Improving integral attacks against Rijndael-256 up to 9 rounds

Samuel Galice[1] M. Minier[1]

[1]Laboratoire CITI, INSA de Lyon, France

24th october 2007

- **A Brief Outline of Rijndael-256**
- **Integral Properties**
  - The Original one
  - The new one
- **Attacking up to 9 rounds of Rijndael-256**
  - The 7 rounds attack
  - The 8 rounds attack
  - The 9 rounds attack
- **Conclusion**

A brief outline of Rijndael-256

- Rijndael-256: parallel and byte oriented block cipher with 14 rounds
- Block length = 256 bits. Keylength = 128, 192 or 256 bits
- Current block at input of round r = a 4×8 matrix of bytes:

$$A^{(r)} = \begin{pmatrix} a_{0,0}^{(r)} & a_{0,1}^{(r)} & a_{0,2}^{(r)} & a_{0,3}^{(r)} & a_{0,4}^{(r)} & a_{0,5}^{(r)} & a_{0,6}^{(r)} & a_{0,7}^{(r)} \\ a_{1,0}^{(r)} & a_{1,1}^{(r)} & a_{1,2}^{(r)} & a_{1,3}^{(r)} & a_{1,4}^{(r)} & a_{1,5}^{(r)} & a_{1,6}^{(r)} & a_{1,7}^{(r)} \\ a_{2,0}^{(r)} & a_{2,1}^{(r)} & a_{2,2}^{(r)} & a_{2,3}^{(r)} & a_{2,4}^{(r)} & a_{2,5}^{(r)} & a_{2,6}^{(r)} & a_{2,7}^{(r)} \\ a_{3,0}^{(r)} & a_{3,1}^{(r)} & a_{3,2}^{(r)} & a_{3,3}^{(r)} & a_{3,4}^{(r)} & a_{3,5}^{(r)} & a_{3,6}^{(r)} & a_{3,7}^{(r)} \end{pmatrix}$$

- The key schedule derives 15 256-bits round keys K_0 to K_{14} from the master key K

The round function F

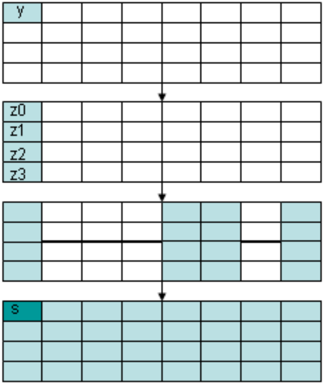
- The round function F repeats 13 times 4 mappings:
  - **SubBytes:** applies on each byte a non linear S-box S
  - **ShiftRows:** rotates on the left all the rows of the current matrix (0 for the first row, 1 for the second, 3 for the third and 4 for the fourth)
  - **MixColumns:** Each column of the input matrix is multiplied by the MixColumns matrix M
  - **AddRoundKey:** x-or between the block and the subkey K_r.

The round function F

- The round function F repeats 13 times 4 mappings:
  - **SubBytes:** applies on each byte a non linear S-box S
  - **ShiftRows:** rotates on the left all the rows of the current matrix (0 for the first row, 1 for the second, 3 for the third and 4 for the fourth)
  - **MixColumns:** Each column of the input matrix is multiplied by the MixColumns matrix M
  - **AddRoundKey:** x-or between the block and the subkey K_r.
- At the top, an initial key addition with K_0
- At the bottom, a final transformation = a round function without MixColumns.

The First integral property

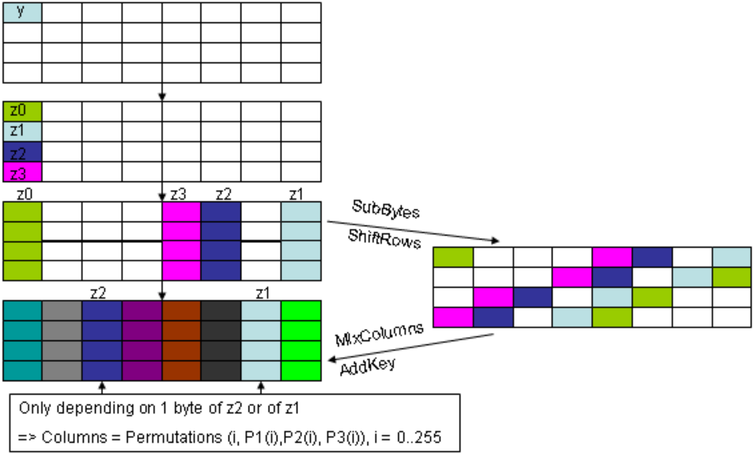
- introduced in [Rijndael - 99], works on three rounds using one active byte:



- $\bigoplus_{y \in \{0..255\}} s = 0$, for all bytes.

