Yet another attack on whitebox AES implementation

Patrick Derbez [1], Pierre-Alain Fouque[1], Baptiste Lambin[1], Brice Minaud[2]

[1]Univ Rennes, CNRS, IRISA

[2]Royal Holloway University of London

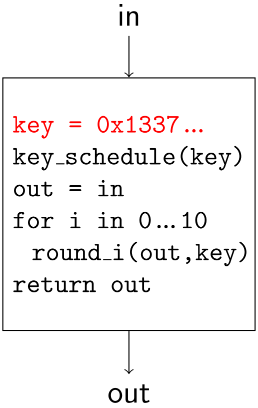
Black box vs. White box

Black box model

Black box vs. White box

Black box model

White box model



White box implementation

Attacker:

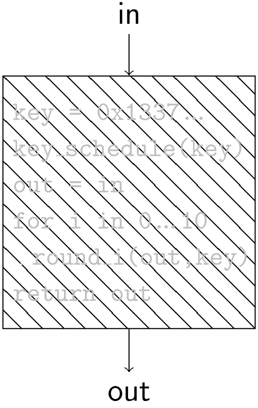
- extracting key information from the implementation
- computing decryption scheme from encryption scheme

Designer:

- provide sound and secure implementation

Main application:

- Digital Rights Management
- Fast (post-quantum 😇) public-key encryption scheme



Two main design strategies

- **Table lookup**
  - First proposal by Chow *et al.* in 2002: broken
  - Xiao and Lai in 2009: broken
  - Karroumi *et al.* in 2011: broken
  - Baek *et al.* in 2016: our target
  - *WhiteBlock* from Fouque *et al.*: secure (but weird model)
- **ASASA-like designs**
  - SASAS construction: broken in 2001 by Biryukov *et al.*
  - ASASA proposals (Biryukov *et al.*, 2014): broken
  - Recent proposals at ToSC'17 by Biryukov *et al.* to use more layers, leading to SA. . . SAS

CEJO Framework

- Derived from Chow *et al.* first white-box candidate constructions.
- Block cipher decomposed into R round functions.
- Round functions obfuscated using encodings.
- Obfuscated round functions implemented and evaluated using several tables (of reasonable size)

$$\cdots \circ \underbrace{{f^{(r+1)}}^{-1} \circ E^{(r)} \circ f^{(r)}}_{\texttt{table}} \circ \underbrace{{f^{(r)}}^{-1} \circ E^{(r-1)} \circ f^{(r-1)}}_{\texttt{table}} \circ \cdots$$

- Increase security with external encodings

Baek *et al.*'s toolbox

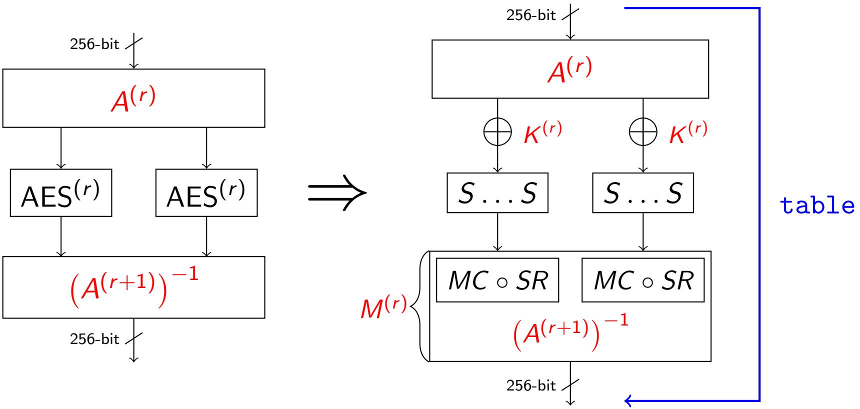
- Proposed by Baek, Cheon and Hong in 2016.
- Toolbox dedicated to SPN under CEJO framework
  - Generic method to recover non-linear part of encodings
  - Generic algorithm to recover the linear component of encodings

Finding non-linear part not higher than recovering linear part

- New AES white-box construction
  - Based on CEJO framework
  - Parallel AES
  - Resisting their toolbox (110 bits of security)
  - **Our target**

The Baek, Cheon and Hong proposal

Round function of AES : $\text{AES}^{(r)} = \text{MC} \circ \text{SR} \circ \text{SB} \circ \text{ARK}$



