A new representation of the AES Key Schedule
Application to mixFeed, ALE, and AES

Gaëtan Leurent, Clara Pernot
Inria, Paris

Friday, April 16th 2021

Table of contents

Table of contents

Introduction

National Institute of Standards and Technology (NIST) initiated processes to solicit, evaluate, and standardize cryptographic algorithms:

Introduction

National Institute of Standards and Technology (NIST) initiated processes to solicit, evaluate, and standardize cryptographic algorithms:

- 1997 - 2000: **Advanced Encryption Standard (AES) [FIPS-197].**
  - Rijndael is a block cipher designed by Rijmen and Daemen that had been selected by the NIST.
  - Block size: 128 bits. Key size: 128, 192, 256 bits.
  - The AES is the most widely used block cipher today.

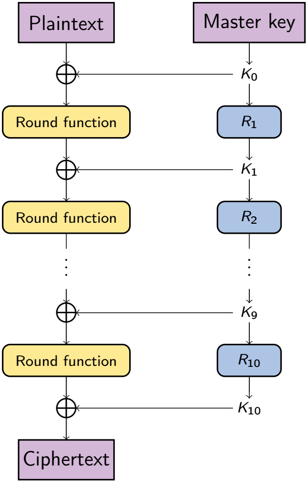
Introduction

National Institute of Standards and Technology (NIST) initiated processes to solicit, evaluate, and standardize cryptographic algorithms:

- 1997 - 2000: **Advanced Encryption Standard (AES) [FIPS-197].**
  - Rijndael is a block cipher designed by Rijmen and Daemen that had been selected by the NIST.
  - Block size: 128 bits. Key size: 128, 192, 256 bits.
  - The AES is the most widely used block cipher today.

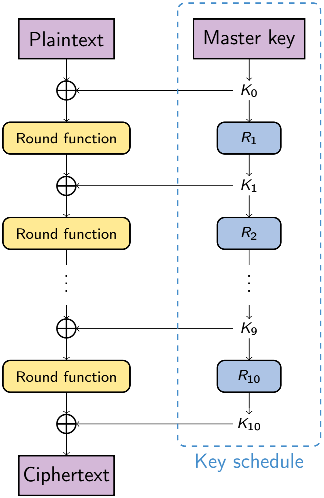
- 2019 - ... : **Lightweight Cryptography**.
  - 57 submissions.
  - 56 were selected as Round 1 Candidates.
  - 32 were selected as Round 2 Candidates.
  - 10 finalists.

AES: Advanced Encryption Standard [FIPS-197]



Description of the AES-128.

AES: Advanced Encryption Standard [FIPS-197]

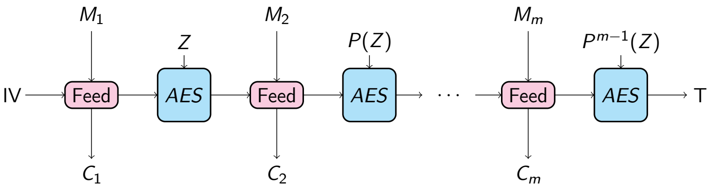


Description of the AES-128.

mixFeed [Chakraborty and Nandi, NIST LW Submission]

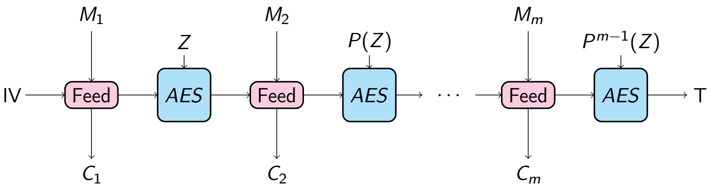
- mixFeed is a **second-round candidate** in the NIST Lightweight Standardization Process which was **not selected as a finalist**.
- It was submitted by **Bishwajit Chakraborty** and **Mridul Nandi**.
- It is an **AEAD** (Authenticated Encryption with Associated Data) algorithm.
- It is based on the AES block cipher.

mixFeed

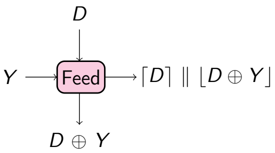


Simplified scheme of mixFeed encryption.

mixFeed

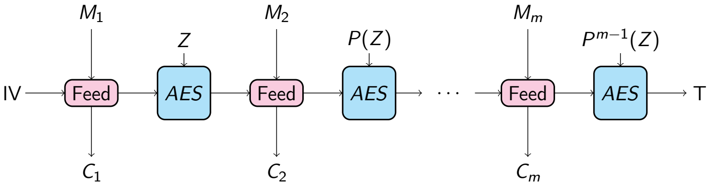


Simplified scheme of mixFeed encryption.

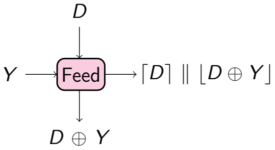


Function Feed in the case where $|D| = 128$.

mixFeed



Simplified scheme of mixFeed encryption.



Function Feed in the case where $|D| = 128$.

P: it is the permutation corresponding to eleven rounds of AES-128 key schedule.

Mustafa Khairallah's observation [ToSC'19]

000102030405060708090a0b0c0d0e0f
00020406080a0c0e10121416181a1c1e
0004080c1014181c2024282c3034383c
00081018202830384048505860687078
00102030405060708090a0b0c0d0e0f0
101112131415161718191a1b1c1d1e1f
20222426282a2c2e30323436383a3c3e
4044484c5054585c6064686c7074787c
80889098a0a8b0b8c0c8d0d8e0e8f0f8
303132333435363738393a3b3c3d3e3f
707172737475767778797a7b7c7d7e7f
000306090c0f1215181b1e2124272a2d
00050a0f14191e23282d32373c41464b
00070e151c232a31383f464d545b6269
000d1a2734414e5b6875828f9ca9b6c3
00152a3f54697e93a8bdd2e7fc11263b
00172e455c738aa1b8cfe6fd142b4259
00183048607890a8c0d8f00820385068
001c3854708ca8c4e0fc1834506c88a4
001f3e5d7c9bbad9f81736557493b2d1

Using brute-force and out of 33 tests, Khairallah found **20 cycles of length**

$$\mathbf{14018661024} \approx 2^{33.7}$$

for the P permutation[1].

Surprising facts:

→ all cycles found are of the same length.

→ this length is much smaller than the cycle length expected for a 128-bit permutation.

[1] Khairallah actually reported the length as 1133759136, probably because of a 32-bit overflow

AES Key Schedule

The AES key schedule is used to derive **11 subkeys** from a master key K.

AES Key Schedule

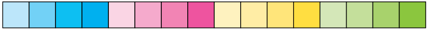
The AES key schedule is used to derive **11 subkeys** from a master key K.

Division of the key into words and representation of the words in a matrix.

AES Key Schedule

The AES key schedule is used to derive **11 subkeys** from a master key K.

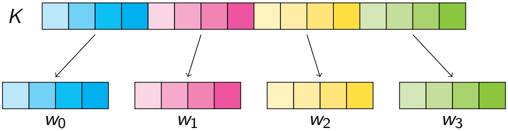
K



Division of the key into words and representation of the words in a matrix.

AES Key Schedule

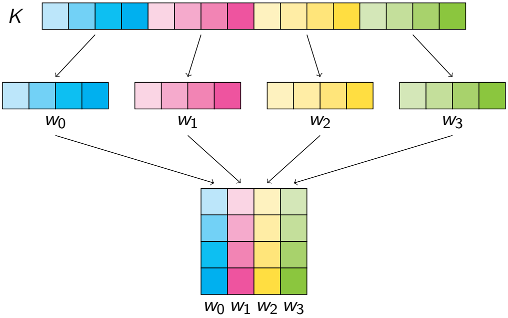
The AES key schedule is used to derive **11 subkeys** from a master key K.



Division of the key into words and representation of the words in a matrix.

AES Key Schedule

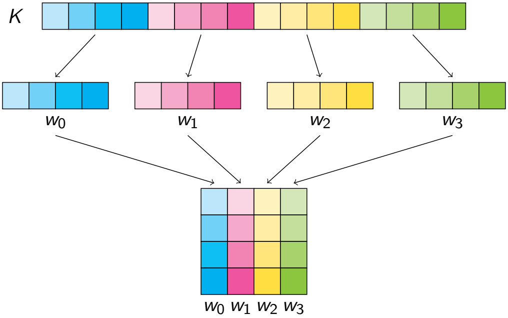
The AES key schedule is used to derive **11 subkeys** from a master key K.



Division of the key into words and representation of the words in a matrix.

AES Key Schedule

The AES key schedule is used to derive **11 subkeys** from a master key K.

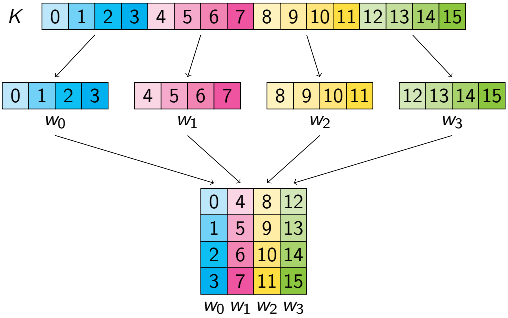


Division of the key into words and representation of the words in a matrix.

