New representations of the AES Key Schedule

Gaëtan Leurent, Clara Pernot
Inria, Paris

Monday, October 18th 2021
The AES is the most widely used block cipher today.

Winner of the AES competition.

Subset of Rijndael block cipher.

Designed by Rijmen and Daemen.

**Block size:** 128 bits.

**Key size:** 128, 192, 256 bits.

\[\text{plaintext} \xrightarrow{\text{Round function}} \text{ciphertext}\]

\[\text{Key schedule}\]

\[K_0, K_1, K_9, K_{10}, R_1, R_2, R_{10}, \ldots\]

\[\oplus\]

\[\oplus\]

\[\oplus\]

\[\oplus\]
The AES is the most widely used block cipher today.
Winner of the AES competition.
Subset of Rijndael block cipher.
Designed by Rijmen and Daemen.
Block size: 128 bits.
Key size: 128, 192, 256 bits.

Description of the AES-128.
The AES is the most widely used block cipher today.
Winner of the AES competition.
Subset of Rijndael block cipher.
Designed by Rijmen and Daemen.
Block size: 128 bits.
Key size: 128, 192, 256 bits.

After 20 years of cryptanalysis:
- only 7 rounds out of 10 are broken.
- the key schedule is known to cause issues in the related-key setting.
The AES is the most widely used block cipher today.

- Winner of the AES competition.
- Subset of **Rijndael** block cipher.
- Designed by Rijmen and Daemen.
- **Block size**: 128 bits.
- **Key size**: 128, 192, 256 bits.

**After 20 years of cryptanalysis:**
- only **7 rounds out of 10** are broken.
- the **key schedule** is known to cause issues in the related-key setting.
One round of the AES key schedule.

- Non-linear part: Sbox
- Feistel network structure

Impression:
all bytes are mixed!
Our results

*Alternative representations* of the AES key schedules

Even after a large number of rounds, the key schedule does not mix all the bytes!
Our results

- **Alternative representations** of the AES key schedules
  
  Even after a large number of rounds, the key schedule does not mix all the bytes!

- **Short length cycles** when iterating an odd number of rounds of key schedule
  - Attacks on *mixFeed* and ALE
Our results

- **Alternative representations** of the AES key schedules
  - Even after a large number of rounds, the key schedule does not mix all the bytes!

- **Short length cycles** when iterating an odd number of rounds of key schedule
  - Attacks on *mixFeed* and ALE

- **Efficient combination of information** from subkeys
  - Improvement of *Impossible Differential* and Square attacks against the AES
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion

Gaëtan Leurent and Clara Pernot
New representations of the AES Key Schedule
Difference diffusion

**Invariant subspaces**: a subspace $A$ s.t. it exists an offset $u$ that verifies:

$$F(A + u) = A + F(u)$$

Subspace trails: a subspace $A$ s.t. for all offset $u$:

$$F(A + u) = B + F(u)$$

→ 4 families of subspace trails whose linear parts are:

**$E_0$**

$$E_0 = \{(a, b, c, d, 0, b, 0, d, a, 0, 0, 0, d) \mid a, b, c, d \in \mathbb{F}_2^8\}$$

**$E_1$**

$$E_1 = \{(a, b, c, d, a, 0, c, 0, 0, 0, c, d) \mid a, b, c, d \in \mathbb{F}_2^8\}$$

**$E_2$**

$$E_2 = \{(a, b, c, d, 0, b, 0, d, 0, b, 0, 0, 0, d) \mid a, b, c, d \in \mathbb{F}_2^8\}$$

**$E_3$**

$$E_3 = \{(a, b, c, d, a, 0, c, 0, a, b, 0, 0, 0, 0) \mid a, b, c, d \in \mathbb{F}_2^8\}$$

∀ $u \in (\mathbb{F}_2^8)^{16}$, $F(E_i + u) = E_i + 1 + F(u)$

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

$$(\mathbb{F}_2^8)^{16} = E_0 \oplus E_1 \oplus E_2 \oplus E_3$$
Difference diffusion

**Invariant subspaces:** a subspace $A$ s.t. it exists an offset $u$ that verifies:

\[ F(A + u) = A + F(u) \]

