New representations of the AES Key Schedule

Gaëtan Leurent, Clara Pernot
Inria, Paris

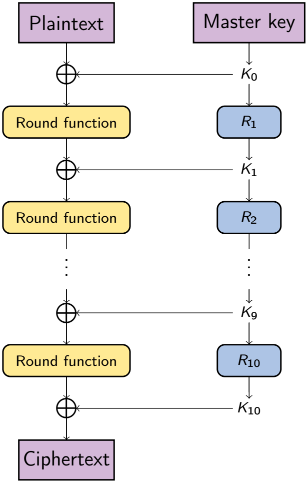
Thursday, April 14th 2022

AES: Advanced Encryption Standard [FIPS-197]

- 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.

AES: Advanced Encryption Standard [FIPS-197]

- 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.



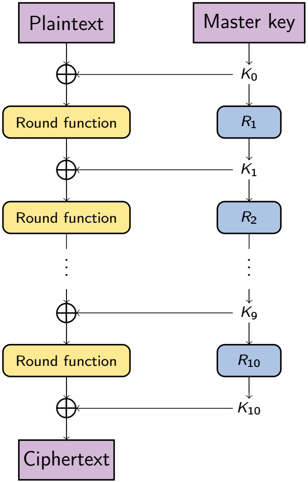
Description of the AES-128.

AES: Advanced Encryption Standard [FIPS-197]

- 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.



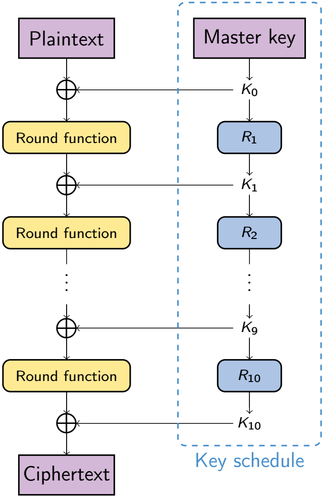
Description of the AES-128.

AES: Advanced Encryption Standard [FIPS-197]

- 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.

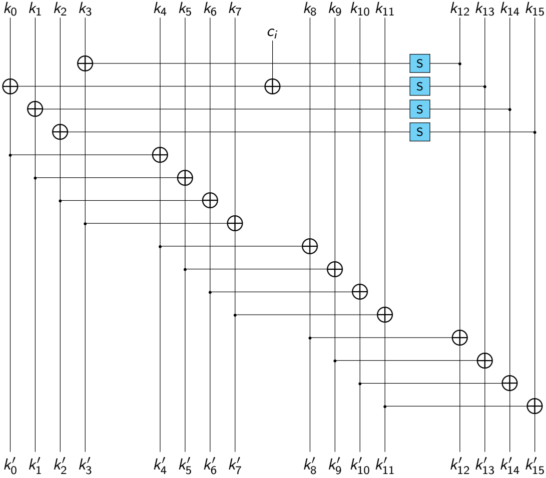
After 20 years of cryptanalysis:

- only **7 rounds out of 10** are broken.
- the **key schedule** is known to **cause issues** in the related-key setting.



Description of the AES-128.

AES key schedule



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

Difference diffusion

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

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

Difference diffusion

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

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

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

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

Difference diffusion

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

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

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

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

$\rightarrow$ **4 families of subspace trails** whose linear parts are:

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

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

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

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

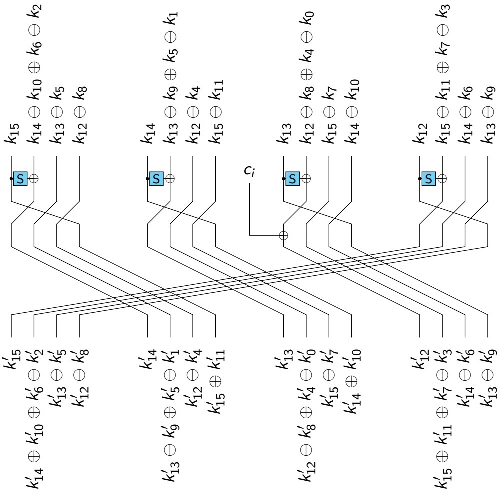
New representation of the AES Key Schedule

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

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

$\Rightarrow$ The **4 subspaces** appear more clearly!

New representation of the AES Key Schedule

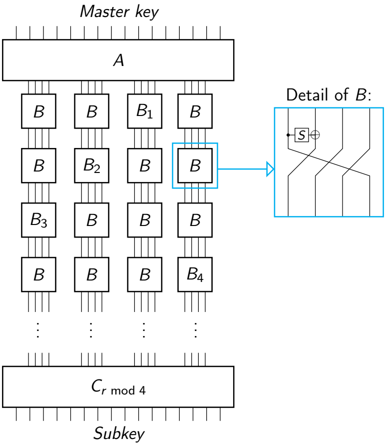


One round of the AES key schedule
(alternative representation).

- 4 subspace trails
- 4 independent functions

The key schedule does not mix all the bytes!

New representation of the AES Key Schedule



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.

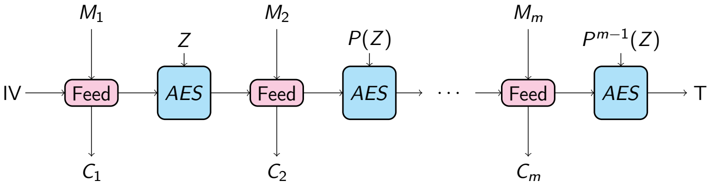
Table of contents

Table of contents

mixFeed [Chakraborty and Nandi, NIST LW Submission]

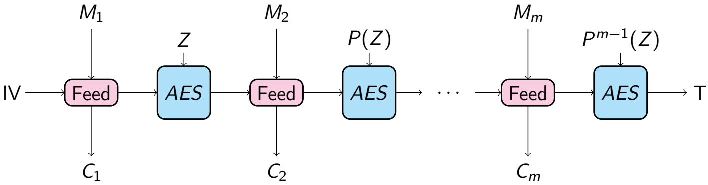
- mixFeed was a **second-round candidate** in the NIST Lightweight Standardization Process which was **not selected as a finalist**
- Submitted by Bishwajit **Chakraborty** and Mridul **Nandi**
- **AEAD** (Authenticated Encryption with Associated Data) algorithm
- Based on the AES block cipher

mixFeed

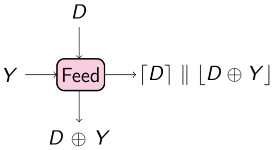


Simplified scheme of mixFeed encryption.

mixFeed



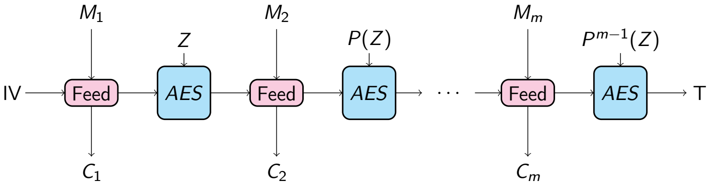
Simplified scheme of mixFeed encryption.



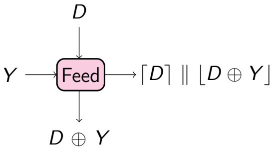
Function Feed in the case where

$|D| = 128$

mixFeed



Simplified scheme of mixFeed encryption.