→ The subkey at round i is the concatenation of the words w_{4i} to w_{3+4i}.

AES Key Schedule

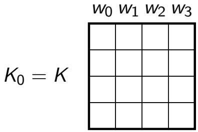
The AES key schedule is used to derive **11 subkeys** from a master key K.



Division of the key into words and representation of the words in a matrix.

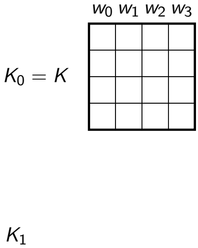
→ The subkey at round i is the concatenation of the words w_{4i} to w_{3+4i}.

AES Key Schedule



Construction of words $\mathrm{w_i}$ for $i \geq 4$.

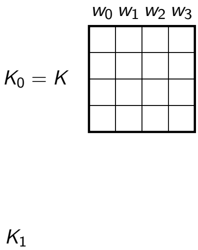
AES Key Schedule



Construction of words w_i for $i \geq 4$.

AES Key Schedule

The leftmost column:

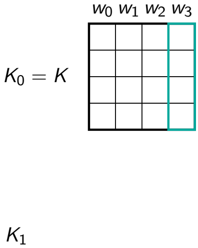


$$\mathsf{w_i} = \mathsf{SubWord}(\mathsf{RotWord}(\mathsf{w_{i-1}})) \oplus \mathsf{RCon}(i/4) \oplus \mathsf{w_{i-4}}$$

Construction of words $\mathsf{w_i}$ for $i \geq 4$.

AES Key Schedule

The leftmost column:



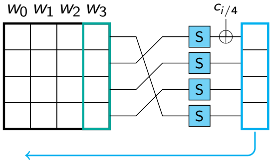
$$w_i = \mathrm{SubWord}(\mathrm{RotWord}(w_{i-1})) \oplus \mathrm{RCon}(i/4) \oplus w_{i-4}$$

Construction of words w_i for $i \geq 4$.

AES Key Schedule

The leftmost column:

$K_0 = K$



K_1

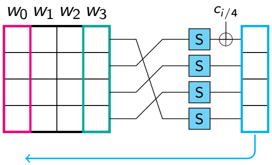
$w_i = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}(i/4) \oplus w_{i-4}$

Construction of words w_i for $i \geq 4$.

AES Key Schedule

The leftmost column:

$K_0 = K$



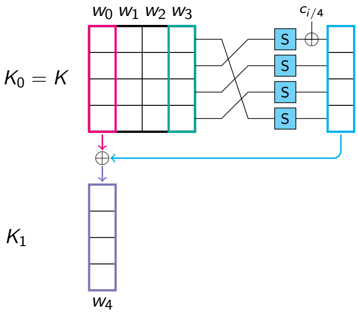
K_1

$w_i = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}(i/4) \oplus w_{i-4}$

Construction of words w_i for $i \geq 4$.

AES Key Schedule

The leftmost column:

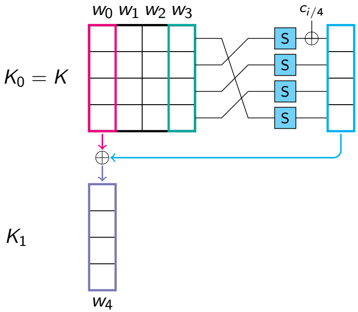


$$w_i = \mathrm{SubWord}(\mathrm{RotWord}(w_{i-1})) \oplus \mathrm{RCon}(i/4) \oplus w_{i-4}$$

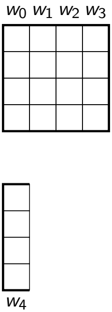
Construction of words w_i for $i \geq 4$.

AES Key Schedule

The leftmost column:

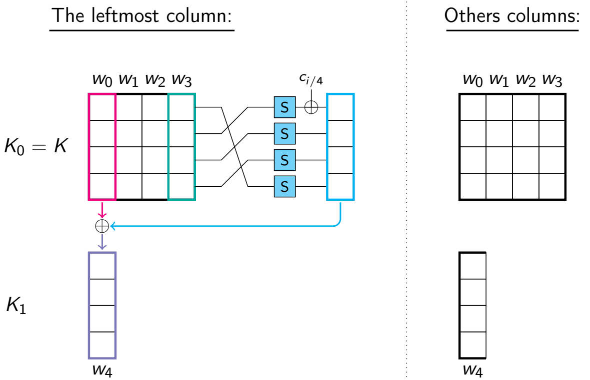


$$\mathsf{w_i} = \mathsf{SubWord}(\mathsf{RotWord}(\mathsf{w_{i-1}})) \oplus \mathsf{RCon}(i/4) \oplus \mathsf{w_{i-4}}$$



Construction of words $\mathsf{w_i}$ for $i \geq 4$.

AES Key Schedule



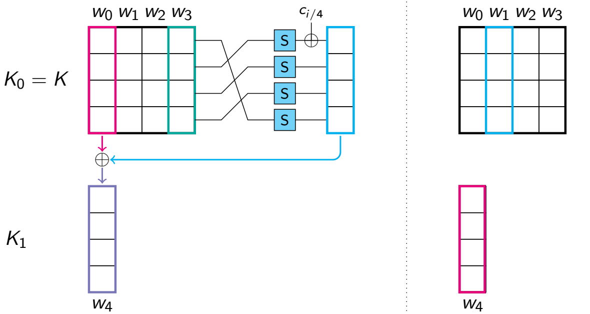
$\mathrm{w_i} = \mathrm{SubWord}(\mathrm{RotWord}(\mathrm{w_{i-1}})) \oplus \mathrm{RCon}(i/4) \oplus \mathrm{w_{i-4}}$

$\mathrm{w_i} = \mathrm{w_{i-1}} \oplus \mathrm{w_{i-4}}$

Construction of words $\mathrm{w_i}$ for $i \geq 4$.

AES Key Schedule

The leftmost column: | Others columns:

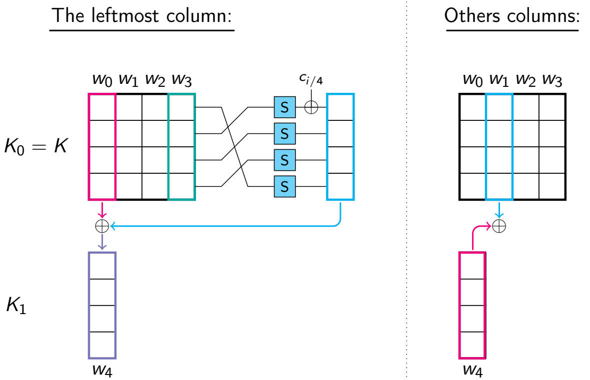


$w_i = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}(i/4) \oplus w_{i-4}$

$w_i = w_{i-1} \oplus w_{i-4}$

Construction of words w_i for $i \geq 4$.

AES Key Schedule

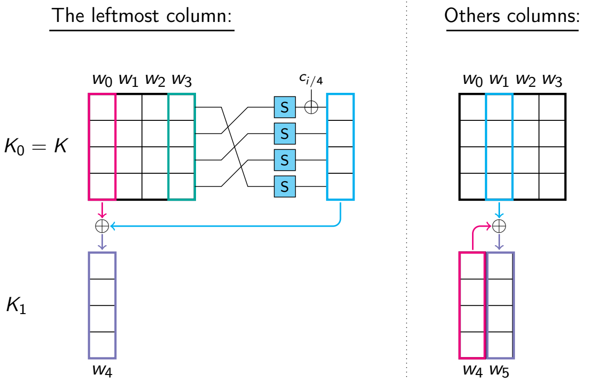


$\mathrm{w_i} = \mathrm{SubWord}(\mathrm{RotWord}(\mathrm{w_{i-1}})) \oplus \mathrm{RCon}(i/4) \oplus \mathrm{w_{i-4}}$

$\mathrm{w_i} = \mathrm{w_{i-1}} \oplus \mathrm{w_{i-4}}$

Construction of words $\mathrm{w_i}$ for $i \geq 4$.

AES Key Schedule

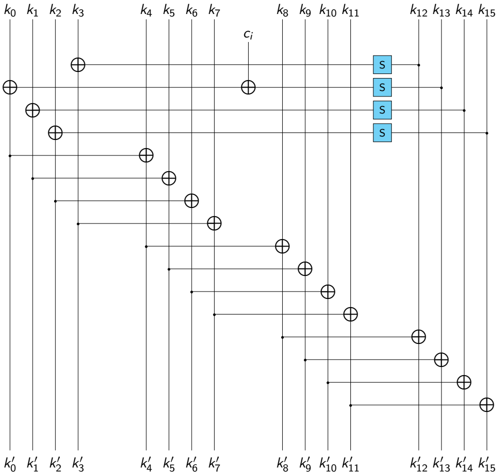


$\mathrm{w_i} = \mathrm{SubWord}(\mathrm{RotWord}(\mathrm{w_{i-1}})) \oplus \mathrm{RCon}(i/4) \oplus \mathrm{w_{i-4}}$

$\mathrm{w_i} = \mathrm{w_{i-1}} \oplus \mathrm{w_{i-4}}$

Construction of words $\mathrm{w_i}$ for $i \geq 4$.

