Yet another attack on whitebox AES implementation

Patrick Derbez¹, Pierre-Alain Fouque¹, Baptiste Lambin¹, Brice Minaud²

¹Univ Rennes, CNRS, IRISA
²Royal Holloway University of London
1 Introduction

2 The Baek, Cheon and Hong proposal

3 Dedicated Attack

4 Generic attack
Introduction

The Baek, Cheon and Hong proposal

Dedicated Attack

Generic attack
Black box vs. White box

Black box model

\[ \text{in} \rightarrow \text{AES}_K \rightarrow \text{out} \]
Introduction

Black box vs. White box

Black box model

\[ \text{in} \rightarrow \text{AES}_K \rightarrow \text{out} \]

White box model

\[
\begin{align*}
\text{key} &= 0x1337...
\text{key}_\text{schedule}(\text{key}) \\
\text{out} &= \text{in}
\text{for } i \text{ in } 0...10 \\
&\quad \text{round}_i(\text{out}, \text{key}) \\
&\quad \text{return out}
\end{align*}
\]

\[ \text{in} \rightarrow \text{out} \]
Introduction

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)

\[
\cdots \circ f(r+1)^{-1} \circ E(r) \circ f(r) \circ f(r)^{-1} \circ E(r-1) \circ f(r-1) \circ \cdots
\]

- Increase security with external encodings
**Introduction**

**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**
1 Introduction

2 The Baek, Cheon and Hong proposal

3 Dedicated Attack

4 Generic attack
The Baek, Cheon and Hong proposal

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

\[
\begin{align*}
256\text{-bit} & \quad A^{(r)} \\
\text{AES}^{(r)} & \quad \text{AES}^{(r)} \\
(A^{(r+1)})^{-1} & \\
256\text{-bit} &
\end{align*}
\quad \implies \quad
256\text{-bit} & \quad A^{(r)} \\
\text{MC} \circ \text{SR} & \quad \text{MC} \circ \text{SR} \\
(S \ldots S) & \quad (A^{(r+1)})^{-1} \\
\text{table} & \quad \text{table} \\
256\text{-bit} & \quad \text{table} \\
\end{align*}
\]
Sparse input encoding

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

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

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

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

Memory requirement

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

≈ 160MB
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 \( \mathcal{O}(2^{25}) \) using the affine equivalence algorithm from Biryukov et al.
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 $A$ and $B$ such that $S_2 = B \circ S_1 \circ A$, if they exist.

- Ascertain whether such mappings exist
- Enumerate all solutions
- Time complexity in $O(n^32^n)$
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 $A$ and $B$ such that $S_2 = B \circ S_1 \circ A$, if they exist.

- Ascertain whether such mappings exist
- Enumerate all solutions
- Time complexity in $O(n^3 2^n)$
- Time complexity for linear version in $O(n^3 2^n)$
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
Introduction

The Baek, Cheon and Hong proposal

Dedicated Attack

Generic attack
Overview of the attack

From encoded round functions $F \simeq M \circ S \circ A$ with $A \simeq \left( \begin{array}{ccc} * & * & * \\ * & \ddots & * \\ * & * & * \end{array} \right)$

1. Reduce the problem to block diagonal encodings:
   \[ \tilde{F} = M \circ S \circ B \] 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 \( A = B \circ \tilde{A} \) with:

- \( B \) block diagonal affine mapping built from \( B_i \)'s (unknown)
- \( \tilde{A} \) with same structure as \( A \), built from blocks \((0_8 \text{ Id}_8) \circ E^{-1}_i\) (known)
Reducing the problem to block diagonal encodings

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

- $B$ block diagonal affine mapping built from $B_i$’s (unknown)
- $\tilde{A}$ with same structure as $A$, built from blocks $(0_8 \text{ Id}_8) \circ E^{-1}_i$ (known)