Sparse input encoding

$$A(x) = \begin{pmatrix} A_{0,0} & A_{0,1} & & & \\ & A_{1,1} & A_{1,2} & & \\ & & \ddots & \ddots & \\ A_{31,0} & & & & A_{31,31} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{31} \end{pmatrix} \oplus \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{31} \end{pmatrix}$$

$$M = A^{-1} \circ MC \circ SR$$

1. Split M in columns blocks of size 8 s.t. $M = (M_0|\dots|M_{31})$
2. $M.x = \bigoplus_{i=0}^{31} M_i.x_i$
3. 16-bit to 256-bit mappings: $F_i = M_i \circ S \circ \oplus_{(k_i \oplus a_i)} \circ (A_{i,i}, A_{i,i+1})$
4. Round function:

$$F^{(r)}(x_0, \dots, x_{31}) = \bigoplus_{i=0}^{31} F_i(x_i, x_{i+1})$$

Complexity

Time complexity

- R AES rounds: $32R$ table lookups + $31R$ xor of 256-bits words.
- For $R = 10$: 320 table lookups + 310 xor of 256-bit words.

Very fast

Memory requirement

- R AES rounds: $32R$ 16-bit to 256-bit mappings.
- For $R = 10$: 320 16-bit to 256-bit mappings

$\approx$ 160MB

Issue

16-bit to 256-bit mappings: $F_i = M_i \circ S \circ \oplus_{(k_i \oplus a_i)} \circ (A_{i,i}, A_{i,i+1})$

Remark

$F_i(x, 0) = M_i \circ S \circ \oplus_{(k_i \oplus a_i)} \circ A_{i,i}(x)$ is a 8-bit to 256-bit mapping.

- Composing with right projection $\Rightarrow$ affine equivalent to AES Sbox.

Issue

16-bit to 256-bit mappings: $F_i = M_i \circ S \circ \oplus_{(k_i \oplus a_i)} \circ (A_{i,i}, A_{i,i+1})$

Remark

$F_i(x, 0) = M_i \circ S \circ \oplus_{(k_i \oplus a_i)} \circ A_{i,i}(x)$ is a 8-bit to 256-bit mapping.

- Composing with right projection $\Rightarrow$ affine equivalent to AES Sbox.

Possible to recover affine mappings in $\mathcal{O}\left(2^{25}\right)$ using the affine equivalence algorithm from Biryukov *et al.*.

Affine Equivalence Algorithm

In 2003, Biryukov, De Cannière, Braeken and Preneel proposed an algorithm to solve the following problem:

Given two bijections S_1 and S_2 on n bits, find affine mappings $\mathcal{A}$ and $\mathcal{B}$ such that $S_2 = \mathcal{B} \circ S_1 \circ \mathcal{A}$, if they exist.

- Ascertain whether such mappings exist
- Enumerate all solutions
- Time complexity in $\mathcal{O}\left(n^3 2^{2n}\right)$

Affine Equivalence Algorithm

In 2003, Biryukov, De Cannière, Braeken and Preneel proposed an algorithm to solve the following problem:

Given two bijections S_1 and S_2 on n bits, find affine mappings $\mathcal{A}$ and $\mathcal{B}$ such that $S_2 = \mathcal{B} \circ S_1 \circ \mathcal{A}$, if they exist.

- Ascertain whether such mappings exist
- Enumerate all solutions
- Time complexity in $\mathcal{O}\left(n^3 2^{2n}\right)$
- Time complexity for linear version in $\mathcal{O}\left(n^3 2^n\right)$

Baek *et al.* Proposal

To avoid this weakness, take 32 random 8-bit to 256-bit mappings h_i.
The 16-bit to 256-bit tables are defined as

$$T_i(x, y) = F_i(x, y) \oplus h_i(x) \oplus h_{i+1}(y)$$

And we can evaluate the encoded round function with

$$\bigoplus_{i=0}^{31} T_i(x_i, x_{i+1}) = \bigoplus_{i=0}^{31} F_i(x_i, x_{i+1}) = F^{(r)}(x_0, \ldots, x_{31})$$

Security claim : 110-bit

Overview of the attack

From encoded round functions $F \simeq M \circ S \circ A$ with $A \simeq \begin{pmatrix} * & * & & \\ & * & * & \\ & & \ddots & \\ * & & & * \end{pmatrix}$