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

P: **11 rounds** of key schedule

P is **iterated** $\rightarrow$ we study its **cycles**!

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**

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

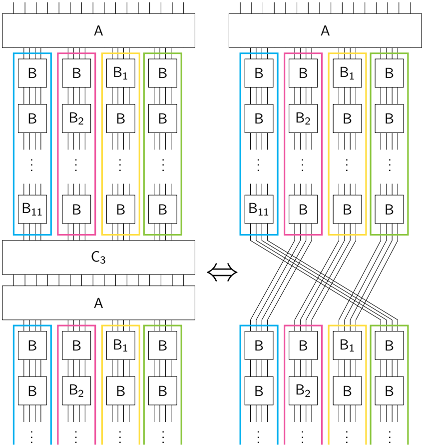
for the P permutation.

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

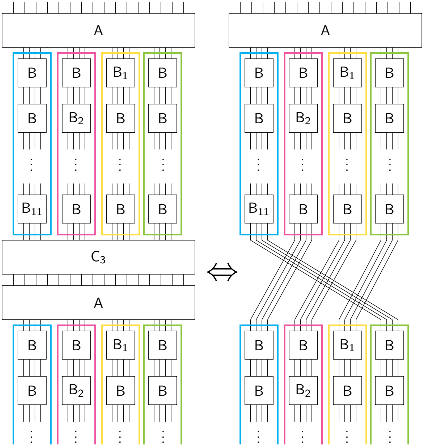
Table of contents

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:

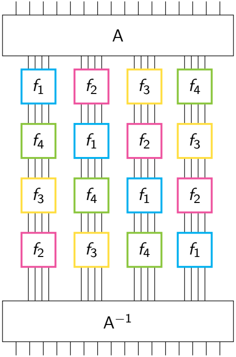
$f_1 = B_{11} \circ B \circ B \circ B \circ B_7 \circ B \circ B \circ B \circ B_3 \circ B \circ B$

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

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

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

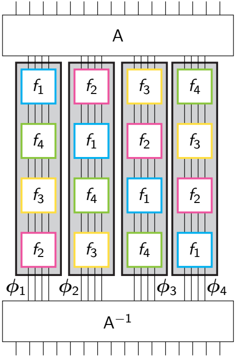
Cycle analysis of 11-round AES key schedule



4 iterations of P in the new model.

Cycle analysis of 11-round AES key schedule

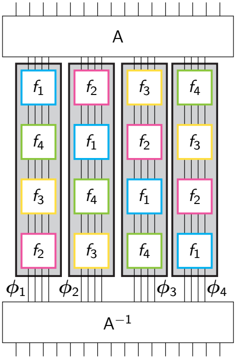
$$\widetilde{P}^4 : (a, b, c, d) \mapsto (\phi_1(a), \phi_2(b), \phi_3(c), \phi_4(d))$$



4 iterations of P in the new model.

Cycle analysis of 11-round AES key schedule

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

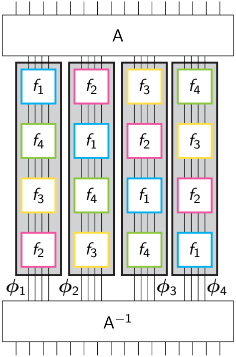


4 iterations of P in the new model.

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

Cycle analysis of 11-round AES key schedule

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

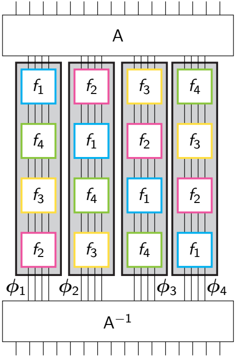


4 iterations of P in the new model.

- 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.

Cycle analysis of 11-round AES key schedule

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



4 iterations of P in the new model.

- 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 ϕ_i functions have the **same cycle structure**:

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

Cycle analysis of 11-round AES key schedule

We study the 32-bit permutation ϕ_1 and we obtain that:

→ With probability 82%: a is in the **largest cycle** of ϕ_1 of length ℓ
The same for ϕ_2, ϕ_3, and ϕ_4.

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

→ With probability 45%: (a, b, c, d) is in a cycle of length 4ℓ for 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 $\approx 2^{33.7}$.

→ 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.

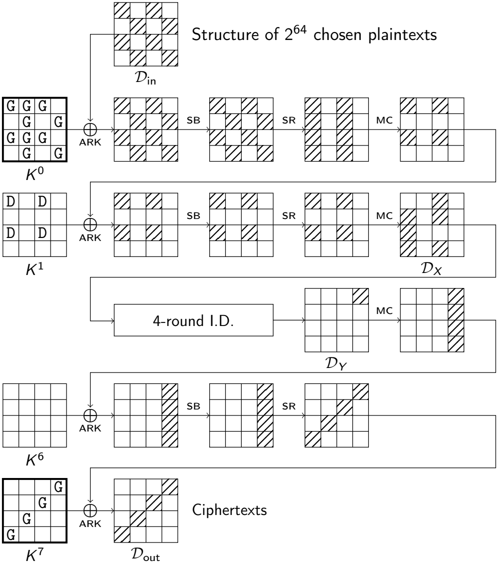
→ This contradicts the assumption made in a security proof of mixFeed:

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 ℓ is at most $\ell / 2^{n/2}$.

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 candidates for the key bytes marked G.

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

Matching bytes from K^0 and K^7

Given 10 bytes of K^0 and 4 bytes of K^7,
how to find the corresponding master keys?

Matching bytes from K^0 and K^7

Given 10 bytes of K^0 and 4 bytes of K^7,
how to find the corresponding master keys?

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

Given 10 bytes of K^0 and 4 bytes of K^7,
how to find the corresponding master keys?

Naively:

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

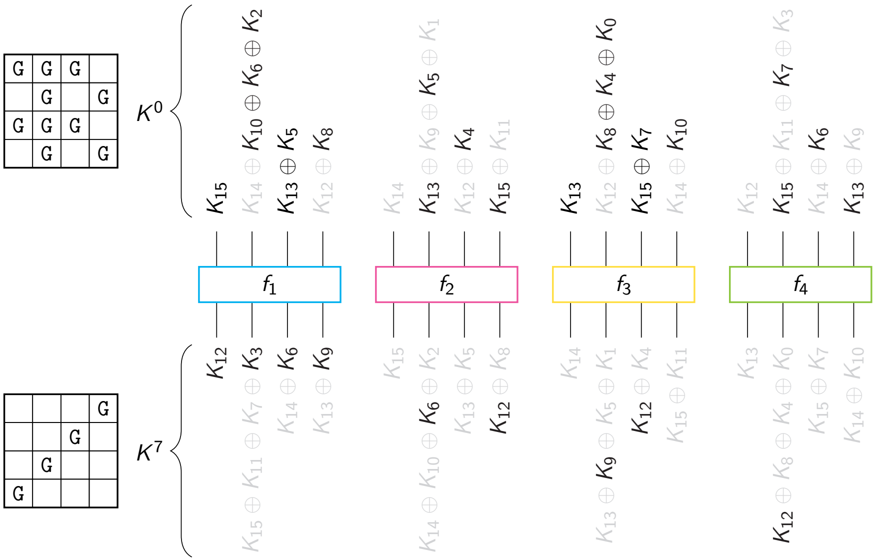
Complexity: 2^{48}

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

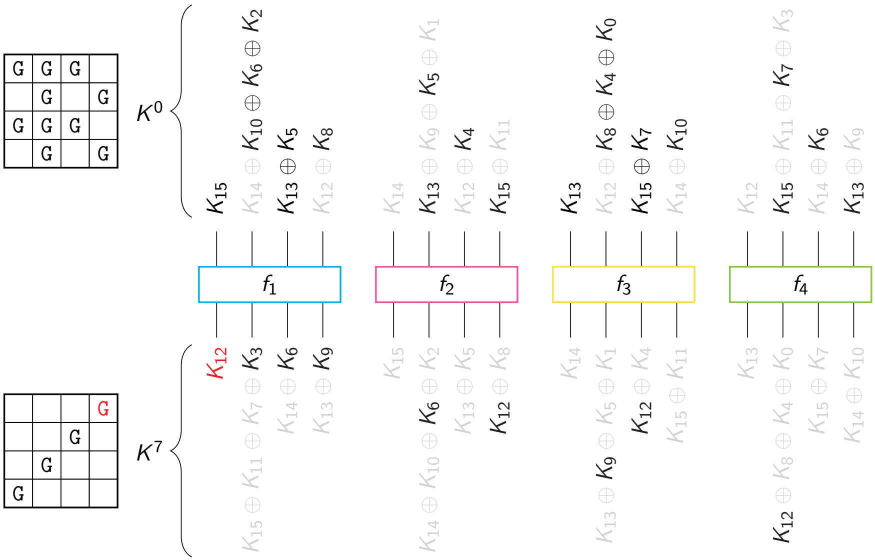
Complexity: 4×2^{16}

Matching bytes from K^0 and K^7



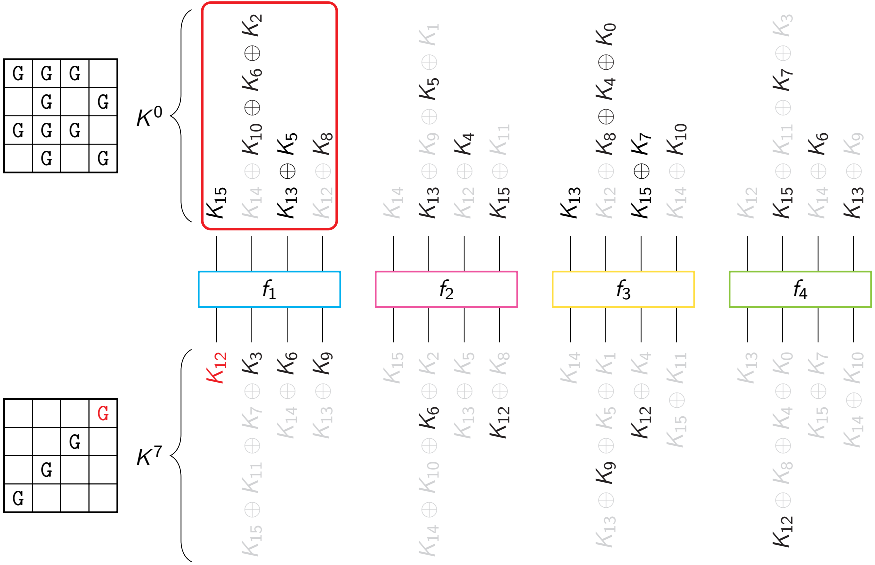
Matching bytes from K^0 and K^7

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



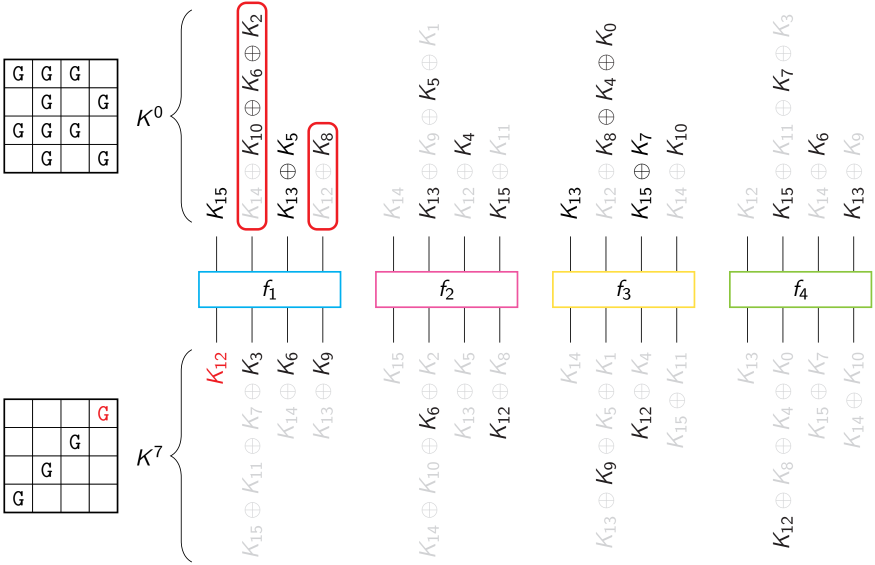
Matching bytes from K^0 and K^7

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



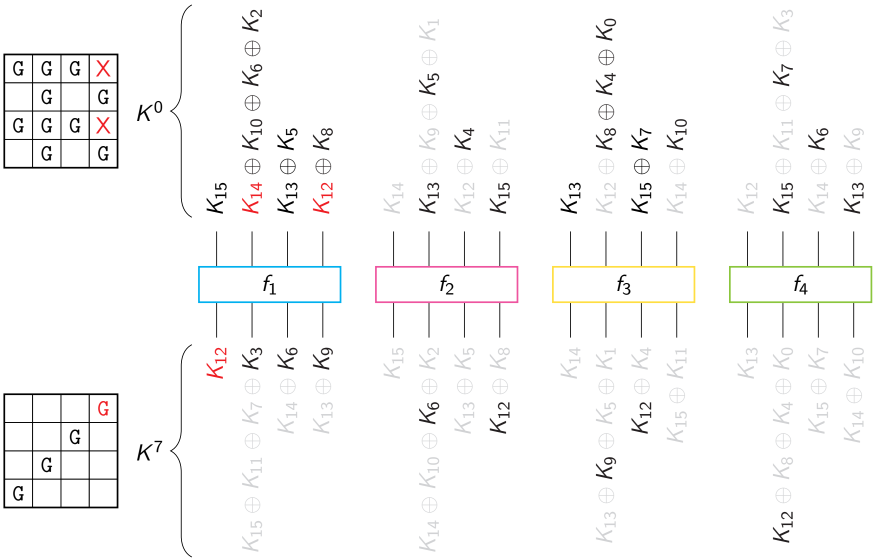
Matching bytes from K^0 and K^7

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

