# Security of the AES with a Secret S-box

**Tyge Tiessen** Lars R. Knudsen Stefan Kölbl
Martin M. Lauridsen

DTU Compute
Technical University of Denmark

22nd International Workshop on Fast Software Encryption, 2015

## Why bother looking at secret S-boxes?

Potential reasons for using a secret S-box in AES

- Increase size of the secret (128–256 bits $\rightarrow$ 1812–1940 bits)
- Legal obligation to use "secret" cipher but lack of resources to develop dedicated one

Why else might we want to cryptanalyze this (apart from the pure joy of cryptanalysis)?
We might gain

- Insight into the structural security of AES
- Potential applications in whitebox cryptography and SCARE (side-channel reverse engineering)

DTU

## The cryptanalytic scenario

### The Target

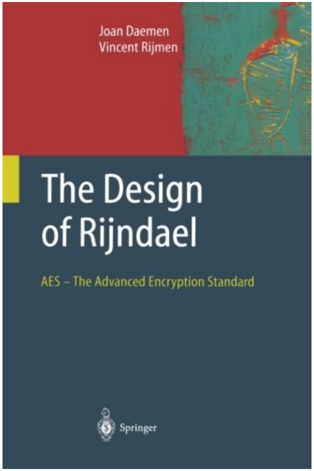
The Advanced Encryption Standard (AES) where the standard (Rijndael) S-box has been substituted everywhere it appears with a randomly chosen S-box about which the adversary has no knowledge.

### The Goal

Retrieve both the S-box and the key. The goal is thus not to just find a decryption algorithm.

## Know your AES

I assume you all know that $\rightarrow$ by heart.



## Differential and linear cryptanalysis

- A random 8-bit S-box is already very likely to have low maximum differential probability and maximum square correlation. [O'C95][O'C94]
- Additional filtering can guarantee good differential and linear probabilities.

$\Rightarrow$ Due to the strong diffusion of AES, good differential and linear attacks remain unlikely even with a random S-box. (How to find good differentials or linear hulls is another question by itself.)

$\Rightarrow$ Integral cryptanalysis seems to be our best shot.

## Integral attacks

**Idea**

Instead of looking at single plaintexts or pairs of plaintexts, look at the properties of a whole set of plaintexts as it propagates through a cipher.

- original attack is Square attack by Knudsen
- generalized by Lucks to saturation attacks
- and by Shamir and Biryukov to SASAS structures
- can break 4-6 rounds of AES-128
- can be viewed as a clever way of calculating higher-order differentials

## The boring notations and definitions slide

**Definition**

A $\Lambda$-set is a set of 256 messages that differs only in one byte but takes for this byte all possible 256 values.

Properties of sets of 256 bytes, as used in the Square attack

| | | |
|---|---|---|
| $\mathcal{P}$ | : | each possible value appears once |
| $\mathcal{B}$ | : | all values sum up to zero |
| $\cdot$ | : | all bytes are the same value |
| ? | : | no clue |

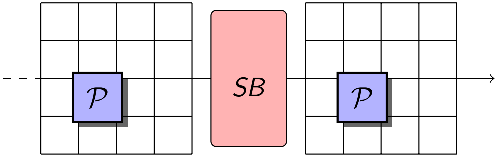
To save me and you the pain, I will say:

"Rijndael field" for $\mathbb{F}_{256}$

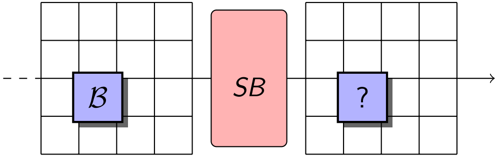
"The vector space" for $\mathbb{F}_2^8$

# Effect of the SubBytes operation on multisets

Effect in $\mathcal{P}$ sets

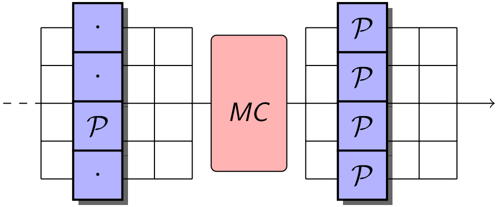


Effect on $\mathcal{B}$ sets

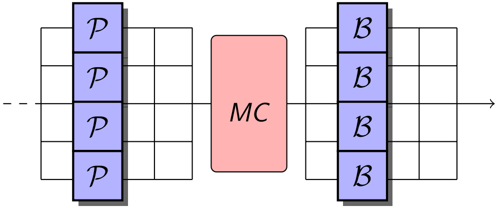


# Effect of the MixColumns operation on multisets

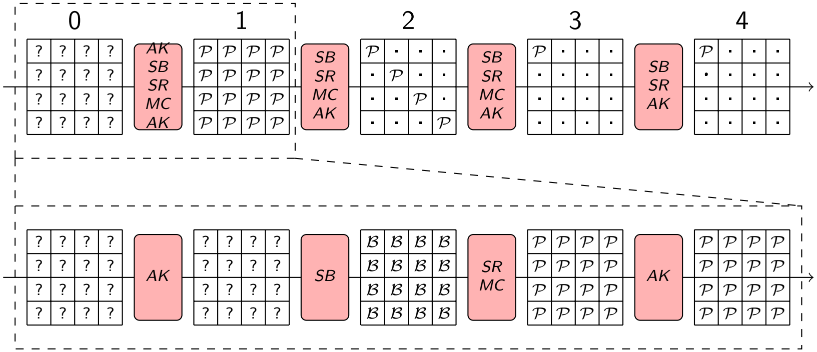
Effect on a column with 3 bytes constant, one byte $\mathcal{P}$