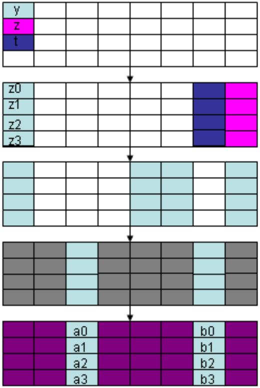
A three round property of Rijndael-256

- A stronger property for Rijndael-256: for the 3d and the 7th columns after 3 rounds, the distribution is always a set of permutations $(i, P1(i), P2(i), P3(i))$ with $i \in \{0, \cdots, 255\}$



How to use it ?

- On four rounds, by saturating more bytes to exploit the permutations at the end of the third round



- Then, $\forall i, \bigoplus_{y,z,t \in \{0..255\}} a_i = 0$ and $\bigoplus_{y,z,t \in \{0..255\}} b_i = 0$

The four rounds distinguisher

- Thus, you could use this equality to build a distinguisher between 4 Rijndael-256 rounds and a random permutation
  - testing if for a given i:

$$\bigoplus_{y,z,t\in\{0..255\}} a_i = 0 \quad (1)$$

or

$$\bigoplus_{y,z,t\in\{0..255\}} b_i = 0 \quad (2)$$

  - requiring $(256)^3$ plaintexts with three active bytes

The four rounds distinguisher

- Thus, you could use this equality to build a distinguisher between 4 Rijndael-256 rounds and a random permutation
  - testing if for a given i:

$$\bigoplus_{y,z,t\in\{0..255\}} a_i = 0 \qquad (1)$$

or

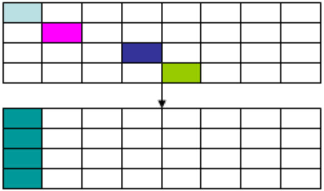
$$\bigoplus_{y,z,t\in\{0..255\}} b_i = 0 \qquad (2)$$

  - requiring $(256)^3$ plaintexts with three active bytes

- We perform some computer experiments which confirm the existence of such properties.

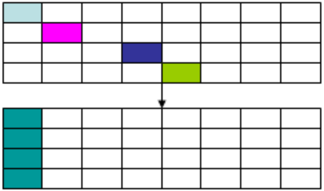
Extension by one round at the beginning

- as proposed in [Ferguson et al - 00], extension of the previous 4 rounds distinguisher by one round at the beginning
  - considering that we sum on all the 2^{32} plaintexts that represent 2^8 particular set with 3 active bytes



Extension by one round at the beginning

- as proposed in [Ferguson et al - 00], extension of the previous 4 rounds distinguisher by one round at the beginning
  - considering that we sum on all the 2^{32} plaintexts that represent 2^8 particular set with 3 active bytes



- Thus you could attack 5 rounds of Rijndael-256 using 2^{32} plaintexts and testing if always the equality (2) occurs for a particular i.

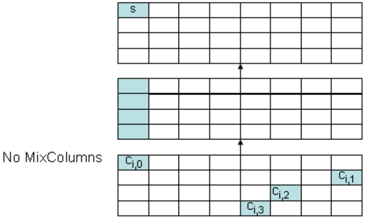
Extension by two rounds at the end

- [Ferguson et al - 00]: extension of the 5 rounds distinguisher by adding two rounds at the end

Extension by two rounds at the end

- [Ferguson et al - 00]: extension of the 5 rounds distinguisher by adding two rounds at the end
  - s directly deduced from the bytes $(c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3})$ by computing:

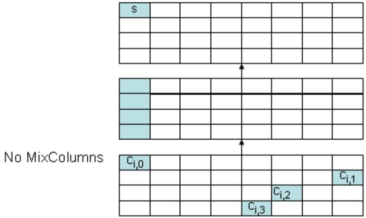
$$\bigoplus_i S^{-1}\left[S_0\left[c_{i,0} \oplus k_0\right] \oplus S_1\left[c_{i,1} \oplus k_1\right] \oplus S_2\left[c_{i,2} \oplus k_2\right] \oplus S_3\left[c_{i,3} \oplus k_3\right] \oplus k_4\right] \tag{3}$$



Extension by two rounds at the end

- [Ferguson et al - 00]: extension of the 5 rounds distinguisher by adding two rounds at the end
  - s directly deduced from the bytes $(c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3})$ by computing:

$$\bigoplus_i S^{-1}\left[S_0\left[c_{i,0} \oplus k_0\right] \oplus S_1\left[c_{i,1} \oplus k_1\right] \oplus S_2\left[c_{i,2} \oplus k_2\right] \oplus S_3\left[c_{i,3} \oplus k_3\right] \oplus k_4\right] \quad (3)$$



- Associate the partial sum to each ciphertext c:

$$x_k := \sum_{j=0}^{k} S_j\left[c_j \oplus k_j\right] \text{ for } k \text{ from } 0 \text{ to } 3$$

- Use $(c_0, c_1, c_2, c_3) \longrightarrow (x_k, c_{k+1}, \cdots, c_3)$ to sequentially determine k_k
- Share the global computation in 4 steps

The seven rounds attack

- Using the two extensions, build a 7 rounds attack:
  - Cipher a set of 2^{32} plaintexts with 4 active bytes at good positions

The seven rounds attack

- Using the two extensions, build a 7 rounds attack:
  - Cipher a set of 2^{32} plaintexts with 4 active bytes at good positions
  - Decipher the two last rounds using the partial sum technique for four bytes of K_7 and one byte of K'_6

The seven rounds attack

- Using the two extensions, build a 7 rounds attack:
  - Cipher a set of 2^{32} plaintexts with 4 active bytes at good positions
  - Decipher the two last rounds using the partial sum technique for four bytes of K_7 and one byte of K'_6
  - test if equality (2) occurs at the end of the 5th round

The seven rounds attack

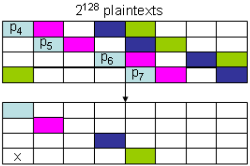
- Using the two extensions, build a 7 rounds attack:
  - Cipher a set of 2^{32} plaintexts with 4 active bytes at good positions
  - Decipher the two last rounds using the partial sum technique for four bytes of K_7 and one byte of K'_6
  - test if equality (2) occurs at the end of the 5th round
- **Total cost:**
  - For a set of 2^{32} ciphertexts, cost of the four steps $\approx 2^{50}$ S-box lookups.
  - Repeat the process with 6 different sets of 2^{32} ciphertexts to detect false alarms
  - The total number of S-box lookups is $2^{52} \approx 2^{44}$ encryptions (considering that 2^8 S-box applications $\approx$ one trial encryption).

The herd techniques

- [Ferguson et al - 00]: extension by one more round at the top

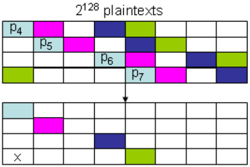
The herd techniques

- [Ferguson et al - 00]: extension by one more round at the top
  - Naively require 2^{128} plaintexts: divide into 2^{96} packs of 2^{32} sets with 4 active bytes but in this case wrong keys pass the test !
  - Instead, use a particular byte x with a fixed value. Obtain a herd a set of 2^{120} possible encryptions composed of 2^{88} structures.
  - Test equality (2) on a herd, only correct key !



The herd techniques

- [Ferguson et al - 00]: extension by one more round at the top
  - Naively require 2^{128} plaintexts: divide into 2^{96} packs of 2^{32} sets with 4 active bytes but in this case wrong keys pass the test !
  - Instead, use a particular byte x with a fixed value. Obtain a herd a set of 2^{120} possible encryptions composed of 2^{88} structures.
  - Test equality (2) on a herd, only correct key !