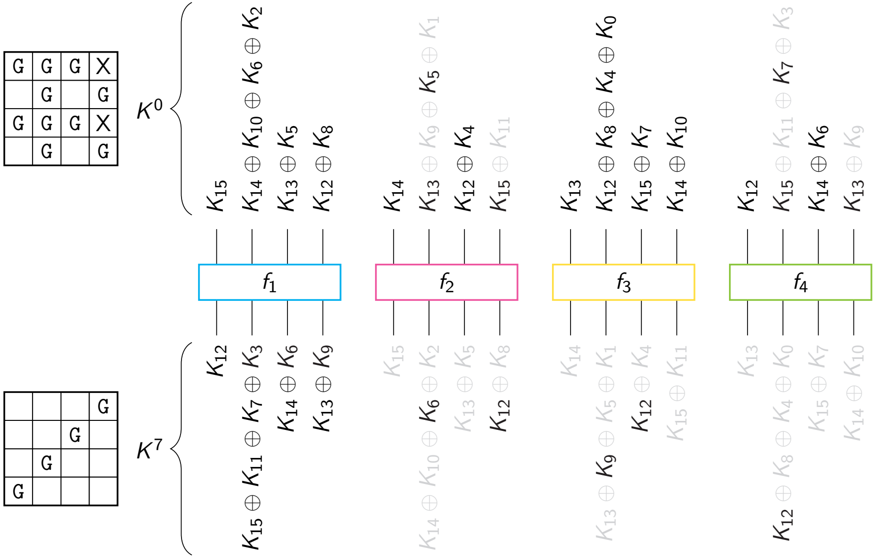


Matching bytes from K^0 and K^7

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

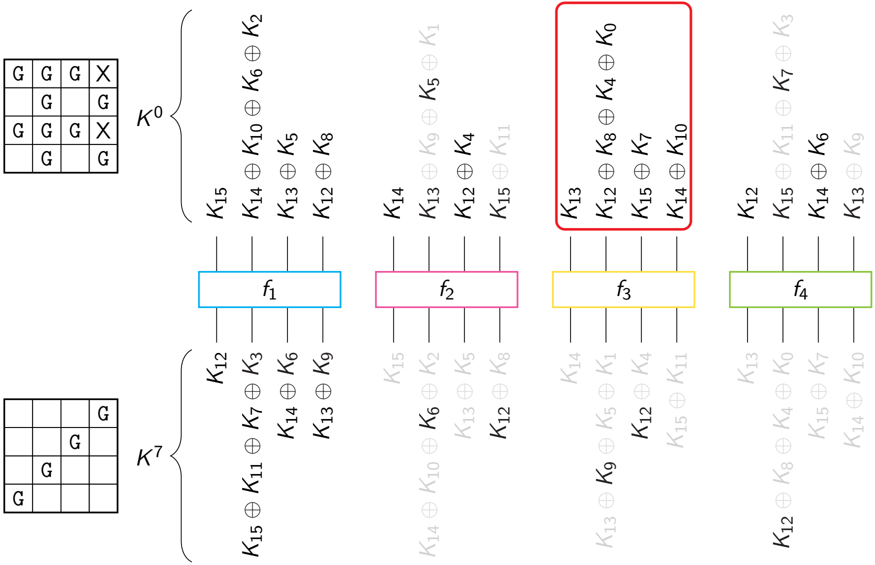


Matching bytes from K^0 and K^7



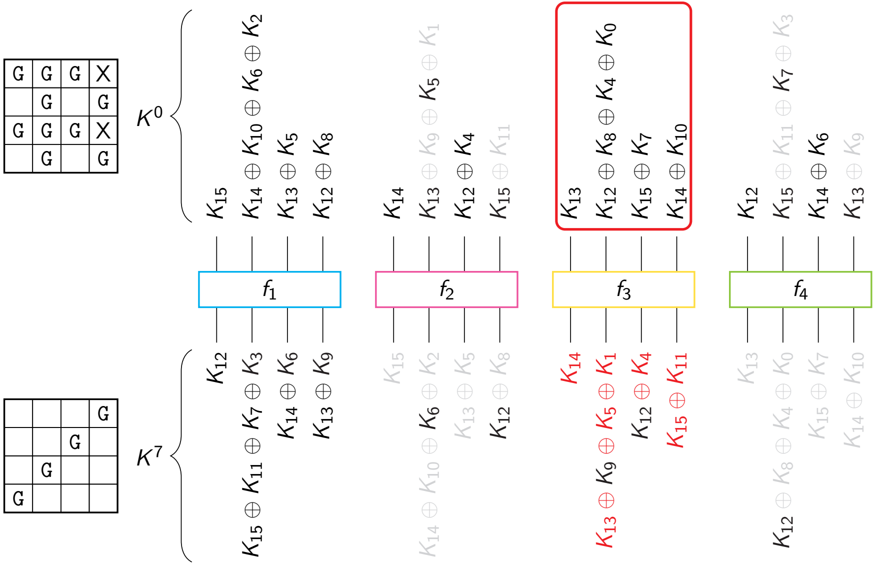
Matching bytes from K^0 and K^7

All the input of f_3 is known, so the output is also known

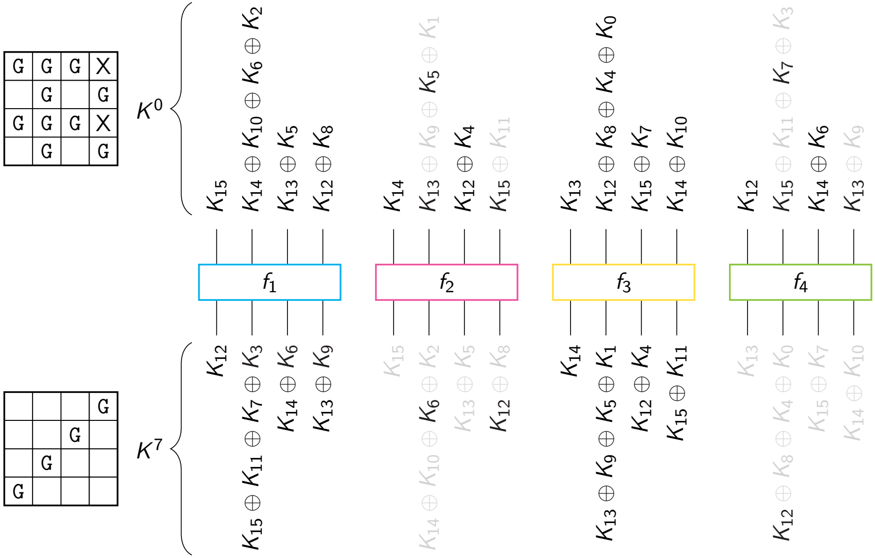


Matching bytes from K^0 and K^7

All the input of f_3 is known, so the output is also known



Matching bytes from K^0 and K^7



Matching bytes from K^0 and K^7

We are also able to filter according to $K^7_6 = (K^7_{14} \oplus K^7_6) \oplus K^7_{14}$