1. Reduce the problem to block diagonal encodings :
   $\Rightarrow \widetilde{F} = M \circ S \circ B$ with B block diagonal.
2. Compute candidates for each block:
   1. Using a projection, $P \circ M \circ S \circ B_i$ is affine equivalent to S.
   2. Use the affine equivalence algorithm from [BCBP03] to get some candidates for B_i.
3. Identify the correct blocks :
   Use a MITM technique to filter the wrong candidates

Reducing the problem to block diagonal encodings

Decompose A in $A = B \circ \widetilde{A}$ with:

- B block diagonal affine mapping built from B_i's (unknown)
- $\widetilde{A}$ with same structure as A, built from blocks $(0_8 \ \mathrm{Id}_8) \circ E_i^{-1}$ (known)

Reducing the problem to block diagonal encodings

Decompose A in $A = B \circ \widetilde{A}$ with:

- B block diagonal affine mapping built from B_i's (unknown)
- $\widetilde{A}$ with same structure as A, built from blocks $(0_8 \ \mathrm{Id}_8) \circ E_i^{-1}$ (known)

For all $0 \leq i \leq 31$:

1. compute Ker L_i with $L_i = (A_{i,i} \ A_{i,i+1})$ (8×16 matrix)
2. get a basis $(e_1, \ldots, e_8)$ of Ker L_i
3. complete this basis $\Rightarrow E_i = (e_1 \ldots e_{16})$
4. $\exists$ B_i 8x8 invertible matrix s.t. $L_i = B_i \circ (0_8 \ \mathrm{Id}_8) \circ E_i^{-1}$

Find Ker L_i with $L_i = (A_{i,i}\ A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

Find Ker L_i with $L_i = (A_{i,i}\ A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

$$\begin{aligned}
& T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) \\
&= f_i\left[A_{i,i}(a \oplus \mathbf{x}) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y) \\
&\quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a) \oplus h_{i+1}(b \oplus y)
\end{aligned}$$

Find Ker L_i with $L_i = (A_{i,i}\ A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

$$
\begin{aligned}
& T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) \\
& = f_i\left[A_{i,i}(a \oplus \mathbf{x}) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \oplus \cancel{h_{i+1}(b \oplus y)} \\
& \quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a) \oplus \cancel{h_{i+1}(b \oplus y)}
\end{aligned}
$$

Find Ker L_i with $L_i = (A_{i,i}\ A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

$$
\begin{aligned}
& T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) \\
&= f_i\left[A_{i,i}(a \oplus \mathbf{x}) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&\quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&= f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \\
&\quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a)
\end{aligned}
$$

Find Ker L_i with $L_i = (A_{i,i} \; A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

$$\begin{aligned}
& T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) \\
&= f_i\left[A_{i,i}(a \oplus \mathbf{x}) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&\quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&= \cancel{f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right]} \oplus h_i(a \oplus x) \\
&\quad \oplus \cancel{f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right]} \oplus h_i(a)
\end{aligned}$$

Find Ker L_i with $L_i = (A_{i,i}\ A_{i,i+1})$

For any $(a, b) \in \mathbb{F}_2^8 \times \mathbb{F}_2^8$:

1. $x \in \text{Ker } A_{i,i} \Rightarrow y \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$ is constant
2. $y \in \text{Ker } A_{i,i+1} \Rightarrow x \mapsto T_i(a \oplus x, b \oplus y) \oplus T_i(a \oplus x, y)$ is constant
3. $(x, y) \in \text{Ker } L_i \Rightarrow T_i(a, b) \oplus T_i(a \oplus x, b) \oplus T_i(a, b \oplus y) \oplus T_i(a \oplus x, b \oplus y) = 0$

If $\mathbf{x} \in \text{Ker } A_{i,i}$ then :

$$
\begin{aligned}
& T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) \\
&= f_i\left[A_{i,i}(a \oplus \mathbf{x}) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a \oplus x) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&\quad \oplus f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right] \oplus h_i(a) \oplus \cancel{h_{i+1}(b \oplus y)} \\
&= \cancel{f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right]} \oplus h_i(a \oplus x) \\
&\quad \oplus \cancel{f_i\left[A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i\right]} \oplus h_i(a) \\
&= h_i(a \oplus x) \oplus h_i(a)
\end{aligned}
$$

Computing candidates for each block B_i

We decomposed A into $B \circ \widetilde{A}$ where B is a block diagonal affine mapping. Hence

$$\sum_{j=0}^{31} T_j \circ \widetilde{A}^{-1}(0, \dots, x_i, \dots, 0)$$

is a 8-bit to 256-bit mapping of the form $M_i \circ S \circ B_i$.