One round of key schedule at byte level



One round of the AES key schedule.

Table of contents

Table of contents

Difference diffusion

Leander, Minaud and Rønjom ([EC'15]) introduced an algorithm in order to **detect invariant subspaces for a permutation**, *i.e.* a subspace A and an offset u such as:

$$\boxed{F(A+u) = A + F(u)}$$

Let's recall how the generic algorithm works for a permutation F : $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$:

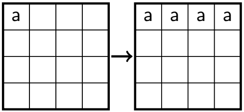
1) Guess an offset $u' \in \mathbb{F}_2^n$ and a one-dimensional subspace A_0.
2) Compute $A_{i+1} = span\{(F(u'+A_i) - F(u')) \cup A_i\}$
3) If the dimension of A_{i+1} equals the dimension of A_i, we found an invariant subspace and exit.
4) Else, we go on step 2.

Difference diffusion

a			

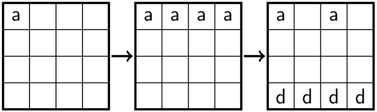
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



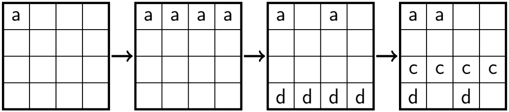
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



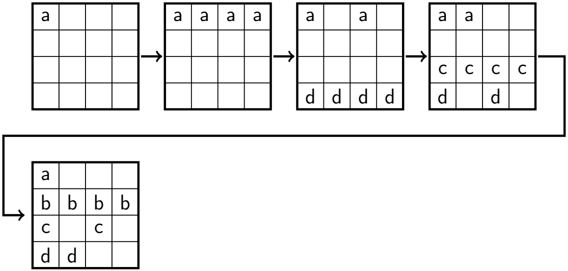
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



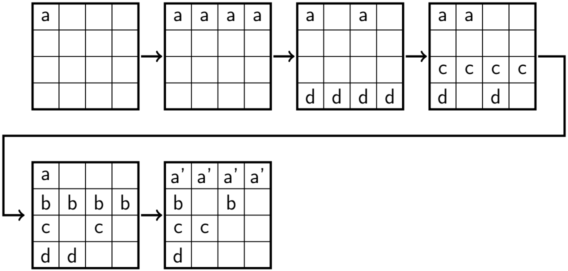
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



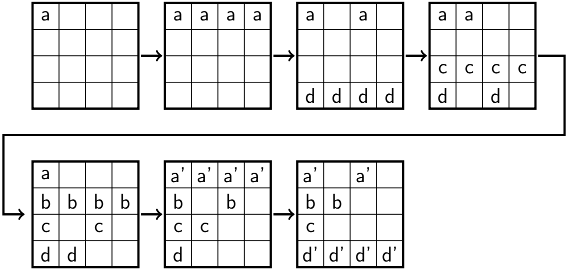
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



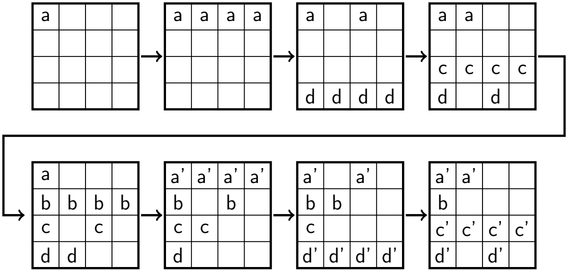
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



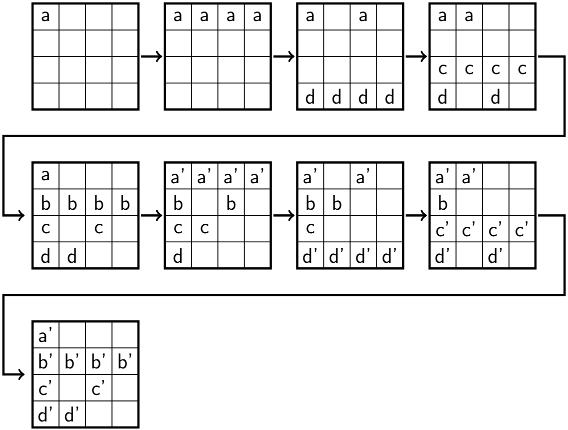
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



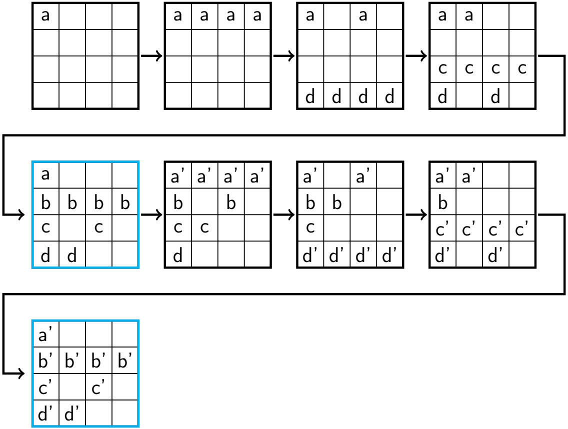
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



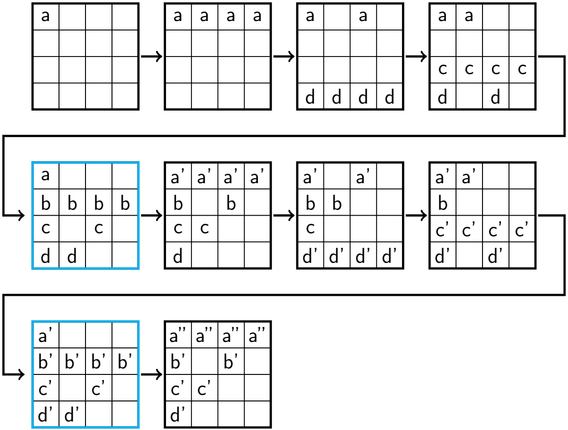
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



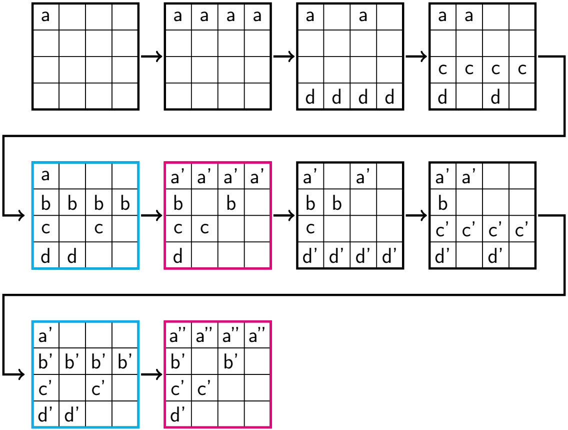
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



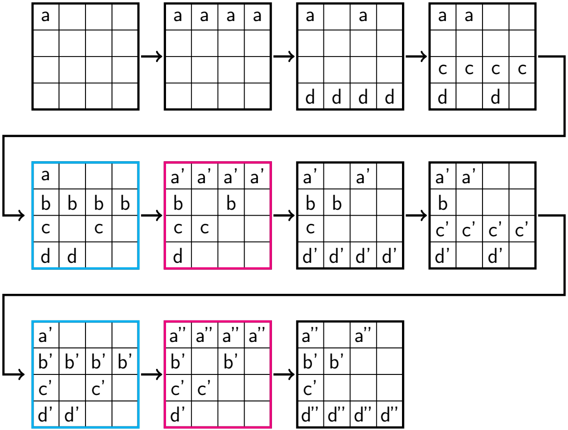
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



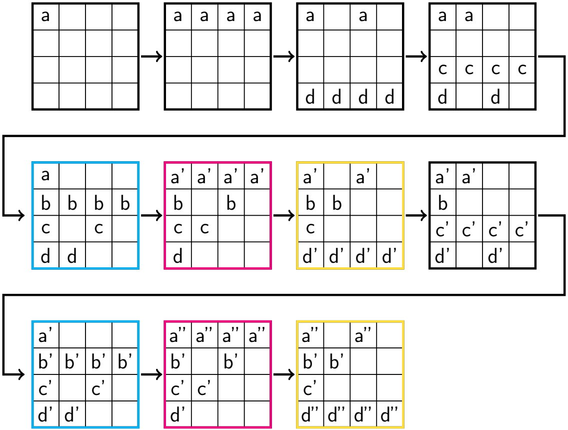
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



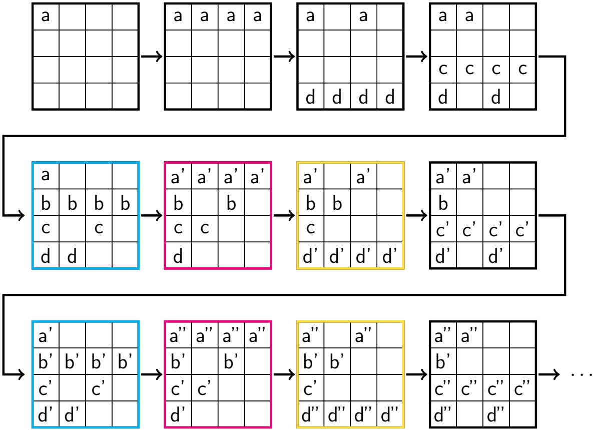
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