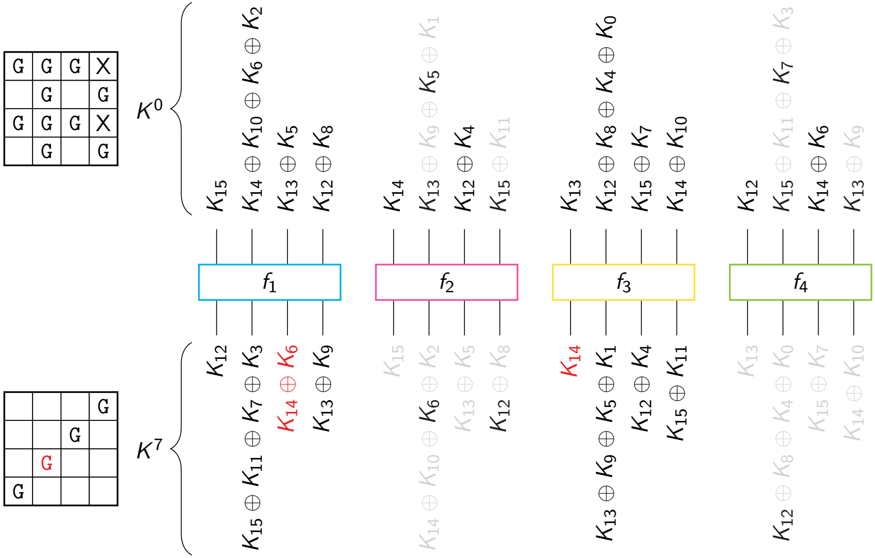


Table of contents

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.

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.

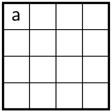
→ **Improvement of Impossible Differential** and **Square** attacks against the **AES** by **combining efficiently** information from subkeys.

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

For more details:

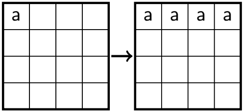
https://eprint.iacr.org/2020/1253

Difference diffusion



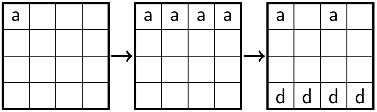
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



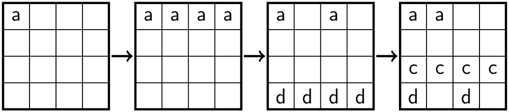
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



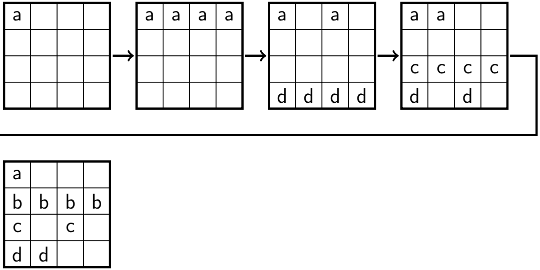
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



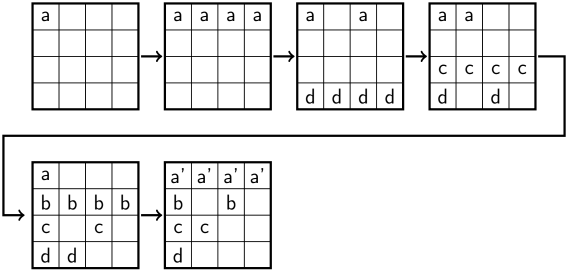
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



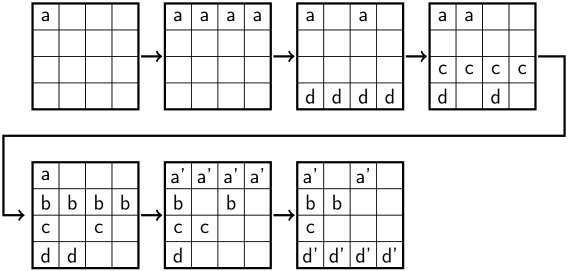
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



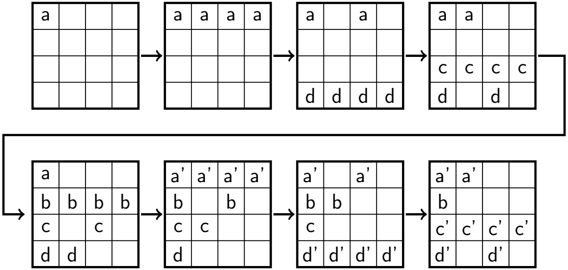
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



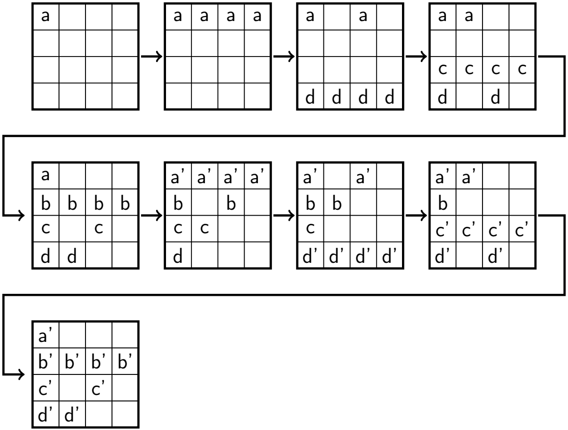
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



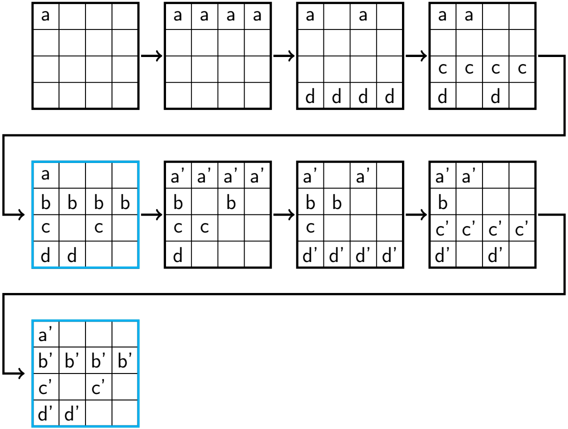
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



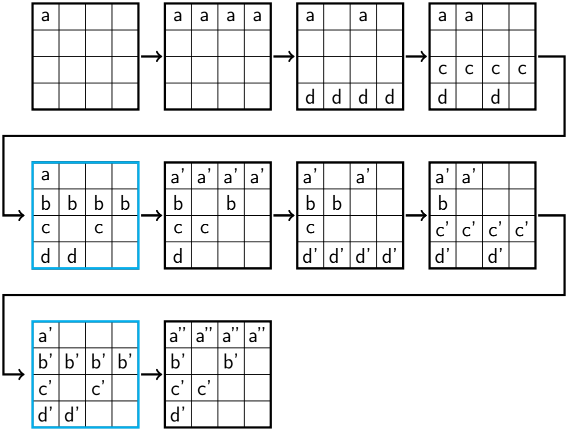
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



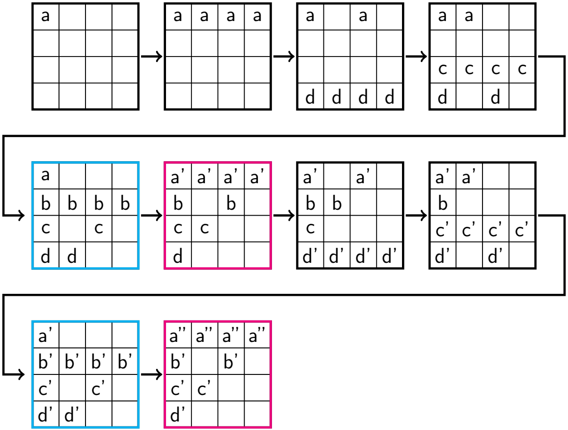
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