1. Compute a projection P_i such that $P_i \circ M_i \circ S \circ B_i$ is a bijection over $\mathbb{F}_2^8$.
2. Use Biryukov *et al.* affine equivalence algorithm to recover all possible candidates for B_i ($\approx 2^{11}$ candidates for AES Sbox).

Identifying the correct blocks

$$\left(A^{(r+1)}\right)^{-1} \quad \circ \quad \mathrm{MC} \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \quad A^{(r)}$$

Identifying the correct blocks

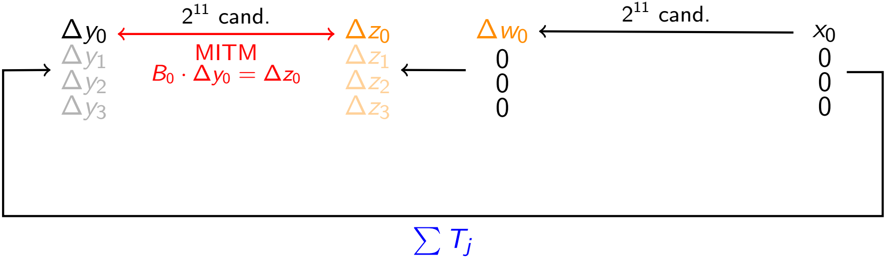
$$\widetilde{A}^{-1} \circ \begin{pmatrix} \mathbf{B}_0^{-1} & & & \\ & \mathbf{B}_1^{-1} & & \\ & & \mathbf{B}_2^{-1} & \\ & & & \mathbf{B}_3^{-1} \end{pmatrix} \circ \quad \mathrm{MC} \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \qquad A^{(r)}$$

Identifying the correct blocks

$$\widetilde{A}^{-1} \circ \begin{pmatrix} \mathbf{B}_0^{-1} & & & \\ & \mathbf{B}_1^{-1} & & \\ & & \mathbf{B}_2^{-1} & \\ & & & \mathbf{B}_3^{-1} \end{pmatrix} \circ \quad \mathrm{MC} \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \mathbf{C}_0 & & & \\ & \mathbf{C}_5 & & \\ & & \mathbf{C}_{10} & \\ & & & \mathbf{C}_{15} \end{pmatrix} \circ \widehat{A}$$

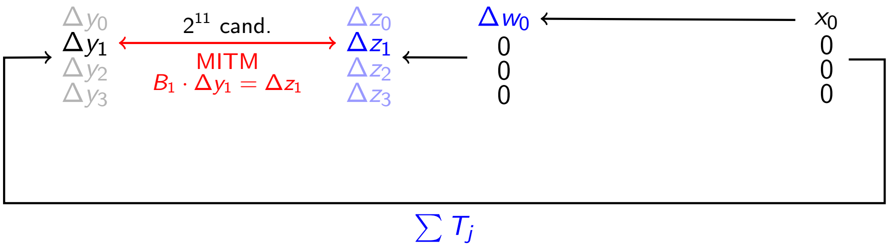
Identifying the correct blocks

$$\widetilde{A}^{-1} \circ \begin{pmatrix} \mathbf{B}_0^{-1} & & & \\ & \mathbf{B}_1^{-1} & & \\ & & \mathbf{B}_2^{-1} & \\ & & & \mathbf{B}_3^{-1} \end{pmatrix} \circ \quad \mathrm{MC} \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \mathbf{C}_0 & & & \\ & \mathbf{C}_5 & & \\ & & \mathbf{C}_{10} & \\ & & & \mathbf{C}_{15} \end{pmatrix} \circ \widehat{A}$$



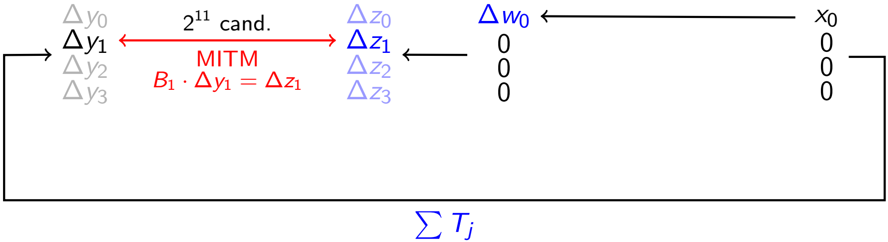
Identifying the correct blocks

$$\widetilde{A}^{-1} \circ \begin{pmatrix} \mathbf{B}_0^{-1} & & & \\ & \mathbf{B}_1^{-1} & & \\ & & \mathbf{B}_2^{-1} & \\ & & & \mathbf{B}_3^{-1} \end{pmatrix} \circ \quad \mathrm{MC} \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \mathbf{C}_0 & & & \\ & \mathbf{C}_5 & & \\ & & \mathbf{C}_{10} & \\ & & & \mathbf{C}_{15} \end{pmatrix} \circ \widehat{A}$$