**Subspace trails:** a subspace $A$ s.t. for all offset $u$:

\[ F(A + u) = B + F(u) \]
Difference diffusion

**Invariant subspaces**: a subspace $A$ s.t. it exists an offset $u$ that verifies:

$$F(A + u) = A + F(u)$$

**Subspace trails**: a subspace $A$ s.t. for all offset $u$:

$$F(A + u) = B + F(u)$$

→ 4 families of subspace trails whose linear parts are:

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

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

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

$$(\mathbb{F}_{2^8})^{16} = E_0 \oplus E_1 \oplus E_2 \oplus E_3$$
New representation of the AES Key Schedule

We perform a linear transformation $A$, which corresponds to a basis change:

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

$\Rightarrow$ The 4 subspaces appear more clearly!
New representation of the AES Key Schedule

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

- 4 subspace trails
- 4 independent functions

The key schedule does not mix all the bytes!
New representation of the AES Key Schedule

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

$r$ rounds of the key schedule in the new representation.
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion
mixFeed [Chakraborty and Nandi, NIST LW Submission]

- mixFeed was a **second-round candidate** in the NIST Lightweight Standardization Process which was **not selected as a finalist**
- Submitted by Bishwajit Chakraborty and Mridul Nandi
- **AEAD** (Authenticated Encryption with Associated Data) algorithm
- Based on the AES block cipher
Simplified scheme of mixFeed encryption.
Function Feed in the case where $|D| = 128$
Simplified scheme of mixFeed encryption.

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

$P$: 11 rounds of key schedule

$P$ is iterated $\rightarrow$ we study its cycles!
Mustafa Khairallah’s observation [ToSC’19]

Using brute-force and out of 33 tests, Khairallah found 20 cycles of length \(14018661024 \approx 2^{33.7}\) for the P permutation.

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
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion
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:

- $f_1 = B_{11} \circ B \circ B \circ B \circ B_7 \circ B \circ B \circ B \circ B_3 \circ B \circ B$
- $f_2 = B \circ B_{10} \circ B \circ B \circ B \circ B \circ B_6 \circ B \circ B \circ B \circ B_2 \circ B$
- $f_3 = B \circ B \circ B_9 \circ B \circ B \circ B \circ B_5 \circ B \circ B \circ B \circ B_1$
- $f_4 = B \circ B \circ B \circ B_8 \circ B \circ B \circ B \circ B_4 \circ B \circ B \circ B \circ B$
Cycle analysis of 11-round AES key schedule

4 iterations of $P$ in the new model.
Cycle analysis of 11-round AES key schedule

\[ \tilde{P}^4 : (a, b, c, d) \mapsto (\phi_1(a), \phi_2(b), \phi_3(c), \phi_4(d)) \]

4 iterations of P in the new model.
Cycle analysis of 11-round AES key schedule

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

- The length of the small cycles divide the length of the big cycle.

4 iterations of \( P \) in the new model.
Cycle analysis of 11-round AES key schedule

\[ \tilde{P}^4 : (a, b, c, d) \mapsto (\phi_1(a), \phi_2(b), \phi_3(c), \phi_4(d)) \]

- The length of the small cycles divide the length of the big cycle.
- The length of the big cycles is the lowest common multiple of the length of the small cycles.

4 iterations of P in the new model.
Cycle analysis of 11-round AES key schedule

\[ \tilde{P}^4 : (a, b, c, d) \mapsto (\phi_1(a), \phi_2(b), \phi_3(c), \phi_4(d)) \]

- The length of the small cycles divide the length of the big cycle.
- The length of the big cycles is the lowest common multiple of the length of the small cycles.
- The \( \phi_i \) functions have the same cycle structure:
  \[
  \begin{align*}
  \phi_2 &= f_2^{-1} \circ \phi_1 \circ f_2 \\
  \phi_3 &= f_3^{-1} \circ \phi_2 \circ f_3 \\
  \phi_4 &= f_4^{-1} \circ \phi_3 \circ f_4
  \end{align*}
  \]

4 iterations of \( P \) in the new model.
Cycle analysis of 11-round AES key schedule

We study the 32-bit permutation $\phi_1$ and we obtain that:

$\rightarrow$ With probability $82\%$: $a$ is in the largest cycle of $\phi_1$ of length $\ell$

The same for $\phi_2$, $\phi_3$, and $\phi_4$.