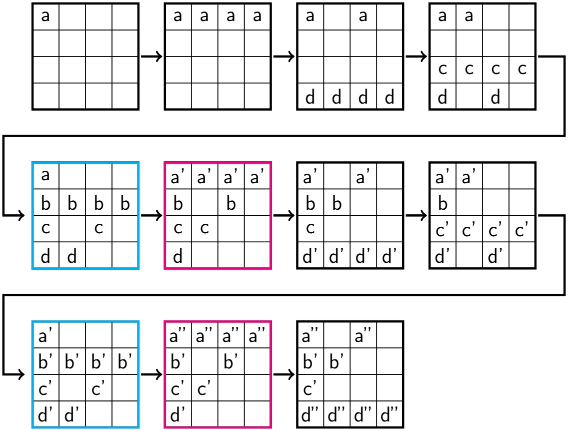
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



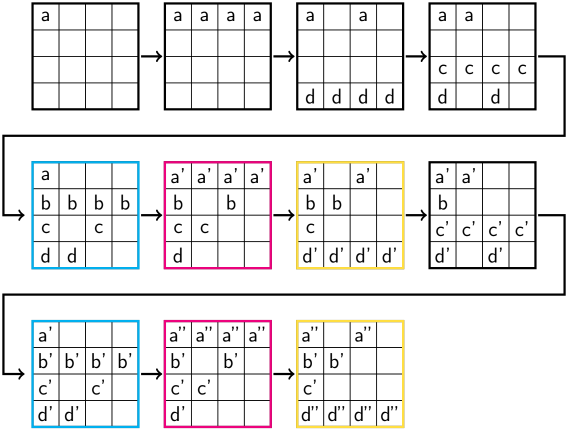
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



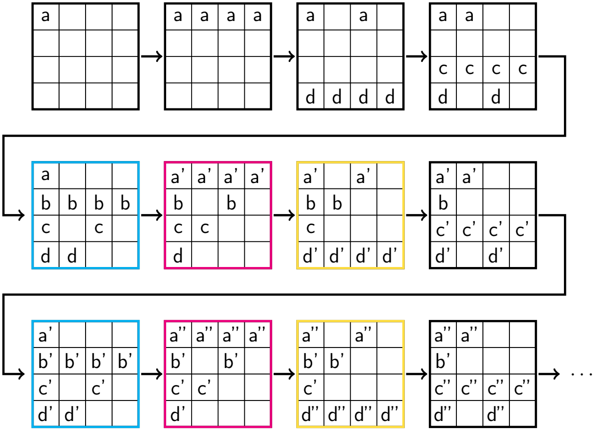
Diffusion of a difference on the first byte after several rounds of key schedule.

Difference diffusion



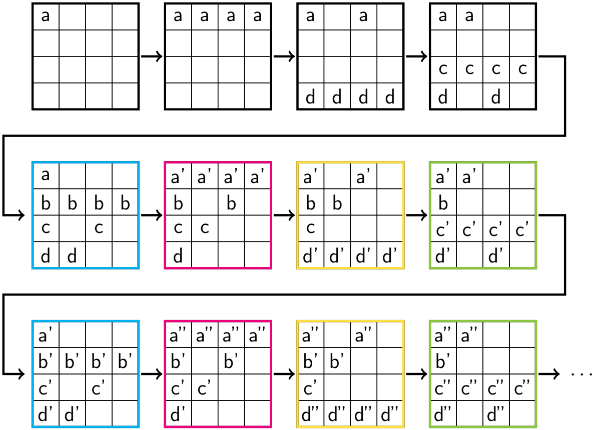
Diffusion of a difference on the first byte after several rounds of key schedule.

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.

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 ℓ
- we choose a message M made of m blocks, with $m > \ell$

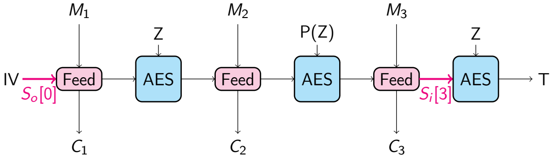
Forgery attack against mixFeed [Khairallah, ToSC'19]

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 ℓ
- we choose a message M made of m blocks, with $m > \ell$

(1) Cut



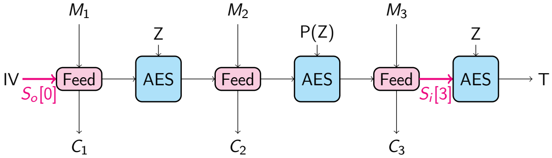
Forgery attack against mixFeed [Khairallah, ToSC'19]

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

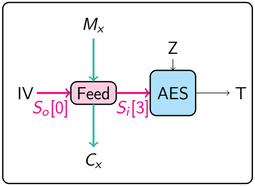
Khairallah proposed a forgery attack against mixFeed:

- we assume that Z belongs to a **cycle** of length ℓ
- 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

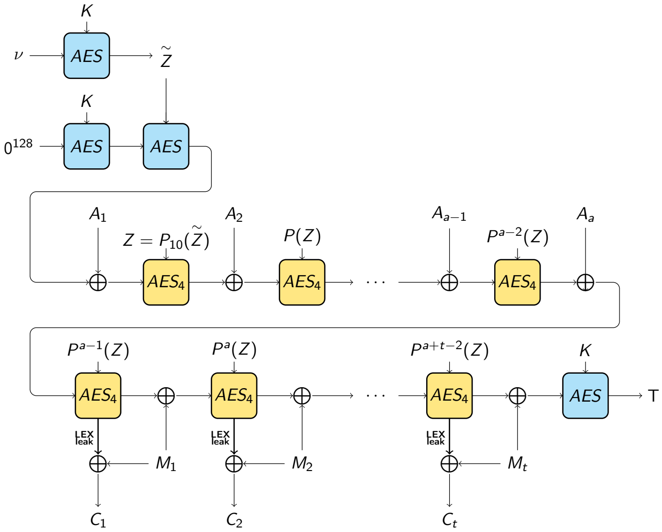
→ Memory complexity: negligible

→ Time complexity: negligible

→ Success rate: 45%

⇒ Verified using the mixFeed reference implementation

Application to ALE [BMRRT, FSE'13]



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 $16043203220 \approx 2^{33.9}$.

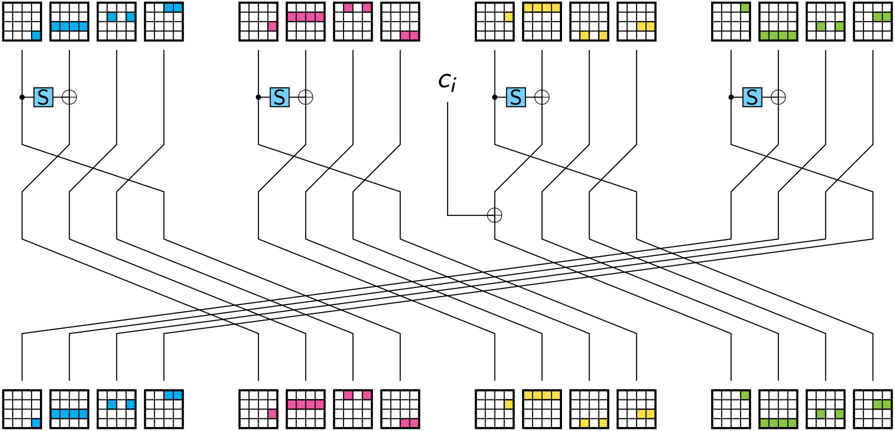
→ Short length cycles allows us to easily find states encrypted under the same key.