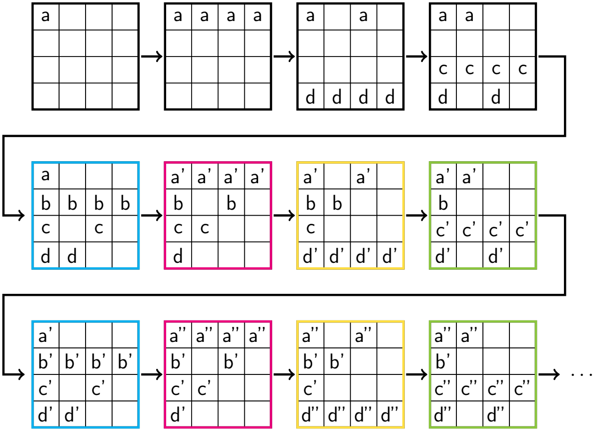
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion

We obtain **4 families of invariant affine subspaces** whose linear parts are:

$$E_0 = \{(a, b, c, d, 0, b, 0, d, a, 0, 0, d, 0, 0, 0, d) \text{ with } a, b, c, d \in \mathbb{F}_{2^8}\}$$
$$E_1 = \{(a, b, c, d, a, 0, c, 0, 0, 0, c, d, 0, 0, c, 0) \text{ with } a, b, c, d \in \mathbb{F}_{2^8}\}$$
$$E_2 = \{(a, b, c, d, 0, b, 0, d, 0, b, c, 0, 0, b, 0, 0) \text{ with } a, b, c, d \in \mathbb{F}_{2^8}\}$$
$$E_3 = \{(a, b, c, d, a, 0, c, 0, a, b, 0, 0, a, 0, 0, 0) \text{ with } a, b, c, d \in \mathbb{F}_{2^8}\}$$

$$\boxed{\forall u \in (\mathbb{F}_{2^8})^{16}, R(E_i + u) = E_{i+1} + R(u)}$$

The full space is the direct sum of those four vector spaces:

$$(\mathbb{F}_{2^8})^{16} = E_0 \oplus E_1 \oplus E_2 \oplus E_3$$

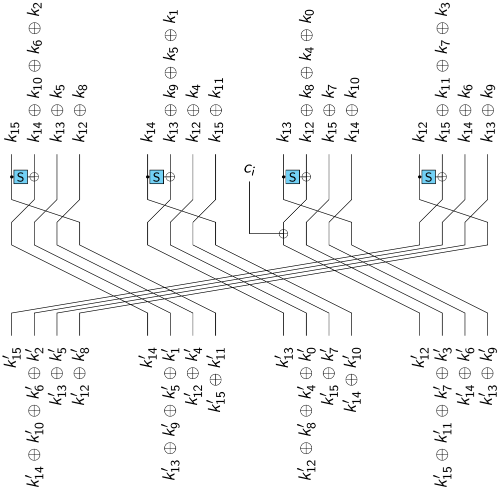
Table of contents

New representation of the AES Key Schedule

To describe a representation that makes the **4 subspaces** appear more clearly, we will perform a **linear transformation** $\mathsf{A} = \mathsf{C}_0^{-1}$, which corresponds to a base change:

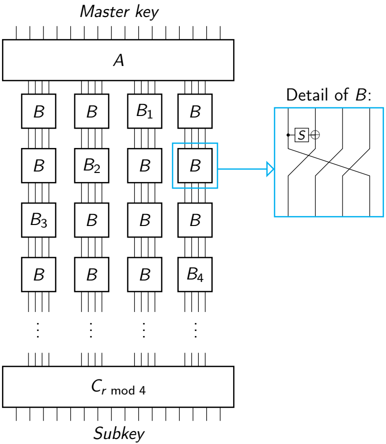
$$
\begin{array}{cccc}
s_0 = k_{15} & s_1 = k_{14} \oplus k_{10} \oplus k_6 \oplus k_2 & s_2 = k_{13} \oplus k_5 & s_3 = k_{12} \oplus k_8 \\
s_4 = k_{14} & s_5 = k_{13} \oplus k_9 \oplus k_5 \oplus k_1 & s_6 = k_{12} \oplus k_4 & s_7 = k_{15} \oplus k_{11} \\
s_8 = k_{13} & s_9 = k_{12} \oplus k_8 \oplus k_4 \oplus k_0 & s_{10} = k_{15} \oplus k_7 & s_{11} = k_{14} \oplus k_{10} \\
s_{12} = k_{12} & s_{13} = k_{15} \oplus k_{11} \oplus k_7 \oplus k_3 & s_{14} = k_{14} \oplus k_6 & s_{15} = k_{13} \oplus k_9
\end{array}
$$

New representation of the AES Key Schedule



One round of the AES key schedule (alternative representation).

New representation of the AES Key Schedule



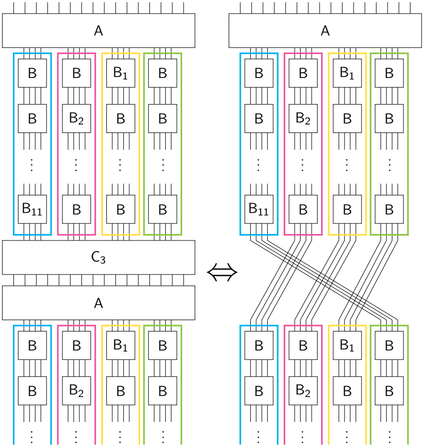
r rounds of the key schedule in the new representation.

- B_i is similar to B but the round constant c_i is XORed to the output of the S-box.
- $C_i = A^{-1} \times \mathrm{SR}^i$, with SR the matrix corresponding to rotation of 4 bytes to the right.

Table of contents

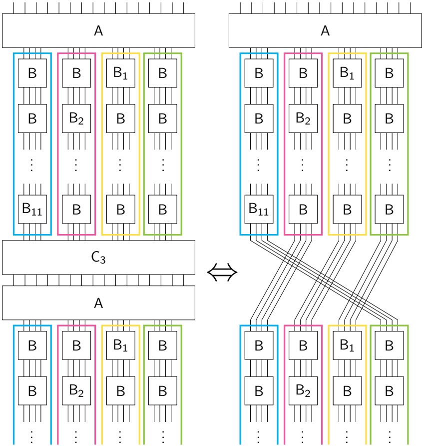
Table of contents

Cycle analysis of 11-round AES key schedule



Two iterations of 11 rounds of the key schedule in the new representation.

Cycle analysis of 11-round AES key schedule



Two iterations of 11 rounds of the key schedule in the new representation.

We define:

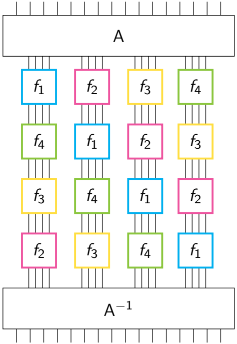
$f_1 = B_{11} \circ B \circ B \circ B \circ B_7 \circ B \circ B \circ B \circ B_3 \circ B \circ B$

$f_2 = B \circ B_{10} \circ B \circ B \circ B \circ B_6 \circ B \circ B \circ B \circ B_2 \circ B$

$f_3 = B \circ B \circ B_9 \circ B \circ B \circ B \circ B_5 \circ B \circ B \circ B \circ B_1$

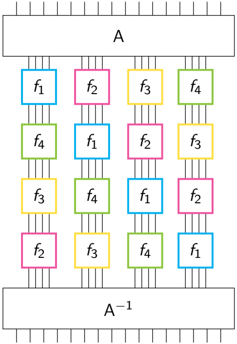
$f_4 = B \circ B \circ B \circ B_8 \circ B \circ B \circ B \circ B_4 \circ B \circ B \circ B$

Cycle analysis of 11-round AES key schedule

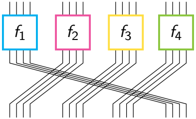


4 iterations of P in the new model.

Cycle analysis of 11-round AES key schedule



4 iterations of P in the new model.



$$\widetilde{P} = A \circ P \circ A^{-1}$$

$$\widetilde{P} : (a, b, c, d) \mapsto (f_2(b), f_3(c), f_4(d), f_1(a))$$

$$\widetilde{P}^4 : (a, b, c, d) \mapsto (\phi_1(a), \phi_2(b), \phi_3(c), \phi_4(d))$$

$$\phi_1(a) = f_2 \circ f_3 \circ f_4 \circ f_1(a)$$

$$\phi_2(b) = f_3 \circ f_4 \circ f_1 \circ f_2(b)$$

$$\phi_3(c) = f_4 \circ f_1 \circ f_2 \circ f_3(c)$$

$$\phi_4(d) = f_1 \circ f_2 \circ f_3 \circ f_4(d)$$

Cycle analysis of 11-round AES key schedule

- If (a, b, c, d) is in a cycle of length ℓ of $\widetilde{P}^4$, we have:

$$\phi_1^\ell(a) = a \qquad \phi_2^\ell(b) = b \qquad \phi_3^\ell(c) = c \qquad \phi_4^\ell(d) = d$$

In particular, a, b, c and d must be in cycles of ϕ_1, ϕ_2, ϕ_3, ϕ_4 (respectively) of **length dividing** ℓ.