$\rightarrow$ With probability $45\%$: $(a, b, c, d)$ is in a cycle of length $\ell$ for $\tilde{P}^4$

$\rightarrow$ With probability $45\%$: $(a, b, c, d)$ is in a cycle of length $4\ell$ for $P$
Cycle analysis of 11-round AES key schedule

Summary: 45% of keys belong to cycles of length $14018661024 \approx 2^{33.7}$. This explains the observation on mixFeed [Khairallah, ToSC’19]. This allows to make a forgery against mixFeed. 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 $\ell$ is at most $\ell/2^n/2$. 
Cycle analysis of 11-round AES key schedule

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

→ 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 $14018661024 \approx 2^{33.7}$.

→ This explains the observation on mixFeed [Khairallah, ToSC’19].

→ This allows to make a forgery against mixFeed.
Cycle analysis of 11-round AES key schedule

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

→ This explains the observation on mixFeed [Khairallah, ToSC’19].

→ This allows to make a forgery against mixFeed.

→ 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 $\ell$ is at most $\ell/2^{n/2}$.
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   - Description of mixFeed
   - The Explanation of Short Cycles
   - Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   - Application to AES - Impossible Differential
   - Generalisation and Results

5 Conclusion
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 $(M, C, T)$.

Khairallah proposed a forgery attack against mixFeed:

- we assume that $Z$ belongs to a **cycle** of length $\ell$
- we choose a message $M$ made of $m$ blocks, with $m > \ell$
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 $(M, C, T)$.

Khairallah proposed a forgery attack against mixFeed:
- we assume that $Z$ belongs to a **cycle** of length $\ell$
- we choose a message $M$ made of $m$ blocks, with $m > \ell$

![Diagram of the forgery attack](image)
The goal of a **forgery attack** is to forge a valid tag $T'$ for a new ciphertext $C'$ using $(M, C, T)$.

Khairallah proposed a forgery attack against mixFeed:
- we assume that $Z$ belongs to a **cycle** of length $\ell$
- we choose a message $M$ made of $m$ blocks, with $m > \ell$

(1) Cut

(2) Paste
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

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   - Description of mixFeed
   - The Explanation of Short Cycles
   - Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   - Application to AES - Impossible Differential
   - Generalisation and Results

5 Conclusion
Impossible Differential – AES

The attack is in 2 parts:

(1) find candidates for the key 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

The attack is in 2 parts:

1. find candidates for the key 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.

7-round impossible differential attack ([MDRM, IC’10]).

Figure adapted from Tikz for Cryptographers [Jean].
Matching bytes from $K^0$ and $K^7$

Naively:

- Guess 6 bytes of $K^0$
- Filter using 4 bytes of $K^7$

__________________________

Complexity: $2^{48}$
Matching bytes from $K^0$ and $K^7$

Naively:
- Guess 6 bytes of $K^0$
- Filter using 4 bytes of $K^7$

Complexity: $2^{48}$

Improvement:
- Guess 2 bytes of $K^0$
- Filter using 2 bytes of $K^7$
- Guess 2 bytes of $K^0$
- Filter using 1 byte of $K^7$
- Guess 1 byte of $K^0$
- Deduce 1 byte of $K^0$ from $K^7$

Complexity: $4 \times 2^{16}$
Matching bytes from $K^0$ and $K^7$
Matching bytes from $K^0$ and $K^7$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

$K^0$

<table>
<thead>
<tr>
<th>$K_{15}$</th>
<th>$K_{14}$</th>
<th>$K_{13}$</th>
<th>$K_{12}$</th>
<th>$K_{11}$</th>
<th>$K_{10}$</th>
<th>$K_{9}$</th>
<th>$K_{8}$</th>
<th>$K_{7}$</th>
<th>$K_{6}$</th>
<th>$K_{5}$</th>
<th>$K_{4}$</th>
<th>$K_{3}$</th>
<th>$K_{2}$</th>
<th>$K_{1}$</th>
<th>$K_{0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$K^7$

<table>
<thead>
<tr>
<th>$K_{15}$</th>
<th>$K_{14}$</th>
<th>$K_{13}$</th>
<th>$K_{12}$</th>
<th>$K_{11}$</th>
<th>$K_{10}$</th>
<th>$K_{9}$</th>
<th>$K_{8}$</th>
<th>$K_{7}$</th>
<th>$K_{6}$</th>
<th>$K_{5}$</th>
<th>$K_{4}$</th>
<th>$K_{3}$</th>
<th>$K_{2}$</th>
<th>$K_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How to compute $K^7_{12}$ from $K^0$?