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

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

Comparison of attacks against ALE.

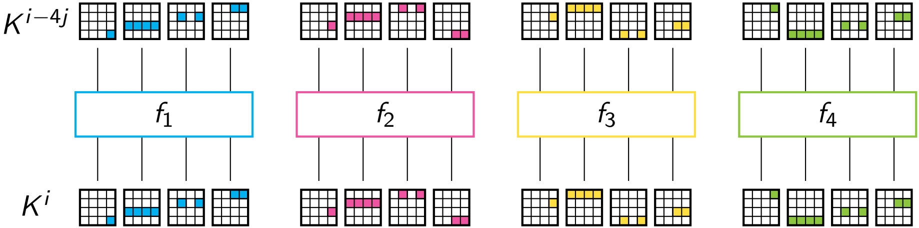
Property on the AES Key Schedule



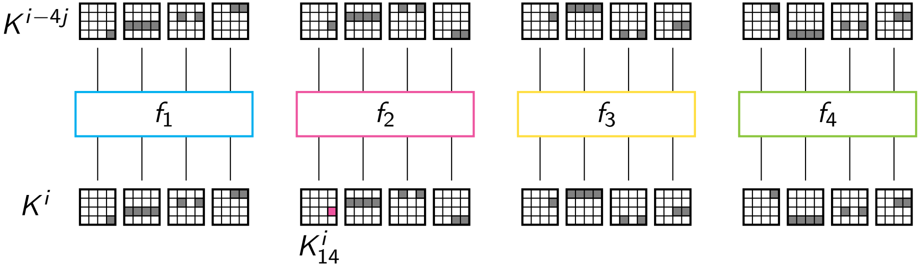
One round of the AES key schedule with graphic representations of bytes positions (alternative representation).

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

Property on the AES Key Schedule

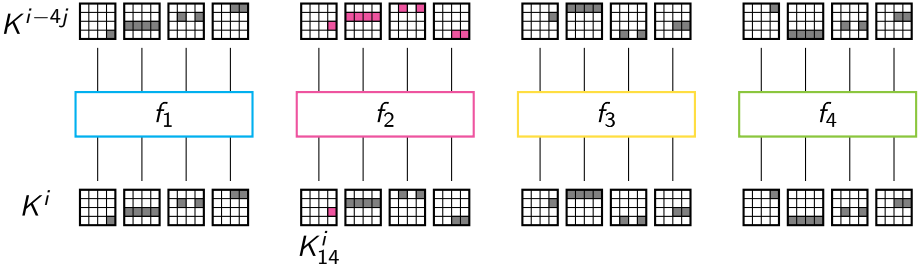


Property on the AES Key Schedule



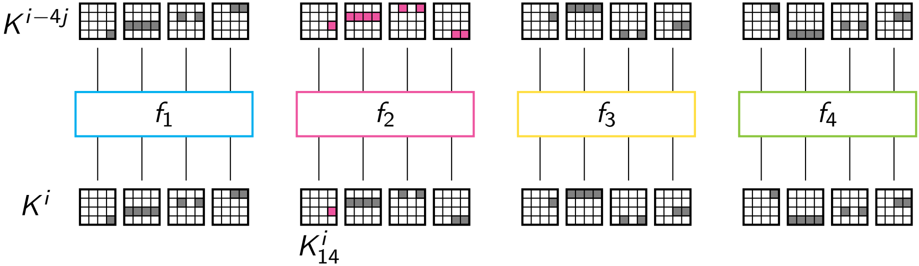
How to compute K^i_{14}?

Property on the AES Key Schedule



How to compute K^i_{14}?

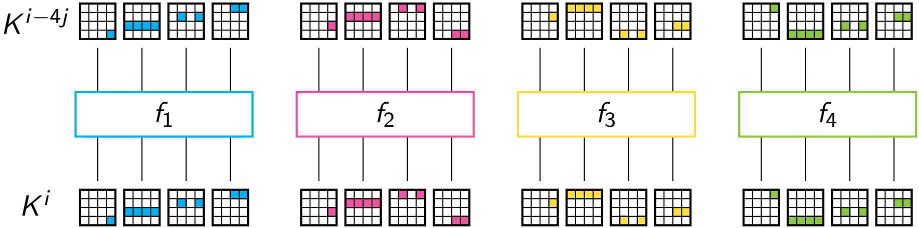
Property on the AES Key Schedule



How to compute K^i_{14}?

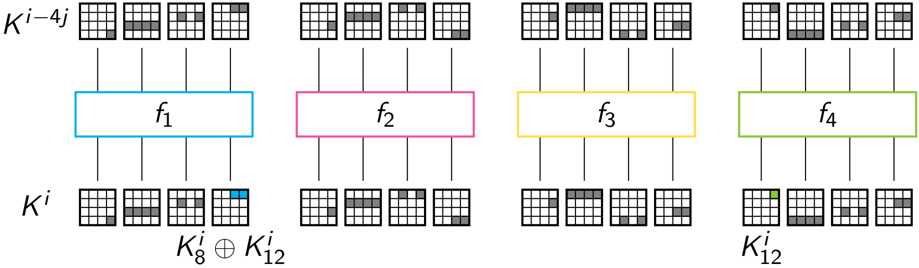
→ **A byte in the last column depends on only 32 bits of information.**

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

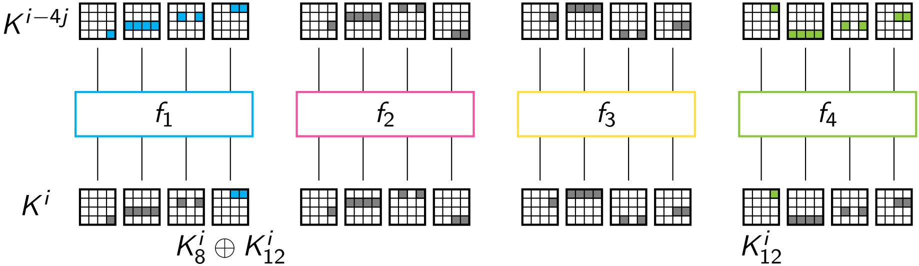


How to compute K^i_8?

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

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

Property on the AES Key Schedule

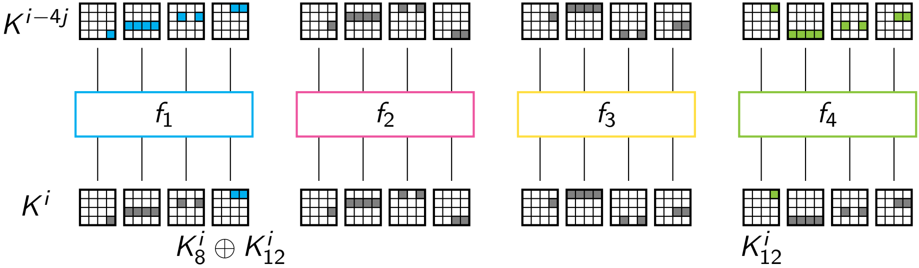


How to compute K_8^i?

