Introduction to Cryptology

4.3 - AES

Federico Pintore

Mathematical Institute, University of Oxford (UK)

The Advanced Encryption Standard (AES)

- In 1997, the US agency NIST[1] opened a competition for a new block cipher, to be called AES.
- In 2000, Rijndael, a new block cipher designed by Vincent Rijmen and Joan Daemen, won the competition.
- The are three standardised versions of AES. For each of them, the block length ℓ is 128.
- The key length n can be either 128 (AES-128), 192 (AES-192) or 256 (AES-256).

[1] National Institute of Standards and Technology

The Advanced Encryption Standard (AES)

- Given an input $x = (x_0, \ldots, x_{15})$, where each x_i is a byte, a 4x4 matrix s is initialised as:

$$s = \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}.$$

- The number of rounds Nr depends on the key length: Nr is 10 for AES-128, 12 for AES-192 and 14 for AES-256.

- The key schedule takes the key k and constructs $Nr + 1$ 4x4 matrices, $sk_0, \ldots, sk_{Nr}$, where each entry is a byte.

The Advanced Encryption Standard (AES)

Input : x, Nr, $(sk_0, \ldots, sk_{Nr})$.

Output : y.

$s = (x_0, x_4, x_8, x_{12}; x_1, x_5, x_9, x_{13}; x_6, x_8, x_{10}, x_{14}; x_3, x_7, x_{11}, x_{15})$

$s = s \oplus sk_0$

for $j = 1, \ldots, Nr - 1$;

$s = SubBytes(s)$

$s = ShiftRows(s)$

$s = MixColumns(s)$

$s = s \oplus sk_j$

$s = SubBytes(s)$

$s = ShiftRows(s)$

$s = s \oplus sk_{Nr}$

return s

AES - SubBytes

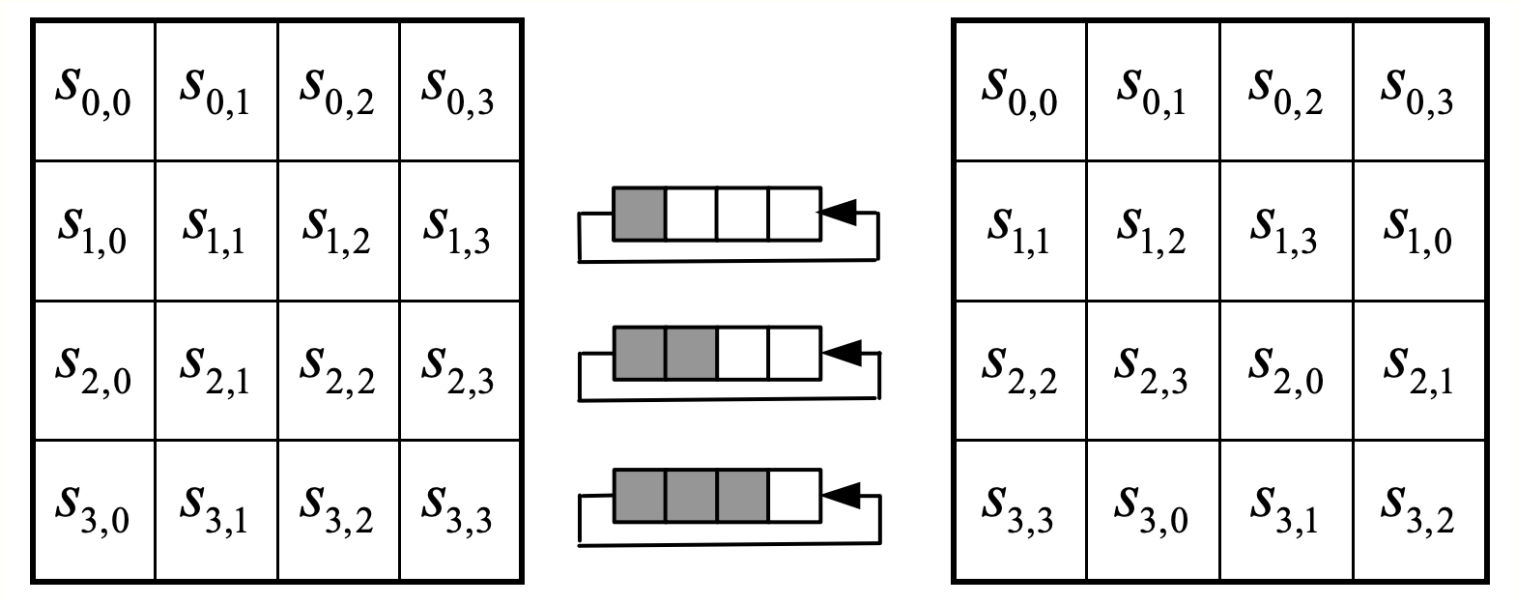
It operates independently on each byte of s.