<table>
<thead>
<tr>
<th>$K_{15}$</th>
<th>$K_{14}$</th>
<th>$K_{13}$</th>
<th>$K_{12}$</th>
<th>$K_{11}$</th>
<th>$K_{10}$</th>
<th>$K_{9}$</th>
<th>$K_{8}$</th>
<th>$K_{7}$</th>
<th>$K_{6}$</th>
<th>$K_{5}$</th>
<th>$K_{4}$</th>
<th>$K_{3}$</th>
<th>$K_{2}$</th>
<th>$K_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{15}$</th>
<th>$K_{14}$</th>
<th>$K_{13}$</th>
<th>$K_{12}$</th>
<th>$K_{11}$</th>
<th>$K_{10}$</th>
<th>$K_{9}$</th>
<th>$K_{8}$</th>
<th>$K_{7}$</th>
<th>$K_{6}$</th>
<th>$K_{5}$</th>
<th>$K_{4}$</th>
<th>$K_{3}$</th>
<th>$K_{2}$</th>
<th>$K_{1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Matching bytes from $K^0$ and $K^7$

How to compute $K_{12}^7$ from $K^0$?

$K^0$

$K^7$
Matching bytes from $K^0$ and $K^7$

How to compute $K_{12}^7$ from $K^0$?

\[
\begin{array}{c}
K^0 \\
\hline
G & G & G \\
G & G & G \\
G & G & G \\
G & G & G \\
\hline
\end{array}
\]

\[
\begin{array}{c}
K^7 \\
\hline
G & & \\
G & & \\
G & & \\
\hline
\end{array}
\]

\[
\begin{array}{c}
K_{14} \\ K_{12} \\ K_{13} \\ K_2 \\
\hline
K_{14} \\ K_{12} \\ K_{13} \\ K_2 \\
\hline
\end{array}
\]

\[
\begin{array}{c}
f_1 \\
\hline
f_2 \\
\hline
f_3 \\
\hline
f_4 \\
\hline
\end{array}
\]

Gaëtan Leurent and Clara Pernot
New representations of the AES Key Schedule
Matching bytes from $K^0$ and $K^7$

We can filter using $K^7_{12}$ by guessing only 2 bytes of $K^0$!
Matching bytes from $K^0$ and $K^7$
Matching bytes from $K^0$ and $K^7$

All the input of $f_3$ is known, so the output is also known.
Matching bytes from $K^0$ and $K^7$

All the input of $f_3$ is known, so the output is also known
Matching bytes from $K^0$ and $K^7$

<table>
<thead>
<tr>
<th>G</th>
<th>G</th>
<th>G</th>
<th>X</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

$K^0$

- $K_{15} \oplus K_{10} \oplus K_{13} \oplus K_{12} \oplus K_2$
- $K_{14} \oplus K_6 \oplus K_5 \oplus K_8$

$K^7$

- $K_{15} \oplus K_{14} \oplus K_7 \oplus K_3 \oplus K_6 \oplus K_9$
- $K_{13} \oplus K_{12} \oplus K_2 \oplus K_5 \oplus K_8$

$f_1$

$f_2$

$f_3$

$f_4$
Matching bytes from $K^0$ and $K^7$

We are also able to filter according to $K_6^7 = (K_{14}^7 \oplus K_6^7) \oplus K_{14}^7$

$$
\begin{array}{c|cc|c|c|c|c}
 & G & G & G \\
G & G & G & X \\
G & G & G & X \\
G & G & G & & \\
\end{array}

\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c}
K^0 & K_15 & K_{14} & K_{13} & K_{12} & K_6 & K_{10} & K_5 & K_8 & K_2 \\
\end{array}