Cycle analysis of 11-round AES key schedule

- If (a, b, c, d) is in a cycle of length ℓ of $\widetilde{P}^4$, we have:

$$\phi_1^\ell(a) = a \qquad \phi_2^\ell(b) = b \qquad \phi_3^\ell(c) = c \qquad \phi_4^\ell(d) = d$$

  In particular, a, b, c and d must be in cycles of ϕ_1, ϕ_2, ϕ_3, ϕ_4 (respectively) of **length dividing** ℓ.

- Conversely, if a, b, c, d are in small cycles of the corresponding ϕ_i, then (a, b, c, d) is in a cycle of $\widetilde{P}^4$ of length the **lowest common multiple of the small cycle lengths**.

Cycle analysis of 11-round AES key schedule

- If (a, b, c, d) is in a cycle of length ℓ of $\widetilde{P}^4$, we have:

$$\phi_1^\ell(a) = a \qquad \phi_2^\ell(b) = b \qquad \phi_3^\ell(c) = c \qquad \phi_4^\ell(d) = d$$

  In particular, a, b, c and d must be in cycles of ϕ_1, ϕ_2, ϕ_3, ϕ_4 (respectively) of **length dividing** ℓ.

- Conversely, if a, b, c, d are in small cycles of the corresponding ϕ_i, then (a, b, c, d) is in a cycle of $\widetilde{P}^4$ of length the **lowest common multiple of the small cycle lengths**.

- Due to the structure of the ϕ_i functions, all of them have the **same cycle structure**:

$$\phi_2 = f_2^{-1} \circ \phi_1 \circ f_2; \qquad \phi_3 = f_3^{-1} \circ \phi_2 \circ f_3; \qquad \phi_4 = f_4^{-1} \circ \phi_3 \circ f_4$$

Cycle analysis of 11-round AES key schedule

Length	# cycles	Proba	Smallest element
3504665256	1	0.82	00 00 00 01
255703222	1	0.05	00 00 00 0b
219107352	1	0.05	00 00 00 1d
174977807	1	0.04	00 00 00 00
99678312	1	0.02	00 00 00 21
13792740	1	0.003	00 00 00 75
8820469	1	$2^{-8,93}$	00 00 00 24
7619847	1	$2^{-9,14}$	00 00 00 c1
5442633	1	$2^{-9,63}$	00 00 02 78
4214934	1	2^{-10}	00 00 05 77
459548	1	$2^{-13,2}$	00 00 38 fe
444656	1	$2^{-13,24}$	00 00 0b 68
14977	1	$2^{-18,13}$	00 06 82 5c
14559	1	$2^{-18,18}$	00 04 fa b1
5165	1	$2^{-19,67}$	00 0a d4 4e
4347	1	$2^{-19,92}$	00 04 94 3a
1091	1	$2^{-21.91}$	00 21 4b 3b
317	1	$2^{-23,7}$	00 28 41 36
27	1	$2^{-27,25}$	01 3a 0d 0c
6	1	$2^{-29,42}$	06 23 25 51
5	3	$3 \cdot 2^{-29,68}$	06 1a ea 18
4	2	$2 \cdot 2^{-30}$	23 c6 6f 2b
2	3	$3 \cdot 2^{-31}$	69 ea 63 75
1	2	$2 \cdot 2^{-32}$	7e be d1 92

Cycle structure of ϕ_1 for 11-round AES-128 key schedule.

Cycle analysis of 11-round AES key schedule

Length	# cycles	Proba	Smallest element
3504665256	1	0.82	00 00 00 01
255703222	1	0.05	00 00 00 0b
219107352	1	0.05	00 00 00 1d
174977807	1	0.04	00 00 00 00
99678312	1	0.02	00 00 00 21
13792740	1	0.003	00 00 00 75
8820469	1	$2^{-8,93}$	00 00 00 24
7619847	1	$2^{-9,14}$	00 00 00 c1
5442633	1	$2^{-9,63}$	00 00 02 78
4214934	1	2^{-10}	00 00 05 77
459548	1	$2^{-13,2}$	00 00 38 fe
444656	1	$2^{-13,24}$	00 00 0b 68
14977	1	$2^{-18,13}$	00 06 82 5c
14559	1	$2^{-18,18}$	00 04 fa b1
5165	1	$2^{-19,67}$	00 0a d4 4e
4347	1	$2^{-19,92}$	00 04 94 3a
1091	1	$2^{-21.91}$	00 21 4b 3b
317	1	$2^{-23,7}$	00 28 41 36
27	1	$2^{-27,25}$	01 3a 0d 0c
6	1	$2^{-29,42}$	06 23 25 51
5	3	$3 \cdot 2^{-29,68}$	06 1a ea 18
4	2	$2 \cdot 2^{-30}$	23 c6 6f 2b
2	3	$3 \cdot 2^{-31}$	69 ea 63 75
1	2	$2 \cdot 2^{-32}$	7e be d1 92

Cycle structure of ϕ_1 for 11-round AES-128 key schedule.

With probability $0.82^4 \simeq 0.45$, we have a, b, c and d in a cycle of length $\ell = 3504665256$, resulting in:

- $\rightarrow$ a cycle of length ℓ for $\widetilde{P}^4$,
- $\rightarrow$ a cycle of length at most $4\ell = 14018661024$ for $\widetilde{P}$ and P.

Cycle analysis of 11-round AES key schedule

Summary: 45% of keys belong to cycles of length 14018661024 $\approx 2^{33.7}$.

Cycle analysis of 11-round AES key schedule

Summary: 45% of keys belong to cycles of length 14018661024 $\approx 2^{33.7}$.

$\rightarrow$ This explains the observation on mixFeed [Khairallah, ToSC'19].

Cycle analysis of 11-round AES key schedule

Summary: 45% of keys belong to cycles of length 14018661024 $\approx 2^{33.7}$.

$\rightarrow$ This explains the observation on mixFeed [Khairallah, ToSC'19].

$\rightarrow$ This contradicts the assumption made in a security proof of mixFeed:

Assumption [Chakraborty and Nandi, NIST LW Workshop]

For any $K \in \{0,1\}^n$ chosen uniformly at random, probability that K has a period at most ℓ is at most $\ell/2^{n/2}$.

Table of contents

Forgery attack against mixFeed [Khairallah, ToSC'19]

The goal of a **forgery attack** is to forge a valid tag T' for a new ciphertext C' using (M, C, T).

Forgery attack against mixFeed [Khairallah, ToSC'19]

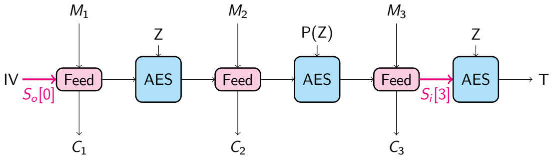
The goal of a **forgery attack** is to forge a valid tag T' for a new ciphertext C' using (M, C, T).

Assuming that Z belongs to a cycle of length ℓ, we have the following attack considering a message M made of m blocks, with $m > \ell$:

1) Encrypt the message M, and obtain the corresponding ciphertext C and tag T.
2) Calculate $S_o[0] = IV$ and $S_i[\ell+1]$ using M_r and C_r for $r = 1$ and $r = \ell + 1$.
3) Choose M_x and C_x such that $(S_i[\ell+1], \mathrm{C}_x) = \mathrm{Feed}\ (S_o[0], M_x)$.
4) The T tag will also authenticate the new ciphertext $C' = C_x \| C_{\ell+2} \| \cdots \| C_m$.

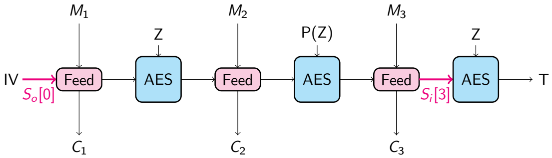
Forgery attack against mixFeed

1) Encrypt a message M, and obtain the corresponding ciphertext C and tag T.
2) Calculate $S_o[0] = IV$ and $S_i[\ell + 1]$ using M_r and C_r for $r = 1$ and $r = \ell + 1$.

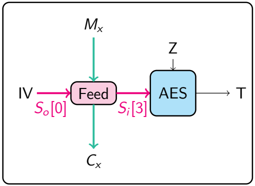


Forgery attack against mixFeed

1) Encrypt a message M, and obtain the corresponding ciphertext C and tag T.
2) Calculate $S_o[0] = IV$ and $S_i[\ell + 1]$ using M_r and C_r for $r = 1$ and $r = \ell + 1$.



3) Choose M_x and C_x such that $(S_i[\ell + 1], \mathrm{C}_x) = \mathrm{Feed}\ (S_o[0], M_x)$.
4) The T tag will also authenticate the new ciphertext $C' = C_x \| C_{\ell+2} \| \cdots \| C_m$.