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

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

Property on the AES Key Schedule



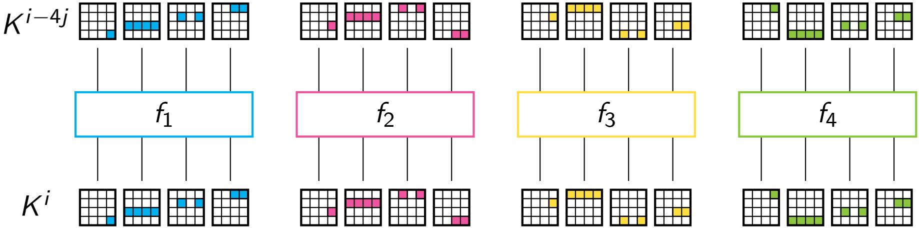
How to compute K_8^i?

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

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

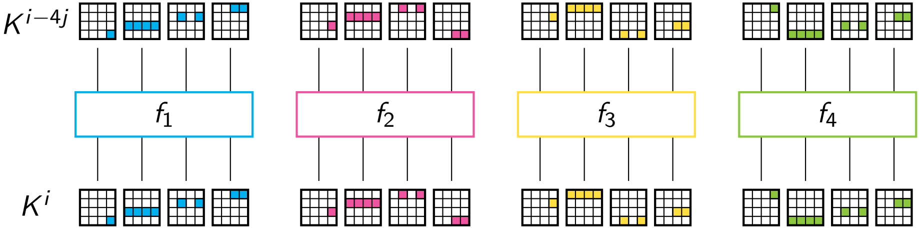
→ **A byte in the 3rd column depends on only 64 bits of information.**

Property on the AES Key Schedule



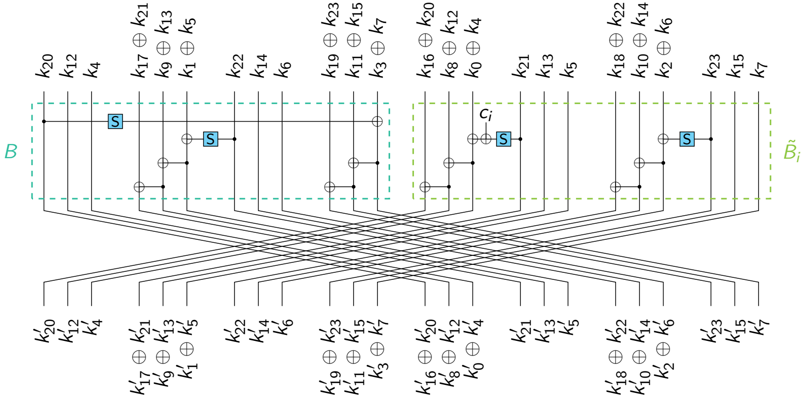
→ A byte in the last column depends on only 32 bits of information.
→ A byte in the 3rd column depends on only 64 bits of information.
→ **A byte in the 2nd column depends on only 64 bits of information.**

Property on the AES Key Schedule



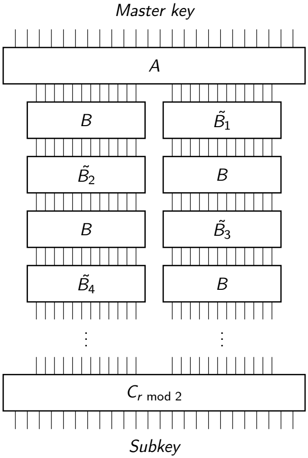
→ A byte in the last column depends on only 32 bits of information.
→ A byte in the 3rd column depends on only 64 bits of information.
→ A byte in the 2nd column depends on only 64 bits of information.
→ **A byte in the first column depends on 128 bits of information.**

New Representation of the AES-192 Key Schedules



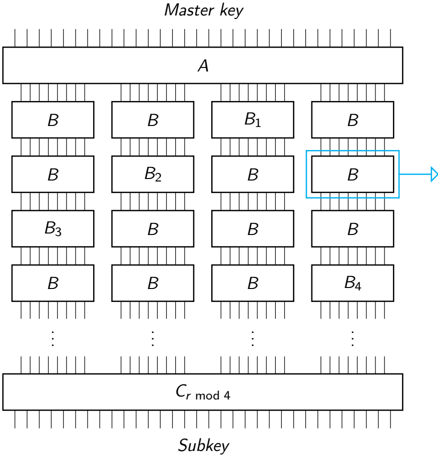
One round of the AES-192 key schedule (alternative representation).

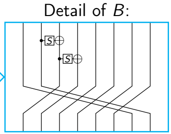
New Representation of the AES-192 Key Schedules



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

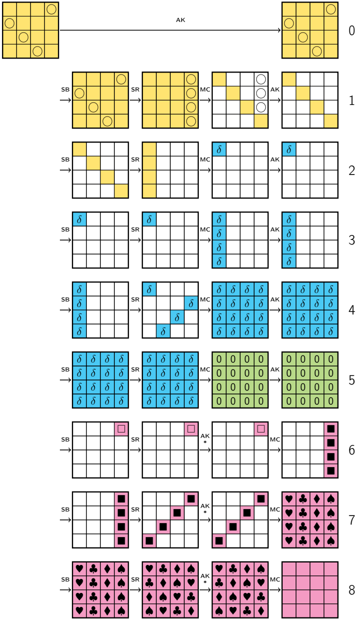
New Representation of the AES-256 Key Schedules





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



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

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]
	$2^{106.1}$	$2^{112.1}$	$2^{73.1}$	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.

Results

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]
	$2^{106.1}$	$2^{112.1}$	$2^{73.1}$	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.

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

Other Results

Attack	Cipher	Rounds	Data	Time	Reference
Square	AES-192	8/12	$2^{128} - 2^{119}$	2^{188}	[FKL+, FSE'00]
			$2^{128} - 2^{119}$	$2^{187.3}$	Variant of [FKL+, FSE'00]
			$2^{128} - 2^{119}$	$2^{185.7}$	Variant of [DKS, AC'10]
			$2^{128} - 2^{119}$	$2^{185.1}$	New
Related-Key Impossible Differential	AES-192	7/12	$2^{64.5}$	2^{177}	[ZWZ+, SAC'06]
			$2^{63.5}$	2^{175}	New
Impossible Differential	Rijndael-256/256	9/14	$2^{229.3}$	2^{194}	[WGR+, ICISC'12]
			$2^{228.1}$	$2^{192.9}$	Variant of [WGR+, ICISC'12]
			$2^{227.7}$	$2^{192.5}$	New
Impossible Differential	Rijndael-256/256	10/14	$2^{244.2}$	$2^{253.9}$	[WGR+, ICISC'12]
			$2^{243.9}$	$2^{253.6}$	Variant of [WGR+, ICISC'12]
			2^{243}	$2^{252.7}$	New