A byte, represented by two hexadecimal digits UV, is sent into the byte of the cell in row U and column V.

		v														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
u	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES - ShiftRows

The bytes in the last three rows of the state s are cyclically shifted, while the first row remains unchanged:



AES - MixColumns

It operates on the state s column-by-column, transforming each column by means of the matrix transformation:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Given a byte $b = (b_0, b_1, \ldots, b_7)$, we have that:

- $(01) \cdot b = b$;
- $(02) \cdot b = (b_7, b_0 + b_7, b_1, b_2 + b_7, b_3 + b_7, b_4, b_5, b_6)$;
- $(03) \cdot b = (b_7 + b_0, b_0 + b_7 + b_1, b_1 + b_2, b_2 + b_7 + b_3, b_3 + b_7 + b_4, b_4 + b_5, b_5 + b_6, b_6 + b_7)$.

AES - Key schedule

The key k is divided into 4-byte words $k_1, \ldots, k_{Nk}$.

The functions/conventions used are:

- SubWord: takes a 4-byte input word and applies the S-box to each of the four bytes.
- RotWord: takes a 4-byte word (a_0, a_1, a_2, a_3) as input and returns (a_1, a_2, a_3, a_0).
- Rcon[i] is the 4-byte word $(x^{i-1}, 00, 00, 00)$, where x^{i-1} is the byte corresponding to the reduction of x^{i-1} modulo $x^8 + x^4 + x^3 + x + 1$ (the reduction is a polynomial of degree smaller than 8 in $\mathbb{F}_2[x]$).

AES - Key schedule

Input : $k = (k_1, \ldots, k_{Nk})$, Nr.

Output : $sk_0, \ldots, sk_{Nr}$.

$w = [], \text{temp} = [], i = 0$

while $i < Nk$

  $w[i] = k_{i+1}$

  $i = i + 1$

$i = Nk$

while $i < 4(Nr + 1)$

  $\text{temp} = w[i - 1]$

  if $i \equiv 0 \pmod{Nk}$

    $\text{temp} = \text{SubWord}(\text{RotWord}(\text{temp})) \oplus \text{Rcon[i/Nk]}$

  elseif $Nk > 6 \wedge i \equiv 4 \pmod{Nk}$

    $\text{temp} = \text{SubWord}(\text{temp})$

  $w[i] = w[i - Nk] \oplus \text{temp}$

  $i = i + 1$

return w

Security of AES

- No practical attacks that are notably better than brute-force search for the key.
- A practical instantiation of a (strong) pseudo-random permutation.
- Free, standardised, and efficient.

Further Reading I


Nadhem J AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt.
On the security of RC4 in TLS.
In 22nd USENIX Security Symposium (USENIX Security 13), pages 305–320, 2013.


Boaz Barak and Shai Halevi.
A model and architecture for pseudo-random generation with applications to/dev/random.
In Proceedings of the 12th ACM conference on Computer and communications security, pages 203–212. ACM, 2005.


Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway.
A concrete security treatment of symmetric encryption.
In Proceedings 38th Annual Symposium on Foundations of Computer Science, 1997, pages 394–403, 1997.

Further Reading II


Daniel J Bernstein.
The Salsa20 Family of Stream Ciphers.
In New stream cipher designs, pages 84–97. Springer, 2008.


Lenore Blum, Manuel Blum, and Mike Shub.
A simple unpredictable pseudo-random number generator.
SIAM Journal on computing, 15(2):364–383, 1986.


Christian Cachin.
Entropy measures and unconditional security in cryptography.
PhD thesis, ETH Zurich, 1997.


Scott Fluhrer, Itsik Mantin, and Adi Shamir.
Weaknesses in the key scheduling algorithm of RC4.
In Selected areas in cryptography, pages 1–24. Springer, 2001.

Further Reading III

Christina Garman, Kenneth G Paterson, and Thyla Van der Merwe.
Attacks only get better: Password recovery attacks against RC4 in TLS.
In 24th USENIX Security Symposium (USENIX Security 15), pages 113–128, 2015.

Itsik Mantin and Adi Shamir.
A practical attack on broadcast RC4.
In Fast Software Encryption, pages 152–164. Springer, 2002.