Forgery attack against mixFeed

Summary of the forgery attack:

→ Data complexity: a known plaintext of length higher than $2^{37.7}$ bytes

→ Memory complexity: negligible

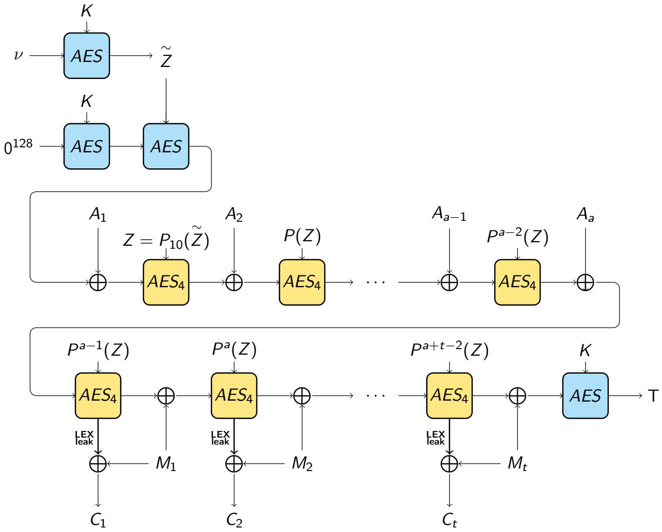
→ Time complexity: negligible

→ Success rate: 45%

⇒ Verified using the mixFeed reference implementation

Table of contents

Application to ALE [Bog+14]



Authenticated encryption with ALE.

Application to ALE

ALE has been designed so that **each AES encryption is performed with different keys**, to avoid attacks that use pairs of messages encrypted with the same key.

$\rightarrow$ Using the same approach as for mixFeed, we find that 76% of the keys belong to cycles of length $16043203220 \approx 2^{33.9}$.

$\rightarrow$ Short length cycles allows us to easily find states encrypted under the same key.

$\rightarrow$ We used the tool developed by Bouillaguet, Derbez, and Fouque [Crypto'11] in order to find an attack against ALE.

Application to ALE

ALE has been designed so that **each AES encryption is performed with different keys**, to avoid attacks that use pairs of messages encrypted with the same key.

$\rightarrow$ Using the same approach as for mixFeed, we find that 76% of the keys belong to cycles of length $16043203220 \approx 2^{33.9}$.

$\rightarrow$ Short length cycles allows us to easily find states encrypted under the same key.

$\rightarrow$ We used the tool developed by Bouillaguet, Derbez, and Fouque [Crypto'11] in order to find an attack against ALE.

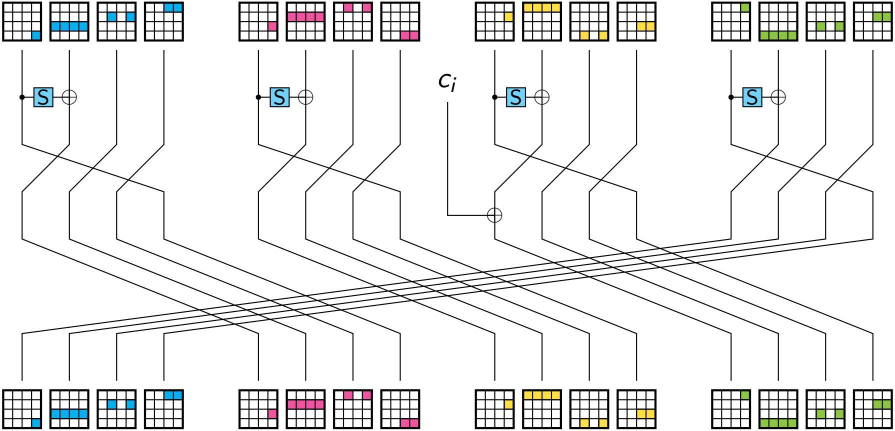
Attack		Enc	Verif	Time	Ref
Existential Forgery	Known Plaintext	$2^{110.4}$	2^{102}	$2^{110.4}$	[Wu+13]
Existential Forgery	Known Plaintext	2^{103}	2^{103}	2^{104}	[KR14]
Existential Forgery	Known Plaintext	1	2^{120}	2^{120}	[KR14]
State Recovery, Almost Univ. Forgery	Known Plaintext	1	2^{121}	2^{121}	[KR14]
State Recovery, Almost Univ. Forgery	Chosen Plaintext	$2^{57.3}$	0	$2^{104.4}$	New

Comparison of attacks against ALE.

Table of contents

Table of contents

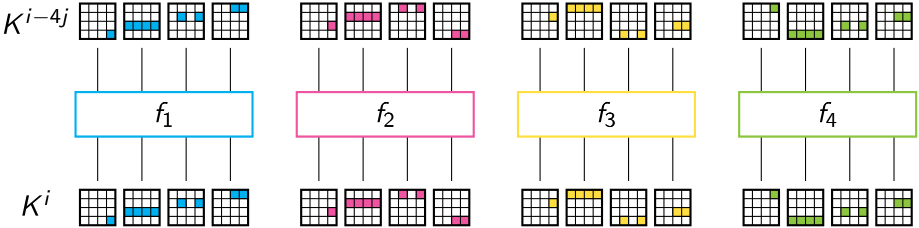
Property on the AES Key Schedule



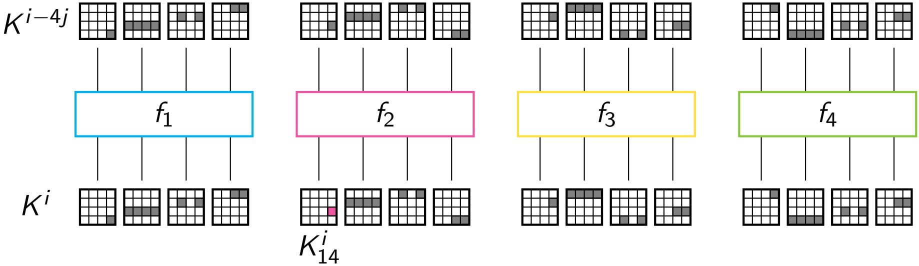
One round of the AES key schedule with graphic representations of bytes positions (alternative representation).

Only the XOR of the colored bytes is required for each state.

Property on the AES Key Schedule

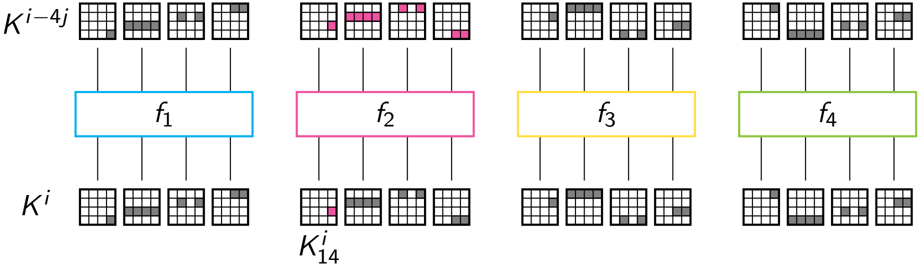


Property on the AES Key Schedule



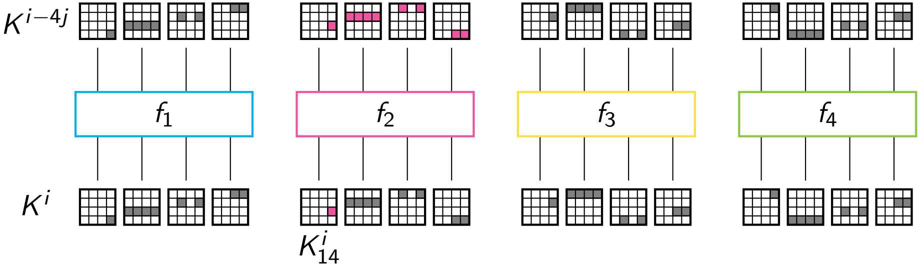
How to compute K^i_{14}?

Property on the AES Key Schedule



How to compute K^i_{14}?

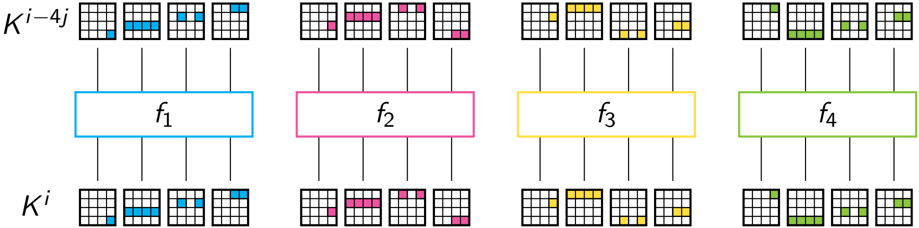
Property on the AES Key Schedule



How to compute K^i_{14}?

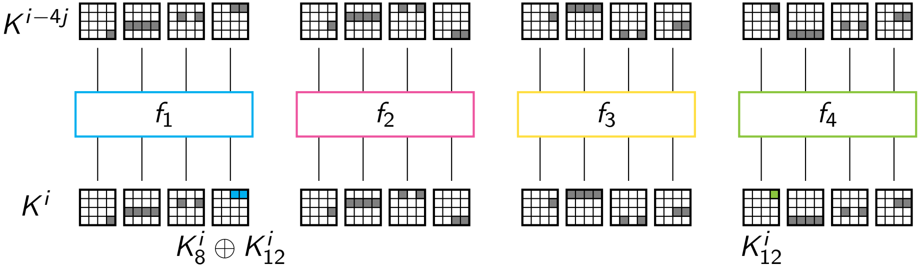
→ **A byte in the last column depends on only 32 bits of information.**

Property on the AES Key Schedule



$\rightarrow$ A byte in the last column depends on only 32 bits of information.

