools

We want to find Guess-n-determine/Meet-in-the-middle attacks

**Problems**

- We are lazy
- It is delicate and complicated, and nearly made us crazy

Standard Solution: build a **tool** to do the job for you!

**We are not alone!** *E.g.*, Tools to find **differential paths**:

**DES** [Matsui, 93], **SHA-1** [de Cannière et. al, 06], **Grindhal** [Peyrin et al., 07], **RadioGatùn** [Fuhr et al., 09], **MD4/MD5** [Leurent et al., 07], **AES** [Biryukov et al., 10], etc.

## The AES Has a Clean Description over $\mathbb{F}_{256}$

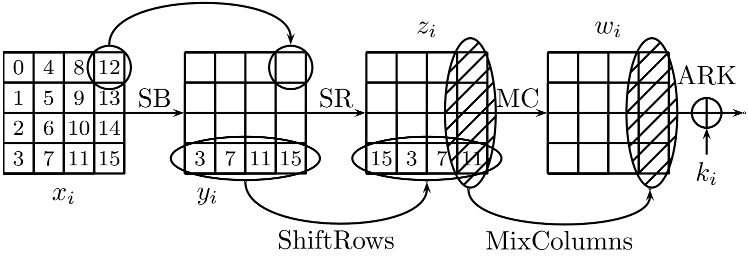
### Is it a Problem?

- Concerns about the AES's algebraic simplicity have been expressed several times
- But so far, no attack directly exploited this property...

...Until now

# The AES Has a Clean Description over $\mathbb{F}_{256}$

## Round Function



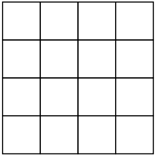

$$y_i[\ell] = S(x_i[\ell])$$

$$x_{i+1} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} y_i[0] & y_i[4] & y_i[8] & y_i[12] \\ y_i[5] & y_i[9] & y_i[13] & y_i[1] \\ y_i[10] & y_i[14] & y_i[2] & y_i[6] \\ y_i[15] & y_i[3] & y_i[7] & y_i[11] \end{pmatrix} + k_i$$

# The AES Has a Clean Description over $\mathbb{F}_{256}$

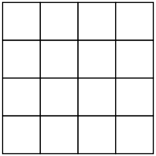
## Key-Schedule

$k_i$

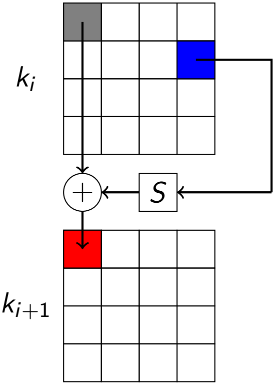


- $k_0 = K$ (the master-key)

$k_{i+1}$

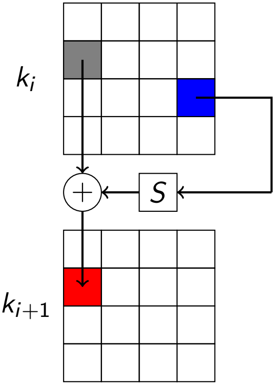


# The AES Has a Clean Description over $\mathbb{F}_{256}$
## Key-Schedule



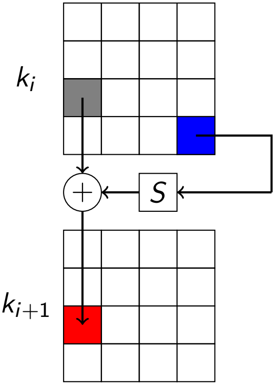

- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$

# The AES Has a Clean Description over $\mathbb{F}_{256}$
## Key-Schedule




- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$

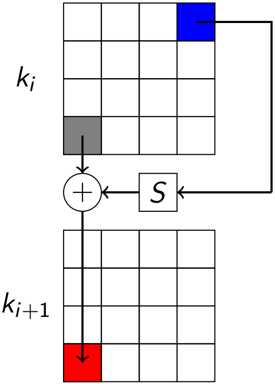
# The AES Has a Clean Description over $\mathbb{F}_{256}$
## Key-Schedule




- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$
- $k_{i+1}[2] = k_i[2] + S(k_i[15])$