Identifying the correct blocks

$$\widetilde{A}^{-1} \circ \begin{pmatrix} \mathbf{B}_0^{-1} & & & \\ & \mathbf{B}_1^{-1} & & \\ & & \mathbf{B}_2^{-1} & \\ & & & \mathbf{B}_3^{-1} \end{pmatrix} \circ \quad MC \quad \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \mathbf{C}_0 & & & \\ & \mathbf{C}_5 & & \\ & & \mathbf{C}_{10} & \\ & & & \mathbf{C}_{15} \end{pmatrix} \circ \widehat{A}$$



Knowledge of each B_i and C_i $\Rightarrow$ extract the key

Implementation (`Intel Core i7-6600U CPU @ 2.60GHz`):

- $\sim$ 2000 `C++` code lines
- Decomposition $A = B \circ \widetilde{A}$: < 1s
- Get candidates for each B_i, C_i : ~ 10s $\quad \left(64 \times \mathcal{O}\left(2^{25}\right)\right)$
- Recovering the correct B_i and C_i : < 1s
- Recovering the externals encodings : < 1s

Total time : $\sim$ 12s

Theorical time complexity : $\mathcal{O}(2^{31})$

Negligible memory

Implementation (`Intel Core i7-6600U CPU @ 2.60GHz`):

- $\sim$ 2000 `C++` code lines
- Decomposition $A = B \circ \widetilde{A}$: < 1s
- Get candidates for each B_i, C_i : ~ 10s $\quad \left(64 \times \mathcal{O}\left(2^{25}\right)\right)$
- Recovering the correct B_i and C_i : < 1s
- Recovering the externals encodings : < 1s

Total time : ~ 12s
Theorical time complexity : $\mathcal{O}(2^{31})$
Negligible memory

Fixing the construction for 60-bit security would require $n = 2^{13}$ parallel AES, leading to an implementation of size $\sim 2^{12}\,TB$

Generic Problem

Problem

Let F be an n-bit to n-bit permutation such that $F = \mathcal{B} \circ S \circ \mathcal{A}$, where:

1. $\mathcal{A}$ and $\mathcal{B}$ are n-bit affine layers;
2. $S = (S_1, \dots, S_k)$ consists of the parallel application of k permutations S_i on m bits each (called S-boxes). Note that $n = km$.

Knowing S, and given oracle access to F (but not F^{-1}), find affine $\mathcal{A}'$, $\mathcal{B}'$ such that $F = \mathcal{B}' \circ S \circ \mathcal{A}'$.

Solving this problem

$\Longleftrightarrow$

Breaking white-box implementations (of SPN) following the CEJO framework

Remarks

- **Remark 1:** F^{-1} can be built from F in 2^n operations
- **Remark 2:** *a priori* the problem has many solutions
- **Remark 3:** When S is composed of a single S-box, this is precisely the affine equivalence problem tackled by Biryukov *et al.* (with the caveat that F^{-1} is not accessible)

Overview of the algorithm

- Similar to our dedicated attack (but generic)
- **2-step algorithm:**
  1. Isolate the input and output subspaces of each Sbox
  2. Apply the generic affine equivalence algorithm by Biryukov *et al.* to each Sbox separately

Finding input subspace of each S-box

Goal

Build a subspace of dimension m of the input space, such that this subspace spans all 2^m possible values at the input of a single fixed Sbox, and yields a constant value at the input of all other Sboxes.

Idea:

1. Recover k subspaces of dimension $n - m$, each yielding a zero difference at the input of a distinct S-box
2. Pick any $k - 1$ of these spaces and compute their intersection
3. Result is a subspace of dimension m that yields a zero difference at the input of $k - 1$ Sboxes, and spans all values at the input of the remaining Sbox.

Finding input subspace of each S-box

New goal

Build a subspace of dimension $n - m$ of the input space that yields a zero difference at the input of one Sbox.

