On Recovering Affine Encodings in White-Box Implementations

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

[1]Univ Rennes, CNRS, IRISA

[2]Royal Holloway University of London

1. Introduction

2. Generic algorithm

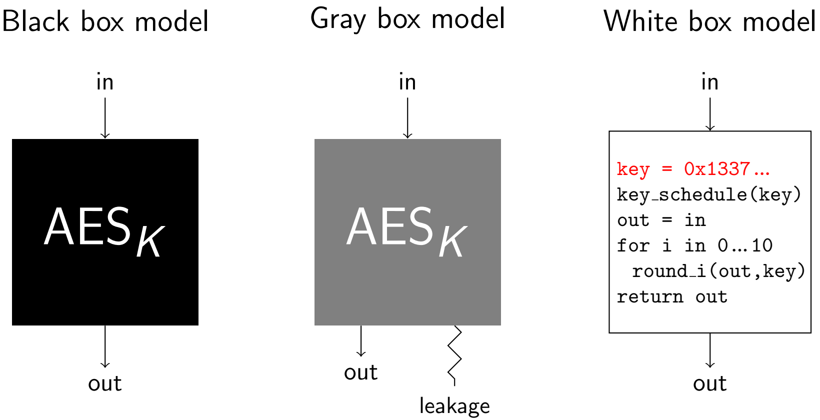
3. Dedicated attack on Baek et al.'s scheme

1. Introduction

2. Generic algorithm

3. Dedicated attack on Baek et al.'s scheme

Black box vs. White box



```
key = 0x1337...
key_schedule(key)
out = in
for i in 0...10
 round_i(out,key)
return out
```

White box implementation

Attacker:

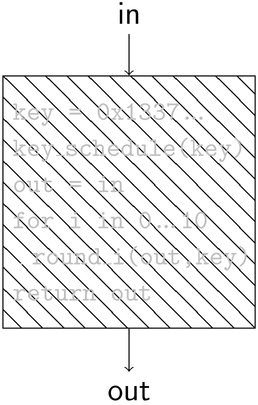
- extracting key information from the implementation
- computing decryption scheme from encryption scheme

Designer:

- provide sound and secure implementation

Main application:

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



```
key = 0x1337...
key_schedule(key)
out = in
for i in 0...10
  round_i(out,key)
return out
```

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 and Shamir
  - 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
- The affine and non-linear part of all $f^{(r)}$ is often structured for efficient implementations !

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 $\mathcal{O}\left(n^3 2^{2n}\right)$, $\mathcal{O}\left(n^3 2^n\right)$ if $\mathcal{A}, \mathcal{B}$ linears

Improved by Dinur at Eurocrypt'18 to $\mathcal{O}\left(n^3 2^n\right)$ in the affine case, but with a few limitations

1. Introduction
2. Generic algorithm
3. Dedicated attack on Baek et al.'s scheme

Problem to solve for the attacker

$$\text{Given}\quad \underset{\textit{known}}{F} \quad = \quad \underset{\textit{secret}}{\overset{\textit{affine}}{\mathcal{B}}} \quad \circ \quad \underset{\textit{known}}{\overset{\textit{non-linear}}{\begin{bmatrix} S_1 \\ \vdots \\ S_k \end{bmatrix}}} \quad \circ \quad \underset{\textit{secret}}{\overset{\textit{affine}}{\mathcal{A}}} \quad \text{without knowing } F^{-1}$$

- Find an equivalent representation $\tilde{F}$ of F such that $\tilde{F}^{-1}$ is easily computable (leads to a decryption function).