Property on the AES Key Schedule

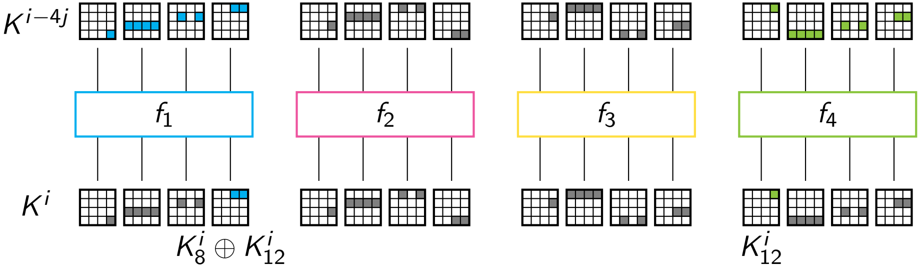


How to compute K_8^i?

$$K_8^i = (K_8^i \oplus K_{12}^i) \oplus K_{12}^i$$

→ A byte in the last column depends on only 32 bits of information.

Property on the AES Key Schedule

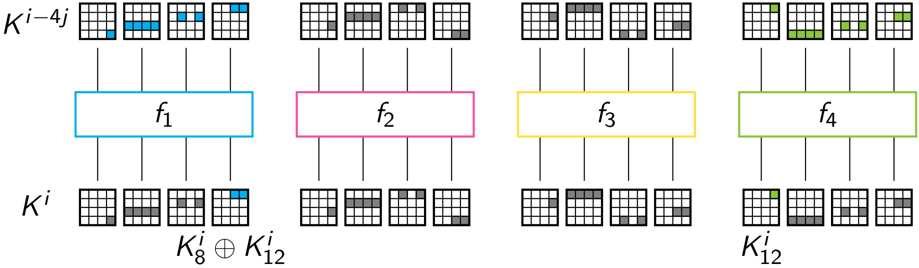


How to compute K_8^i?

$$K_8^i = (K_8^i \oplus K_{12}^i) \oplus K_{12}^i$$

$\rightarrow$ A byte in the last column depends on only 32 bits of information.

Property on the AES Key Schedule



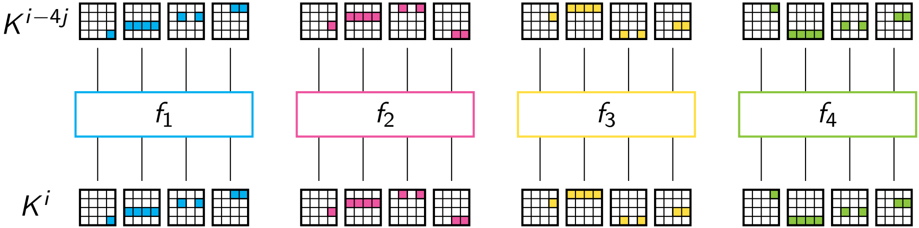
How to compute K^i_8?

$$K^i_8 = (K^i_8 \oplus K^i_{12}) \oplus K^i_{12}$$

→ A byte in the last column depends on only 32 bits of information.

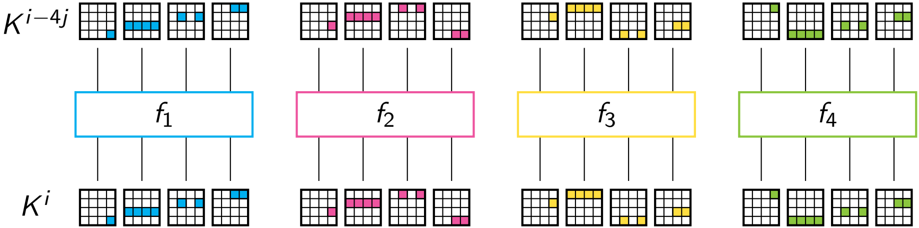
→ **A byte in the 3rd column depends on only 64 bits of information.**

Property on the AES Key Schedule



→ A byte in the last column depends on only 32 bits of information.
→ A byte in the 3rd column depends on only 64 bits of information.
→ **A byte in the 2nd column depends on only 64 bits of information.**

Property on the AES Key Schedule



→ A byte in the last column depends on only 32 bits of information.
→ A byte in the 3rd column depends on only 64 bits of information.
→ A byte in the 2nd column depends on only 64 bits of information.
→ **A byte in the first column depends on 128 bits of information.**

Property on the AES Key Schedule

Summary: even after a large number of rounds, the key schedule does not mix all the bytes!

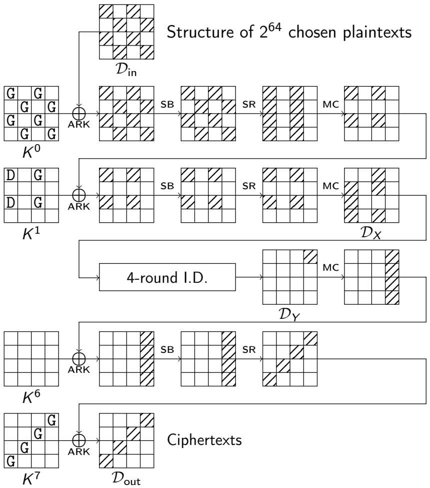
Computing the value of a byte of a subkey does not necessarily require to know the whole master key:

→ A byte in the first column depends on at most 128 bits of information
→ A byte in the second column depends on at most 64 bits of information
→ A byte in the third column depends on at most 64 bits of information
→ A byte in the last column depends on at most 32 bits of information

⇒ This allows to **combine more efficiently** information on the first and the last subkeys.

Table of contents

Impossible Differential - AES



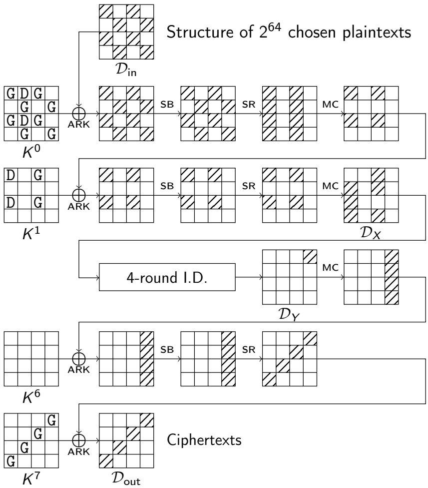
7-round impossible differential attack ([MDRM, IC'10]).
Figure adapted from Tikz for Cryptographers [Jean].

The attack is in 2 parts:

(1) find the possible candidates for the bytes marked G.

(2) find the master keys corresponding to these bytes.

Impossible Differential - AES



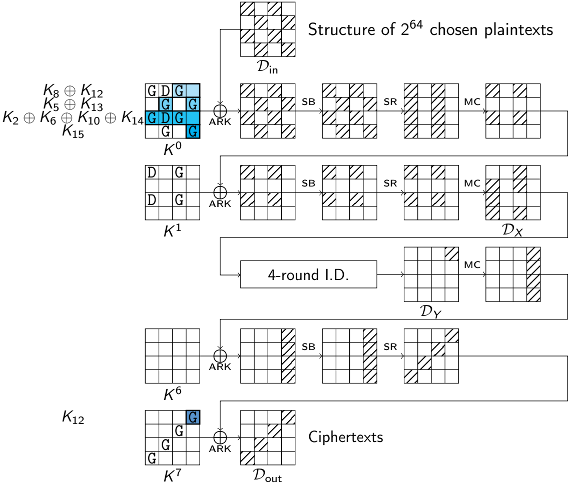
The attack is in 2 parts:

(1) find the possible candidates for the bytes marked G.

(2) find the master keys corresponding to these bytes.

7-round impossible differential attack ([MDRM, IC'10]).
Figure adapted from Tikz for Cryptographers [Jean].

Impossible Differential - AES



7-round impossible differential attack ([MDRM, IC'10]). Figure adapted from Tikz for Cryptographers [Jean].

The attack is in 2 parts:

(1) find the possible candidates for the bytes marked G.

(2) find the master keys corresponding to these bytes.

We improve (2) by **combining information from K^0 and K^7 more efficiently** thanks to properties related to our new representation.