\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c}
K^7 & K_15 & K_{14} & K_{13} & K_{12} & K_6 & K_{10} & K_5 & K_8 & K_2 \\
\end{array}

\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c}
K^0 & f_1 & f_2 & f_3 & f_4 \\
\end{array}

Gaëtan Leurent and Clara Pernot
New representations of the AES Key Schedule
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   - Description of mixFeed
   - The Explanation of Short Cycles
   - Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   - Application to AES - Impossible Differential
   - Generalisation and Results

5 Conclusion
Generalisation

Using our **new representation** of the key schedule, we demonstrate that:

→ 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

**Even after a large number of rounds, the key schedule does not mix all the bytes!**
## Results

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

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

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

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

We also slightly improve the time and data complexities of:

- Related-Key Impossible Differential Attacks against AES-192
- Impossible Differential against Rijndael-256/256
- Square Attack against AES-192
Table of contents

1 Introduction

2 A New Representation of the AES-128 Key Schedule

3 Short Length Cycles
   • Description of mixFeed
   • The Explanation of Short Cycles
   • Forgery Attack against mixFeed

4 Combining Efficiently Information from Subkeys
   • Application to AES - Impossible Differential
   • Generalisation and Results

5 Conclusion
Conclusion

→ **Alternatives representations** of AES key schedules:
  - 128 bits: 4 chunks of 4 bytes
  - 192 bits: 2 chunks of 12 bytes
  - 256 bits: 4 chunks of 8 bytes
Conclusion

→ **Alternatives representations** of AES key schedules:
   - 128 bits: 4 chunks of 4 bytes
   - 192 bits: 2 chunks of 12 bytes
   - 256 bits: 4 chunks of 8 bytes

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

→ **Alternatives representations** of AES key schedules:
  - 128 bits: 4 chunks of 4 bytes
  - 192 bits: 2 chunks of 12 bytes
  - 256 bits: 4 chunks of 8 bytes

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

→ **Improvement of Impossible Differential** and **Square** attacks against the **AES** by combining efficiently information from subkeys.

→ For more details: [https://eprint.iacr.org/2020/1253](https://eprint.iacr.org/2020/1253)
Conclusion

→ **Alternatives representations** of AES key schedules:
  - 128 bits: 4 chunks of 4 bytes
  - 192 bits: 2 chunks of 12 bytes
  - 256 bits: 4 chunks of 8 bytes

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

→ **Improvement of Impossible Differential** and **Square** attacks against the **AES** by combining efficiently information from subkeys.

→ It confirms that the key schedule should not be considered as a random permutation.
Conclusion

→ **Alternatives representations** of AES key schedules:
  - 128 bits: 4 chunks of 4 bytes
  - 192 bits: 2 chunks of 12 bytes
  - 256 bits: 4 chunks of 8 bytes

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

→ **Improvement of Impossible Differential** and **Square** attacks against the AES by combining efficiently information from subkeys.

→ It confirms that the key schedule should not be considered as a random permutation.

For more details:

https://eprint.iacr.org/2020/1253
Difference diffusion

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

<table>
<thead>
<tr>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
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.
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.

Gaëtan Leurent and Clara Pernot
New representations of the AES 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.
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.

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

→ Short length cycles allows us to easily find states encrypted under the same key.

→ 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.

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

→ Short length cycles allows us to easily find states encrypted under the same key.

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

<table>
<thead>
<tr>
<th>Attack</th>
<th>Enc</th>
<th>Verif</th>
<th>Time</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existential Forgery Known Plaintext</td>
<td>$2^{110.4}$</td>
<td>$2^{102}$</td>
<td>$2^{110.4}$</td>
<td>[WWHWW, AC’13]</td>
</tr>
<tr>
<td>Existential Forgery Known Plaintext</td>
<td>$2^{103}$</td>
<td>$2^{103}$</td>
<td>$2^{104}$</td>
<td>[KR, SAC’13]</td>
</tr>
<tr>
<td>Existential Forgery Known Plaintext</td>
<td>1</td>
<td>$2^{120}$</td>
<td>$2^{120}$</td>
<td>[KR, SAC’13]</td>
</tr>
<tr>
<td>State Recovery, Almost Univ. Forgery</td>
<td>Known Plaintext</td>
<td>1</td>
<td>$2^{121}$</td>
<td>[KR, SAC’13]</td>
</tr>
<tr>
<td>State Recovery, Almost Univ. Forgery</td>
<td>Chosen Plaintext</td>
<td>$2^{57.3}$</td>
<td>0</td>
<td>$2^{104.4}$</td>
</tr>
</tbody>
</table>