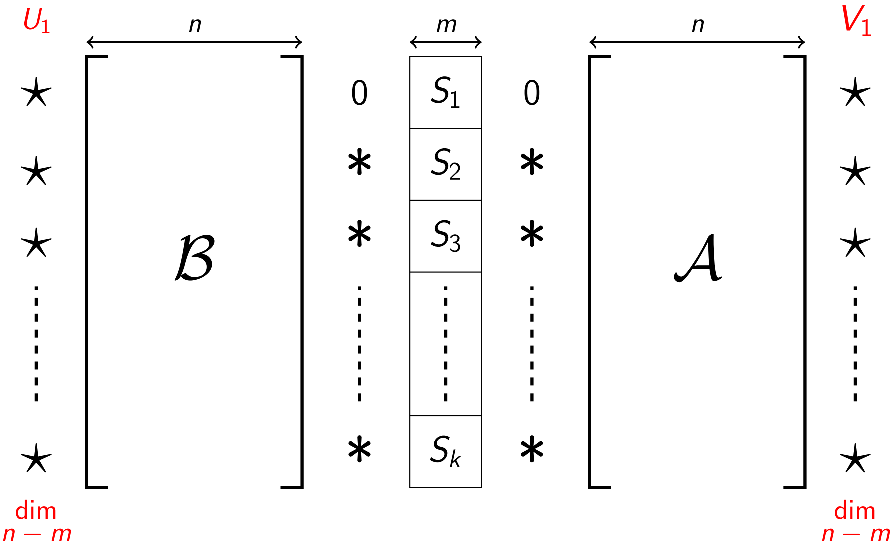
- Find which $\mathcal{A}$ and $\mathcal{B}$ were used (leads to a key recovery).

Overview of the algorithm

2-step algorithm:

1. Isolate the input and output subspaces of each Sbox (essentially the technique from Biryukov and Shamir in their SASAS cryptanalysis)
2. Apply the generic affine equivalence algorithm to each Sbox separately

Finding input subspace of each S-box



Building V_1

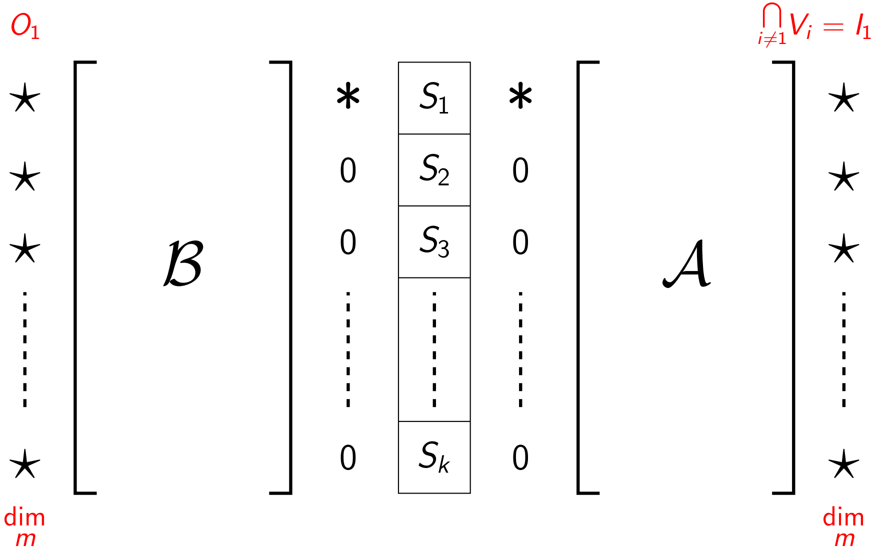
Testing if $\Delta \in V_1$:

- $X = \{x_i \in \mathbb{F}_2^n, x_i \text{ random}\}$ "big enough"
- $U = \{F(x_i) \oplus F(x_i \oplus \Delta), x_i \in X\}$ (output difference space)
- If $\dim(\mathrm{Span}(U)) = n - m$, then $\Delta \in V_1$ w.h.p.

Build a basis of V_1 by doing the same test on independent vectors, and by testing if the resulting output difference space is the same.

Do this k times to build all $V_1, \ldots, V_k$.

Finding input subspace of each S-box



Recovering affine layers

$$\begin{tikzcd}
\mathbb{F}_2^m & O_i \arrow[l, "\mathcal{Q}_i"'] & I_i \arrow[l, "\mathcal{B} \circ \begin{bmatrix} S_1 \\ \vdots \\ S_k \end{bmatrix} \circ \mathcal{A}"'] & \mathbb{F}_2^m \arrow[l, "\mathcal{P}_i"'] \\
 & \begin{array}{c}\text{dim}\\ m\end{array} & \begin{array}{c}\text{dim}\\ m\end{array} &
\end{tikzcd}$$