- x depends on 4 plaintext bytes $(p_4, \cdots, p_7)$ and on 4 K_0 bytes
- Share the key exhaustive search on the 4 K_0 bytes in a 3-phase attack
  - **First phase:** using 2^{64} counters m_y
  - **2nd phase:** 2^{32} counters n_z
  - **3d phase:** filter information for key guesses

The 8 rounds attack

- **the attack:**
  - **First phase:** Increment m_y at bit level according the 64-bit value $y = (c_0, \cdots, c_3, p_4, \cdots, p_7)$

The 8 rounds attack

- **the attack:**
  - **First phase:** Increment m_y at bit level according the 64-bit value $y = (c_0, \cdots, c_3, p_4, \cdots, p_7)$
  - **2nd phase:** Guess the 4 K_0 bytes, compute x, share the counters into herds, select a single herd, update n_z by adding $z = (c_0, \cdots, c_3)$ for each y that is in the good herd

The 8 rounds attack

- **the attack:**
  - **First phase:** Increment m_y at bit level according the 64-bit value $y = (c_0, \cdots, c_3, p_4, \cdots, p_7)$
  - **2nd phase:** Guess the 4 K_0 bytes, compute x, share the counters into herds, select a single herd, update n_z by adding $z = (c_0, \cdots, c_3)$ for each y that is in the good herd
  - **3d phase:** guess the five key bytes of K_7 and of K'_6 on the two last rounds to decrypt each z to a single byte a^6, sum this byte over all the 2^{32} values of z (with multiplicities) and check for zero. Repeat it for each guess of the four K_0 bytes

The 8 rounds attack

- **the attack:**
  - **First phase:** Increment m_y at bit level according the 64-bit value $y = (c_0, \cdots, c_3, p_4, \cdots, p_7)$
  - **2nd phase:** Guess the 4 K_0 bytes, compute x, share the counters into herds, select a single herd, update n_z by adding $z = (c_0, \cdots, c_3)$ for each y that is in the good herd
  - **3d phase:** guess the five key bytes of K_7 and of K_6' on the two last rounds to decrypt each z to a single byte a^6, sum this byte over all the 2^{32} values of z (with multiplicities) and check for zero. Repeat it for each guess of the four K_0 bytes

  - **A trick:** The 4 plaintext bytes $(p_4, \cdots, p_7)$ and the four K_0 bytes provide four bytes. => Create 2^{24} smaller herds with 2^{104} elements by fixing three more bytes. => Reduce to $2^{128} - 2^{119}$ plaintexts

The 8 rounds attack

- **the attack:**
  - **First phase:** Increment m_y at bit level according the 64-bit value $y = (c_0, \cdots, c_3, p_4, \cdots, p_7)$
  - **2nd phase:** Guess the 4 K_0 bytes, compute x, share the counters into herds, select a single herd, update n_z by adding $z = (c_0, \cdots, c_3)$ for each y that is in the good herd
  - **3d phase:** guess the five key bytes of K_7 and of K'_6 on the two last rounds to decrypt each z to a single byte a^6, sum this byte over all the 2^{32} values of z (with multiplicities) and check for zero. Repeat it for each guess of the four K_0 bytes

  - **A trick:** The 4 plaintext bytes $(p_4, \cdots, p_7)$ and the four K_0 bytes provide four bytes. => Create 2^{24} smaller herds with 2^{104} elements by fixing three more bytes. => Reduce to $2^{128} - 2^{119}$ plaintexts

- **Total cost:** $2^{128} - 2^{119}$ trial encryptions + 2^{120} trial encryptions for the attack itself

The 9 rounds attack [Ferguson et al - 00] (1/2)

- **Add one more round at the end:** using partial sum techniques + exhaustive search of the 16 bytes of K_9
- Need to guess four K_0 bytes + 21 subkey bytes (16 bytes of K_9, 4 bytes of K_8 and one byte of K_7') to add three rounds at the end of the 5 rounds distinguisher.