# The AES Has a Clean Description over $\mathbb{F}_{256}$

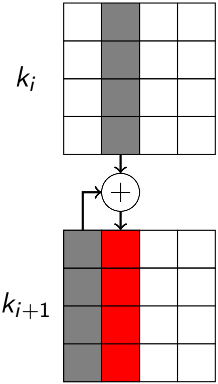
## Key-Schedule




- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$
- $k_{i+1}[2] = k_i[2] + S(k_i[15])$
- $k_{i+1}[3] = k_i[3] + S(k_i[12])$

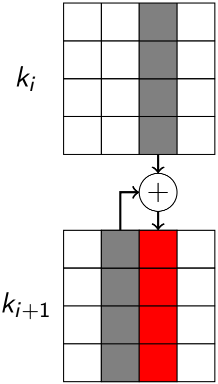
# The AES Has a Clean Description over $\mathbb{F}_{256}$

## Key-Schedule




- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$
- $k_{i+1}[2] = k_i[2] + S(k_i[15])$
- $k_{i+1}[3] = k_i[3] + S(k_i[12])$
- $k_{i+1}[4..7] = k_{i+1}[4..7] + k_i[0..3]$

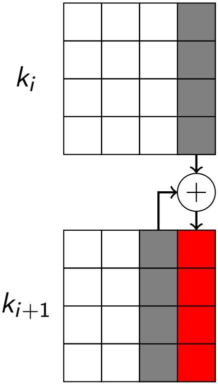
# The AES Has a Clean Description over $\mathbb{F}_{256}$
## Key-Schedule




- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \mathtt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$
- $k_{i+1}[2] = k_i[2] + S(k_i[15])$
- $k_{i+1}[3] = k_i[3] + S(k_i[12])$
- $k_{i+1}[4..7] = k_{i+1}[4..7] + k_i[0..3]$
- $k_{i+1}[8..11] = k_{i+1}[8..11] + k_i[4..7]$

# The AES Has a Clean Description over $\mathbb{F}_{256}$

## Key-Schedule



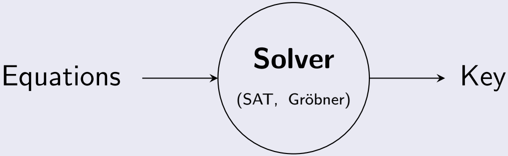

- $k_0 = K$ (the master-key)
- $k_{i+1}[0] = k_i[0] + S(k_i[13]) + \texttt{RCON}_i$
- $k_{i+1}[1] = k_i[1] + S(k_i[14])$
- $k_{i+1}[2] = k_i[2] + S(k_i[15])$
- $k_{i+1}[3] = k_i[3] + S(k_i[12])$
- $k_{i+1}[4..7] = k_{i+1}[4..7] + k_i[0..3]$
- $k_{i+1}[8..11] = k_{i+1}[8..11] + k_i[4..7]$
- $k_{i+1}[12..15] = k_{i+1}[12..15] + k_i[8..11]$

## Working With the Equations

The equations describing the AES are:

- sparse: each equation relates, at most, five variables
- structured: each variable appears in, at most, four equations
- linear over $\mathbb{F}_{256}$ in $x_i$ and $S(x_i)$

**Algebraic Cryptanalysis: have a go at the equations**



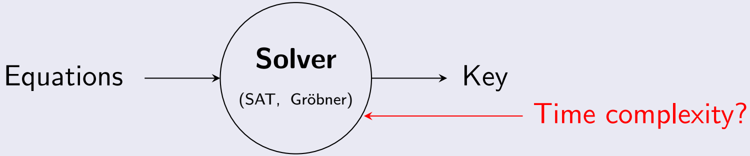

- Solving systems of AES-like equations would break the cipher

## Working With the Equations

The equations describing the AES are:

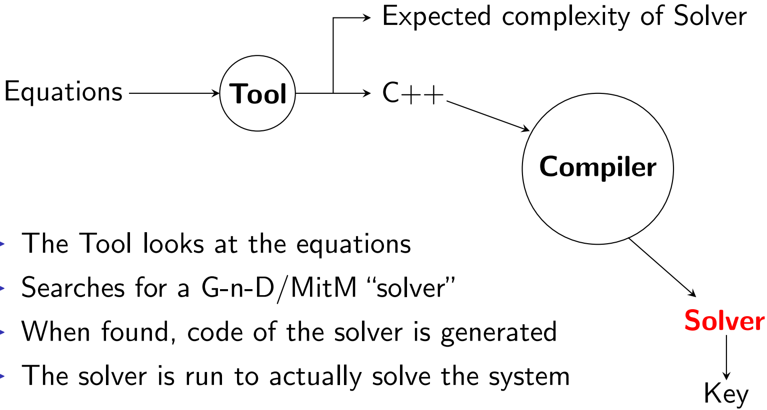
- sparse: each equation relates, at most, five variables
- structured: each variable appears in, at most, four equations
- linear over $\mathbb{F}_{256}$ in $x_i$ and $S(x_i)$

**Algebraic Cryptanalysis: have a go at the equations**




- Solving systems of AES-like equations would break the cipher
- No interesting result at this point

## Our Approach to Solve Systems of AES-like equations




- The Tool looks at the equations
- Searches for a G-n-D/MitM "solver"
- When found, code of the solver is generated
- The solver is run to actually solve the system

The structure of the equations makes:

- the search procedure (somewhat) easy
- the results (sometimes) interesting

# Harnessing The Algebraic Simplicity
## Guess-and-Determine Attacks

### The equations are **sparse**

All terms known except one: knowledge propagation

$$e.g. \quad x_i + S(z_j) + 03 \cdot z_k = 0$$

### The equations are **linear** over $\mathbb{F}_{256}$ in $x_i$ and $S(x_i)$

Gaussian elimination allows more knowledge propagation:

$$e.g. \quad \begin{cases} x_i + S(z_j) & +03 \cdot z_k & & +7\mathtt{f} \cdot u_\ell & = & 0 \\ 3\mathtt{d} \cdot x_j & +56 \cdot z_k & +S(v_r) & +9\mathtt{a} \cdot u_\ell & = & 0 \\ \mathtt{c}2 \cdot y_s & +84 \cdot z_k & +\mathtt{cf} \cdot S(v_r) & & = & 0 \end{cases}$$

All terms known except one in a linear combination

# Harnessing The Algebraic Simplicity

## Guess-and-Determine Attacks

### A Tentative Guess-and-determine Attack Search Procedure

- **For all** possible subset $X$ of the variables
  - **Assume** $X$ is known
  - **While** knowledge propagation gives a new variable $y$ **do**
    - $X \leftarrow Y \cup \{y\}$
  - **If** $X$ contains all the variables, **then** report possible solver.
- When done (or timeout) **return** best solver found so far

## Harnessing The Algebraic Simplicity

### Meet-in-the-Middle Attacks

**The equations are linear over $\mathbb{F}_{256}$ in $x_i$ and $S(x_i)$**

$$\begin{matrix} f_1(x,y,z,u,v,t) = 0 \\ f_2(x,y,z,u,v,t) = 0 \\ f_3(x,y,z,u,v,t) = 0 \\ f_4(x,y,z,u,v,t) = 0 \end{matrix} \implies \underbrace{\begin{pmatrix} g_1(x,y,z) \\ g_2(x,y,z) \\ g_3(x,y,z) \\ g_4(x,y,z) \end{pmatrix}}_{G(x,y,z)} = \underbrace{\begin{pmatrix} h_1(u,v,t) \\ h_2(u,v,t) \\ h_3(u,v,t) \\ h_4(x,y,z) \end{pmatrix}}_{H(u,v,t)}$$

MitM solver:

- **for all** $x, y, z$, store $G(x,y,z) \mapsto (x,y,z)$ in a hash table
- **for all** $u, v, t$, look-up $H(u,v,t)$ in the hash table
- We expect one value of $(x,y,z)$ per value of $(u,v,t)$.