Effect on a column with all bytes $\mathcal{P}$



# The inverted Square attack on four rounds



## Attacking four rounds with the SASAS attack

Looking into the attack on SASAS, we find a solution:

Generate balanced sets after the first S-box layer

$\rightarrow$ Corresponds to a linear equation for the S-box

$\rightarrow$ Create system of linear equations to find S-box

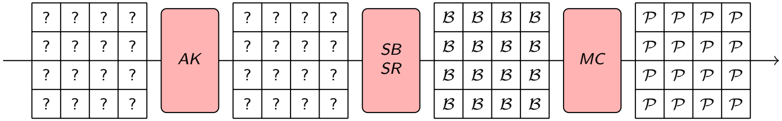
**Problem**

This can only determine the S-box up to affine equivalence over $\mathbb{F}_2^8$
$\rightarrow$ $2^{72}$ candidates

Can we continue with the SASAS attack?
Not if we want to recover the key and the S-box.

## What do we do with the box now?

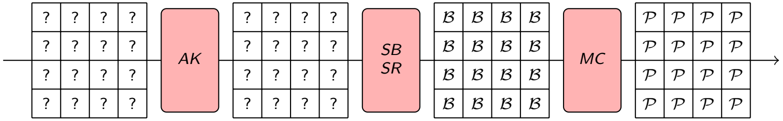
## Picking up where the SASAS attack leaves us



**Idea**

Let us use the fact that a set of texts has the $\mathcal{P}$ property in every byte after the MixColumns operation to filter out wrong S-box candidates.

## Steps of the attack



- Find one S-box (out of the $2^{72}$ options) for the first byte (assume the whitening key byte is zero)
- Determine the remaining key bytes just as in the Square attack
- Determine the intermediate texts after the ShiftRows operation up to affine equivalence over $\mathbb{F}_2^8$.
- Now find an affine transformation that assures the $\mathcal{P}$ property after the MixColumns operation
- We have then determined the S-box up to affine equivalence over $\mathbb{F}_{256}$ ($2^{16}$ remaining candidates)

## Affine transformations over $\mathbb{F}_{256}$ commute with the MixColumns matrix

Applying an invertible affine transformation over $\mathbb{F}_{256}$ to a byte vector before multiplication with the MixColumns matrix is the same as applying the transformation on the resulting vector:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} av_0 + b \\ av_1 + b \\ av_2 + b \\ av_3 + b \end{pmatrix}$$

$$= a \cdot \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} b \\ b \\ b \\ b \end{pmatrix}$$

## Affine transformations over $\mathbb{F}_2^8$ generally do not commute with the MixColumns matrix

Let $A$ be an affine transformation over $\mathbb{F}_2^8$. With

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}, \quad B = \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & A & 0 & 0 \\ 0 & 0 & A & 0 \\ 0 & 0 & 0 & A \end{pmatrix}$$

we generally have

$$MB \neq BM.$$

This is because linear mappings over $\mathbb{F}_{256}$ generally do not commute with linear mappings over $\mathbb{F}_2^8$ that are not linear over $\mathbb{F}_{256}$.

Can we prove this? Yes!

## General affine transformation do not commute with field multiplication

For $a \in \mathbb{F}_{256}$ let $L_a$ denote the $8 \times 8$ $\mathbb{F}_2$-matrix that corresponds to multiplication with $a$: $a \cdot b = L_a b$.

**Lemma**

*Let $g$ be primitive in $\mathbb{F}_{256}$. Let $B$ be an $8 \times 8$ matrix over $\mathbb{F}_2$ which commutes with $L_g$. Then there exists $b \in \mathbb{F}_{256}$ such that $L_b = B$.*

**Proof.**

Let $c \in \mathbb{F}_{256}^*$. As $g$ primitive, $c = g^k$ and $L_c = L_g^k$ for some $k$. By induction $B$ commutes with $L_c$. Thus $B$ commutes with all of $\mathbb{F}_{256}$. Let $b = B1$. We then have for any $c \in \mathbb{F}_{256}^*$:

$$Bc = L_c L_{c^{-1}} Bc = L_c B L_{c^{-1}} c = L_c B1 = L_c b = L_b c.$$

As this is true for any $c \in \mathbb{F}_{256}^*$ and for 0, we have $B = L_b$. □

## How to improve the efficiency of finding the affine equivalent

Using the $\mathcal{P}$ property, we still have to test $2^{56}$ affine mappings ($2^{72}$ affine mappings modulo affine equivalence over $\mathbb{F}_{256}$). This can still be improved:

**$\mathcal{R}$ property**

We say that a set of bytes has the $\mathcal{R}$ property if in each bit position the values 1 and 0 appear an equal number of times.