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

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

$$T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$$

$$= f_i [A_{i,i}(a \oplus x) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y)$$

$$\oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a) \oplus h_{i+1}(b \oplus y)$$
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 $x \in \text{Ker } A_{i,i}$ then:

$$T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$$

$$= f_i [A_{i,i}(a \oplus x) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y)$$

$$\oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a) \oplus h_{i+1}(b \oplus y)$$
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 $x \in \text{Ker } A_{i,i}$ then:

$$T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$$

$$= f_i [A_{i,i}(a \oplus x) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y)$$

$$\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a) \oplus h_{i+1}(b \oplus y)$$

$$= f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x)$$

$$\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a)$$
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 $x \in \text{Ker } A_{i,i}$ then:

\[
T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y) = \\
= f_i [A_{i,i}(a \oplus x) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y) \\
\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a) \oplus h_{i+1}(b \oplus y) \\
= f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \\
\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a)
\]
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 $x \in \text{Ker } A_{i,i}$ then:

$$T_i(a \oplus x, b \oplus y) \oplus T_i(a, b \oplus y)$$

$$= f_i [A_{i,i}(a \oplus x) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x) \oplus h_{i+1}(b \oplus y)$$

$$\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a) \oplus h_{i+1}(b \oplus y)$$

$$= f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a \oplus x)$$

$$\quad \oplus f_i [A_{i,i}(a) \oplus A_{i,i+1}(b \oplus y) \oplus c_i] \oplus h_i(a)$$

$$= h_i(a \oplus x) \oplus h_i(a)$$
Computing candidates for each block $B_i$

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

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

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

\[ (A^{(r+1)})^{-1} \circ MC \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ A^{(r)} \]
Identifying the correct blocks

\[
\tilde{A}^{-1} \circ \begin{pmatrix}
B_0^{-1} \\
B_1^{-1} \\
B_2^{-1} \\
B_3^{-1}
\end{pmatrix} \circ \text{MC} \circ \begin{bmatrix}
S \\
\vdots \\
S
\end{bmatrix} \circ A^{(r)}
\]
Identifying the correct blocks

\[
\tilde{A}^{-1} \circ \begin{pmatrix}
B_0^{-1} & \quad & B_1^{-1} \\
B_2^{-1} & \quad & B_3^{-1}
\end{pmatrix} \circ MC \circ \begin{bmatrix}
S \\
S
\end{bmatrix} \circ \begin{pmatrix}
C_0 \\
C_5 \\
C_{10} \\
C_{15}
\end{pmatrix} \circ \hat{A}
\]
Identifying the correct blocks

\[
\tilde{A}^{-1} \circ \begin{pmatrix} B_0^{-1} & \delta_1 & \ldots & \delta_{11} \\
\end{pmatrix} \circ \begin{pmatrix} S \\
\end{pmatrix} \circ \begin{pmatrix} C_0 & C_5 & C_{10} & C_{15} \\
\end{pmatrix} \circ \hat{A}
\]

Knowledge of each \( B_i \) and \( C_i \) \( \Rightarrow \) extract the key

\[
\Delta y_0 \quad \Delta y_1 \quad \Delta y_2 \quad \Delta y_3
\]

\( 2^{11} \) cand.

\[
\begin{align*}
\Delta z_0 \quad \Delta z_1 \quad \Delta z_2 \quad \Delta z_3
\end{align*}
\]

MITM

\( B_0 \cdot \Delta y_0 = \Delta z_0 \)

\[
\begin{align*}
\Delta w_0 \\
\end{align*}
\]

\( 2^{11} \) cand.

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
\end{array}
\]

\( x_0 \)

\[
\sum T_j
\]
Identifying the correct blocks

\[ \tilde{A}^{-1} \circ \begin{pmatrix} B_0^{-1} & B_1^{-1} & B_2^{-1} & B_3^{-1} \\ \end{pmatrix} \circ \text{MC} \circ \begin{pmatrix} S \\ S \end{pmatrix} \circ \begin{pmatrix} C_0 & C_5 & C_{10} & C_{15} \end{pmatrix} \circ \hat{A} \]

\[ \begin{align*}
\Delta y_0 & \quad 2^{11} \text{ cand.} \\
\Delta y_1 & \quad \text{MITM} \\
\Delta y_2 & \\
\Delta y_3 & \\
\Delta z_0 & \\
\Delta z_1 & \quad B_1 \cdot \Delta y_1 = \Delta z_1 \\
\Delta z_2 & \\
\Delta z_3 & \\
\Delta w_0 & \quad 0 \\
\Delta w_1 & \quad 0 \\
\Delta w_2 & \quad 0 \\
\Delta w_3 & \quad 0 \\
\sum T_j & \\
\end{align*} \]

\[ x_0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]
Identifying the correct blocks