- Apply the Affine Equivalence Algorithm on each $F_i = \mathcal{Q}_i \circ F \circ \mathcal{P}_i$
- Lead to 2 affine mappings $\mathcal{A}_i, \mathcal{B}_i$ such that $F_i = \mathcal{B}_i \circ S_i \circ \mathcal{A}_i$
- Build $\mathcal{A}'$ from all $\mathcal{A}_i$'s and $\mathcal{P}_i$'s, $\mathcal{B}'$ from all $\mathcal{B}_i$'s and $\mathcal{Q}_i$'s such that $\mathcal{B}' \circ (S_1, \dots, S_k) \circ \mathcal{A}' = F$

We can now inverse F easily as $F^{-1} = \mathcal{A}'^{-1} \circ \left(S_1^{-1}, \dots, S_k^{-1}\right) \circ \mathcal{B}'^{-1}$!

Complexities

Complexity of solving the problem:

- Biryukov *et al.*: $\mathcal{O}(n^3 2^{2n})$, Dinur : $\mathcal{O}(n^3 2^n)$
- Baek *et al.*: $\mathcal{O}\left(\min(n^{m+4} 2^{2m}/m, n\log(n) 2^{n/2})\right)$
- Our (best case): $\mathcal{O}\left(2^m n^3 + \frac{n^4}{m} + 2^m m^2 n\right)$
- Our (different Sboxes): $\mathcal{O}\left(2^m n^3 + \frac{n^4}{m} + 2^m mn^2\right)$
- Our (worst case, e.g. AES S-box): $\mathcal{O}\left(2^m n^3 + \frac{n^4}{m} + 2^{2m} m^2 n\right)$

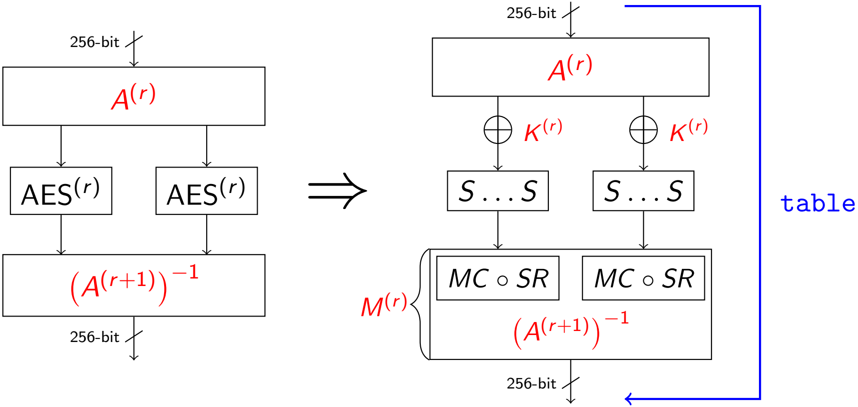
Applications:

- 128-bit block cipher, AES S-box (8 bits) : $\sim 2^{30}$ operations
- Baek *et al.* proposal (256-bit block, AES S-box) : $\sim 2^{35}$ operations

1. Introduction
2. Generic algorithm
3. Dedicated attack on Baek et al.'s scheme

The Baek, Cheon and Hong proposal

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



Security claim : 110 bits

Overview of the attack

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

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

See our paper for more details !

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

- $\sim$ 2000 `C++` code lines
- Main cost : 64 calls to the affine equivalence algorithm ($\sim 64 \times 2^{25}$)
- Generic algorithm complexity : $\sim 2^{35}$ (Decryption function)
- Dedicated attack complexity : $\sim 2^{31}$ (Key-recovery)
- Total time : $\sim$ 12s, negligible memory

Implementation available at `http://wbcheon.gforge.inria.fr/`.

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

Conclusion

- Given $F = \mathcal{B} \circ (S_1, \ldots, S_k) \circ \mathcal{A}$, with $\mathcal{A}$ and $\mathcal{B}$ secret, we provide a generic algorithm to efficiently compute F^{-1}.
  This *efficiently* solve a critical step when attacking table-based white box implementations.
- Best case complexity : $\mathcal{O}\left(2^m n^3 + \frac{n^4}{m} + 2^m m^2 n\right)$
  In practice with AES parameters : $\sim 2^{30}$
  Scale linearly if S-boxes are different
- We mounted a dedicated attack on Baek *et al.*'s scheme, leading to a key recovery in about 2^{31} operations.