This allows us to reconstruct a correct affine mapping part by part, reducing the overall complexity. Note that $\mathcal{P} \Rightarrow \mathcal{R} \Rightarrow \mathcal{B}$.

## How to improve the efficiency of finding the affine equivalent

Let us take a closer look at the specific form of matrix $M$. When written as a linear function from $F_{256}^4$ to $F_{256}^4$, it has the form

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}.$$

If we associate the multiplication with 01, 02, and 03 with their respective linear mappings from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$, we get the following representations:

$$01 = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix} \quad 02 = \begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&0&1&0&0&0 \\ 1&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix} \quad 03 = \begin{pmatrix} 1&1&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 1&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 1&0&0&0&0&0&1&1 \\ 1&0&0&0&0&0&0&1 \end{pmatrix}$$

## How to improve the efficiency of finding the affine equivalent

First row of $M$ in binary notation:

$$\begin{pmatrix} 0\,\mathbf{1}\,0\,0\,0\,0\,0\,0 & \mathbf{1}\,\mathbf{1}\,0\,0\,0\,0\,0\,0 & \mathbf{1}\,0\,0\,0\,0\,0\,0\,0 & \mathbf{1}\,0\,0\,0\,0\,0\,0\,0 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

If we now write $a_0, a_1, \ldots, a_7$ for the rows of $A$ we can write the first row of

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & A & 0 & 0 \\ 0 & 0 & A & 0 \\ 0 & 0 & 0 & A \end{pmatrix}$$

as

$$(a_1, \quad a_0 \oplus a_1, \quad a_0, \quad a_0)\,.$$

## How to improve the efficiency of finding the affine equivalent

- Because we reduce $A$ to affine equivalence over $\mathbb{F}_{256}$, we can fix one row.
- Thus we can fix $a_0$ and only need to try all options for $a_1$.
- Thus we need to test only $2^8$ values at once, compared to $2^{56}$ before.
- Interestingly, when using a chosen-plaintext attack and working with the inverse MixColumns matrix, the equation involves four rows of $A$ increasing the complexity of this step by $2^{16}$.

## What are the complexities of the attack?

Complexities given in encryption equivalents, plaintexts/ ciphertexts, and bytes respectively.

| Cipher | Rnds | Time | Data | Mem | Reference |
|---|---|---|---|---|---|
| SASAS | 3 | $2^{21}$ | $2^{16}$ | $2^{20}$ | [BS01] |
| AES-128 sec. S-box | 4 | $2^{17}$ | $2^{16}$ | $2^{16}$ | *This work* |
| AES-128 | 4 | $2^{14}$ | $2^{9}$ | – | [DR02] |
| AES-128 sec. S-box | 5 | $2^{38}$ | $2^{40}$ | $2^{40}$ | *This work* |
| AES-128 | 5 | $2^{38}$ | $2^{33}$ | – | [DR02] |
| AES-128 sec. S-box | 6 | $2^{90}$ | $2^{64}$ | $2^{69}$ | *This work* |
| AES-128 | 6 | $2^{44}$ | $2^{34}$ | $2^{36}$ | [FKL+00] |

Implemented attack for four rounds runs in less than one second.

## Conclusions

- Gain in security by using a secret S-box is low for up to six rounds
- Using the $\mathcal{R}$ property instead of the $\mathcal{P}$ property can significantly reduce the complexity of an attack
- Example of where complexity of attack depends on the direction (encryption/decryption)

- Open problems
  - What if all S-boxes are different (and still secret)?
    $\rightarrow$ Closer to SASAS attack
  - What about more than 6 rounds?

## Conclusions

- Gain in security by using a secret S-box is low for up to six rounds
- Using the $\mathcal{R}$ property instead of the $\mathcal{P}$ property can significantly reduce the complexity of an attack
- Example of where complexity of attack depends on the direction (encryption/decryption)

- Open problems
  - What if all S-boxes are different (and still secret)?
    $\rightarrow$ Closer to SASAS attack
  - What about more than 6 rounds?

**Questions?**

## References I

Alex Biryukov and Adi Shamir.
Structural cryptanalysis of SASAS.
In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 394–405, 2001.

Joan Daemen and Vincent Rijmen.
*The Design of Rijndael: AES - The Advanced Encryption Standard*.
Information Security and Cryptography. Springer, 2002.

Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting.
Improved cryptanalysis of Rijndael.
In Bruce Schneier, editor, *Fast Software Encryption, FSE 2000*, volume 1978 of *LNCS*, pages 213–230, 2000.

## References II

Luke O'Connor.
On the Distribution of Characteristics in Bijective Mappings.
In Tor Helleseth, editor, *EUROCRYPT '93*, volume 765 of *LNCS*, pages 360–370. Springer, 1994.

Luke O'Connor.
Properties of linear approximation tables.
In Bart Preneel, editor, *Fast Software Encryption, FSE '94*, volume 1008 of *LNCS*, pages 131–136. Springer, 1995.