\[
\tilde{A}^{-1} \circ \begin{pmatrix} B_0^{-1} & B_1^{-1} & B_2^{-1} & B_3^{-1} \end{pmatrix} \circ \text{MC} \circ \begin{bmatrix} S \\ S \end{bmatrix} \circ \begin{pmatrix} C_0 & C_5 & C_{10} & C_{15} \end{pmatrix} \circ \hat{A}
\]

\[
\begin{align*}
\Delta y_0 & \quad \Delta y_1 & \quad \Delta y_2 & \quad \Delta y_3 \\
\Delta z_0 & \quad \Delta z_1 & \quad \Delta z_2 & \quad \Delta z_3 \\
\Delta w_0 & \quad \Delta w_1 & \quad \Delta w_2 & \quad \Delta w_3 \\
x_0 & \quad 0 & \quad 0 & \quad 0
\end{align*}
\]

\[\sum T_j\]

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 $A = B \circ \tilde{A} : < 1s$
- Get candidates for each $B_i, C_i : \sim 10s \ (64 \times O\left(2^{25}\right))$
- Recovering the correct $B_i$ and $C_i : < 1s$
- Recovering the externals encodings : $< 1s$

**Total time : $\sim 12s$**

*Theoretical time complexity : $O(2^{31})$*

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

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

**Total time : \( \sim 12s \)**

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

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
1 Introduction

2 The Baek, Cheon and Hong proposal

3 Dedicated Attack

4 Generic attack
Let $F$ be an $n$-bit to $n$-bit permutation such that $F = B \circ S \circ A$, where:

1. $A$ and $B$ are $n$-bit affine layers;
2. $S = (S_1, \ldots, S_k)$ 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 $A'$, $B'$ such that $F = B' \circ S \circ A'$.

Solving this problem

$\iff$

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 S-box, 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 $\Delta$
2. With probability $2^{-m}$, $\Delta$ yields a zero difference at the input of a particular Sbox.
3. Check that the set of output differences generated by input difference $\Delta$ spans a subspace of dimension $n - m$.
4. Repeat this process few times to find $n - m$ independent difference $\Delta$. 
From previous step, we know $A'$ such that:

$$F \circ A'^{-1} = \left( \begin{array}{c|c|c} \cdots & B_i & \cdots \end{array} \right) \circ \begin{bmatrix} S \\ \vdots \\ S \end{bmatrix} \circ \begin{pmatrix} \cdots & D_i & \cdots \end{pmatrix}$$
1. From previous step, we know $A'$ such that:

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

2. Compose with projections and run affine equivalence algorithm to recover $D_i$’s
Recovering affine layers

1. From previous step, we know $A'$ such that:

$$F \circ A'^{-1} \circ \begin{pmatrix} \ddots & D_i^{-1} & \ddots \\ \vdots & \vdots & \ddots \\ \end{pmatrix} \circ \begin{bmatrix} S^{-1} \\ \vdots \\ S^{-1} \end{bmatrix} = \begin{pmatrix} \ddots & B_i & \ddots \end{pmatrix}$$

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

3. Retrieve $B_i$'s
Complexity of solving the problem:

- Biryukov et al.: $O(n^32^{2n})$
- Baek et al.: $O(2^n + n^42^{3m}/m)$
- Our (identical Sboxes): $O\left(2^m n^3 + 2^m \ln^3 + \frac{n^4}{m} + 2^{2m} m^2 n\right)$
- Our (different Sboxes): $O\left(2^m n^3 + 2^m \ln^3 + \frac{n^4}{m} + 2^{2m} mn^2\right)$

Application to Baek et al. proposal:

- generic attack: $O\left(2^{35}\right)$ (allows to decrypt but do not recover the key)
- dedicated attack: $O\left(2^{31}\right)$ (recover the key)
Thank you for your attention!
1-round attack

From $\tilde{M} \circ (S, \ldots, S) \circ \tilde{B} \circ \tilde{A}$,
give an equivalent representation $\tilde{M} \circ (S, \ldots, S) \circ \tilde{B} \circ \tilde{A}$
Suppose that we know the key
Remains external encodings:

\[ M_{\text{out}} \circ (\text{AES}, \text{AES}) \circ M_{\text{in}} \]
Suppose that we know the key and $A^{(1)}$
Remains externals encodings:

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

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

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

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