The 9 rounds attack [Ferguson et al - 00] (1/2)

- **Add one more round at the end:** using partial sum techniques + exhaustive search of the 16 bytes of K_9
- Need to guess four K_0 bytes + 21 subkey bytes (16 bytes of K_9, 4 bytes of K_8 and one byte of K'_7) to add three rounds at the end of the 5 rounds distinguisher.
- Then the attack works as follows:
  - construct 2^{23} undamaged herds of 2^{104} elements using $2^{128} - 2^{119}$ plaintexts
  - guess the four key bytes of K_0 to determine a particular herd
  - apply the partial sum technique to this set and obtain a single byte of $A^{(7)}$ depending on 16 bytes of the ciphertext and 21 subkey bytes
  - Use the fact that summing the 2^{104} values on a single byte of $A^{(7)}$ will yield zero (from equality (2)) for the good key

The 9 rounds attack [Ferguson et al - 00] (2/2)

- **Total cost**:
  - required storage is about 2^{104} bits
  - Total complexity about $2^{32} \cdot 2^{170} = 2^{202}$ trial encryptions for one herd and a 256-bit key.
  - We need to test four herds before discarding the first bad keys and at least 26 herds to get exactly the good key (with a decreasing complexity).
  - Total complexity $\approx 2^{204}$ trial encryptions.

The 9 rounds attack [Ferguson et al - 00] (2/2)

- **Total cost**:
  - required storage is about 2^{104} bits
  - Total complexity about $2^{32} \cdot 2^{170} = 2^{202}$ trial encryptions for one herd and a 256-bit key.
  - We need to test four herds before discarding the first bad keys and at least 26 herds to get exactly the good key (with a decreasing complexity).
  - Total complexity $\approx 2^{204}$ trial encryptions.

  - In the case of a 192-bit key, weakness of the key-schedule [Lucks - 00], we preserve 2 bytes of K_9 determined by the 14 others. Total complexity $\approx 2^{204-16} = 2^{188}$ trial encryptions

Conclusion

- New particular integral property on 4 rounds of Rijndael-256
- leads to the best known attack against a 9 rounds version of Rijndael-256 requiring for a 192-bit keys 2^{188} trial encryptions with $2^{128} - 2^{119}$ plaintexts.

Conclusion

- New particular integral property on 4 rounds of Rijndael-256
- leads to the best known attack against a 9 rounds version of Rijndael-256 requiring for a 192-bit keys 2^{188} trial encryptions with $2^{128} - 2^{119}$ plaintexts.

- No way to extend related key rectangle attacks against Rijndael-256: the number of 32-bit key words that must be generated to construct 256-bit subkeys is higher: we do not find a key pattern that sufficiently preserves an integral property.

Comparative Table

Cipher	nb rounds	Key size	Data	Time Complexity	source
AES	6	(all)	2^{32} CP	2^{72}	[DR98] (Integral)
	7	(all)	$2^{128}-2^{119}$ CP	2^{120}	[Ferg.] (Part. Sum)
	8	(192)	$2^{128}-2^{119}$ CP	2^{188}	[Ferg.] (Part. Sum)
	8	(256)	$2^{128}-2^{119}$ CP	2^{204}	[Ferg.] (Part. Sum)
	9	(256)	2^{85} RK-CP	2^{224}	[Ferg.] (Related-key)
	9	(192)	2^{86} RK-CP	2^{125}	[BihamDK05] Related-key Rectangle
	10	(256)	$2^{114.9}$ RK-CP	$2^{171.8}$	[BihamDK05] Related-key Rectangle
Rijndael-192	6	(all)	2^{32} CP	2^{72}	[DR98] (Integral)
	7	(all)	$2^{128}-2^{119}$ CP	$2^{128}-2^{119}$	[Ferg.] (Part. Sum)
	8	(192)	$2^{128}-2^{119}$ CP	2^{188}	[Ferg.] (Part. Sum)
	8	(256)	$2^{128}-2^{119}$ CP	2^{204}	[Ferg.] (Part. Sum)
Rijndael-256	6	(all)	2^{32} CP	2^{72}	[DR98] (Integral)
	7	(all)	$2^{128}-2^{119}$ CP	$2^{128}-2^{119}$	[Ferg.] (Part. Sum)
	8	(all)	$2^{128}-2^{119}$ CP	$2^{128}-2^{119}$	this paper
	9	(192)	$2^{128}-2^{119}$ CP	2^{188}	this paper
	9	(256)	$2^{128}-2^{119}$ CP	2^{204}	this paper

Table: Summary of Attacks on Rijndael-b - CP: Chosen plaintexts, RK: Related-key