## Harnessing The Algebraic Simplicity
### Meet-in-the-Middle Attacks

- Idea: partition the set of variables in two halves

$$F(x, y, z, t, u, v) = 0 \iff G(x, y, z) = H(t, u, v)$$

- We may choose the partition as we please

**Objective:**

Find a partition $X_1 \cup Y_1$ such that some linear combinations of the equations only contain $x_1, S(x_1), x_2, S(x_2), \ldots$ [respectively $y_1, S(y_1), \ldots$].

$$F(x, y, z, t, u, v) = 0 \iff \begin{cases} G_1(x, y, z) &= H_1(t, u, v) \\ G_2(x, y, z) &= 0 \\ 0 &= H_2(t, u, v) \end{cases}$$

# Harnessing The Algebraic Simplicity

## Recursive Meet-in-the-Middle Attacks

$$F(x,y,z,t,u,v)=0 \iff \begin{cases} G_1(x,y,z) & = & H_1(t,u,v) \\ G_2(x,y,z) & = & 0 \\ 0 & = & H_2(t,u,v) \end{cases}$$

### Improved Solving Algorithm

- for all $(x,y,z)$ such that $G_2(x,y,z)=0$
  - Store $G_1(x,y,z) \to (x,y,z)$ in a hash table
- for all $(u,v,t)$ such that $H_2(u,v,t)=0$
  - Look-up $H_1(u,v,t)$ in the hash table
- Each collision suggests a complete solution

# Harnessing The Algebraic Simplicity
## Recursive Meet-in-the-Middle Attacks

$$F(x,y,z,t,u,v) = 0 \iff \begin{cases} G_1(x,y,z) & = & H_1(t,u,v) \\ G_2(x,y,z) & = & 0 \\ 0 & = & H_2(t,u,v) \end{cases}$$

### Improved Solving Algorithm

- for all $(x,y,z)$ such that $G_2(x,y,z) = 0$
  - Store $G_1(x,y,z) \rightarrow (x,y,z)$ in a hash table
- for all $(u,v,t)$ such that $H_2(u,v,t) = 0$
  - Look-up $H_1(u,v,t)$ in the hash table
- Each collision suggests a complete solution

A solver for the full problem can be **constructed recursively** from two solvers for smaller sub-problems.

## Results

### Attacks on round reduced version of the AES-128

| | | Tool-found | | Human-found |
|---|---|---|---|---|
| #Rounds | Data | Time | Memory | Time |
| **1** | **1 KP** | $\mathbf{2^{32}}$ | $\mathbf{2^{16}}$ | $\mathbf{2^{48}}$ |
| 2 | 1 KP | $2^{64}$ | $2^{48}$ | $2^{80}$ |
| **2** | **2 KP** | $\mathbf{2^{32}}$ | $\mathbf{2^{24}}$ | $\mathbf{2^{48}}$ |
| **2** | **2 CP** | $\mathbf{2^{8}}$ | $\mathbf{2^{8}}$ | $\mathbf{2^{28}}$ |
| 3 | 1 KP | $2^{96}$ | $2^{72}$ | |
| **3** | **2 CP** | $\mathbf{2^{16}}$ | $\mathbf{2^{8}}$ | $\mathbf{2^{32}}$ |
| 4 | 1 KP | $2^{120}$ | $2^{80}$ | |
| 4 | 2 CP | $2^{80}$ | $2^{80}$ | $2^{104}$ |
| **4** | **4 CP** | $\mathbf{2^{32}}$ | $\mathbf{2^{24}}$ | |
| 4 | 5 CP | | | $2^{64}$ |
| 4.5 | 1 KP | $2^{120}$ | $2^{96}$ | |

The attacks that are practical have been implemented and verified

## Results (cont'd)

The method is somewhat generic, and applies to AES, SQUARE, PHOTON, SkipJack, LEX, Alpha-MAC, Pelican-MAC, etc.

### Pelican-MAC

Recovers the internal state (allows forgery) given an internal state collision, by solving in practice:

$$AES_4(x) + AES_4(x + \Delta_i) = \Delta_o.$$

Allows to break the MAC in $2^{64}$ queries (fastest known attack).

### LEX

Instantly rediscovers the best known differential attack in time $2^{100}$. Finds a higher-order differential attack of complexity $2^{80}$ (fastest known attack, but success probability $= 1/32$ if keystream size is restricted according to specification).

## Conclusion

### Summary

- New process to solve equations describing the AES
- Find automatically the best low data complexity attacks on round-reduced AES, Pelican-MAX, LEX
- Can generate the C++ code of the attacks

### More importantly

- Tool available at:

  `http://www.di.ens.fr/~bouillaguet/`

- Long version of this paper, with more attacks descriptions, soon to be released.