Comparison of attacks against ALE.
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

\[ K^{i-4j} \]

\[ f_1 \]

\[ f_2 \]

\[ f_3 \]

\[ f_4 \]

\[ K^i \]

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.

How to compute \( K^{i-4j} \)?

How to compute \( K^i \)?
Property on the AES Key Schedule

How to compute $K_{14}^i$?

Gaëtan Leurent and Clara Pernot
New representations of the AES Key Schedule
Property on the AES Key Schedule

How to compute $K_{14}^i$?

$K_{i-4j}$

$K^i$

$K_{14}^i$

Gaëtan Leurent and Clara Pernot

New representations of the AES Key Schedule
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

How to compute $K^i_{14}$?

How to compute $K^i_8$?

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

$K^i_j$ depends on:
- 128 bits of information.
- 64 bits of information for columns 1-2.
- 32 bits of information for column 3.

→ A byte in the last column depends on only 32 bits of information.
Property on the AES Key Schedule

How to compute $K^i_i$?

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

→ A byte in the last column depends on only 32 bits of information.
Property on the AES Key Schedule

How to compute $K^i_{14}$?

$K^i_{14} = (K^i_8 \oplus K^i_{12}) \oplus K^i_{12}$

→ A byte in the last column depends on only 32 bits of information.
Property on the AES Key Schedule

\[ K^{i-4j} \]

\[ f_1 \]

\[ f_2 \]

\[ f_3 \]

\[ f_4 \]

\[ K^i \]

\[ K^i_8 \oplus K^i_{12} \]

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

→ 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

$K^{i-4j}$

$K^i$

→ 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

$K^{i-4j}$

- $f_1$
- $f_2$
- $f_3$
- $f_4$

$K^i$

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

- 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.
- 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.
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.
Square Attack
### Other Results

<table>
<thead>
<tr>
<th>Attack</th>
<th>Cipher</th>
<th>Rounds</th>
<th>Data</th>
<th>Time</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>AES-192</td>
<td>8/12</td>
<td>$2^{128} - 2^{119}$</td>
<td>$2^{188}$</td>
<td>[FKL+, FSE'00]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{128} - 2^{119}$</td>
<td>$2^{187.3}$</td>
<td>Variant of [FKL+, FSE'00]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{128} - 2^{119}$</td>
<td>$2^{187.5}$</td>
<td>Variant of [DKS, AC'10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{128} - 2^{119}$</td>
<td>$2^{185.1}$</td>
<td>New</td>
</tr>
<tr>
<td>Related-Key Impossible Differential</td>
<td>AES-192</td>
<td>7/12</td>
<td>$2^{64.5}$</td>
<td>$2^{177}$</td>
<td>[ZWZ+, SAC’06]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{63.5}$</td>
<td>$2^{175}$</td>
<td>New</td>
</tr>
<tr>
<td>Impossible Differential</td>
<td>Rijndael-256/256</td>
<td>9/14</td>
<td>$2^{229.3}$</td>
<td>$2^{194}$</td>
<td>[WGR+, ICISC’12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{228.1}$</td>
<td>$2^{192.9}$</td>
<td>Variant of [WGR+, ICISC’12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{227.7}$</td>
<td>$2^{192.5}$</td>
<td>New</td>
</tr>
<tr>
<td>Impossible Differential</td>
<td>Rijndael-256/256</td>
<td>10/14</td>
<td>$2^{244.2}$</td>
<td>$2^{253.9}$</td>
<td>[WGR+, ICISC’12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{243.9}$</td>
<td>$2^{253.6}$</td>
<td>Variant of [WGR+, ICISC’12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$2^{243}$</td>
<td>$2^{252.7}$</td>
<td>New</td>
</tr>
</tbody>
</table>