Impossible Differential - AES

Attack	Data	Time	Mem.	Ref.
Meet-in-the-middle	2^{97}	2^{99}	2^{98}	[Derbez, Fouque, Jean, EC'13]
	2^{105}	2^{105}	2^{90}	[Derbez, Fouque, Jean, EC'13]
	2^{105}	2^{105}	2^{81}	[Bonnetain, Naya-Plasencia, Schrottenloher, ToSC'19]
	2^{113}	2^{113}	2^{74}	[Bonnetain, Naya-Plasencia, Schrottenloher, ToSC'19]
Impossible differential	2^{113}	2^{113}	2^{74}	[Boura, Lallemand, Naya-Plasencia, Suder, JC'18]
	$2^{105.1}$	2^{113}	$2^{74.1}$	[Boura, Lallemand, Naya-Plasencia, Suder, JC'18][2]
	$2^{106.1}$	$2^{112.1}$	$2^{73.1}$	Variant of [Boura, Lallemand, Naya-Plasencia, Suder, JC'18]
	$2^{104.9}$	$2^{110.9}$	$2^{71.9}$	**New**

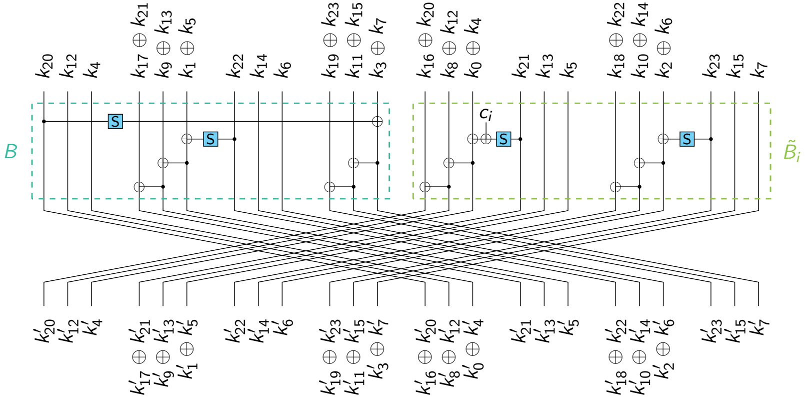
Best single-key attacks against 7-round AES-128.

[2] The time complexity is incorrectly given as $2^{106.88}$ in [Boura, Lallemand, Naya-Plasencia, Suder, JC'18].

Table of contents

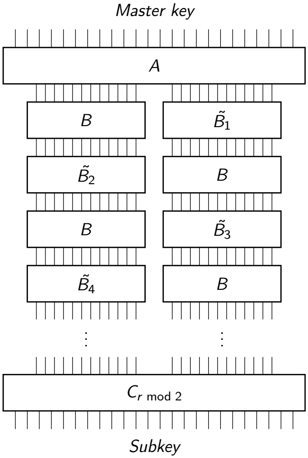
Table of contents

New Representation of the AES-192 Key Schedules



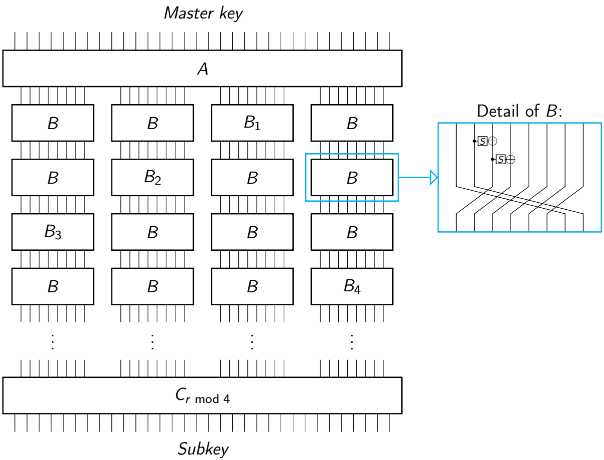
One round of the AES-192 key schedule (alternative representation).

New Representation of the AES-192 Key Schedules



r rounds of the AES-192 key schedule in the new representation.

New Representation of the AES-256 Key Schedules



r rounds of the AES-256 key schedule in the new representation. B_i is similar to B but the round constant c_i is XORed to the output of the first S-box.

Table of contents

Properties on the AES Key Schedule (1)

Proposition

Let P_r and P'_r defined in one of the following ways:

- *AES-128:* $P_r = (k_r[5], k_r[7], k_r[13], k_r[15])$, *and* $P'_r = (k_r[4], k_r[6], k_r[12], k_r[14])$.
- *AES-192:* $P_r = (k_r[5], k_r[7], k_r[13], k_r[15], k_r[21], k_r[23])$, *and* $P'_r = (k_r[4], k_r[6], k_r[12], k_r[14], k_r[20], k_r22])$.
- *AES-256:* $P_r = (k_r[5], k_r[7], k_r[13], k_r[15], k_r[21], k_r[23], k_r[29], k_r[31])$, *and* $P'_r = (k_r[4], k_r[6], k_r[12], k_r[14], k_r[20], k_r22], k_r[28], k_r[30])$.

If there exists an r_0 such as $P_{\mathbf{r_0}}$ and $P'_{\mathbf{r_0}\pm 1}$ are known, then for all $i \in \mathbb{Z}$, the bytes $P_{\mathbf{r_0}+2i}$ and $P'_{\mathbf{r_0}+2i+1}$ are known (and they are easily computable).

Properties on the AES Key Schedule (2)

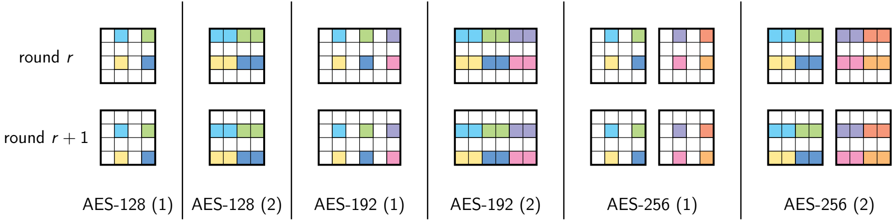
Proposition

Let P_r and P'_r defined in one of the following ways:

- *AES-128:* $P_r = (k_r[0] \oplus k_r[4], k_r[2] \oplus k_r[6], k_r[8] \oplus k_r[12], k_r[10] \oplus k_r[14])$, *and* $P'_r = (k_r[1] \oplus k_r[5], k_r[3] \oplus k_r[7], k_r[9] \oplus k_r[13], k_r[11] \oplus k_r[15])$.
- *AES-192:* $P_r = (k_r[0] \oplus k_r[4], k_r[2] \oplus k_r[6], k_r[8] \oplus k_r[12], k_r[10] \oplus k_r[14], k_r[16] \oplus k_r[20], k_r[18] \oplus k_r[22])$, *and* $P'_r = (k_r[1] \oplus k_r[5], k_r[3] \oplus k_r[7], k_r[9] \oplus k_r[13], k_r[11] \oplus k_r[15], k_r[17] \oplus k_r[21], k_r[3] \oplus k_r[23])$.
- *AES-256:* $P_r = (k_r[0] \oplus k_r[4], k_r[2] \oplus k_r[6], k_r[8] \oplus k_r[12], k_r[10] \oplus k_r[14], k_r[16] \oplus k_r[20], k_r[18] \oplus k_r[22], k_r[24] \oplus k_r[28], k_r[26] \oplus k_r[30])$, *and* $P'_r = (k_r[1] \oplus k_r[5], k_r[3] \oplus k_r[7], k_r[9] \oplus k_r[13], k_r[11] \oplus k_r[15], k_r[17] \oplus k_r[21], k_r[3] \oplus k_r[23], k_r[25] \oplus k_r[29], k_r[27] \oplus k_r[31])$.

If there exists an r_0 such as $P_{\mathbf{r_0}}$ and $P'_{\mathbf{r_0}\pm 1}$ are known, then for all $i \in \mathbb{Z}$, the bytes $P_{\mathbf{r_0}+2i}$ and $P'_{\mathbf{r_0}+2i+1}$ are known (and they are easily computable).

Properties on the AES Key Schedule



Representation of the position of the bytes of the proposition.

In cases (2), only the XOR of the two bytes of the same color must be known.

Table of contents

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

→ **Attacks on mixFeed and ALE**: they exploit the presence of short length cycles when we iterate an odd number of rounds of key schedule.

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

→ **Attacks on mixFeed and ALE**: they exploit the presence of short length cycles when we iterate an odd number of rounds of key schedule.

→ **Properties** on the AES key schedule.

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

→ **Attacks on mixFeed and ALE**: they exploit the presence of short length cycles when we iterate an odd number of rounds of key schedule.

→ **Properties** on the AES key schedule.

→ **Improvement of the Impossible Differential** cryptanalysis against the **AES** by **combining more efficiently** information from subkeys.

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

→ **Attacks on mixFeed and ALE**: they exploit the presence of short length cycles when we iterate an odd number of rounds of key schedule.

→ **Properties** on the AES key schedule.

→ **Improvement of the Impossible Differential** cryptanalysis against the **AES** by **combining more efficiently** information from subkeys.

→ It confirms that the **key schedule is probably the least safe part** of AES, and should not be considered as a random permutation.

Conclusion

→ **Alternatives representations** of AES 128, 192 and 256 key schedule.

→ **Attacks on mixFeed and ALE**: they exploit the presence of short length cycles when we iterate an odd number of rounds of key schedule.

→ **Properties** on the AES key schedule.

→ **Improvement of the Impossible Differential** cryptanalysis against the **AES** by **combining more efficiently** information from subkeys.

→ It confirms that the **key schedule is probably the least safe part** of AES, and should not be considered as a random permutation.

For more details:

`https://eprint.iacr.org/2020/1253`