1. Pick uniformly at random an input difference Δ
2. With probability 2^{-m}, Δ yields a zero difference at the input of a particular Sbox.
3. Check that the set of output differences generated by input difference Δ spans a subspace of dimension $n - m$.
4. Repeat this process few times to find $n - m$ independent difference Δ.

Recovering affine layers

1. From previous step, we know $\mathcal{A}'$ such that:

$$F \circ \mathcal{A}'^{-1} = \left(\cdots \middle| B_i \middle| \cdots \right) \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \ddots & & \\ & D_i & \\ & & \ddots \end{pmatrix}$$

Recovering affine layers

1. From previous step, we know $\mathcal{A}'$ such that:

$$F \circ \mathcal{A}'^{-1} = \left(\cdots \middle| B_i \middle| \cdots \right) \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \ddots & & \\ & D_i & \\ & & \ddots \end{pmatrix}$$

2. Compose with projections and run affine equivalence algorithm to recover D_i's

Recovering affine layers

1. From previous step, we know $\mathcal{A}'$ such that:

$$F \circ \mathcal{A}'^{-1} \circ \begin{pmatrix} \ddots & & \\ & D_i^{-1} & \\ & & \ddots \end{pmatrix} \circ \begin{bmatrix} S^{-1} \\ \vdots \\ S^{-1} \end{bmatrix} = \left(\cdots \middle| B_i \middle| \cdots \right)$$

2. Compose with projections and run affine equivalence algorithm to recover D_i's
3. Retrieve B_i's

Complexities

Complexity of solving the problem:

- Biryukov *et al.*: $\mathcal{O}(n^3 2^{2n})$
- Baek *et al.*: $\mathcal{O}(2^n + n^4 2^{3m}/m)$
- Our (identical Sboxes): $\mathcal{O}\left(2^m n^3 + 2^m l n^3 + \frac{n^4}{m} + 2^{2m} m^2 n\right)$
- Our (different Sboxes): $\mathcal{O}\left(2^m n^3 + 2^m l n^3 + \frac{n^4}{m} + 2^{2m} m n^2\right)$

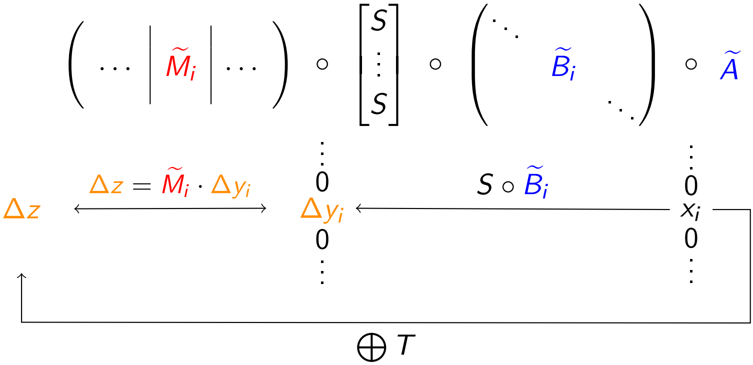
Application to Baek et al. proposal:

- generic attack: $\mathcal{O}\left(2^{35}\right)$ (allows to decrypt but do not recover the key)
- dedicated attack: $\mathcal{O}\left(2^{31}\right)$ (recover the key)

Thank you for your attention!

1-round attack

From $M \circ (S, \ldots, S) \circ B \circ \widetilde{A}$,
give an equivalent representation $\widetilde{M} \circ (S, \ldots, S) \circ \widetilde{B} \circ \widetilde{A}$



Get the external encodings from the key

Suppose that we know the key

Remains externals encodings :

$$M_{out} \circ (\text{AES}, \text{AES}) \circ M_{in}$$

Get the external encodings from the key

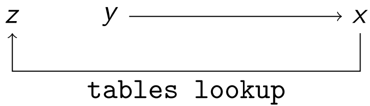
Suppose that we know the key and $A^{(1)}$

Remains externals encodings :

$$M_{out} \circ (\text{AES}, \text{AES}) \circ A^{(1)} \circ \widetilde{M}_{in}$$

$\widetilde{M}_{in}$ is known, built as $\widetilde{M}_{in} = \left(A^{(1)}\right)^{-1} \circ M_{in} \Rightarrow$ extract M_{in}

$$M_{out} \circ (\text{AES}, \text{AES}) \circ A^{(1)}$$



Use 256+1 values of y to recover M_{out}